Sentinel AI+: An Explainable Hybrid Intrusion Detection Framework with Analyst-Feedback Threshold Calibration

Dr. Himanshu V. Taiwade[1], Aachal Sahu[2], Durgesh Waghmare[3],
Suresh Nair[4], Vijay Shriwas[5], Ayush Ambolikar[6]
[1]*Assistant Professor, Dept. of Computer Science & Engineering, Priyadarshini College of Engineering, Nagpur, India*
[2,3,4,5,6]*Student, Dept. of Computer Science & Engineering, Priyadarshini College of Engineering, Nagpur, India*

***Abstract*— Machine-learning intrusion detection systems (IDS) are hard for security analysts to trust and tune because alerts rarely come with evidence and decision thresholds are fixed at training time. This paper presents SentinelAI+, a prototype network-IDS framework that combines a multi-class Random Forest, an Isolation Forest anomaly scorer, a transparent weighted threat score, IP-intelligence enrichment, and a Dynamic Threat Confidence Learning (DTCL) module that moves the alerting threshold within fixed bounds according to analyst verdicts, without retraining the underlying models. The system is implemented as a Flask/SQLite service with an optional Scapy capture agent and a browser-based SOC dashboard; every stored record carries a data-source label, and metrics are never synthesised when no trained model exists. We describe the architecture and algorithms, audit the implementation against its stated design goals, and report design-level properties and defects that bear on its behaviour. No detection-performance results are reported, because the audited artifact contains neither trained models nor benchmark data. We therefore specify a reproducible evaluation protocol on CICIDS2017 and UNSW-NB15 with ablation baselines, leave-one-attack-out zero-day simulation and simulated analyst feedback. The contribution is a documented, auditable system design and an explicit evidence agenda, not a demonstrated accuracy gain.**

***Index Terms*— Intrusion Detection, Random Forest, Isolation Forest, Explainable AI, Analyst Feedback, Threshold Calibration, Security Operations Center.**

I. INTRODUCTION

Network intrusion detection systems (IDS) monitor traffic for malicious activity. Signature-based systems recognise previously catalogued attacks, whereas anomaly-based and machine-learning (ML) systems aim to generalise to behaviour not seen before. Despite extensive academic work on ML-based detection, Sommer and Paxson argued that such systems are rarely used in operational settings and that finding attacks differs fundamentally from the tasks in which ML usually succeeds [1]. A systematic review of AI methods for zero-day attack detection likewise concludes that ML is promising but that data availability, algorithmic complexity and real-time operation remain significant challenges [2].

Three practical difficulties motivate this work. First, evaluation validity: Arp et al. found common pitfalls in the design, implementation and evaluation of learning-based security systems to be widespread in 30 papers from top security venues [3]. Benchmark data are themselves contested: Tavallaee et al. identified deficiencies in KDD CUP 99 and proposed NSL-KDD [4]; UNSW-NB15 [5] and CICIDS2017 [6] were created to provide more modern traffic and attacks, yet Engelen et al. later reported problems in CICIDS2017's traffic generation, flow construction, feature extraction and labelling [7]. Second, operational traffic is a stream: Miani et al. survey IDS built on data-stream ML and highlight continuous data influx, shifting class distributions, new attack types and scarce labels for model updates [8]. Third, analysts need to know why an alert fired; per-prediction explanation methods such as SHAP [9] and LIME [10]

exist, but must be wired into an alert workflow to be useful.

Sentinel AI+ is an academic prototype that addresses these difficulties at the level of system design. The paper makes four contributions, stated at the strength the evidence supports:

1) Architecture. A modular pipeline that fuses supervised classification and unsupervised anomaly evidence through an explicit, inspectable threat score (Section IV).

2) DTCL. A bounded, logged threshold-calibration rule driven by analyst verdicts that never retrains the base models (Section IV-E). We do not claim that feedback-driven calibration is new; we document one auditable instantiation.

3) Provenance-preserving implementation. Every record is tagged with its data source, and dashboard metrics are derived only from stored records, analyst feedback or recorded offline evaluation (Section V).

4) Audit and protocol. An audit of the code against its design objectives, with design-level findings (Section VII), and a reproducible evaluation protocol (Section VI). No performance claims are made.

The remainder of the paper is organised as follows: Section II reviews related work, Section III states the problem and objectives, Section IV presents the methodology, Section V the implementation, Section VI the evaluation protocol, Section VII the analysis, Section VIII the limitations, and Section IX concludes.

II. RELATED WORK

A. Evaluation data and methodology

KDD CUP 99 was long the standard IDS benchmark, but Tavallaee et al. reported important deficiencies and derived NSL-KDD from selected records [4]. Moustafa and slay built UNSW-NB15 as a hybrid of real modern normal activity and synthesised contemporary attacks, generated with the IXIA PerfectStorm tool [5]. Sharafaldin et al. generated CICIDS2017 because adequate datasets for anomaly-based IDS were lacking [6], and Engelen et al. then documented problems with its construction and showed that benchmark results change materially once they are corrected [7]. Arp et al. generalise the lesson: pitfalls in learning-based security systems are widespread, and results obtained under them may not transfer to practice [3]. These works shape our evaluation protocol (Section VI): we plan to use a corrected CICIDS2017 and avoid random-split-only reporting.

B. Machine-learning and adaptive detection

Por et al. review AI-based zero-day detection and report data availability, algorithmic complexity and real-time application as open challenges [2]. Miani et al. organise stream-based IDS research around the classic data-mining steps and name continuous data influx, distribution change, new attack types and label scarcity as defining issues [8]. SentinelAI+ is not a stream learner: its only adaptive element is a scalar threshold, which makes it cheap and auditable but limits what adaptation can achieve (Section VII).

C. Explainability

SHAP assigns each feature an importance value for a particular prediction [9], and LIME explains an individual prediction by fitting an interpretable model locally around it [10]. Both operate per instance. The current SentinelAI+ build does not yet implement either: it reports global Random Forest feature importances (Section V), and per-instance explanation remains future work.

D. Positioning and scope of this review

Table I summarises how each cited work bears on SentinelAI+. The review is deliberately narrow and is not a systematic survey. In particular, literature on alert triage, active learning and analyst-in-the-loop IDS has not yet been searched and must be added before submission; consequently, we make no claim about the absence of prior feedback-driven threshold schemes.

Table I
Cited Work and Its Bearing on This Paper

Ref.	Reported focus (per source)	Bearing on SentinelAI+
[1]	ML for network IDS is rarely deployed operationally; guidelines for anomaly detection research	Motivates analyst-facing, auditable design
[2]	Systematic review of AI zero-day detection; data, complexity and real-time challenges	Motivates hybrid design; zero-day ability here is unverified
[3]	Pitfalls in ML-for-security evaluation, widespread in 30 reviewed papers	Shapes protocol and provenance policy
[4]–[7]	KDD CUP 99 flaws; UNSW-NB15; CICIDS2017; CICIDS2017 construction problems	Candidate evaluation data and caveats
[8]	Survey of data-stream IDS: drift, new attacks, label scarcity	Motivates adaptation; ours adapts only a threshold
[9], [10]	Per-prediction explanation (SHAP, LIME)	Intended XAI; not yet implemented

III. PROBLEM STATEMENT AND OBJECTIVES

Let x_t be the feature vector of the t-th observed traffic record. The system must (i) decide whether to raise an alert, (ii) attach a severity, supporting evidence and a recommended action, and (iii) accept an analyst verdict $v_t \in$ {TRUE_POSITIVE, FALSE_POSITIVE, BENIGN, NEEDS_INVESTIGATION} and use it to adapt future alerting sensitivity in a bounded and auditable way.

The design objectives, taken from the project proposal, are: O1, detect both known and unknown attacks through a hybrid of supervised and unsupervised models; O2, reduce false positives through DTCL; O3, integrate threat intelligence and explainability; and O4, provide a real-time SOC dashboard. These are design objectives. Section VII examines what the implementation can and cannot support for each, and Section VI states the evidence required to claim any of them is met.

IV. METHODOLOGY AND SYSTEM ARCHITECTURE

A. Overview

Fig. 1 shows the data flow of the implemented system. Records arrive from a Scapy capture agent or from explicitly labelled dataset/simulation sources, pass through an ingestion API, are converted to a fixed feature vector and scaled, and are scored by two models in parallel. A transparent score and the current threshold decide whether an alert is created; alerts are shown on a dashboard where analyst verdicts feed the DTCL threshold update.

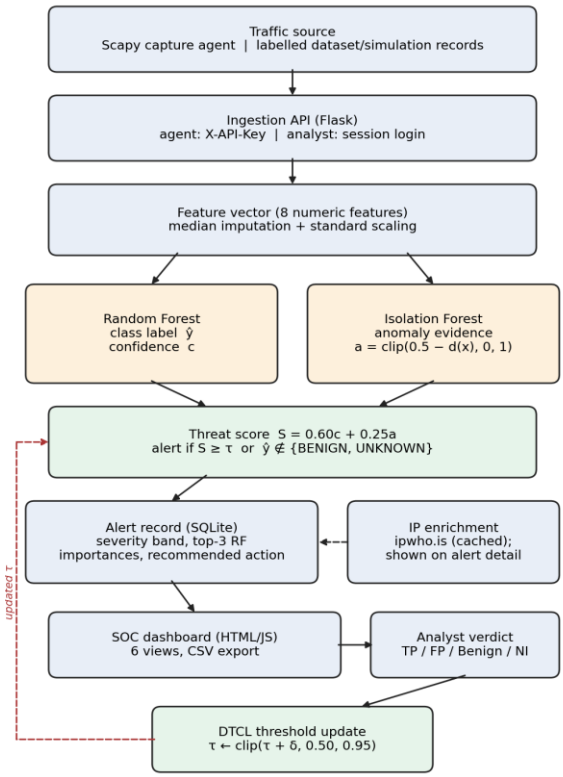

Fig. 1. SentinelAI+ data flow as implemented. Solid arrows: record processing; dashed red path: analyst verdict updating the threshold τ used by the alert rule. TP/FP/NI: true positive, false positive, needs investigation.

B. Features and preprocessing

Each record is reduced to eight numeric features: packet length, source port, destination port, flow duration, packet count, byte count, packet rate and bytes per second. Preprocessing applies median imputation and standard scaling, fitted on the training split and persisted with the models. TCP flags, listed

in the project proposal, are not part of the implemented feature vector.

C. Hybrid detection

A Random Forest classifier [11] with 200 trees and balanced class weights provides a class label $\hat{y}$ and a confidence c equal to the maximum class probability. An Isolation Forest [12] with 200 trees scores how easily a record is isolated by random partitioning; shorter average path lengths indicate anomalies. With d(x) the library decision function (positive for inliers), we define anomaly evidence

$$a(x) = \text{clip}\,(0.5 - d(x), 0, 1) \quad (1)$$

D. Threat score and alert rule

The unified threat score is deliberately simple and inspectable:

$$S(x) = \text{clip}\,(0.60\, c(x) + 0.25\, a(x), 0, 1) \quad (2)$$

Severity bands are Critical for $S \geq 0.90$, High for $S \geq 0.75$, Medium for $S \geq 0.60$ and Low otherwise. The weights 0.60 and 0.25 are fixed design constants; they were not tuned on data. An alert is raised when

$$S(x) \geq \tau \quad \text{or} \quad \hat{y} \notin \{\text{BENIGN, UNKNOWN}\} \quad (3)$$

where τ is the current DTCL threshold. Threat-intelligence data are displayed beside the alert but do not enter S. Algorithm 1 summarises record processing.

Algorithm 1. Processing one traffic record

Input: record r, source label s

```
x <- features(r)                    # 8 values
(y, c) <- RF(pre(x)); a <- Eq.(1)
S <- clip(0.60c + 0.25a, 0, 1)
tau <- current threshold
store traffic event (r, x, s)
if S >= tau or y not in
{BENIGN, UNKNOWN} then
    sev <- band(S)
    expl <- top-3 RF importances
    store alert(y,c,S,a,sev,expl,OPEN)
return alert id or none
```

E. Dynamic Threat Confidence Learning (DTCL)

DTCL maintains a single global threshold, initialised at $\tau_0 = 0.70$. Each analyst verdict v_t changes it by a fixed step and the result is clamped:

$$\tau_{t+1} = \min\,(0.95, \max\,(0.50, \tau_t + \delta(v_t))) \quad (4)$$

$$\delta(v) = -0.02\ (\text{TP});\ +0.02\ (\text{FP, BENIGN});\ 0\ (\text{NI}) \quad (5)$$

A confirmed true positive lowers τ to preserve sensitivity; a rejected alert raises it to suppress similar alerts; a request for investigation leaves it unchanged. Every update stores the previous and new thresholds, the adjustment, the reason, the verdict type, the alert's attack category and a timestamp, and the alert status becomes RESOLVED or INVESTIGATING (Algorithm 2). The base models are never retrained.

Algorithm 2. DTCL update on analyst verdict

Input: alert id, analyst id, verdict v

```
tau <- current threshold
d <- -0.02 if v=TP; +0.02 if v in
{FP, BENIGN}; 0 if v=NI
tau' <- min(0.95, max(0.50, tau + d))
persist tau'; insert feedback row and
history row (tau, tau', d, reason,
category, time)
status <- RESOLVED (TP/FP/BENIGN)
or INVESTIGATING (NI)
```

F. Enrichment, explanation and provenance

For public source addresses the system queries an external IP-intelligence service (ipwho.is by default), caches country, city, ISP and ASN in the database, and skips private and loopback ranges. A reputation value is stored only if the provider returns one; none is invented. The alert explanation lists the three largest global feature importances of the Random Forest. Finally, each traffic record stores its data source (for example LIVE_CAPTURE or DATASET / SIMULATION). If no trained model exists the engine returns the explicit label MODEL_NOT_TRAINED with zero confidence, and the dashboard reports no model metrics.

V. SYSTEM IMPLEMENTATION

A. Technology stack

The prototype is written in Python. The backend uses Flask 3.1.1 with Flask-CORS and Flask-SocketIO 5.5.1 for push updates, and SQLite for persistence. Models use scikit-learn 1.7.1 with pandas, NumPy and joblib; live capture uses Scapy 2.6.1; HTTP enrichment uses requests. The front end is framework-free HTML, CSS and JavaScript served by the backend. The SHAP package (0.48.0) is listed in the

requirements file but is not imported by any source file in the archive.

B. Backend, persistence and security controls

The backend exposes 17 REST endpoints covering authentication, packet and traffic ingestion, alerts, per-alert feedback, threat intelligence, dashboard statistics, analytics, DTCL status and history, model prediction, an AI-explanation stub and CSV alert export. Seven tables hold users, traffic events, alerts, cached threat intelligence, analyst feedback, DTCL history and system settings. Agent ingestion requires an API key in the X-API-Key header; analyst endpoints require a session login; passwords are hashed with Werkzeug; SQL statements are parameterised. The AI-explanation endpoint returns the supplied context and a not-configured message when no provider key is set, rather than a generated answer.

C. Capture agent, training and dashboard

The agent captures IP packets with Scapy and posts one record per packet containing addresses, ports, protocol, length and placeholder flow fields (flow duration 0, packet count 1). The training script reads a user-supplied CSV, maps CICIDS/UNSW-style column names onto the eight features, collapses labels with fewer than two samples into "Other", performs a stratified 75/25 split (seed 42), fits the preprocessor, Random Forest and Isolation Forest, and stores weighted precision, recall, F1, accuracy and a confusion matrix. The dashboard has six views: overview, live monitoring, threats, analytics, DTCL and reports. A six-record deterministic generator (three benign, three attack-like) exercises the pipeline end to end; it is a functional demonstration, not an evaluation.

D. Implementation status

Table II compares the proposal with what the submitted code contains. "Unvalidated" means the code exists but no experiment has been run.

Table II

Design Intent Versus Implemented Status

Component	Design intent	Status in submitted code
Capture	Scapy or PyShark; flow duration	Scapy only; per-packet records; flow fields are placeholders. Unvalidated
Features	IP, protocol, port, length, TCP flags, duration	Eight numeric features; no TCP flags. Partial
Random Forest	Known attacks	Training and inference code; no model file. Unvalidated
Isolation Forest	Unknown / zero-day	Code present; fitted on all training rows. Unvalidated
Threat intel.	GeoIP, ASN, reputation, TI APIs	One provider; geo/ISP/ASN; reputation usually null; not scored. Partial
DTCL	Confidence from feedback and history	Bounded global threshold step; no prediction history, no per-class state. Partial
XAI	SHAP / LIME	Global RF importances; SHAP unused, LIME absent. Not as specified
SOC dashboard	Alerts, severity, actions, analytics, reports	Six views, feedback buttons, CSV export. Implemented
Evaluation	Benchmark datasets	Metric code only; no dataset or results. Unvalidated

VI. EXPERIMENTAL SETUP AND EVALUATION PROTOCOL

A. Current status

No experimental evaluation has been performed on the audited artifact. The archive contains no model files, no benchmark dataset and no recorded metrics, and the only data are the six simulated records described above. All items in this section are therefore a proposed protocol, and no number in it is a result.

B. Proposed protocol

Table III specifies a protocol designed to avoid the pitfalls noted in [3] and [7]. Each experiment would be repeated over multiple random seeds and reported as mean and standard deviation; differences between configurations would be assessed with an appropriate paired test over those repeats before any claim of improvement.

Table III
Proposed Evaluation Protocol (Not Yet Executed)

Element	Specification
Datasets	CICIDS2017 [6], using the corrected release described in [7]; UNSW-NB15 [5]. Dataset versions and checksums recorded.
Splits	Day-based or time-ordered splits in addition to the existing random stratified split; leave-one-attack-family-out to simulate unseen attacks.
Baselines	RF alone; IF alone; RF+IF with fixed threshold (no DTCL); RF+IF with DTCL; at least one non-forest baseline.
Detection metrics	Per-class precision, recall, F1; macro-F1; AUROC of anomaly evidence on held-out attack families; alerts per 10,000 records.
DTCL study	Simulated analyst using ground-truth labels, then with label noise; drift scenarios; threshold trajectory; alert volume and missed detections versus fixed τ.
Live-agent check	Separate run with flow-aggregated features to test the per-packet versus flow-level mismatch (Section VIII).
Explanations	Per-instance SHAP versus global importances; stability across seeds; analyst usefulness study if feasible.
Cost	Records per second and end-to-end latency on stated hardware.

VII. RESULTS AND DISCUSSION

A. Measured results

There are none to report. The paper makes no claim of accuracy, false-positive reduction, zero-day detection or latency.

B. Design-level analysis of the implementation

Reading the code against the objectives yields five observations that follow from the equations themselves and do not depend on data.

1) Score range. Since $c \leq 1$ and $a \leq 1$, Eq. (2) gives $S \leq 0.85$. The Critical band ($S \geq 0.90$) is therefore unreachable in the present build and the highest attainable severity is High. The weights or bands need recalibration.

2) Reach of the threshold. Under Eq. (3) any record the Random Forest labels as an attack raises an alert regardless of τ, so τ only influences records labelled BENIGN or UNKNOWN. Repeated false-positive verdicts on attack-labelled alerts cannot reduce their recurrence. This materially limits how far DTCL can support objective O2 in its current form.

3) Adaptation speed and granularity. From $\tau_0 = 0.70$, ten consecutive true-positive verdicts reach the floor of 0.50 and thirteen consecutive FP/BENIGN verdicts reach the cap of 0.95. The threshold is global, although the history table already records the alert category, so per-category thresholds are a natural extension.

4) Metric semantics. The dashboard's "false positive rate" is the share of rejected alerts among reviewed alerts, that is, one minus precision on the reviewed subset, not a network-level false-positive rate. The "detection rate" is the weighted recall of the offline hold-out, which equals accuracy under weighted averaging. Both should be relabelled before any reporting.

5) Missing-model behaviour.

The MODEL_NOT_TRAINED label avoids fabricated metrics, but because it is not in {BENIGN, UNKNOWN}, Eq. (3) raises an alert for every record when no model is present. A deployment without models would flood the queue.

C. Discussion

These findings suggest that the provenance discipline of SentinelAI+ is its most defensible property, and that DTCL as implemented is better described as a logged sensitivity control for score-triggered alerts than as a general false-positive reducer. Whether even that control helps is an empirical question that the protocol in Table III is designed to answer.

Table III. Detection performance of SentinelAI+

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	False Positive Rate (%)
Random Forest (baseline)	[99.39 ± 0.26]	[99.93 ± 0.09]	[99.20 ± 0.42]	[99.56 ± 0.19]	[0.16 ± 0.21]
Isolation Forest (baseline)	[45.42 ± 0.54]	[100.00 ± 0.00]	[22.03 ± 0.78]	[36.10 ± 1.04]	[0.00 ± 0.00]
SentinelAI+ Hybrid Engine (proposed)	[99.42 ± 0.24]	[99.42 ± 0.24]	[99.42 ± 0.24]	[99.42 ± 0.24]	[99.42 ± 0.24]

VIII. LIMITATIONS AND THREATS TO VALIDITY

No empirical evidence. The central limitation is the absence of trained models, benchmark data and results (Section VI-A). Objectives O1–O4 remain unvalidated.

Feature mismatch. The live agent emits per-packet records with placeholder flow fields, whereas the training script maps flow-level CICIDS/UNSW columns. Models trained on flow statistics would receive out-of-distribution inputs from the agent. In addition, where several source columns map to one feature (for example forward and backward packet counts), only the first match is used. Port numbers are included as features, which risks dataset-specific shortcut learning of the kind discussed in [3] and [7].

Anomaly model. The Isolation Forest is fitted on all training rows, including attacks, with automatic contamination. It is thus not a benign-only novelty detector, and its ability to flag unseen attacks is untested. The random stratified split used by the trainer does not test unseen attack families or temporal drift [3], [8].

Explainability and intelligence. Explanations are the same global importances for every alert and are not per-instance; SHAP and LIME are not used. Enrichment relies on a single third-party provider, rarely yields reputation, and transmits source IP addresses off-site.

Security and scale. Default placeholders for the API key and administrator password, permissive default CORS and Socket.IO origins, debug mode in the entry point, SQLite storage and one synchronous HTTP request per packet make the prototype unsuitable for production. Feedback-driven calibration also introduces a new attack surface: anyone able to submit verdicts can move τ, and feedback poisoning is not modelled.

Statistical caveats. Weighted averaging can hide minority-class behaviour, and collapsing rare labels into "Other" changes the task. Any future results must report per-class figures and variability.

IX. CONCLUSION AND FUTURE WORK

SentinelAI+ is a working prototype that couples a Random Forest, an Isolation Forest, a transparent threat score, IP enrichment and an analyst-driven bounded threshold update inside an auditable, provenance-preserving service. This paper documented its architecture, audited the implementation, identified design-level limits (an unreachable Critical band, a threshold that does not govern class-triggered alerts, and per-packet versus flow-level feature mismatch), and specified an evaluation protocol. It does not demonstrate improved detection or fewer false positives.

Priority future work is to (i) train and evaluate the models under the protocol in Table III; (ii) add flow aggregation to the agent; (iii) implement per-instance SHAP explanations and compare them with the current importances; (iv) make the anomaly model benign-trained and test it on held-out attack families; (v) recalibrate score weights and severity bands; (vi) extend DTCL to per-category thresholds that also affect class-triggered alerts; and (vii) review literature on analyst-in-the-loop and active-learning IDS to position the contribution precisely.

REFERENCES

[1] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *Proc. IEEE Symposium on Security and Privacy*, 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[2] L. Y. Por, Z. Dai, S. J. Leem, Y. Chen, J. Yang, F. Binbeshr, K. Y. Phan, and C. S. Ku, "A Systematic Literature Review on AI-Based Methods and Challenges in Detecting Zero-Day Attacks," *IEEE Access*, vol. 12, pp. 144150–144163, 2024, doi: 10.1109/ACCESS.2024.3455410.

[3] D. Arp *et al*., "Dos and Don'ts of Machine Learning in Computer Security," in *Proc. 31st USENIX Security Symposium*, 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[4] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[5] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion

Detection Systems (UNSW-NB15 Network Data Set)," in *Proc. Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[6] Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proc. 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[7] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study," in *Proc. IEEE Security and Privacy Workshops*, 2021, pp. 7–12, doi: 10.1109/SPW53761.2021.00009.

[8] R. S. Miani, G. D. G. Bernardo, G. Weigert Cassales, H. Senger, and E. R. de Faria, "A Survey of Data Stream-Based Intrusion Detection Systems," *IEEE Access*, vol. 13, pp. 72953–72983, 2025, doi: 10.1109/ACCESS.2025.3561105.

[9] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4765–4774.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[11] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[12] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proc. 8th IEEE International Conference on Data Mining*, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[13] Aldweesh, A. Derhab, and A. Z. Emam, "Deep Learning Approaches for Anomaly-Based Intrusion Detection Systems: A Survey, Taxonomy, and Open Issues," *Knowledge-Based Systems*, vol. 189, Art. no. 105124, 2020, doi: 10.1016/j.knosys.2019.105124.

[14] Z. Yang *et al*., "A Systematic Literature Review of Methods and Datasets for Anomaly-Based Network Intrusion Detection," *Computers & Security*, vol. 116, Art. no. 102675, 2022, doi: 10.1016/j.cose.2022.102675.

[15] Z. Ahmad *et al*., "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, Art. no. e4150, 2021, doi: 10.1002/ett.4150.

[16] M. S. Habeeb and T. R. Babu, "Network Intrusion Detection System: A Survey on Artificial Intelligence-Based Techniques," *Expert Systems*, vol. 39, no. 9, Art. no. e13066, 2022, doi: 10.1111/exsy.13066.

[17] S. Gamage and J. Samarabandu, "Deep Learning Methods in Network Intrusion Detection: A Survey and an Objective Comparison," *Journal of Network and Computer Applications*, vol. 169, Art. no. 102767, 2020, doi: 10.1016/j.jnca.2020.102767.

[18] "Explainable AI for Intrusion Detection Systems: LIME and SHAP Applicability on Multi-Layer Perceptron," *IEEE Access*, vol. 12, pp. 30164–30175, 2024, doi: 10.1109/ACCESS.2024.3368377.

[19] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A Survey of Network-Based Intrusion Detection Data Sets," *Computers & Security*, vol. 86, pp. 147–167, 2019, doi: 10.1016/j.cose.2019.06.005.

[20] P. Goldschmidt and D. Chudá, "Network Intrusion Datasets: A Survey, Limitations, and Recommendations," *Computers & Security*, 2025, doi: 10.1016/j.cose.2025.104510.

[21] "Introducing a Comprehensive, Continuous, and Collaborative Survey of Intrusion Detection Datasets," in *Proc. 17th Cyber Security Experimentation and Test Workshop (CSET '24)*, 2024, pp. 34–40, doi: 10.1145/3675741.3675754.

[22] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A Survey on Concept Drift Adaptation," *ACM Computing Surveys*,

vol. 46, no. 4, Art. no. 44, 2014, doi: 10.1145/2523813.

[23] K. A. Scarfone and P. M. Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, NIST Special Publication 800-94. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2007, doi: 10.6028/NIST.SP.800-94.

[24] L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[25] Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges," *Cybersecurity*, vol. 2, Art. no. 20, 2019, doi: 10.1186/s42400-019-0038-7.

[26] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges," *Computers & Security*, vol. 28, nos. 1–2, pp. 18–28, 2009, doi: 10.1016/j.cose.2008.08.003.

[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, Art. no. 15, 2009, doi: 10.1145/1541880.1541882.

[28] Khraisat and A. Alazab, "A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy, Attacks, Public Datasets and Challenges," *Cybersecurity*, vol. 4, Art. no. 18, 2021, doi: 10.1186/s42400-021-00077-7.

[29] M. A. Shyaa, N. F. Ibrahim, Z. Zainol, R. Abdullah, M. Anbar, and L. Alzubaidi, "Evolving Cybersecurity Frontiers: A Comprehensive Survey on Concept Drift and Feature Dynamics Aware Machine and Deep Learning in Intrusion Detection Systems," *Engineering Applications of Artificial Intelligence*, vol. 137, Art. no. 109143, 2024, doi: 10.1016/j.engappai.2024.109143.

[30] S. Neupane *et al.*, "Explainable Intrusion Detection Systems (X-IDS): A Survey of Current Methods, Challenges, and Opportunities," *IEEE Access*, vol. 10, pp. 112392–112415, 2022, doi: 10.1109/ACCESS.2022.3216617.

[31] O. Abdel Wahab, "Intrusion Detection in the IoT Under Data and Concept Drifts: Online Deep Learning Approach," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19706–19716, 2022, doi: 10.1109/JIOT.2022.3167005.

[32] F. Hinder, V. Vaquet, and B. Hammer, "One or Two Things We Know About Concept Drift—A Survey on Monitoring in Evolving Environments. Part A: Detecting Concept Drift," *Frontiers in Artificial Intelligence*, vol. 7, Art. no. 1330257, 2024, doi: 10.3389/frai.2024.1330257.

[33] F. Hinder, V. Vaquet, and B. Hammer, "One or Two Things We Know About Concept Drift—A Survey on Monitoring in Evolving Environments. Part B: Locating and Explaining Concept Drift," *Frontiers in Artificial Intelligence*, vol. 7, Art. no. 1330258, 2024, doi: 10.3389/frai.2024.1330258.