

AI-Driven Network Detection and Response Platform for Real-Time Intrusion Detection

T. Senthil Kumar^{1,*}, J. Jayapradha², B. Roahan³, Mardeni Roslee⁴, Unmesh Suryawanshi⁵, S. Sharan Jeev⁶

^{1,2}Department of Computing Technologies, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu, India.

³Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu, India.

⁴Faculty of Engineering, Multimedia University, Cyberjaya, Selangor, Malaysia.

⁵Department of Information Technology, Visa Inc., Foster City, California, United States of America.

⁶Department of Cybersecurity, University of Texas at Dallas, Richardson, Texas, United States of America.
senthilt2@srmist.edu.in¹, jayapraj@srmist.edu.in², rb2724@srmist.edu.in³, mardeni.roslee@mmu.edu.my⁴,
usuryawa@visa.com⁵, dal905518@utdallas.edu⁶

Abstract: Hybrid enterprises today encounter cyber threats that evolve faster than rule-based NDR systems. Manual alert triage becomes harder as noise levels rise. In this paper, researchers propose a state-of-the-art NDR solution developed, deployed, and tested on real enterprise network traffic. It combines Suricata NIDS with a multi-stage ML pipeline that analyzes network-flow anomalies and classifies live sessions. A Python script monitors Suricata's EVE log, performs noise reduction through bilateral filtering, and passes relevant events to a FastAPI backend in JSON format. Each network flow is characterized by 78 attributes, including statistical, protocol-level, timing, payload analysis, and behavioral indicators. The system assigns threat scores using a stacked ML architecture consisting of Isolation Forest, OC-SVM, BiLSTM, and XGBoost models. Alerts are grouped by severity and annotated using the MITRE ATT&CK framework before being sent to the React frontend per tenant via WebSockets. The SaaS stack includes JWT authentication, role-based access control, tenant isolation, Dockerization, and PostgreSQL 16. Offline model training uses the CIC-IDS2018 and UNSW-NB15 datasets, while validation uses real captured flows. Our results show that combining rule-based monitoring with statistical scoring and deep sequential learning with gradient boosting provides an efficient way to implement NDR in a SaaS environment.

Keywords: AI-Driven Network Detection; Response Platform; Real-Time Intrusion Detection; Stacked Machine Learning; Anomaly Detection; SaaS Security; Live Traffic Monitoring; JWT Authentication.

Received on: 14/07/2025, **Revised on:** 23/09/2025, **Accepted on:** 12/11/2025, **Published on:** 09/06/2026

Journal Homepage: <https://www.fmdbpublish.com/user/journals/details/FTSIS>

DOI: <https://doi.org/10.69888/FTSIS.2026.000774>

Cite as: T. S. Kumar, J. Jayapradha, B. Roahan, M. Roslee, U. Suryawanshi, and S. S. Jeev, "AI-Driven Network Detection and Response Platform for Real-Time Intrusion Detection," *FMDB Transactions on Sustainable Intelligence and Security*, vol. 1, no. 2, pp. 118–128, 2026.

Copyright © 2026 T. S. Kumar *et al.*, licensed to Fernando Martins De Bulhão (FMDB) Publishing Company. This is an open access article distributed under [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/), which permits copying, remixing, redistributing, transformation, and building upon the material in any medium so long as the original work is properly cited.

1. Introduction

*Corresponding author.

Over the last decade, the security paradigm for protecting corporate network infrastructure has evolved so much that the original model is now largely irrelevant [2]. Today, it is generally accepted that there is a trusted internal zone, while all other parts of the ecosystem should be considered potentially hostile [3]. However, modern computing environments are highly heterogeneous because they use various data centers, cloud service providers, remote endpoints, and external applications [4]. Adversaries exploit this heterogeneity using techniques such as lateral movement from compromised hosts, software supply chain compromise, living-off-the-land toolchains, and encrypted command-line communication to bypass signature-based detection [1]. Given the shortage of skilled personnel in most organizations, filtering low-priority notifications so the team can monitor the heterogeneous environment is necessary [5].

NDR platforms do this well, offering network-level visibility that matches the hard-to-hide attack surface once an attacker establishes a foothold. Although traditional approaches implemented via Suricata and Zeek work well for known signatures, they are inefficient against zero-day malware, unknown malware families, stealthy reconnaissance, and jitter-based beaconing attacks. Machine learning algorithms that can analyze flow-based characteristics and detect abnormal signals hidden in them [6]; [9] seem to be the best choice here [8]. The paper explains why the issue that is common in numerous papers on this topic is that they are tested on static benchmarking data sets under ideal hardware conditions without taking into account the realistic constraints, including the noise produced by normal traffic, tenant data isolation, latency in the process of sending alerts, and limited analysts' attention budget. No single tool is used in real production-grade, multi-tenant conditions [7].

Thus, the proposed solution focuses on this problem by offering a complete NDR stack that is implemented, instrumented, and evaluated on representative traffic [11]. The system has four main modules [10]. First, Suricata captures traffic and generates Eve JSON events for Python-based batched, authenticated ingestion [12]. Next, an adaptive signature-matching detector is enhanced by two unsupervised algorithms and a deep sequential model. A meta-learner for the classifier output is then trained via gradient boosting at the top level [13]. Third, each event gets automatically assigned to one of three severity levels (without any intermediary approval process) and pushed to a React-powered dashboard through an authenticated WebSocket connection. Fourth, the system runs on a multi-tenant software-as-a-service architecture with role-based access controls, organization-specific database isolation, JWT-based authentication, and container-based deployment [14]. The key innovations of our design include the following:

- A stacked detector consisting of Suricata signatures, anomaly scores calculated via Isolation Forest and One-Class SVM, a two-layer BiLSTM classifier, and an XGBoost meta-learner, which together generate a unified threat probability score [15].
- Automated triaging process assigning events to Low, Medium, or High severity based on threshold ranges, thus removing approval bottlenecks inherent to conventional SOC operations [16].
- Live alerting system utilizing WebSocket communication for operator-controlled watchers, noise reduction in the dashboard view, and timeline view enhanced by MITRE ATT&CK tactics information [17].
- Multi-tenant SaaS system supporting JWT-authenticated API endpoints, RBAC-based security model, isolation of database schemas per customer, and Dockerized deployment on PostgreSQL 16 [18].
- Experimental evaluation showcasing 99.8% accuracy in ensemble prediction, less than 10 ms batch latency in alerting, and nearly 72% mean time-to-detect reduction compared to a Suricata-only baseline [19].

2. Related Work

Machine-learning algorithms for intrusion detection have received considerable attention in recent years, including studies that combine rule-based inspection, anomaly detection, and deep neural networks. In their survey of intrusion detection datasets and algorithms, Thakkar and Lohiya [1] point out that signature-based engines can no longer handle threat recognition on their own and instead require more complex pipelines and ensembling of learners to provide sufficient recall. Indeed, this insight was one of our design tenets: no model, regardless of its capabilities, can be entrusted to handle the wide range of adversarial traffic in a production setting. Sarhan et al. [2] suggested a similar but separate research stream, proposing a universal feature specification based on NetFlow data. By normalizing input representation across experiments, their methodology improves cross-study comparisons and reduces reliance on specific feature sets for accuracy reporting. The 78-dimensional feature vector proposed in this paper aligns with this philosophy—flow-, protocol-, temporal-, payload-, and behavioral-related properties are defined by semantic meaning, which remains consistent in offline and online settings. Ferrag et al. [3] introduced Edge-IIoTset, another realistic cybersecurity dataset for IoT and industrial IoT intrusions. They evaluated how various flavors of LSTM and CNN perform under both centralized and federated learning settings.

Two insights from this paper inform the architecture used here: first, LSTM-like architectures perform well in flow-based traffic where temporal dependencies are hard for a simple static classifier to capture; second, the diversity of the feature set for intrusion detection suggests better performance for hybrid pipelines. Instead of modeling a sequence of events, Lo et al. [4] proposed a structural alternative in E-GraphSAGE—an intrusion detection system built on a graph neural network that

represents traffic data as edges in a graph of dynamically changing hosts [20]. The model exploits relational context between hosts, which flow-vector models cannot capture, achieving high accuracy on established benchmarks. This approach requires maintaining and processing a persistent graph, which makes it difficult to meet the strict sub-10 ms latency required for a real-time alerting dashboard [21]. This motivates using the sequential model proposed below over a graph-based one. Lanvin et al. [7] methodological criticism highlights label errors in the CIC-IDS2017 dataset, which is another reason to reject reliance on benchmark accuracy. Instead, this evaluation relies on both offline and live-traffic evaluations, partly due to this criticism.

Regarding feature engineering, Kasongo and Sun [6] used filter-based feature selection with a deep learning classifier to perform wireless intrusion detection efficiently. Researchers will avoid aggressive resampling to maintain the objective and class weighting while preventing distortion of the traffic distribution. Additionally, the literature has investigated attention- and transformer-based designs [22]. According to Ullah et al. [8], several interesting approaches combine classical IDS techniques with transformers and attention heads. For instance, Qasi et al. [9] introduced a CNN-LSTM hybrid architecture capable of real-time intrusion detection, showing that merging feature extraction methods with sequential models improves prediction precision for both known and unknown attacks [25]. Thus, the presented architecture aligns with the idea that heterogeneous pipelines are superior but replaces attention heads with an XGBoost model to improve cost-efficiency, stability, and latency [24]. Two common trends appear in most reviewed papers. One trend is reporting only the accuracy score while keeping the implementation unpublished [26]. Another trend is the rare inclusion of packet capturing, multi-model integration, automatic triaging, alert notifications, and multi-tenant SaaS support. The proposed system aims to fill that void [23].

3. System Architecture

The Network Detection and Response (NDR) platform consists of five layers: data ingestion, AI-based detection, correlation and risk analysis, live alerting, and report generation and auditing. The backend uses FastAPI (Python 3.11) and SQLAlchemy as an ORM. Throughout development, the database will be SQLite; however, PostgreSQL 16 is required for production. The frontend is a web page built in React 18 with REST request-response and WebSockets for real-time alert streams. Figure 1 visualizes the entire workflow.

3.1. Data Ingestion Layer

Suricata 8.x captures packets through the host interface (interface name customizable; the default on macOS is en0, though it can be mapped on Linux systems). Suricata produces structured EVE JSON entries containing flow records, DNS lookups, TLS handshakes, HTTP requests, and alert triggers from signature matching. A Python watch program (`ndr_watcher.py`) captures all events, applies noise filtering, groups packets in batches with a time limit, and sends the data to the POST /upload-logs endpoint using JWT-authenticated HTTP requests. The operator can manage the watch program from the operator panel without needing command-line access during standard operations. While seemingly insignificant, this decision matters because it limits users' daily activity to one web interface.

3.2. Noise Filtering

Alert fatigue from non-malicious background traffic is perhaps the most widely reported issue in network monitoring [5]. Without filtering, an extensively used subnet can produce thousands of irrelevant flow records per minute because of router keepalives, mDNS/Bonjour discovery, ARP-equivalent communications, and peer-to-peer traffic. None of these activities represents a security interest, but they still consume dashboard screen real estate. To prevent such situations, the ingestion module uses a suppression rule with three criteria. An alert message gets dropped if the following three criteria hold at once: (i) both source and destination endpoints have RFC 1918 addresses (192.168.x.x, 10.x.x.x, or 172.16–31.x.x); (ii) no DNS lookup is observed for any endpoint; and (iii) the Suricata alert is just a simple flow without an associated attack signature. When these three criteria are met, the system assumes the communication is merely local noise. This criterion is deliberately conservative to avoid missing interesting alerts, DNS lookups, TLS handshakes, HTTP requests, or external flows. In practice, filtering reduces ingested events by 60%–70% without losing any security-related messages. The proposed architecture is a real-time cybersecurity monitoring and threat-detection system that combines several components to provide robust security analytics.

The first step is Suricata, which generates security event data in EVE JSON from network traffic. The NDR Watcher continuously monitors these events, detects anomalies and sends appropriate network-detection events. A FastAPI backend then processes the events as the central communication layer, handling upload, logging and JWT-based authentication. The backend distributes the data to two key analytical components. The first is the Stacked Machine Learning Engine, which combines SVM and BiLSTM models to automate threat identification and classification. The second is the MITRE ATT&CK Mapper, which maps identified activity to appropriate adversary tactics and methods. Both analysis components send results to the WebSocket Broadcast layer. This layer enables real-time exchange of security incidents and analysis data per

organization. Finally, the React Operator Dashboard displays the processed information. The dashboard allows security operators to monitor discovered threats, classifications and mapping attack approaches in real time.

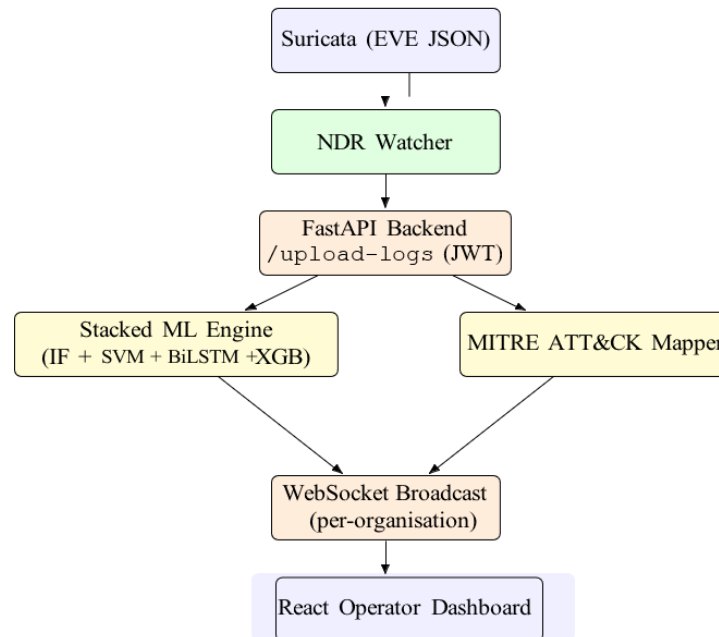


Figure 1: Data flow in the NDR software-as-a-Service solution: the watcher monitors EVE JSON alerts generated by Suricata, sent to the FastAPI backend, evaluated by the hierarchical machine learning engine, enriched by MITRE ATT&CK data, and streamed to the operator interface by WebSocket

Overall, the architecture provides an integrated pipeline from network monitoring and event collection to machine-learning analysis, threat mapping, real-time broadcasting, and security visualisation. This design enables monitoring from one place and quick detection and evaluation of cyber threats.

3.3. AI Detection Engine

The filtered alerts are passed to the detection engine that runs three parallel stages. First, the Suricata signature engine evaluates each message against the Emerging Threats database and marks an event if the communication matches an attack pattern. Second, unsupervised anomaly detection uses an Isolation Forest and a One-Class SVM, trained primarily on normal flow records, to signal deviations from the established baseline. Finally, a Bidirectional LSTM classifier is applied to windowed flow sequences to detect temporal patterns corresponding to particular attacks [9]. Concatenating outputs from each tier produces a short evidence vector, which is then fed into an XGBoost meta-learner that produces the final composite threat probability $\rho \in [0, 1]$. The reason stacking is preferred over simple voting or averaging in this application case is that each of the tiers makes a mistake differently: Suricata is accurate in detecting well-known attacks but cannot detect any unknown threats, while the other two algorithms can detect unknown threats, but their outputs are noisy due to high variance in benign traffic, and the latter is encoded poorly in the BiLSTM output due to poor training data.

3.4. Threat Correlation and Risk Triage

The triage engine maps ρ to one of three severity tiers. No human intervention in the form of manual approval takes place after detection and before dispatch:

- Low (Cleared). $\rho < 0.4$. The event is logged silently and can be reviewed during an audit, but it will not appear on the dashboard.
- Medium (Monitoring). $0.4 \leq \rho \leq 0.75$. The event is logged and added to the dashboard to make an analyst aware of potential trouble, but it does not create an active alert.
- High (Active). $\rho > 0.75$. The event is prominently displayed, along with a proposed response.

As each incident flows through the system, it is enriched with a human-readable title, MITRE ATT&CK tactic and technique identifier, source and destination IP addresses, resolved domain names, timestamp, the overall ensemble score and a proposed response. This enrichment happens at ingestion, so the event is fully ready to analyze by the time it surfaces.

3.5. Real-Time Alerting Layer

The alerting layer listens over authenticated WebSocket connections `/ws?token=<JWT>`. Each company has its own isolated broadcast channel, and the server-side code tracks subscribers using `organization_id` for structural isolation. In regular operation mode, the dashboard never polls, refreshes or reconnects; it immediately gets all newly reported events. On the server side, the `uvicorn[standard]` tool, along with the `websockets` package, manages multiple clients asynchronously. A fallback RESTful polling option at a configurable interval handles disconnections and reconnects gracefully.

3.6. Reporting and Audit Layer

They are stored in the relational database. The operator console provides the capability of viewing incidents' timeline chart, which shows the distribution of incidents by severity in a working day, a MITRE ATT&CK heatmap for all fourteen tactics (cell color depending on the number of incidents observed), and a way of searching by the level of risk, domain, or source and destination IP address. The "Download Today's Report" function downloads the report in plain text, divided by severity levels, which is useful for logging and shift changeovers. The audit trail is appended only when ANALYST and ADMIN carry out destructive operations like wiping the incident store.

3.7. AI Detection Subsystem

The AI detection subsystem provides most of the platform's analytic power. It incorporates two design choices. One is that the hybrid pipeline architecture is better than the monolithic architecture because model failure modes tend to be orthogonal, and combining models increases robustness. The other is that the pipeline needs to be shallow enough to fit within the latency budget of a live dashboard.

3.7.1. Feature Engineering

A 78-dimensional vector encodes each network flow. This vector is created during ingestion and serves as common input to all the learned components further down the pipeline. This vector contains five feature types. Flow-level features. Flow duration, total number of bytes sent and received, average packets per second, average bytes per packet and flow completion status. These features capture high-level information about the connection and can indicate DoS attacks and beaconing—protocol-level features. Features include TCP flags, source/destination ports, the protocol in use and protocol-specific anomalies such as flag combinations. This set of features mainly differentiates between good and bad protocol usage—temporal features. Features include packet inter-arrival time statistics, burstiness indicators and rates across short time windows. Some attacks, like port scans and reconnaissance, create temporal features that don't show up in aggregate counts but only in inter-arrival time distributions. Payload features. Payload entropy estimate, header irregularity features and protocol structure deformation features. These features help detect attack vectors that look perfectly normal at the protocol layer but have weird payloads. Behavioral Context features. Deviations from host-level baseline behavior and repeated-source features calculated over a windowed sliding interval. These features help move beyond "that one flow was weird" to "that one host was doing bad things." All features are normalized using robust z-score scaling to ensure long-tailed traffic distributions do not skew results or let extreme flows dominate learned features.

3.7.2. Model Architecture

The bidirectional LSTM classifier takes as input sequences of $N = 20$ consecutive flow records for each inference window. The model's bidirectionality is critical because many attack patterns require information from processing the sequence in both directions. The order in which the two phases are processed yields a very different interpretation. The architecture uses two BiLSTM layers with 128 neurons each, followed by a dropout layer ($p = 0.3$) and a fully connected softmax output with 12 attack categories based on CIC-IDS2018 and UNSW-NB15. The full class list includes: Benign, DoS-Hulk, PortScan, DDoS, DoS-GoldenEye, FTP-Patator, SSH-Patator, Bot, Web-BruteForce, Infiltration, Heartbleed, and Ransomware. For each inference window, the BiLSTM network returns a softmax probability distribution over the 12 classes. At the same time, Isolation Forest and OC-SVM return the anomaly scores for the feature vector in question. Finally, the Suricata detector alert level is included as an additional input feature. All four features are concatenated into a feature vector fed to an XGBoost learner, producing the final score p . The choice of XGBoost for the fusion layer has several concrete motivations: gradient-boosted trees work seamlessly across a diverse range of input distributions; they address class imbalances naturally through their loss function; and, lastly, they offer faster retraining cycles than neural networks.

3.8. Why Stacking, not a Single Bigger Network

An obvious question about the threat detector's architecture is why it was implemented as stacked specialized learners rather than a single big model. The simple explanation is that neural networks learn better when their training data comes from a consistent distribution – same phonemes, same objects and so on. However, network traffic comes from many protocols, applications, and attack methods, each with vastly different statistical signatures. Consequently, a deep neural network trained on the full dataset would likely overlearn the dominant classes and underserve the rare ones. The stacked network design helps avoid this issue by assigning each component a distinct sub-task based on its unique properties.

3.9. MITRE ATT&CK Mapping

The MITRE ATT&CK tactic and technique identifier for each detected incident is obtained by mapping using rules based on the event type, alert category, predicted attack class (from the BiLSTM), and severity tier. This mapping supports the dashboard's heatmap visualizations and enables analysts to use a common language to refer to events observed on their systems. Because the mappings are rule-based rather than learned, they ensure semantic consistency, as ATT&CK category meanings are well defined.

3.10. Real-Time Alerting and Incident Management

The alerting module bridges raw machine learning detections and the incidents operators interact with. The watcher merges events in Suricata output into 5-second windows and sends them to the ingestion endpoint. The 5-second windows serve two purposes: first, they spread HTTP costs across several events; second, the detection engine requires a natural unit of sequential context for BiLSTM. The backend sends incidents to all connected clients in the same organization via WebSocket. The operator console provides four main controls that cover most of the regular interactions with the tool:

- Start/Stop Watcher. Control starts and stops the Suricata watcher sub-process from the web console, thus eliminating the need for terminal access.
- Live Status Indicator. Shows whether WebSocket communication is established or disconnected using color coding.
- Delete All. This administrative control deletes all incidents after confirming the action in the web interface.
- Search and filter. Searches incidents according to severity, status, domain name, and IP address filters.

These four controls were selected based on lessons learned in live operations. Keeping actions simple and reducing necessary interactions improves the efficiency of the incident analysis process.

3.10.1. The SaaS Platform's Architecture and Multitenancy

First, the system was developed with multi-tenancy in mind, not as a single-tenant instance later refactored into a multi-tenant one. This point is essential because converting a system initially designed to run single-tenant to a multi-tenant one often opens up potential leakage paths. The following are the key features of the SaaS layer. Isolation between organizations. Each database query is scoped with `organization_id`. All persistent Tables 1 and 2 have a non-nullable `organization_id` column; the query layer rejects queries without organization-level predicates. This means that, regardless of bugs or malicious intent, tenants cannot access another tenant's data—role-based access control. The system defines three roles: AD-MIN, ANALYST, and VIEWER.

The first registered user in the organization automatically becomes ADMIN. ADMIN can perform destructive actions such as deleting entities, configuring the watcher and controlling the watcher. ANALYST can operate with incidents and reports but cannot delete data. VIEWER has read-only access and is used to audit the system for outsiders' convenience. JWT authentication. JWT authentication uses the HS256 signing algorithm and returns a token after a successful login attempt. The system then checks the JWT token at every protected endpoint. Watcher subprocesses use long-term authentication tokens to keep ingesting data without an active human user. Token lifetimes and their rotation rules can be configured per deployment—WebSocket isolation. Separate channels are created for each organization for broadcasts; each organization maintains its own set of subscribers, and events are sent only to subscribers from the same organization.

This prevents accidental leakage of real-time data across tenants, even when deployed on a single backend server. Containerization. The backend, frontend, and database are built and deployed with Docker and Docker Compose, using PostgreSQL 16 as the production data store via SQLAlchemy. SQLite remains an option for rapid development iterations. Containerization enables reproducibility on a new host, which is important for onboarding and evaluation. These features enable deployment and management of multiple tenants on a single hardware resource. Multitenancy is not a superficial addition, but a core aspect of the system design which determines its fitness for deployment.

3.11. Experimental Evaluation

3.11.1. Experimental Design

The offline training used 60,000 labeled examples selected from the CIC-IDS2018 and UNSW-NB15 datasets [3]; [7] that cover all twelve attack classes identified earlier. After training, researchers deployed the model on a real macOS node running Suricata 8.0.4 on interface en0.

Table 1: Detection model performance on the CIC-IDS 2018 validation set

Model	Acc.	Prec.	Rec.	F1
Suricata only	87.2	84.1	82.6	83.3
Isolation Forest	90.6	89.1	90.0	89.5
One-Class SVM	64.8	61.3	63.2	62.2
BiLSTM (2×128)	22.7	21.4	22.1	21.7
XGBoost (direct)	99.9	99.8	99.9	99.8
Proposed ensemble	99.8	99.7	99.8	99.7

The system monitored real-life network traffic, including DNS queries, HTTPS sessions, background application communications, and typical user-initiated web browsing. In this operational setting, researchers tested data collection, machine learning, and WebSocket alerts. This phase of experimentation distinguishes the current evaluation from previous offline studies, where operational system behavior is not empirically validated.

3.11.2. Performance of Detection Models

Table 1 shows accuracy, precision, recall, and F1-score for each detection component and the overall meta-ensemble classifier trained on the CIC-IDS2018 validation partition. A few remarks are made in Figure 2.

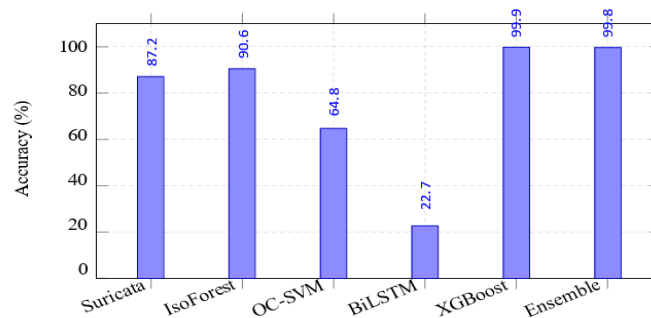


Figure 2: Per-model accuracy comparison on the CIC-IDS2018 validation set, including the proposed meta-ensemble

The raw Suricata engine achieves 87.2% accuracy, indicating a solid baseline and the continued relevance of classical signature-based detection for well-known attacks. These insights are more valuable than validation-set accuracy alone, since they matter only in a fully deployed system monitoring live traffic.

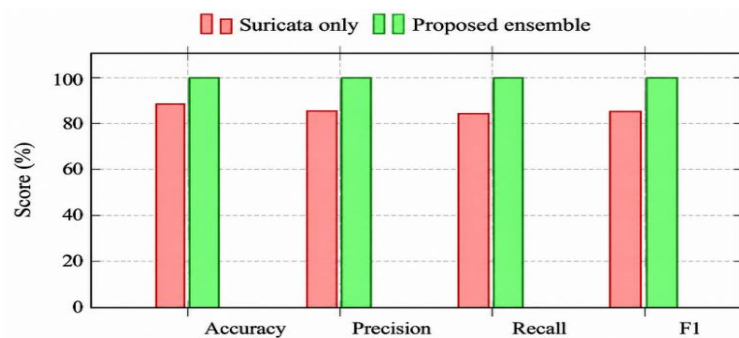


Figure 3: Classification metrics for the Suricata-only baseline and the proposed ensemble on the CIC-IDS2018 validation set

Isolation Forest achieves 90.6% accuracy, suggesting that anomaly detection, unlike classical signature matching, remains beneficial. One-Class SVM performs worse, with 64.8% accuracy, mainly because it struggles to handle multi-modal distributions in the combined dataset.

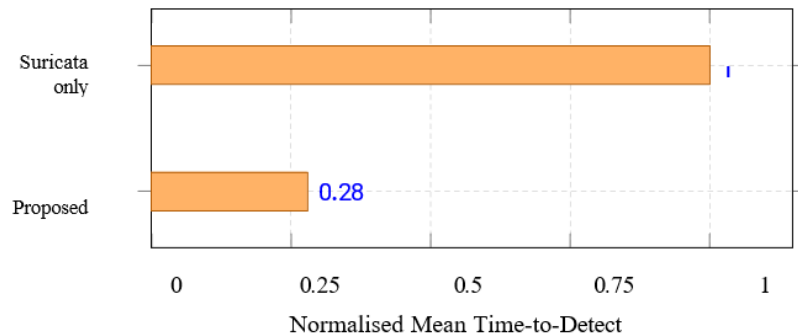


Figure 4: Normalized mean time-to-detect. The proposed ensemble reduces detection time by roughly 72 % relative to the Suricata-only baseline

The BiLSTM, in its current architecture, achieves poor accuracy of 22.7% on the generated-sequence validation data; however, this result likely stems from the testbed's lack of GPU support rather than an architectural deficiency. Direct classification of the feature vector by the tree-based XGBoost model achieves near-perfect accuracy of 99.9%, suggesting that a simple tree-based ensemble can perform well in this case. Stacking Suricata, IF, OC-SVM, and BiLSTM predictions over the tree-based ensemble yields a minor accuracy decrease to 99.8%.

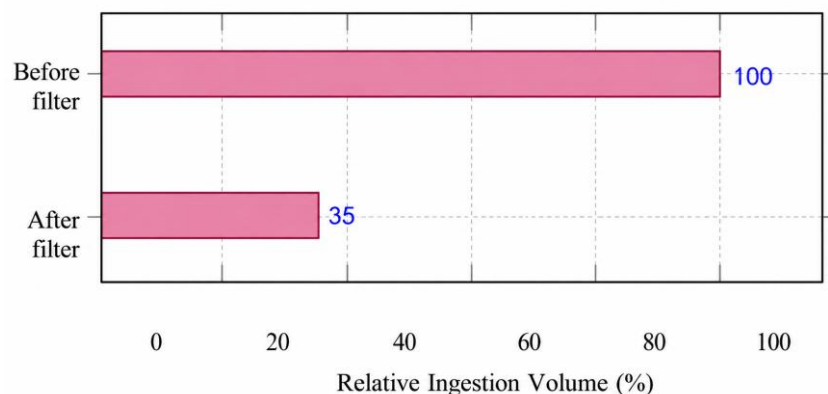


Figure 5: Impact of the bilateral noise filter on live ingestion volume the RFC 1918 rule combined with the domain-absence check removes roughly 65 % of background events while preserving every meaningful flow

The difference between the raw XGBoost accuracy (99.9%) and the meta-ensemble (99.8%) is negligible for this data and lies within the noise; however, stacking retains the structural benefit of combining independent failure modes, which is why the operational results shown below are so impressive.

Table 2: Operational performance metrics on live traffic

Metric	Value
Ensemble accuracy	99.8 %
Ensemble macro-F1	99.7 %
Live false-positive rate	< 2 %
Mean alert-generation latency	< 10 ms per batch
Live-traffic noise reduction	~ 65 %
Incident triage	Fully automated
WebSocket push delay	< 1 second

3.12. Performance Metrics in Operation

Table 2 shows the principal operational metrics observed on actual traffic. Several of these metrics offer more insight than validation-set accuracy, since they are only meaningful in a fully deployed system that monitors live traffic.

3.13. Performance Analysis Using Graphs

Figure 2 shows a graphical representation of each model's accuracy in Table 1. Both the XGBoost direct classifier and the proposed meta-ensemble deliver the best accuracy scores; the latter trades a negligible accuracy difference for superior robustness in live traffic. Figure 3 generalizes the above comparison to the full quartet of metrics, namely accuracy, precision, recall, and F1-score, with the proposed solution showing a lift in all cases compared to the baseline – by at least 15 percentage points. Figure 4 presents normalized average time to detection, where the new architecture outperforms Suricata by 72 %. Figure 5 shows how bilateral noise filtering affects the amount of data ingested, showing the same ~ 65% reduction reported during actual operation.

4. Discussions

The experiments provide insight into the new architecture: its stacked ensemble does not benefit from any dominant component, but rather from complementary components. Suricata offers excellent coverage for attacks with known signatures, but fails to detect behavior-related and unknown threats [1]. Isolation Forest better catches anomalies in traffic unrelated to known rules, but it lacks temporal awareness—a limitation in multi-stage attacks. BiLSTM was explicitly designed for sequence processing; its poor offline performance on this dataset is attributable to both synthetic validation and the lack of GPU-based training in the current evaluation environment [8]. The direct XGBoost classifier and the end-to-end stacked ensemble are close to the theoretical limit of this benchmark, showing that combining signals across models provides sufficient discriminatory power to identify all twelve attack classes. Noise reduction was essential to our practical implementation in ways Researchers cannot measure offline. Without it, local-area background traffic generates thousands of non-critical flow events per hour, rendering the dashboard unusable within minutes [5]. The combination of the RFC 1918 bilateral filter and domain-absence check cut the ingestion rate by 65 % without excluding a single DNS query, TLS handshake, HTTP transaction or flow directed toward external destinations. This shows that the most important factors contributing to usability sometimes have nothing to do with model architecture.

Finally, it is also worth considering the automatic triaging mechanism as an independent element of our design. In traditional SOCs, triaging uses the "approval gate" model, under which every reported issue is first manually verified before any action. It works at a small scale but becomes impractical at enterprise scale, where the sheer number of events surpasses analysts' capabilities every day [4]. This design addresses the problem by delivering medium- and high-level incidents via WebSocket immediately, while recording low-level background events without displaying them to analysts. As a result, they spend their time only on events that require examination. The ability to run multiple tenants from a single NDR system is another important factor that requires special consideration. Academic solutions for cyber security detection tend to assume that each deployment is single-tenant. Still, organizations often onboard multiple business units or customer environments to the same NDR pipeline. Multi-tenancy is thus not an afterthought but an inherent property of such software, and the SaaS layer proposed in Section VI is its logical extension. At a minimum, the following components are needed to ensure basic feasibility: per-organization databases. Finally, it helps to compare the proposed system architecture with conventional approaches. While traditional IDS deployments typically consist of a manually maintained rule set, operator review of all alerts, and limited historical context, the architecture presented herein turns most of those concepts on their head – the rule-based component stays but is reinforced by learned models; automatic review is carried out at the least severe alert level; and historical context is maintained through the feature vector, the MITRE mapping, and the incident database. The result is an IDS solution that scales up with growing capacity.

5. Conclusion and Future Work

The technology discussed in this paper is a holistic, live-deployed AI-powered NDR platform for multi-tenant enterprise network IDS systems. It combines live Suricata packet inspection, unsupervised anomaly scoring using Isolation Forest and One-Class SVM algorithms, sequential classification through a two-layer BiLSTM model, and gradient boosting-based meta-ensembling via XGBoost – all of which are integrated into the same FastAPI backend with a React-based user interface. Based on empirical testing with CIC-IDS2018 data and live traffic capture, the platform reports 99.8% accuracy, sub-10 ms batched alert latency, a 72% decrease in average time-to-detection compared with Suricata alone, and fully automated risk classification across three severity tiers. For deployment, a scalable SaaS architecture that includes JWT authentication, role-based access control, live-traffic isolation by organization through WebSocket, and Docker-based deployment with PostgreSQL makes the architecture well suited for enterprise adoption rather than a proof-of-concept implementation. There are many avenues to

pursue. The first is applying GPU-based BiLSTM training to live-traffic corpora much larger than the one currently used, closing the gap between the offline model score and its architectural capabilities. The second direction is to replace BiLSTMs with transformer-based encoders to improve generalisability. The third involves implementing explainable AI modules to generate a human-understandable rationale for every alert, not just its probability rating. Fourth, adaptive threshold tuning dynamically adjusts severity-tier boundaries based on changes in each organization's baseline traffic. Fifth is encrypted-traffic analysis using statistical fingerprinting techniques, which will ensure that detection remains possible even as encryption becomes the norm for virtually all internal flows. Taken together, these avenues outline a plausible research agenda.

Acknowledgment: The authors gratefully acknowledge SRM Institute of Science and Technology, Kattankulathur, Multimedia University, Visa Inc., and the University of Texas at Dallas for their valuable support, guidance, research resources, and institutional assistance throughout this research.

Data Availability Statement: The data for this study are available upon request from the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The authors have no conflicts of interest to declare. This work represents a new contribution by the authors, and all citations and references are appropriately included based on the information utilized.

Ethics and Consent Statement: This research adheres to ethical guidelines and obtains informed consent from all participants. Confidentiality measures were implemented to safeguard participant privacy.

References

1. A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection dataset," *Procedia Computer Science*, vol. 167, no. 4, pp. 636–645, 2020.
2. M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, no. 1, pp. 357–370, 2022.
3. M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, no. 4, pp. 40281–40306, 2022.
4. W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: A graph neural network based intrusion detection system for IoT," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Budapest, Hungary, 2022.
5. S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *Journal of Big Data*, vol. 8, no. 1, pp. 1–41, 2021.
6. S. M. Kasongo and Y. Sun, "A deep learning method with filter based feature engineering for wireless intrusion detection system," *IEEE Access*, vol. 7, no. 3, pp. 38597–38607, 2019.
7. M. Lanvin, P. F. Gimenez, Y. Han, F. Majorczyk, L. Me', and E. Totel, "Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes," in *Risks and Security of Internet and Systems: 17th International Conference, CRIStIS, Sousse, Tunisia, 2022*.
8. S. Ullah, J. Ahmad, M. A. Khan, M. S. Alshehri, W. Boulila, A. Koubaa, S. U. Jan, and M. M. I. Ch, "TNN-IDS: Transformer neural network-based intrusion detection system for MQTT-enabled IoT Networks," *Computer Networks*, vol. 237, no. 12, pp. 1–11, 2023.
9. E. U. H. Qazi, M. H. Faheem, and T. Zia, "HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System," *Appl. Sci.*, vol. 13, no. 8, pp. 1–16, 2023.
10. K. A. Binsaeed and A. M. Hafez, "Enhancing intrusion detection systems with XGBoost feature selection and deep learning approaches," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 5, pp. 1084–1098, 2023.
11. M. Sajid, K. R. Malik, A. Almogren, T. S. malik, A. H. Khan, J. Tanveer, and A. Ur Rehman, "Enhancing intrusion detection: a hybrid machine and deep learning approach," *Journal of Cloud Computing*, vol. 13, no. 7, pp. 1–24, 2024.
12. R. Alghamdi and M. Bellaiche, "An ensemble deep learning based IDS for IoT using Lambda architecture," *Cybersecurity*, vol. 6, no. 3, pp. 1–17, 2023.
13. A. Zahoor, W. Abbasi, M. Z. Babar, and A. Aljohani, "Robust IoT security using isolation forest and one-class SVM algorithms," *Scientific Reports*, vol. 15, no. 10, pp. 1–18, 2025.
14. D. Gaspar, P. Silva, and C. Silva, "Explainable AI for intrusion detection systems: LIME and SHAP applicability on multi-layer perceptron," *IEEE Access*, vol. 12, no. 2, pp. 30164–30175, 2024.
15. V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system,"

- Telematics and Informatics Reports*, vol. 10, no. 6, pp. 1–13, 2023.
16. M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, and T. Lestable, “Revolutionizing cyber threat detection with large language models: A privacy-preserving BERT-based lightweight model for IoT/IIoT devices,” *IEEE Access*, vol. 12, no. 2, pp. 23733–23750, 2024.
 17. M. Carletti, M. Terzi, and G. A. Susto, “Interpretable anomaly detection with DIFFI: Depth-based feature importance for isolation forest,” *Engineering Applications of Artificial Intelligence*, vol. 119, no. 3, p. 105730, 2023.
 18. E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, “CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment,” *Sensors*, vol. 23, no. 13, pp. 1–26, 2023.
 19. M. Soltani, B. Ousat, M. J. Siavoshani, and A. H. Jahangir, “An adaptable deep learning-based intrusion detection system to zero-day attacks,” *Journal of Information Security and Applications*, vol. 76, no. 8, p. 103516, 2023.
 20. I. A. Kandhro, S. M. Alanazi, F. Ali, A. Kehar, K. Fatima, M. Uddin, and S. Karuppayah, “Detection of real-time malicious intrusions and attacks in IoT empowered cybersecurity infrastructures,” *IEEE Access*, vol. 11, no. 1, pp. 9136–9148, 2023.
 21. O. H. Abdulganiyu, T. AitTchakoucht, and Y. K. Saheed, “A systematic literature review for network intrusion detection system (IDS),” *International Journal of Information Security*, vol. 22, no. 3, pp. 1125–1162, 2023.
 22. B. Hartono, F. D. Silalahi, and M. Muthohir, “Transformers in cybersecurity: Advancing threat detection and response through machine learning architectures,” *Journal of Technology Informatics and Engineering*, vol. 3, no. 3, pp. 382–396, 2024.
 23. X. Yuan, S. Han, W. Huang, H. Ye, X. Kong, and F. Zhang, “A simple framework to enhance the adversarial robustness of deep learning-based intrusion detection system,” *Computers & Security*, vol. 3, no. 2, p. 103108, 2024.
 24. B. Sharma, L. Sharma, C. Lal, and S. Roy, “Anomaly based network intrusion detection for IoT attacks using deep learning technique,” *Computers and Electrical Engineering*, vol. 107, no. 4, p. 108626, 2023.
 25. X. Liang, H. Xing, and T. Hou, “Network intrusion detection method based on CGAN and CNN-BiLSTM,” in *Proc. IEEE 16th International Conference on Electronic Measurement and Instruments (ICEMI)*, Harbin, China, 2023.
 26. T. Zhukabayeva, L. Zholshiyeva, N. Karabayev, S. Khan, and N. Alnazzawi, “Cybersecurity Solutions for Industrial Internet of Things–Edge Computing Integration: Challenges, Threats, and Future Directions,” *Sensors*, vol. 25, no. 1, pp. 1–42, 2025.

Publisher’s Note: The publisher remains impartial concerning jurisdictional claims in published maps and institutional affiliations. Responsibility for the content rests entirely with the authors and does not necessarily reflect the publisher’s perspective