International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 11 November-2026 | Impact Factor: 3.5



An Intelligent SIEM–SOAR Framework Using Isolation Forest

Deevi Radha Rani, B. Varshitha, Ch. Giri Manish Kumar, D. Tejaswini, B. Naveen Kumar
Department of Computer Science and Engineering, GMRIT Deemed to be University, Rajam, India

How to Cite this Article:

Rani, D. R., Varshitha, B., Kumar, C. G. M., Tejaswini, D. & Kumar, B. N. (2026). An Intelligent SIEM–SOAR Framework Using Isolation Forest. International Journal of Creative and Open Research in Engineering and Management, 02(11), 1-9. https://doi.org/10.55041/ijcope.v2i11.049

License:

This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.







https://doi.org/10.55041/ijcope.v2i11.049

Abstract : Security Operations Centres (SOCs) are flooded with huge numbers of events flowing from network and authentication sources, conventional monitoring based on rules fails to cope with unusual activity, and complete automation of response can lead to disruption of legitimate services. This paper proposes a Intelligent SIEM-SOAR framework which integrates unsupervised anomaly detection and human supervised incident response. Three data sources are ingested: CICIDS2017 network flows, Suricata EVE-format intrusion events and Linux SSH authentication logs. The CICIDS2017 records are pre-processed using the following steps: cleaned, reduced to 12 behavioural features, standardized and scored by an Isolation Forest trained on a benign baseline. The detections are then transformed into structured alerts with anomaly scores, severity ratings, managed using an Elasticsearch–Logstash–Kibana (ELK) configuration, and enriched with MITRE ATT&CK techniques, orchestrated via a Shuffle-compatible orchestration client. A React dashboard allows a SOC analyst to view alerts, mark a positive or negative outcome, and approve or disapprove proposed containment – actions approved are executed only in a simulated sandbox and added to an audit trail, which can be used later for retraining. Consideration for deployment of a candidate Isolation Forest model involves a promotion gate to test the candidate model against the baseline. The precision of the baseline model on the bundled test subset was 84.27 %, recall 100 %, F1 score 91.46 % and accuracy 94.40 % on a bundled test set of 500 test and 500 baseline records. The reported candidate model evaluation gave an F1-score of 94.04 % and achieved a decrease in false positives from 28 to 19 while preserving the recall rate. But the archive corresponding to the same evaluation artefact recorded no analyst feedback samples used, so it cannot be said that the improvement was due to analyst feedback.

IndexTerms - SIEM, SOAR, Isolation Forest, Anomaly detection, CICIDS2017, Security Operations Centre, Suricata, MITRE ATT&CK.

1. INTRODUCTION

The Security Operations Centre (SOC) tracks network traffic, authentication activity, and intrusion-detection output, looking for suspicious behaviour and providing a response to cyber threats. All of these sources generate a vast number of events, and doing a manual investigation of each event is slow and tedious. Security Information and Event Management (SIEM) platforms gather and correlate the events, with Security Orchestration, Automation and Response (SOAR) platforms performing the follow-up workflow in an automated way.

Traditional monitoring relies on pre-defined rules and signatures. Known threats are easily recognised by such rules, but unusual behaviours, not matching any of the rules, are difficult to detect. To fill this gap, unsupervised anomaly detection is introduced, which includes the Isolation Forest algorithm [15] which isolates observations that are few and different from a learned baseline without requiring labelled attacks for training. Anomaly detectors, however, trigger false alarms, and automatic blocking of traffic based on an unverified score can affect valid service traffic. Thus, recent research on alert fatigue, trust and human-AI collaboration in SOCs [3, 6, 10] supports the view that decisions that have consequences should be made by human agents with calibrated eyes.

The aim of this work is to develop a prototype of a Intelligent SIEM–SOAR based on that. Network flows are ingested, normalised and scored using an Isolation Forest; detections are structured to become alerts and managed using a SIEM configuration with ELK with MITRE ATT&CK [18] techniques and staged using a Shuffle-compatible orchestration client [20]. An analyst is presented with the alerts on a web dashboard, and must approve or disapprove each proposed response. Approval of the containment is only done in a simulated sandbox, every decision made is recorded on an audit trail, and analyst verdicts are stored for offline retraining of a candidate model that must meet an evaluation gate before it can be promoted.

To develop a Intelligent SIEM–SOAR framework for monitoring and analysing security related events; and to preprocess the CICIDS2017 network traffic and extract behavioural features for anomaly detection, implement the Isolation Forest algorithm to identify unusual pattern in network traffic, to generate structured security alerts and to give a dashboard to investigate and monitor the security alerts with analyst approval, containing and auditing the alerts and retraining the candidate model offline.

The remaining paper is organised as follows. In Section II, related work is briefly reviewed, and the research gap is outlined. In Section III, the methodology is detailed, including how the process will unfold, the system architecture, the working procedure, the models and technologies, human-supervised response and retraining mechanism, and the data and evaluation plan. The results and discussion are presented in Section IV along with the limitations and the conclusion is presented in Section V.



2. LITERATURE REVIEW

The shift in research of SIEM platforms has been from detection of single events to utilization of context. Alboushy et al. [1] describe attacker behaviour using enrichment in an open source SIEM and find that attacks followed a multi-stage web attack are better detected, with fewer false positives, with the approach limited to web-based attacks. Eckhoff et al. [7] model SOC alerts and network entities as graphs to make it easier for analysts to see how alerts are related to each other, thereby facilitating the visualisation and investigation of attack paths; however, the construction of the graph is complicated.

The second set of studies focuses on alert fatigue and triage. Ndichu et al. [3] suggest that an analyst's feedback be used to improve the ranking of rare but critical alerts, while reducing the workload of the analyst in low-prevalence alert streams, a method of triggered active learning. Chowdhury and Tanvir [4] correlate AI trust signals with analyst decision and Turcotte et al. [9] categorize and prioritize alerts automatically to reduce the response time but do not explicitly explain each decision. Singh et al. [13] suggests context during investigation and finds that HIKARI-2021+UNSW-NB15 data with 13 analysts resulted in 21.1 % higher investigation accuracy and 24 % quicker alert validation. Sahay et al. [2] use a large language model, along with security policies, to accelerate threat hunting during Splunk triage. The survey of Ndichu et al. [6] provides an overview of machine-learning, deep-learning, large-language-model, and human-supervised approaches for alert screening and discusses explainable AI and human-supervised operation as recent trends.

Human–AI collaboration is the theme of a third group. The hybrid model of deferring uncertain alerts to analysts and learning from their response was used by Jalalvand et al. [8] and the concept of trusted autonomy with calibrated human supervision is described by Mohsin et al. [10] and Aref et al. [12] used cognitive-hierarchy deep reinforcement learning to model different levels of analyst expertise in cloud security, primarily in simulation.

Lastly, some studies automate the answer. Zhang et al. [5] implement privacy-preserving response playbooks in distributed networks, Ismail et al. [11] use agentic AI to enhance SOAR orchestration for adaptive and autonomous workflows, but with limited explainability, and Liu [14] explores the interaction of multiple agents built on LLM for incident response, albeit based in a tabletop simulation that still requires real-world validation. Table 1 summarises these works.

Table 1: Related work on SOC automation and its limitations

Ref.	Focus of the work	Limitation	Research gap
[1]	Web-attack detection with MITRE ATT&CK behavioural profiling	Limited to web-based attacks	No complete cloud SOC automation
[2]	LLM-assisted Splunk alert investigation	Depends on predefined policies	Limited human-feedback integration
[3]	Alert-fatigue reduction by triggered active learning	Needs labelled analyst feedback	Not integrated with SOAR response
[4]	Aligning AI trust signals with analyst decisions	Focuses mainly on trust signals	No automated incident response
[5]	Response automation with SOAR playbooks	Works only with predefined playbooks	Limited intelligent reasoning
[6]	Survey of AI methods for SOC alert screening	Survey only, no implementation	No unified practical framework
[7]	Connecting related alerts using graphs	Graph creation is complex	No automated response capability
[8]	Alert prioritisation by learning to defer	Requires continuous analyst feedback	No complete SIEM–SOAR implementation
[9]	Automatic alert classification (AACT)	Limited contextual understanding	Cannot explain AI decisions
[10]	Trusted human–AI collaboration framework	Mostly conceptual	Lacks real-time implementation
[11]	Agentic-AI SOAR orchestration	Complex deployment	Limited explainability
[12]	Cloud security by deep reinforcement learning	Tested mainly in simulation	No SIEM integration
[13]	Contextual recommendations for analysts (ContextBuddy)	Learns from previous investigations only	Limited to contextual assistance
[14]	LLM multi-agent incident response	Simulation-based evaluation	Requires real-world validation

The majority of the studies reviewed address only a single aspect of the problem, including detection, prioritisation, contextualisation, or response, with the survey [6] highlighting the lack of a single integrated framework that integrates AI, explainability, human feedback and automated incident response into a single SOC system. Thus, the research gap is a complete and verifiable pipeline from multi-source telemetry to unsupervised detection to SIEM alert management to SOAR orchestration to the analyst's hand to a safe feedback loop. The present work creates such a prototype and test it using curated data sets.

3. METHODOLOGY

3.1 Workflow architecture

The workflow of the framework is shown in Fig. 1 and has seven stages. (1) Security data collection gathers CICIDS2017 network traffic, Suricata EVE logs and SSH authentication logs. (2) Data preprocessing cleans the data, handles missing values and extracts features. (3) Anomaly detection scales the features and applies the Isolation Forest model to identify anomalies. (4) SIEM alert generation creates alerts, classifies their severity and stores them. (5) human-supervised review lets the analyst review the alerts and approve or reject proposed actions. (6) Response and audit executes approved actions in simulation and records the

decisions in audit logs.
(7) Model evaluation trains a candidate model and evaluates its performance.

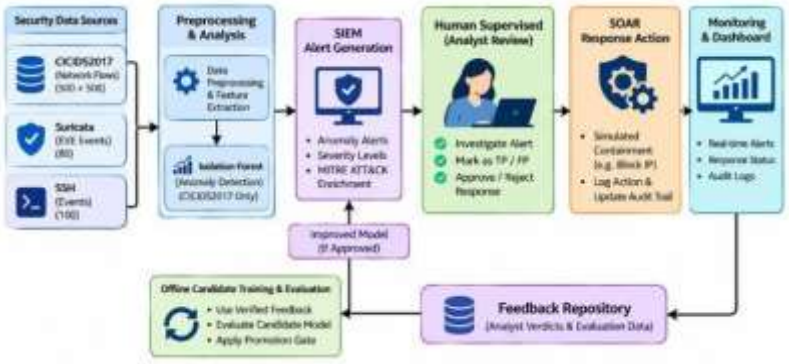


Fig. 1: Workflow of the Intelligent SIEM–SOAR framework (seven stages from security data collection to model evaluation)

3.2 System architecture

The architecture of the framework is the same as the 6-layer one shown in Fig. 2. The CICIDS2017 network traffic, Suricata EVE logs and SSH authentication events make up the L1 (data sources). The second layer (data processing) is implemented in Python using the libraries Pandas [24] and NumPy [25] to process the data, validate it and extract the features. An anomaly score and severity class are generated at Layer 3 (machine learning) using a model called Isolation Forest. Layer 4 (SIEM alert management) is set up with Elasticsearch, Logstash and Kibana [21]. Layer 5 (SOAR and human supervision): It is a combination of a shuffle-compatible workflow client [20] as well as analyst approval (approve or reject) and Mitre ATT&CK enrichment [18]. Layer 6 (audit and model management) provides audit logs of analyst decisions and response actions, and offline candidate-model training and evaluation. The verified feedback from the analysts can be used in the following training cycles, provided that the evaluator and/or the promoter decides to use it. The layered design ensures that detection, alert management and response remain separate, and the interactive dashboard is created using React
[22] with TypeScript and Vite on the backend, running on Express [23].

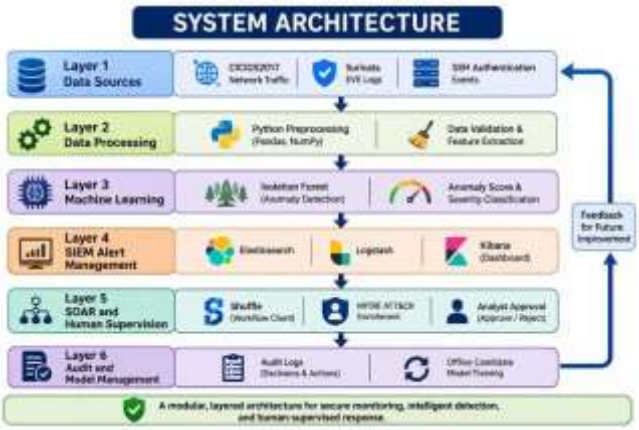


Fig. 2: six-layer architecture of the framework, from data sources to audit and model management, with a feedback path

3.3 Working procedure

1. Data collection: The implementation is comprised of network traffic data from 1) CICIDS2017, and events from Suricata 2) EVE-format events, and 3) SSH authentication events. CICIDS2017 is utilized for the network-traffic analysis by the Isolation Forest, and the Suricata and SSH files are utilized for the security-event analysis.

2. Data preprocessing: the network-traffic records are validated, converted into numerical values and cleaned by handling missing and infinite values.

3. Feature Extraction: Twelve behavioural network-traffic features are chosen for machine-learning analysis. The model does not include labels or identifying fields.

4. Model training and anomaly detection: The numerical features are scaled using a StandardScaler, and the IsolationForest model is trained on the benign baseline data and detects the anomalies in the test data.

5. Alert generation: Structured security alerts generated from the network anomalies detected as well as additional security events. Every alert includes event details, the anomaly score (if applicable) and a severity rating.

6. Alert monitoring: The web dashboard displays the alerts to investigate, and the SIEM configuration files offer the Elasticsearch, Logstash and Kibana components.

7. Response is human-supervised: proposed response actions are reviewed by analysts who approve/disapprove. There are a series of approved containment actions that are carried out in a simulated environment.

8. Audit and feedback management: Analyst decisions and response activities recorded, offline candidate-model training and evaluation supported.



3.4 Models and technologies used

The Isolation Forest [15] constructs an ensemble of random trees which recursively partition the feature space by using a random feature and split value. The anomalies are rare and different, hence they occur after smaller numbers of splits (h(x) is the average path length). For a sample of size n the anomaly score of a record x is given by (1), where c(n) in (2) is the average path length of an unsuccessful search in a binary search tree, H(i) is the i-th harmonic number (approximately ln(i) + 0.5772) and scores close to 1 indicate anomalies. The model is trained using the benign baseline records only, and tested on test records; labels are only for evaluation purposes.

$s(x, n) = 2^{(-E[h(x)] / c(n))}$ (1) $c(n) = 2H(n - 1) - 2(n - 1) / n$ (2)

Include feature scaling and severity mapping. Before training and inference every numerical feature j is standardised as in (3), where μj and σj are the mean and standard deviation of the feature on the training data. The framework transforms the model output to a risk score (0-1) and assigns a severity rating; in the evaluated run, scores in the range 0.54-0.65 were assigned medium severity and scores in the range 0.66-0.70 were assigned high severity (Fig. 5).

$zij = (xij - \mu j) / \sigma j$ (3)

SIEM and SOAR elements: Elasticsearch, Logstash and Kibana are provided in configuration to index alerts, process events and to view monitoring dashboards [21]. A Shuffle-compatible client [20] stages workflows and also supports webhooks, and alerts are enhanced with MITRE ATT&CK [18] technique info. The saved executions are ones against a local mock adapter, not a real Shuffle server. The technologies used are presented in Table 2.

Table 2: Models and technologies used

Category	Technology	Role
Machine learning	Isolation Forest, StandardScaler	Anomaly detection; feature standardisation
Programming and data processing	Python, Pandas, NumPy, Scikit-learn	Back-end processing, dataset handling, numerical operations, model implementation
Front end and back end	React, TypeScript, Vite, Express	Interactive dashboard and application logic
SIEM and SOAR	Elasticsearch, Logstash, Kibana; Shuffle-compatible client; MITRE ATT&CK	Alert indexing, event pipeline, monitoring; workflow staging and webhooks; technique enrichment
Data formats	JSON	Security events, alerts, workflow execution records and audit information
Category	**Technology**	**Role**
Testing and deployment	Python unittest, Docker Compose	Implementation validation; service deployment configuration

3.5 Human-supervised response and offline retraining

The alerts generated are sent to analysts who use the dashboard to go over them and check their severity and supporting information. Proposed changes to the response are submitted for either rejection or approval by an analyst before they are actually performed, whereas containment is only simulated and no firewall changes are made. JSON-based feedback and audit logs are created for analyst decisions, response status and activities. Each feedback record includes the alert ID, analyst identifier, verdict, notes and timestamp and analyst verdicts are stored in a feedback repository that supports potential offline retraining and candidate-model evaluation through a promotion gate.

The framework provides for offline implementation of analytical feedback. It is designed such that confirmed false positive feature records are added to the normal training baseline, and the candidate model is tested on the test dataset of the CICIDS2017. Confirmed true positive verdicts are added to the feedback repository, but not used as candidate training samples.The production baseline is compared to a candidate Isolation Forest through the promotion gate in (4). The reported candidate improvement is not evidence of feedback-driven learning since the saved candidate evaluation artefact reports zero feedback samples used. True positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) are used to calculate the evaluation metrics (5)–(7).

$F1cand \geq F1base - 0.01$ and $FPcand \leq FPbase$ (4) $Precision = TP / (TP + FP)$, $Recall = TP / (TP + FN)$ (5) $F1 = 2 \cdot Precision \cdot Recall / (Precision + Recall)$ (6) $Accuracy = (TP + TN) / (TP + TN + FP + FN)$ (7)

3.6 Data sources and evaluation plan

There are three data sources (Table 3). The sample data (Suricata EVE-format events and SSH authentication events) provide extra security-event inputs and are converted into alerts, while the CICIDS2017 network traffic is the only source scored by the Isolation Forest. The confusion matrix of Section IV, which shows 150 attack and 350 benign records, indicates that 500 records is a test subset. Evaluation of the work has been carried out in three ways: (i) a comparison of the baseline Isolation Forest's detection quality with the candidate on the test subset using precision, recall, F1-score, accuracy and the confusion matrix; (ii) a demonstration of the behaviour of the pipeline in terms of telemetry ingestion, alert generation, investigation, approval, simulated response and audit recording via the dashboard; and (iii) a comparison of a retrained model with the baseline using the promotion gate. A unittest test suite is provided for implementation testing.



Table 3: Dataset configuration

Data item	Records
CICIDS2017 benign baseline records (training)	500
CICIDS2017 test records	500 (150 attack, 350 benign)
Suricata EVE-format events (generated)	80
SSH authentication events (generated)	100
Total telemetry shown in the dashboard	1,180

4. Results and Discussion

4.1 Results

Fig. 3 shows the overview of the running dashboard. The ingestion run provided 1180 telemetry records, all of which were parsed (CICIDS2017 1000, Suricata 80, SSH 100), and indexed 268 alerts, 60 critical, 164 high and
44 medium. The overview also provides details of the four pipeline stages (ingestion, detection, SOAR orchestration, and human-supervised retraining) and 224 alerts ready for the analyst's signature. The model version v1.1.0, the status of ELK and simulated sandbox and the analyst role are reported on the dashboard header.

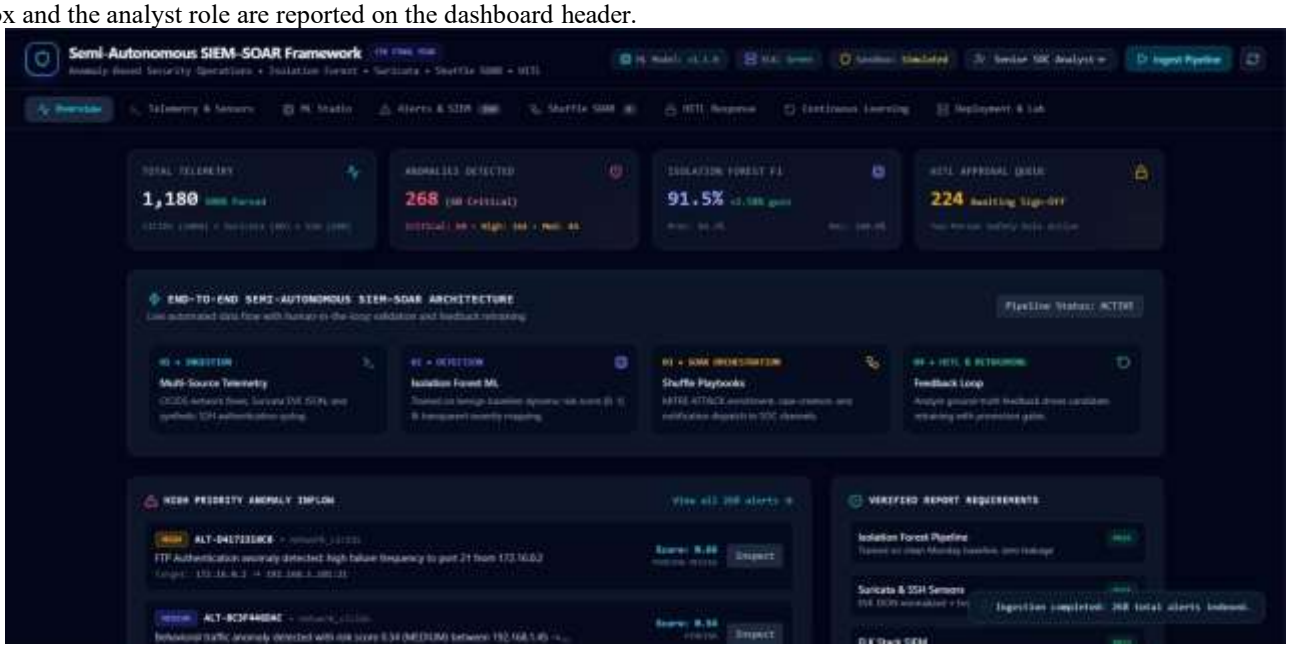


Fig. 3: Overview page of the dashboard showing telemetry, anomaly counts, Isolation Forest metrics and the approval queue

The telemetry page (Fig. 4) normalises the SSH authentication stream with a sliding window that aggregates the failure ratio and flags bursts. The source address 185.220.101.4 made failed password attempts against the accounts root, admin, oracle and postgres within a few seconds; the window failure counts rise from 18 to 21 at a failure ratio of 100 % and every attempt is flagged as a burst attack. In contrast, the accepted public-key logins of the accounts devops (192.168.1.15) and ubuntu (192.168.1.22) show 0 failures and are marked normal. The same page offers the full ingest pipeline, a simulated SSH brute-force burst and a replay of the Suricata EVE events.

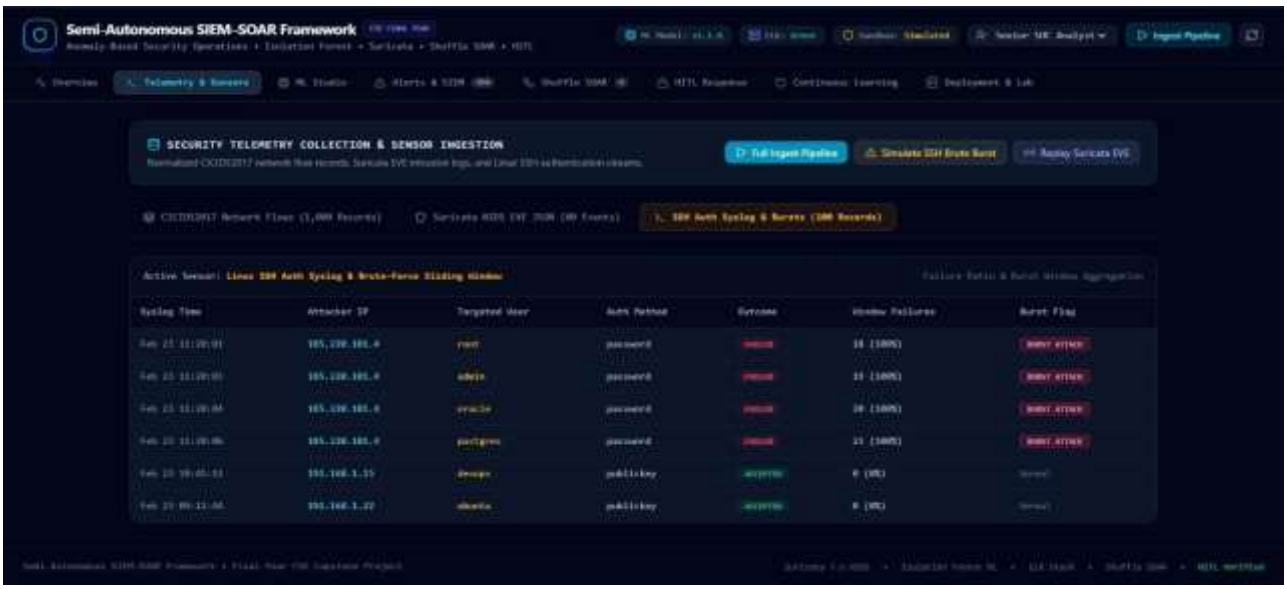


Fig. 4: telemetry and sensor page showing SSH authentication records with sliding-window brute-force detection



The preprocessed test set was used to assess the performance of the Isolation Forest algorithm (Table 4). It identified all attack records (recall 100 %) at the price of 28 false positives, which gives a precision of 84.27 %, an F1-score of 91.46 % and an accuracy of 94.40 %. The confusion matrix and the number of saved alerts are listed in Table 5. The saved alert file of the evaluation artefacts contains 256 alerts (169 CICIDS2017, 47 Suricata and 40 SSH); the last ingestion run of the dashboard generated 268., and the audit trail of Fig. 6 shows that earlier runs generated 260 and 261 alerts, so the alert count depends on the run.

Table 4: Baseline Isolation Forest evaluation (v1.0.0)

Metric	Result
Precision	84.27 %
Recall	100 %
F1-score	91.46 %
Accuracy	94.40 %

Table 5: Baseline confusion matrix and saved alert records

Item	Count
True positives (TP)	150
False positives (FP)	28
True negatives (TN)	322
False negatives (FN)	0
CICIDS2017 alerts	169
Suricata alerts	47
SSH alerts	40
Total saved alerts	256

The alert list is displayed on the left side of the alerts page (Fig. 5) and the details of the selected alert are displayed on the right side: the network and flow evidence (source and destination address, protocol and model version), automated response plan, field for investigation notes, buttons for true-positive or false-positive verdict, button to trigger the Shuffle SOAR playbook, and raw alert schema. This is a medium severity alert with a risk score of 0.54 on 192.168.1.45 to 192.168.1.101:8080, proposed plan to log and monitor; not yet approved. The other alerts listed are high severity SSH brute-force signatures to port 22 (0.68-0.70) and FTP authentication anomaly to port 21 (0.66). There are 4 Shuffle workflow executions.

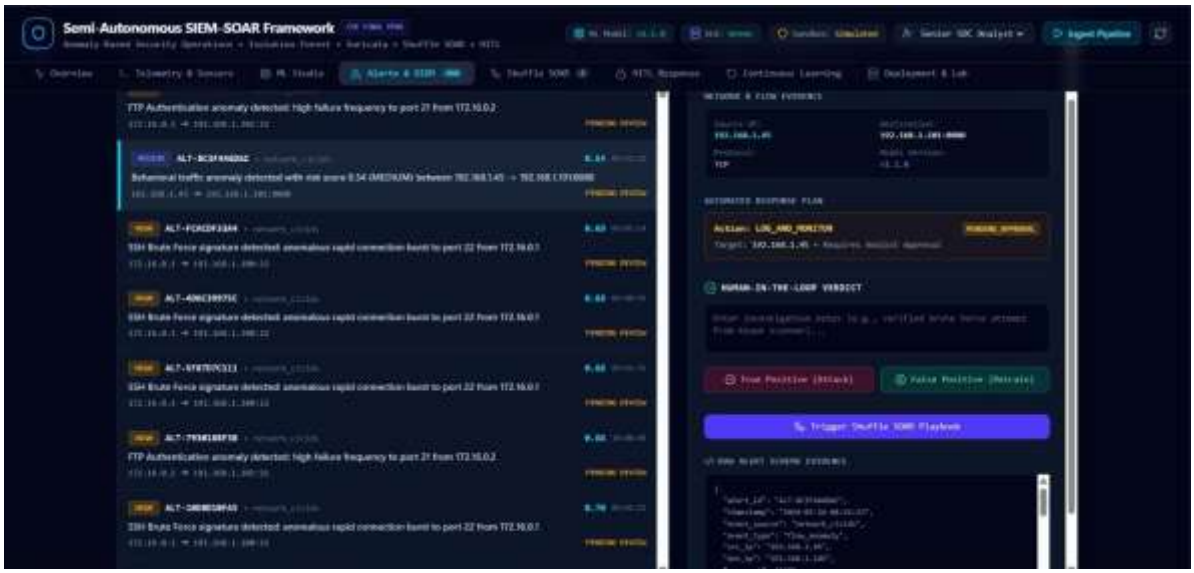


Fig. 5: alerts and SIEM page with flow evidence, automated response plan, analyst verdict controls and raw alert schema

The target address, action and risk score will be listed on the response page (Fig. 6) with the proposed actions. The alert for the FTP authentication anomaly (target 172.16.0.2, action BLOCK_ATTACKER_IP, risk score 0.66) has been executed in simulation, and the SSH brute-force alerts towards 172.16.0.1 (scores 0.69 and 0.68) are still pending approval and have approve and reject buttons. The analyst needs to sign an authorisation dialog (Fig. 7) which displays the target, action and severity, and must include a written justification, before an action is executed. The 2 actions shown on the audit trail on the right of Fig. 6 are the approval of the simulated action (RESPONSE_ACTION_APPROVED, with the reason) and the simulation of the containment action (CONTAINMENT_ACTION_EXECUTED_SIMULATED) itself, along with the operator and the status; the simulated action is an iptables drop rule, which is only echoed and no real firewall is modified. The ingestion runs will be recorded in the same trail.

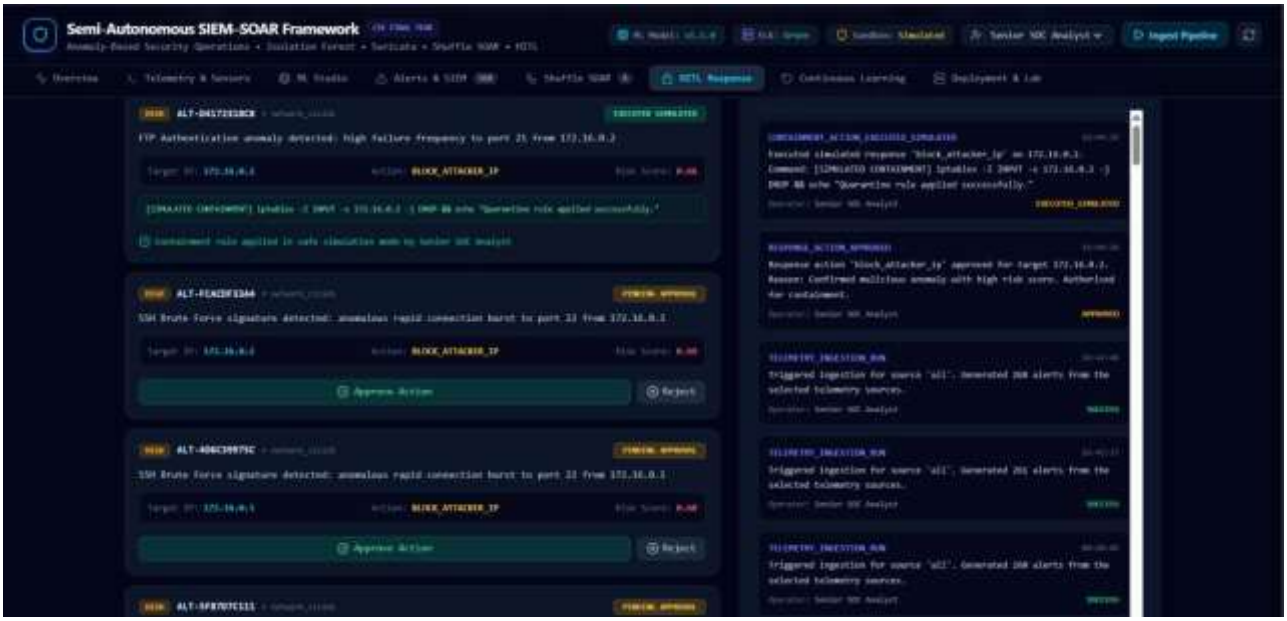


Fig. 6: Human-supervised response page with pending approvals, a simulated containment and the audit trail

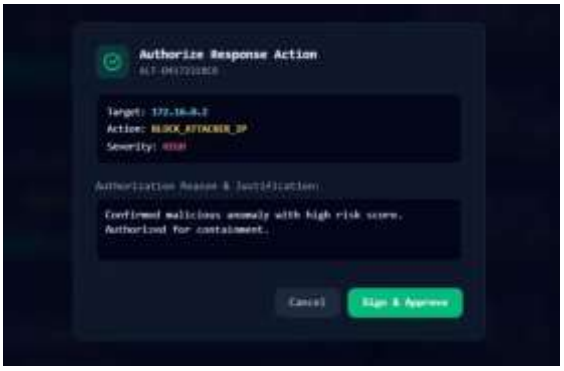


Fig. 7: response authorisation dialog requiring a justification and a signature before an action is approved

The continuous-learning page (Fig. 8) shows a pool of four records of analyst feedback: one confirmed false positive; three confirmed true attacks. However, the saved candidate evaluation artefact states zero number of feedback samples used during the candidate evaluation. Here, the candidate model v1.1.0 is compared to the production baseline v1.0.0 in Table 6.

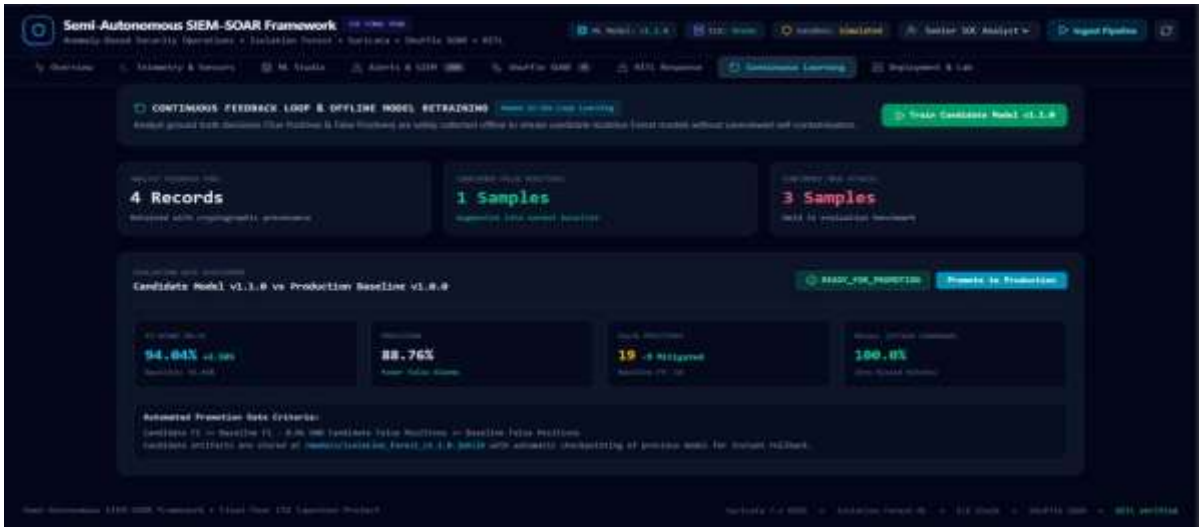
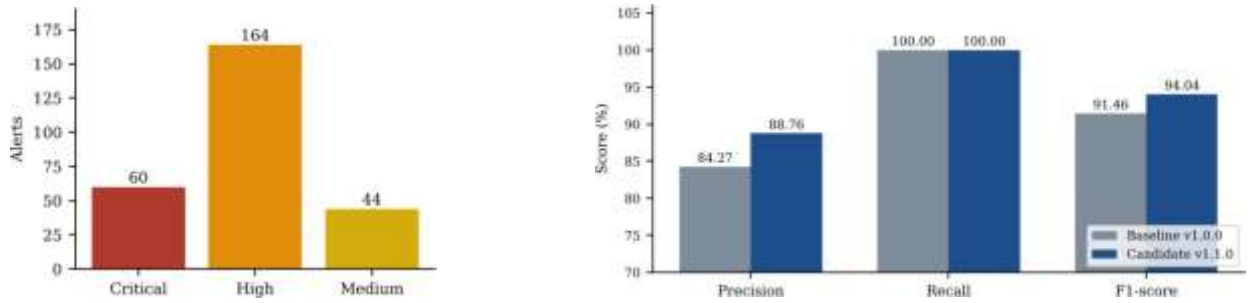


Fig. 8: continuous feedback loop page with the feedback pool, the evaluation of the candidate model and the promotion gate criteria

Table 6: Baseline versus candidate model on the test subset

Metric	Baseline v1.0.0	Candidate v1.1.0	Change
Precision	84.27 %	88.76 %	+4.49 points
Recall	100 %	100 %	none
F1-score	91.46 %	94.04 %	+2.58 points
False positives	28	19	−9



(a) Severity of the 268 alerts (b) Baseline versus candidate

Fig. 9: (a) severity distribution of the alerts of the latest ingestion run; (b) precision, recall and F1-score of the baseline and candidate models

4.2 Discussion

The outcomes back up each of the work objectives. The 1,180 parsed telemetry records and SSH burst detection demonstrate multi-source ingestion, the baseline recall of 100 % F1-Score illustrates anomaly detection, the generation of structured alerts as reflected in the 268 scored and classified alerts, the approval queue, the signed authorisation and the simulated containment reflect analyst-controlled response, and the recorded approval and execution events provide auditability. There are a number of comments to be made on several of these observations. The baseline is a highly biased towards recall and would have missed no attacks in the test set, but identified 28 benign records. This is OK only because the proposed action is subject to a review by a human, and of the 224 alerts waiting for sign-off, none of them are executed without this review. This framework provides for a repository of feedback and an offline retraining system to integrate verified false-positive record into the normal baseline. The saved candidate evaluation artefact, though, gives no indication of how many samples of feedback were used, and confirmed true-positive records are not directly part of the candidate training process. The present results thus show the feedback-storage and candidate-evaluation mechanisms, but not successful feedback-driven learning.

The candidate improvement should be read judiciously. The evaluation artefact saved shows 0 samples of analyst feedback in the candidate evaluation. There is not enough evidence to conclude that these four feedback records led to the reported improvement of candidates. As such, the results provide a proof of the candidate evaluation and promotion-gating mechanism, but not proof of feedback-driven model improvement. Reproducible retraining experiment with a validated feedback dataset must be provided.

The following are limitations of the present work. Selected subsets of CICIDS2017 are used for the evaluation. The Suricata EVE and SSH records contain sample data generated by Suricata, and only the CICIDS2017 flows are scored by the Isolation Forest. No execution artefacts are found to provide live network monitoring. For the saved Shuffle executions, a local mock adapter is used instead of a confirmed external live integration. The execution of the responses is simulated, and has no effect on real firewall configuration. The reported metrics are based on the provided datasets and evaluation artifacts and not on a real enterprise network. Further research will extend the live traffic monitoring by implementing a verified Suricata deployment, set up and test live integrations of SIEM and Shuffle, apply it to larger and wider datasets, apply reproducible feedback-driven retraining, compare with other anomaly-detection algorithms, and test the framework in a controlled SOC environment.

5. Conclusion

This paper introduced a Intelligent SIEM – SOAR framework, which integrates Isolation Forest anomaly detection and human-supervised response. The CICIDS2017 network flows, Suricata EVE events and SSH authentication logs are ingested, network flows are scored with a model trained on a ‘normal' baseline, and the resulting detections are processed into structured alert artefacts which are investigated, approved or rejected by an analyst in a React dashboard. The decisions are logged in an audit trail, and the results of analysts verdicts are placed in a feedback repository, enabling potential offline retraining and candidate-model evaluation via a promotion gate, and approved containment only occurs within a simulated sandbox. The baseline achieved a recall of 100 % and an f1 score of 91.46 % on the bundled test subset while the candidate model evaluation showed a decrease of false positives from 28 to 19 and an increase of f1 score to 94.04 % on the same test subset. The saved evaluation artefact however did not indicate any feedback samples were used, so this improvement was not due to analyst feedback. The work demonstrates that machine-learning detection can complement security monitoring without compromising the ability of humans to take consequential security decisions. The next steps are monitoring in the real world using a verified Suricata deployment, validated SIEM and Shuffle integration, more data, comparing to other anomaly detectors, and testing in a dedicated SOC environment, the evidence is derived from a curated set of datasets, generated sample events, a mock orchestration adapter and very small feedback



pool.

REFERENCES

[1] B. Alboushy, A. Jafar, M. Aljnidi, M. B. Disoki, and A. Shaheed, "Context-Aware Web Attack Detection in Open-Source SIEM Systems via MITRE ATT&CK-Enriched Behavioral Profiling," arXiv:2605.13337, 2026.

[2] R. Sahay, B. Eapen, W. Meng, M. R. Al Mamun, N. K. Dora, M. Sumasadan, S. K. Tetarave, and R. Soto, "Policy-Guided Threat Hunting: An LLM Enabled Framework with Splunk SOC Triage," arXiv:2603.23966, 2026.

[3] S. Ndichu, T. Ban, S. Ozawa, T. Takahashi, and D. Inoue, "PACT: Reducing Alert Fatigue in Low-Prevalence SOC Streams with Triggered Active Learning," arXiv:2605.22324, 2026.

[4] I. J. Chowdhury and M. A. Y. Tanvir, "Decision-Aware Trust Signal Alignment for SOC Alert Triage," arXiv:2601.04486, 2026.

[5] J. Zhang, H. Liu, L. Ren, Y. Zhao, and Z. Song, "Playbook-Guided Executable SOC Automation for Privacy-Preserving Response in Distributed Networked Systems," EAI Endorsed Trans. Scalable Inf. Syst., vol. 13, no. 3, 2026.

[6] S. Ndichu, T. Ban, S. Ozawa, T. Takahashi, and D. Inoue, "AI-Driven Security Alert Screening and Alert Fatigue Mitigation in Security Operations Centers: A Comprehensive Survey," arXiv:2605.08316, 2026.

[7] M. W. Eckhoff, P. M. Flydal, S. Peters, M. Eian, J. Halvorsen, V. Mavroeidis, and G. Grov, "A Graph-Based Approach to Alert Contextualisation in Security Operations Centres," arXiv:2509.12923, 2025.

[8] F. Jalalvand, M. Baruwal Chhetri, S. Nepal, and C. Paris, "Adaptive Alert Prioritisation in Security Operations Centres via Learning to Defer with Human Feedback," arXiv:2506.18462, 2025.

[9] M. Turcotte, F. Labrèche, and S.-O. Paquette, "Automated Alert Classification and Triage (AACT): An Intelligent System for the Prioritisation of Cybersecurity Alerts," arXiv:2505.09843, 2025.

[10] A. Mohsin, H. Janicke, A. Ibrahim, I. H. Sarker, and S. Camtepe, "A Unified Framework for Human–AI Collaboration in Security Operations Centers with Trusted Autonomy," arXiv:2505.23397, 2025.

[11] Ismail, R. Kurnia, Z. A. Brata, G. A. Nelistiani, S. Heo, H. Kim, and H. Kim, "Toward Robust Security Orchestration and Automated Response in Security Operations Centers with a Hyper-Automation Approach Using Agentic Artificial Intelligence," Information, vol. 16, no. 5, p. 365, 2025.

[12] Z. Aref, S. Wei, and N. B. Mandayam, "AI Collaboration in Cloud Security: Cognitive Hierarchy-Driven Deep Reinforcement Learning," arXiv:2502.16054, 2025.

[13] R. Singh, M. Baruwal Chhetri, S. Nepal, and C. Paris, "ContextBuddy: AI-Enhanced Contextual Insights for Security Alert Investigation (Applied to Intrusion Detection)," arXiv:2506.09365, 2025.

[14] Z. Liu, "Multi-Agent Collaboration in Incident Response with Large Language Models," arXiv:2412.00652, 2024.

[15] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422.

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), 2018, pp. 108–116.

[17] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[18] B. E. Strom et al., "MITRE ATT&CK: Design and philosophy," The MITRE Corporation, McLean, VA, USA, Tech. Rep., 2018.

[19] "Suricata: Open source network threat detection engine." [Online]. Available: https://suricata.io

[20] "Shuffle: Open source security automation platform." [Online]. Available: https://shuffler.io

[21] Elasticsearch B.V., "Elastic Stack: Elasticsearch, Logstash and Kibana." [Online]. Available: https://www.elastic.co/elastic-stack

[22] Meta Platforms, Inc., "React: The library for web and native user interfaces." [Online]. Available: https://react.dev

[23] "Express: Fast, unopinionated, minimalist web framework for Node.js." [Online]. Available: https://expressjs.com

[24] W. McKinney, "Data structures for statistical computing in Python," in Proc. 9th Python in Science Conf. (SciPy), 2010, pp. 56–61.

[25] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.