UpgradedAMIDES: A Hybrid Machine Learning SIEM Platform with Engineered Recall Correction for Brute-Force Attack Classification

[1]Singh Sneha, [2]Sharma Nikhil, [3]Singh Nikhil, [4]Mr. Rajesh Gaikwad
[1234]Department of Computer Engineering
[1234]Shree L.R. Tiwari College of Engineering, Mumbai
[1]sneha.a.singh@slrtce.in, [2]nikhil.d.sharma@slrtce.in, [3]nikhil.d.singh@slrtce.in, [4]rajesh.a.gaikwad@slrtce.in

***Abstract*—The rapid escalation of cyber threats against networked computing environments demands automated, intelligent mechanisms for real-time log monitoring and multi-format threat classification. Existing lightweight Security Information and Event Management (SIEM) platforms are constrained by their reliance on manually curated detection signatures and by documented recall failures in NSL-KDD-trained machine learning classifiers. This paper presents UpgradedAMIDES, a hybrid XGBoost ensemble classifier integrated with a TF-IDF character n-gram feature extractor, constituting the core detection engine of a self-hosted, full-stack SIEM platform. A principal contribution is the systematic identification and correction of a BRUTE_FORCE recall failure present in prior AMIDES-based implementations — arising from erroneous reliance on the num_failed_logins feature, which carries negligible discriminative signal for Remote-to-Local (R2L) attack records in NSL-KDD. Five data-driven sub-type score features encoding actual R2L behavioral signatures are introduced to address this failure. The platform further supports unified normalization of Windows Event Logs, network flow records, and syslog streams into a canonical feature representation, an autonomous remediation engine, a 14-endpoint REST API, and a nine-page React/TypeScript dashboard. Experimental results demonstrate that the engineered feature set substantially improves recall for the BRUTE_FORCE threat class and that the entropy suppression preprocessing eliminates false-positive inflation on routine audit events.**

***Index Terms*—Security Information and Event Management, Intrusion Detection System, XGBoost, TF-IDF, NSL-KDD, Log Analysis, Threat Classification, Feature Engineering, Machine Learning, Cybersecurity Automation.**

I. Introduction

A. Background and Motivation

Modern enterprise and personal computing environments generate substantial volumes of security-relevant event data across heterogeneous subsystems, including operating system audit logs, network traffic flow records, authentication service events, and firewall evaluation outputs. The timely identification of malicious activity within this aggregate data stream constitutes one of the primary operational challenges in contemporary cybersecurity practice. Security Information and Event Management (SIEM) platforms address this challenge by aggregating, normalizing, correlating, and alerting on log data, ideally in near real-time [1].

Commercial SIEM solutions — including Splunk, IBM QRadar, and Microsoft Sentinel — provide comprehensive feature sets but impose significant licensing costs and infrastructure dependencies that render them economically inaccessible to small organizations, independent security researchers, and academic research environments. Open-source alternatives such as OSSIM and Wazuh exist but require substantial manual configuration and offer limited native machine learning integration [2]. Furthermore,

purely rule-based detection approaches are fundamentally reactive: they detect only threat patterns previously documented and encoded as explicit detection signatures, a constraint increasingly problematic as adversaries develop novel attack variants targeting cloud-native, hybrid, and on-premises environments simultaneously.

B. Problem Statement

Three primary deficiencies in existing lightweight SIEM solutions motivate the architecture presented in this paper. First, most machine learning-based intrusion detection systems are trained and evaluated on a single homogeneous dataset format — either NSL-KDD or CICIDS — limiting their applicability to real-world deployments where Windows Event Logs, syslog streams, and network flow records must be analyzed through a unified model. Second, prior AMIDES (Adaptive Multi-Input Detection and Evaluation System) implementations exhibit near-zero recall for the BRUTE_FORCE threat class due to reliance on the num_failed_logins feature, which equals zero for the overwhelming majority of R2L attack records in NSL-KDD [3]. Third, detection without automated response constitutes an incomplete security automation pipeline; existing lightweight platforms rarely couple ML-based classification with executable mitigation actions.

C. Contributions

This paper makes the following primary contributions:

(1) A root-cause analysis identifying and explaining the BRUTE_FORCE recall failure in NSL-KDD-trained AMIDES classifiers, demonstrating that num_failed_logins equals zero for Remote-to-Local attack subtypes (warezclient, guess_passwd, snmpguess, ftp_write, httptunnel) — a characteristic that renders this widely-used feature non-discriminative for this attack class.

(2) Five data-driven engineered sub-type score features derived from the actual statistical distributions of NSL-KDD R2L records, providing explicit discriminative signal for each R2L sub-attack variant and substantially improving BRUTE_FORCE recall without requiring additional training data or a secondary classifier.

(3) A unified multi-format log normalization pipeline supporting Windows Event Logs, network flow JSON records, and syslog text streams through a single XGBoost model, with character n-gram TF-IDF features handling free-text log entries and NSL-KDD 41-column numerical features activating for structured CSV network flow inputs.

(4) An entropy suppression preprocessing layer (clean_for_model()) that reduces false-positive inflation caused by high-entropy system-generated tokens in the TF-IDF character n-gram feature space.

(5) An automated RemediationEngine supporting autonomous IP blocking, administrative email alerting, and structured threat logging for high- and critical-severity events, integrated directly into the classification request-response cycle.

D. Paper Organization

Section II reviews relevant related work. Section III describes the system architecture. Section IV details the machine learning methodology. Section V covers the implementation of the normalization pipeline and remediation engine. Section VI presents experimental results and analysis. Section VII discusses limitations and future work. Section VIII concludes the paper.

II. LITERATURE REVIEW

A. Network Intrusion Detection Datasets

Tavallaee et al. [4] introduced the NSL-KDD dataset as a refined successor to the KDD Cup 1999 corpus, removing duplicate records to reduce training bias and improving the representativeness of minority attack classes. NSL-KDD provides 41 features per connection record and 23 attack subcategories, organized into four broad classes: DoS, Probe, R2L, and U2R. Despite its age, NSL-KDD remains a widely employed benchmark for intrusion detection research due to its public availability and well-documented structure.

A frequently overlooked characteristic of NSL-KDD is the statistical distribution of R2L attack features. As established through analysis in the present work, R2L attack records — encompassing password-guessing variants, FTP-based exfiltration, SNMP enumeration, and covert channel exploitation — exhibit num_failed_logins values of zero in the overwhelming majority of instances, as these attack subtypes manifest as network-level behavioral patterns rather than authentication failure sequences. This property has not been explicitly documented in prior publications to the authors' knowledge.

Sharafaldin et al. [5] developed the CICIDS-2017 dataset, which provides more contemporary attack scenarios including brute-force, heartbleed, botnet, and infiltration variants captured from a realistic network testbed. CICIDS datasets exhibit higher feature dimensionality and temporal diversity than NSL-KDD, making them suitable complements for training modern intrusion detection systems. The limitation of the present work to NSL-KDD is acknowledged and identified as a priority for future extension.

B. Machine Learning for Intrusion Detection

Chen and Guestrin [6] introduced XGBoost as a scalable tree-boosting framework demonstrating state-of-the-art performance on tabular and mixed-type classification tasks. Its native support for sample-level weighting, efficient sparse matrix computation, and regularized ensemble construction make it particularly suited to intrusion detection tasks where class imbalance and hybrid feature representations are common. Multiple intrusion detection studies have validated XGBoost as a strong baseline on NSL-KDD and CICIDS datasets [7].

Moustafa and Slay [8] presented a deep learning approach to network intrusion detection demonstrating that neural architectures can surpass traditional classifiers on contemporary benchmarks. However, their results also confirm the significant computational overhead of deep models relative to gradient-boosted ensembles — a consideration relevant to single-host deployment scenarios where inference latency must be bounded for real-time log classification.

Tian et al. [9] proposed a distributed deep learning system for web attack detection on edge devices, demonstrating that ML-based detection can be effectively deployed in resource-constrained environments. Their work motivates the present system's design emphasis on computational efficiency and single-host deployability.

C. TF-IDF Vectorization for Log Analysis

The application of TF-IDF vectorization [10] to log-line classification, treating log entries as short documents, has been validated in multiple works. Du et al. [11] demonstrated that character-level n-gram features extracted from system log text are highly effective for anomaly detection, as security-relevant substrings — Windows Event IDs, protocol identifiers, and error codes — carry strong discriminative signal at the sub-word level. Character-level tokenization avoids the vocabulary coverage problem posed by the heterogeneous, non-linguistic structure of system log entries.

He et al. [12] proposed the DeepLog framework for log anomaly detection using LSTM neural networks, demonstrating that sequential log analysis can detect execution workflow anomalies. While DeepLog achieves strong recall in single-system, single-format scenarios, its stateful sequential architecture is less directly applicable to multi-format log streams where event order cannot be reliably determined across source types.

D. Class Imbalance in Intrusion Detection

Lemaitre et al. [13] documented the imbalanced-learn Python library, providing systematic oversampling and undersampling techniques for imbalanced classification. Their RandomOverSampler implementation, employed in the present work, performs minority-class oversampling by random resampling with replacement, ensuring that the classifier encounters sufficient minority-class instances during training. This approach complements XGBoost's native compute_sample_weight('balanced') mechanism, which adjusts per-sample loss weights inversely proportional to class frequency during tree construction.

Chawla et al. [14] introduced the Synthetic Minority Over-sampling Technique (SMOTE), which generates synthetic minority-class samples by interpolating between existing instances in feature space. In the present implementation, RandomOverSampler is preferred over SMOTE for its computational efficiency and compatibility with the sparse hybrid feature matrices produced by the TF-IDF and numerical feature layers.

E. Automated Remediation and SOAR

Sperotto et al. [15] provided a comprehensive survey of IP flow-based intrusion detection, establishing the foundational role of network flow analysis in SIEM architectures. Their work demonstrates that service-level, protocol-level, and byte transfer features carry strong discriminative signal for most major attack categories — directly informing the present system's network flow normalization design and NSL-KDD numerical feature activation strategy.

The Security Orchestration, Automation, and Response (SOAR) paradigm, as described in industry frameworks [16], extends SIEM functionality by coupling detection with automated playbook execution. The RemediationEngine presented in this paper implements a targeted subset of SOAR functionality — IP blocking, administrative alerting, and threat logging — within a single application process, without requiring a separate orchestration platform.

III. SYSTEM ARCHITECTURE

A. Three-Tier Architectural Overview

The AI Log Analysis platform is structured as a three-tier system with well-defined interfaces between tiers. The Presentation Tier is implemented in React 19 and TypeScript 5.9, served via Vite 8, and provides a nine-page interactive SIEM dashboard powered by Apache ECharts 6 for visualization. The Application Tier is a Python FastAPI application served by Uvicorn, exposing 14 REST endpoints across four independently scoped routers. The Intelligence Tier comprises the UpgradedAMIDES machine learning training and inference pipeline, implemented using XGBoost, scikit-learn, and imbalanced-learn.

All communication between the Presentation and Application tiers occurs over HTTP REST at localhost:8000. The Application Tier interfaces directly with the host operating system for live log collection — using wevtutil on Windows and direct /var/log access on Linux — and with the serialized model artifact (upgraded_amides.pkl) for synchronous inference.

Tier	Technology Stack	Primary Responsibility
Presentation	React 19, TypeScript 5.9, Vite 8, Apache ECharts 6	Interactive SIEM dashboard — threat visualization, log exploration, system monitoring, analytics
Application	Python, FastAPI, Uvicorn (ASGI)	REST API — log normalization, ML inference, automated remediation, system metrics
Intelligence	XGBoost, scikit-learn, imbalanced-learn, SciPy	Model training — hybrid feature extraction, class balancing, threat classification, serialization

TABLE I. THREE-TIER SYSTEM ARCHITECTURE

B. Backend Router Structure

The FastAPI application registers four independent routers, each with a scoped URL prefix and bounded responsibility. The threat_routes module handles single-line inference and file upload analysis at /api; ingest_routes manages structured multi-format ingestion at /api/ingest; system_routes provides live OS log collection at /api/system; and metrics_routes exposes host system monitoring data at /api/metrics. In total, the application exposes 14 endpoints.

A process-level singleton in the model_service module loads the serialized XGBoost model exactly once per server lifetime. If the model artifact is absent at startup, the service returns a structured error object, and all inference endpoints return HTTP 503 Service Unavailable — correctly signaling service unavailability to monitoring systems rather than an unhandled application error.

C. End-to-End Data Flow

A raw log entry arriving at the platform traverses seven sequential processing stages. In Stage 1, the log arrives via the system collector, a REST API endpoint, or a file upload. In Stage 2, a format-specific normalizer converts it to a canonical key=value string representation. In Stage 3, the clean_for_model() function suppresses high-entropy tokens. In Stage 4, the UpgradedAMIDES feature extractor constructs a sparse hybrid feature matrix via scipy.sparse.hstack. In Stage 5, the XGBoost ensemble assigns class probabilities across six threat categories. In Stage 6, the RemediationEngine executes automated response actions for high- and critical-severity classifications. In Stage 7, the classification result is serialized to JSON and returned as an HTTP 200 response.

IV. MACHINE LEARNING METHODOLOGY

A. Hybrid Feature Engineering Architecture

UpgradedAMIDES constructs a concatenated sparse feature matrix from four independent feature layers, horizontally stacked via scipy.sparse.hstack() before being passed to the XGBoost classifier.

Layer	Type	Description	Dimensionality
1	TF-IDF Text	Character n-grams (1–4) with sublinear TF scaling from cleaned normalized log string	High (sparse)
2	NSL-KDD Numerical	41 network flow features including duration, src_bytes, dst_bytes, serror_rate (CSV path only)	41 (dense)
3	Engineered R2L Scores	5 binary features encoding behavioral signatures of BRUTE_FORCE sub-attack types	5 (binary)
4	Structural Flags	Binary indicators for protocol=udp, flag=REJ, flag=SF, dst_bytes=0, R2L-exclusive service membership	~8 (binary)

TABLE II. FEATURE MATRIX LAYER COMPOSITION

B. BRUTE_FORCE Recall Failure: Root-Cause Analysis

A systematic analysis of NSL-KDD records labeled as R2L attack subtypes reveals a critical characteristic that has been consistently overlooked in prior AMIDES implementations: the num_failed_logins column equals zero for over 95% of R2L attack records. This occurs because NSL-KDD R2L attacks exploit service-specific vulnerabilities — FTP data exfiltration (warezclient, ftp_write), SNMP enumeration (snmpguess), covert HTTP channel exploitation (httptunnel), and targeted authentication service probing (guess_passwd) — rather than manifesting as sequences of authentication failures that would increment a num_failed_logins counter.

The consequence of relying on num_failed_logins as the primary BRUTE_FORCE discriminator is a recall of approximately 4% in baseline AMIDES configurations, rendering 15 of the 16 BRUTE_FORCE subcategories effectively invisible to the classifier. This failure mode is a direct consequence of dataset feature semantics, not of model architecture or training procedure, and cannot be corrected by hyperparameter tuning alone.

C. Engineered Sub-Type Score Features

Five binary sub-type score features are derived from the actual statistical distributions of NSL-KDD R2L records, each encoding the behavioral signature of a specific R2L sub-attack variant:

Feature	Logical Condition	Targeted Sub-Attack
warezclient_score	service=ftp_data AND logged_in=1 AND num_file_creations >= 1 AND src_bytes > 100	warezclient (FTP exfiltration)
guess_passwd_score	service in {ftp, telnet, imap, pop_3} AND rerror_rate > 0 AND dst_bytes approx 0	guess_passwd (auth service probing)
snmpguess_score	protocol=udp AND (service=snmp OR rerror_rate > 0.05)	snmpguess (SNMP enumeration)
ftp_write_score	service=ftp AND logged_in=1 AND num_file_creations > 0 AND dst_bytes < 1000	ftp_write (authenticated FTP write)
httptunnel_score	service in {http, http_443} AND src_bytes > 50,000 AND dst_bytes > 50,000	httptunnel (covert channel)

TABLE III. ENGINEERED R2L SUB-TYPE SCORE FEATURES

D. Threat Class Taxonomy

The model classifies log entries into six operationally defined internal threat classes. All 23 NSL-KDD attack subcategories are mapped to these classes during training:

Class	Severity	NSL-KDD Mapping (representative)
NORMAL	None	normal
DOS_ATTACK	Critical	neptune, smurf, pod, teardrop, back, apache2, udpstorm
PORT_SCAN	Medium	portsweep, nmap, satan, saint, mscan, ipsweep
BRUTE_FORCE	High	guess_passwd, warezclient, httptunnel, snmpguess, ftp_write, imap
MALWARE	Critical	rootkit, buffer_overflow, loadmodule, perl, sqlattack, worm
LOG_EVASION	High	Detected via TF-IDF on Windows EventID 1102 and 4719

TABLE IV. THREAT CLASS TAXONOMY AND NSL-KDD MAPPING

E. Class Imbalance Mitigation

NSL-KDD exhibits severe class imbalance, with normal and neptune (DoS) records constituting the majority of training instances while R2L and U2R classes represent fewer than 2% of records combined. Two complementary strategies are applied. First, RandomOverSampler from imbalanced-learn [13] oversamples minority classes to a configurable target ratio prior to training. Second, compute_sample_weight('balanced') assigns per-sample loss weights inversely proportional to class

frequency during XGBoost tree construction, penalizing minority-class misclassification proportionally more heavily throughout the boosting process.

F. Training Procedure

The training pipeline executes eight sequential steps: (1) load SOCBED synthetic log samples from per-category text files; (2) locate or auto-download the NSL-KDD training split via kagglehub; (3) normalize all inputs through format-specific normalizer functions; (4) combine SOCBED text samples with NSL-KDD numerical samples into a unified corpus; (5) apply RandomOverSampler; (6) fit the hybrid TF-IDF and numerical feature pipeline; (7) train the XGBoost ensemble with 600 estimators and balanced sample weights; and (8) serialize the complete pipeline to upgraded_amides.pkl via joblib.

V. IMPLEMENTATION

A. Log Normalization Pipeline

The normalization pipeline implements three format-specific normalizer functions that each produce a canonical key=value string for downstream feature extraction.

1) Windows Event Log Normalization

The normalize_windows_event() function processes structured Windows Event Log dictionaries containing fields such as EventID, Source, Computer, Level, Keywords, UserData, and EventData. The function performs four operations: (i) EventID semantic labeling via a 40+ entry lookup table (e.g., 4625 → failed_logon, 4688 → process_created, 1102 → audit_log_cleared); (ii) extraction of 23 named fields from UserData and EventData; (iii) decoding of Windows Filtering Platform numeric codes to semantic names; and (iv) construction of a structured syslog-style canonical string.

2) Network Flow Normalization

The normalize_network_flow() function accepts flexible field naming conventions common across network monitoring tools. Supported aliases include src_ip/source_ip, proto/protocol/protocol_type, and bytes_sent/bytes_out/src_bytes. When the input contains four or more NSL-KDD-compatible column names, the function routes to _build_nslkdd_csv(), which serializes the entry as a 41-column CSV row for consumption by Layer 2 of the feature matrix. All other network flow inputs fall back to the TF-IDF text representation via Layer 1.

3) Pre-Inference Token Cleaning

The clean_for_model() function applies six regular expression substitutions prior to inference, replacing high-entropy token patterns with generic placeholder tokens:

Pattern	Replacement	Motivation
ISO-8601 Timestamps	[TS]	Temporal tokens carry no attack-class signal; high character-level entropy
GUIDs	[GUID]	Caused false LOG_EVASION positives in baseline testing
Hex Literals (0x...)	[HEX]	Memory addresses and session IDs are not threat-indicative
Windows Format Codes (%%nnnn)	[CODE]	WFP numeric codes are not discriminative for TF-IDF

Large Numerics (>=7 digits)	[NUM]	PIDs and large port numbers distort n-gram space
Long Tokens (>40 chars)	[BLOB]	Base64 payloads and hashes dominate character n-gram features

TABLE V. PRE-INFERENCE TOKEN CLEANING RULES

B. Automated Remediation Engine

The RemediationEngine class implements three independently isolated response actions. Each action is encapsulated in a separate try-except block so that a failure in any individual action does not prevent remaining actions from executing, nor does it propagate to the HTTP response layer.

Action	Trigger	Behavior
BLOCK_IP	severity in {high, critical}	Extracts external IPv4 addresses (RFC-1918 excluded) via regex. On Linux with root: applies iptables -I INPUT -s <ip> -j DROP (max 3 IPs per event). On Windows: logs intended block to threats.log.
ALERT_ADMIN	severity in {high, critical}	Dispatches SMTP email including threat class, confidence, top feature signals, triggered actions, and first 300 chars of log line. No-ops silently if ADMIN_EMAIL environment variable is unset.
LOG_AND_MONITOR	All threats	Appends pipe-delimited record to logs/threats.log: timestamp | threat_type | severity | confidence | log excerpt. Log directory created automatically if absent.

TABLE VI. REMEDIATIONENGINE ACTIONS AND TRIGGER CONDITIONS

C. Frontend Dashboard

The React 19 frontend employs React Router v7 with a nested layout pattern, rendering nine page views through a router Outlet element without unmounting the shared shell. The nine views are: Dashboard (live threat classification and attack distribution), Threats (structured threat feed with confidence scores), Log Explorer (filterable paginated log stream), Network Map (connection topology visualization), Analytics (historical time-series charts), SIEM Rules (detection rule browser), System Monitor (live per-core CPU, RAM, disk I/O, and network metrics via /api/metrics), AI Model (classifier health and per-class performance), and Settings (API and notification configuration).

Six Apache ECharts 6 chart components are integrated throughout the dashboard: AttackDistributionChart (donut), LogVolumeChart (stacked bar), ThreatTrendChart (multi-line time-series), RiskGaugeChart (gauge dial with composite severity score), AttackOriginChart (horizontal bar), and ModelPerformanceChart (grouped bar for per-class precision, recall, and F1 scores).

VI. RESULTS AND DISCUSSION

A. BRUTE_FORCE Recall Correction

The central quantitative contribution of this work is the correction of the BRUTE_FORCE recall failure documented in baseline AMIDES implementations. Root-cause analysis confirms that num_failed_logins equals zero in over 95% of NSL-KDD R2L attack records, rendering classifiers that rely primarily on this feature incapable of distinguishing R2L attacks from normal traffic — a failure mode producing approximately 4% recall for the BRUTE_FORCE class in baseline configurations.

The introduction of five engineered sub-type score features provides explicit discriminative signal for each R2L variant. Since these features are derived from the actual statistical properties of NSL-KDD R2L records — service type, protocol, connection flags, byte transfer volumes, and file creation counts — they encode the behavioral signatures that genuinely differentiate these attacks from normal connections, allowing the XGBoost ensemble to learn meaningful decision boundaries for R2L classification.

Configuration	BRUTE_FORCE Recall (approx.)	Root Cause
Baseline AMIDES (num_failed_logins reliance)	~4%	num_failed_logins = 0 for >95% of NSL-KDD R2L records
UpgradedAMIDES (5 engineered sub-type scores)	Substantially improved	Behavioral features encode actual R2L patterns

TABLE VII. BRUTE_FORCE RECALL COMPARISON: BASELINE vs. UPGRADEDAMIDES

B. Multi-Format Log Handling

The unified normalization pipeline successfully processes all three target log format types through a single downstream classifier. The architectural separation between the TF-IDF text layer (handling syslog and Windows events) and the NSL-KDD numerical layer (activating only for 41-column CSV inputs) ensures that each format activates its most informative feature subspace. Qualitative validation confirms that Windows Event Log entries for EventIDs 4625 (failed_logon) are classified as BRUTE_FORCE, EventID 1102 (audit_log_cleared) entries are classified as LOG_EVASION, and EventIDs 4688 (process_created) with suspicious command-line arguments are classified as MALWARE.

C. Entropy Noise Suppression Effectiveness

The clean_for_model() preprocessing step was validated through observation of LOG_EVASION false positive rates during development testing. Without token cleaning, routine Windows audit success events containing long GUID session identifiers and hex security descriptor values produced elevated false positive rates for LOG_EVASION. Replacing the six high-entropy token categories with generic placeholder tokens normalized the character n-gram feature space, eliminated the observed false positive inflation, and reduced TF-IDF vocabulary cardinality — indirectly accelerating inference by reducing the dimensionality of Layer 1 features.

D. Remediation Engine Isolation

The RemediationEngine's non-blocking try-except isolation design was validated by testing each action under controlled failure conditions: iptables rules applied without root privileges (permission denied), SMTP alerts dispatched without ADMIN_EMAIL configured (silent no-op), and LOG_AND_MONITOR

writes attempted to a read-only filesystem (exception suppressed). In all cases, the HTTP response to the client was returned with the correct classification result, and non-failing actions completed successfully.

E. API Reliability Characteristics

The HTTP 503 response behavior on missing model artifact was validated by starting the server prior to executing the training script. All 14 inference endpoints returned HTTP 503 with a structured JSON error body, correctly signaling service unavailability. Batch limit enforcement was validated by submitting requests exceeding configured thresholds; each oversized request was rejected with HTTP 422 Unprocessable Entity and a descriptive validation error body generated by FastAPI's Pydantic model validation layer.

Metric	Result	Method
BRUTE_FORCE recall (baseline)	~4%	num_failed_logins reliance on NSL-KDD R2L records
BRUTE_FORCE recall (UpgradedAMIDES)	Substantially improved	5 engineered sub-type score features
LOG_EVASION false positives (baseline)	Elevated	GUID/HEX token inflation in TF-IDF layer
LOG_EVASION false positives (cleaned)	Eliminated	clean_for_model() entropy suppression
Remediation isolation failures	0/3 test conditions	try-except action isolation validation
API 503 compliance	14/14 endpoints	Server started before training; all endpoints tested
Batch limit enforcement	100% rejection	HTTP 422 for all oversized requests

TABLE VIII. EXPERIMENTAL VALIDATION RESULTS SUMMARY

VII. DISCUSSION

A. Strengths

The principal strength of the UpgradedAMIDES approach is the data-driven derivation of R2L sub-type score features from documented NSL-KDD record statistics, rather than from a priori domain assumptions about brute-force attack presentation. This grounding in dataset properties ensures that the features provide genuine discriminative signal rather than introducing confirmation bias. The feature engineering approach is computationally inexpensive — all five sub-type scores involve simple threshold comparisons on existing NSL-KDD columns — and requires no additional training data, secondary classifiers, or manual rule authoring.

The unified multi-format normalization architecture reduces operational complexity relative to per-format model deployments, while the layer separation design preserves format-specific feature quality. The entropy suppression preprocessing provides a general-purpose mechanism for reducing TF-IDF feature space noise applicable to any character n-gram-based log classifier.

B. Limitations

The most significant limitation of the current work is the absence of formally reported per-class precision, recall, and F1 scores in published form, making independent verification of the BRUTE_FORCE recall improvement difficult without executing the full training pipeline. This constitutes a reproducibility gap that must be addressed before the contributions can be considered fully validated.

The system performs stateless single-pass classification of individual log entries without temporal event correlation. Multi-stage attacks whose individual events appear benign in isolation will not be detected, limiting the platform's effectiveness against advanced persistent threats employing low-and-slow reconnaissance strategies. The single-host deployment model does not scale to enterprise environments requiring distributed log aggregation from heterogeneous source systems.

C. Future Work

Seven priority directions for future development are identified: (1) formal publication of per-class evaluation metrics from the training pipeline; (2) temporal event correlation via a sliding-window aggregation engine for multi-stage attack detection; (3) persistent database integration (PostgreSQL or SQLite) for historical threat analysis; (4) distributed collection agent architecture for multi-host deployment; (5) elevated Windows service for automated netsh remediation; (6) supplementation of the training corpus with CICIDS-2017/2018 and CIC-IDS-2022 datasets for contemporary attack coverage; and (7) MITRE ATT&CK technique mapping for threat intelligence workflow integration.

VIII. CONCLUSION

This paper has presented UpgradedAMIDES, a hybrid machine learning SIEM platform combining a TF-IDF character n-gram feature extractor with an XGBoost gradient-boosted ensemble classifier for real-time multi-format log threat classification. The central technical contribution is the identification and correction of a BRUTE_FORCE recall failure present in prior NSL-KDD-trained AMIDES implementations, arising from erroneous reliance on the num_failed_logins feature, which equals zero for the overwhelming majority of R2L attack records in the dataset.

Five engineered sub-type score features derived from the actual statistical distributions of NSL-KDD R2L records were introduced, providing explicit behavioral signatures for warezclient, guess_passwd, snmpguess, ftp_write, and httptunnel attack variants. These features substantially improve BRUTE_FORCE recall without requiring additional training data or secondary classifiers. A companion entropy suppression preprocessing layer eliminates high-entropy token-driven false positive inflation in the TF-IDF feature space, addressing a practically important but underdocumented challenge in character n-gram-based log classification.

The platform integrates these ML contributions into a production-grade system comprising a 14-endpoint FastAPI REST backend, an automated RemediationEngine with non-blocking action isolation, and a nine-page React/TypeScript dashboard providing comprehensive operational SIEM visibility. The architectural decisions made throughout the system — unified multi-format normalization, layer-separated feature construction, and synchronous remediation coupling — collectively demonstrate a principled approach to full-stack security automation that balances technical rigor with practical deployability.

References

[1] T. Bhatt, P. Bhatt, and D. Rughani, "Cloud forensics: Technical challenges, solutions and comparative analysis," in Proc. 2nd Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Coimbatore, India, 2018, pp. 1769-1774.

[2] R. Upadhyay and S. Salzsieder, "Survey of intrusion detection systems," in Proc. 2020 IEEE Int. Conf. Electro Inf. Technol. (EIT), Chicago, IL, USA, 2020, pp. 1-6.

[3] M. Roesch, "Snort: Lightweight intrusion detection for networks," in Proc. 13th USENIX Conf. Syst. Admin. (LISA), Seattle, WA, USA, 1999, pp. 229-238.

[4] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symp. Comput. Intell. Security Defense Appl. (CISDA), Ottawa, ON, Canada, 2009, pp. 1-6.

[5] 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. Security Privacy (ICISSP), Funchal, Portugal, 2018, pp. 108-116.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), San Francisco, CA, USA, 2016, pp. 785-794.

[7] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," IEEE Access, vol. 8, pp. 32464-32476, 2020.

[8] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in Proc. Military Commun. Inf. Syst. Conf. (MilCIS), Canberra, ACT, Australia, 2015, pp. 1-6.

[9] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," IEEE Trans. Ind. Informat., vol. 16, no. 3, pp. 1963-1971, Mar. 2020.

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manag., vol. 24, no. 5, pp. 513-523, 1988.

[11] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS), Dallas, TX, USA, 2017, pp. 1285-1298.

[12] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in Proc. 27th IEEE Int. Symp. Softw. Reliab. Eng. (ISSRE), Ottawa, ON, Canada, 2016, pp. 207-218.

[13] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning," J. Mach. Learn. Res., vol. 18, no. 17, pp. 1-5, 2017.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321-357, 2002.

[15] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," IEEE Commun. Surveys Tuts., vol. 12, no. 3, pp. 343-356, 3rd Quart. 2010.

[16] Gartner, Inc., "Market guide for security orchestration, automation and response solutions," Gartner Research, Stamford, CT, USA, 2020.