IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

Explainable Network Intrusion Detection Using Two-Stage Tree Ensembles, Novelty-Aware Thresholding and Risk-Based Alert Prioritisation

Shirin Anna Viju[1*], S. Anitha Jebamani[2], Swetha A[3], A. Sujithra Raj[4], Saranya S[5]

Assistant Professor (O.G), Department of CSE (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[2]

Student, Department of CSE (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[1,3,4,5]

*Corresponding Author

Abstract: Machine-learning intrusion detectors are frequently evaluated in ways hiding their weakness on attack types absent from training. This paper presents an explainable network intrusion detection system evaluated on the official NSL-KDD split, whose test set contains 17 attack types unseen in training. A two-stage tree-ensemble model detects and categorises attacks, an Isolation Forest flags anomalous flows, and each alert carries a calibrated probability, a SHAP explanation and a rule-based severity score; the pipeline is served through FastAPI and Streamlit. Leave-attack-type-out validation on training data alone selected a LightGBM and Extra Trees ensemble with a 0.02 decision threshold. On the held-out test set it reaches 91.9% accuracy, 93.4% recall and 92.7% unseen-type recall, against 77.9%, 63.3% and 31.1% for the original LightGBM, at a 10.2% false-positive rate instead of 2.9%. The gain comes mainly from the threshold (ROC-AUC 0.963 vs. 0.964); calibration lowers the expected calibration error from 0.214 to 0.050.

Keywords: Intrusion detection, LightGBM, Extra Trees, Isolation Forest, unknown attacks, threshold selection, probability calibration, explainable AI, SHAP, alert prioritisation, NSL-KDD

I. INTRODUCTION

Organisations depend on networks for nearly every service they deliver, and attacks such as distributed denial of service (DDoS), port scanning, credential brute-forcing, botnet activity and web exploitation continue to grow in volume and variety. Network intrusion detection systems (NIDS) are a central defensive control. Signature-based NIDS such as Snort [1] match traffic against hand-written rules; they are precise for known attacks but miss new or modified ones. Anomaly-based detection, introduced by Denning [2], models normal behaviour and flags deviations, but has historically suffered from high false-alarm rates [3], [4].

Machine-learning (ML) detectors learn attack behaviour from labelled flows and often report high accuracy on benchmark data sets [5]. Three problems limit their use in a security operations team. First, evaluations frequently use random splits in which every attack type appears in both training and test data, which hides poor generalisation to unseen attacks [6], [7]. Second, raw classifier scores are rarely calibrated, so a reported confidence does not match an observed error rate [8], [9]. Third, alerts are usually raised with equal urgency and without explanation, which contributes to alert fatigue [3].

This paper addresses these problems with an integrated pipeline that combines supervised and unsupervised detection with calibration, explanation and prioritisation, implemented as a working application programming interface (API) and dashboard. It is evaluated on the official NSL-KDD split [10], whose test set deliberately contains attack types absent from training. An earlier version of the system used a default decision threshold and recognised fewer than one in three unseen-type attacks. This revision shows that a validation protocol which imitates unseen attacks, built only from training data, selects a decision threshold and model that generalise much better, and it reports the cost of that choice in false alarms.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

II. RELATED WORK

Rule-based systems such as Snort [1] remain the operational baseline. Denning's statistical model [2] established anomaly detection, and surveys document both its ability to flag new behaviour and its false-alarm problem [4]. Sommer and Paxson [3] argued that ML-based NIDS often fail in practice because of the high cost of errors and the difficulty of evaluation; Arp et al. [6] catalogued common pitfalls, including sampling bias and data snooping.

Lee et al. [11] framed intrusion detection as a data-mining problem, which led to the KDD Cup 99 data; Tavallaee et al. [10] removed its redundant records and released NSL-KDD. Buczak and Guven [5] surveyed decision trees, ensembles, clustering and neural methods. Gradient-boosted trees [12], in particular LightGBM [13], and randomised tree ensembles such as Random Forests [14] and Extra Trees [15] are strong learners for tabular flow features and support exact tree-based SHapley Additive exPlanations (SHAP) [16].

For unknown attacks, Isolation Forest [17] isolates anomalies with random partitions, and autoencoder ensembles such as Kitsune [18] learn normal traffic online. Sarhan et al. [7] studied zero-day detection as a zero-shot problem by holding attack classes out of training and found that detection rates vary widely across held-out classes; the validation protocol in this paper follows the same idea. Platt scaling [19] and isotonic regression [20] map scores to probabilities; the synthetic minority over-sampling technique (SMOTE) [21] and combined re-sampling [22] address class imbalance; SHAP [23] and local interpretable model-agnostic explanations (LIME) [24] explain individual predictions; and MITRE ATT&CK [25] catalogues adversary tactics and techniques. Newer corpora such as CIC-IDS2017 [26] and UNSW-NB15 [27] contain more modern traffic, although CIC-IDS2017 has documented labelling issues [28], and Ring et al. [29] survey such data sets. Table I compares the approach families qualitatively; reported accuracies are not compared because they are not comparable across data sets and protocols [6].

TABLE I QUALITATIVE COMPARISON OF NIDS APPROACH FAMILIES

Approach	Known attacks	Unseen attacks	Calibrated confidence	Per-alert explanation
Signature IDS [1]	Yes	No	No	Rule identifier
Statistical anomaly detection [2], [4]	Partial	Yes	No	No
Supervised ML [5]	Yes	Weak [7]	Rarely	Rarely
Isolation Forest [17]	Partial	Yes	No	No
Autoencoder NIDS [18]	Partial	Yes	No	No
This work	Yes	Improved, not solved	Yes (isotonic)	Yes (SHAP)

III. PROBLEM DEFINITION AND CONTRIBUTIONS

A. Problem Definition

Each flow is described by 41 NSL-KDD features (38 numeric and 3 categorical), a binary label $y \in \{0, 1\}$ (normal, malicious) and, for malicious flows, a category $c \in \{\text{DoS, Probe, R2L, U2R}\}$, i.e. denial of service, probing, remote-to-local and user-to-root attacks. The training set contains 22 attack types; the test set contains 37, of which 20 also occur in training and 17 are unseen. For every flow the system outputs a binary decision, a category when the flow is judged malicious, an anomaly flag, a calibrated probability of attack, a feature-level explanation, and a severity level with a priority score. Known-attack classification means detecting and naming attacks of types seen in training. Unknown-attack flagging means raising any alert for an attack of an unseen type; naming the type is not required, and no zero-day detection capability is claimed.

B. Contributions

(1) An integrated pipeline, implemented as a tested API and dashboard, of two-stage tree-ensemble classification, a parallel Isolation Forest, calibration, per-alert SHAP explanations and a documented severity and priority rule set. (2) A leave-attack-type-out (LOATO) validation protocol, built only from the training data, used to select the model, the decision threshold, the anomaly threshold and the calibrator without touching the test set. (3) A one-shot test evaluation that compares the original and the improved pipeline with bootstrap confidence intervals, separates seen and unseen attack types, and reports the end-to-end hierarchical accuracy. (4) A corrected severity rule set and an explicit account of the remaining weaknesses.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

IV. DATA SET AND PREPROCESSING

A. NSL-KDD and Attack Taxonomy

NSL-KDD [10] provides KDDTrain+ (125,973 flows) and KDDTest+ (22,544 flows). The files used here contain 41 features and an attack-name label (no difficulty-level column). Each attack name was mapped to one of the four categories. The counts in Table II were recomputed directly from the raw files. They depend on how two test-only attack names are mapped: this work maps httptunnel to R2L and worm to DoS, while another common mapping assigns them to U2R and R2L, which gives 7,458 DoS, 2,754 R2L and 200 U2R test flows. All models in this paper use the first mapping. KDDTest+ contains 38 distinct labels, the normal class and 37 attack types, of which 17 attack types are absent from the training set; these unseen types account for 3,750 of the 12,833 malicious test flows. The R2L distribution also shifts: guess_passwd has 53 training flows but 1,231 test flows, and warezmaster has 20 and 944.

TABLE II NSL-KDD CLASS DISTRIBUTION (RECOMPUTED FROM THE RAW FILES)

Category	KDDTrain+	KDDTest+	Test flows of unseen types
Normal	67,343	9,711	–
DoS	45,927	7,460	1,719
Probe	11,656	2,421	1,315
R2L	995	2,885	686
U2R	52	67	30
Total	125,973	22,544	3,750

B. Preprocessing

A single scikit-learn [30] transformer is fitted on KDDTrain+ only and reused unchanged at inference. Numeric features are standardised as in (1), where the mean and standard deviation come from the training data, and protocol_type, service and flag are one-hot encoded, with unseen categories mapped to all-zero vectors. This yields 122 model inputs (38 numeric and 84 binary). No exact duplicates were found in KDDTrain+.

$$z_j = \frac{x_j - \mu_j}{\sigma_j} \qquad (1)$$

C. Feature Ranking

Features were ranked by mutual information and by Random Forest impurity importance, and the top 40 by combined rank were stored; 38 of the 40 appear in both individual top-40 lists. All models, original and improved, use all 122 inputs. On LOATO validation the top-40 subset gave the same accuracy as the full set (Table VII), so the ranking is reported as analysis rather than as a selection step.

V. METHODOLOGY

A. Two-Stage Detector

Gradient boosting [12] builds an additive model of regression trees by fitting each new tree to the gradients of the regularised objective (2), where w are class weights, T is the number of leaves and ω the leaf weights. LightGBM [13] speeds this up with histogram-based split finding and leaf-wise growth. Extra Trees [15] average many fully grown trees whose split thresholds are drawn at random. Stage 1 outputs a probability of attack p(**x**); in the improved system this is the average of a LightGBM model and an Extra Trees model, and a flow is declared malicious when p(**x**) ≥ τ, as in (3). Stage 2 is a LightGBM classifier trained only on the 58,630 malicious training flows; it assigns the category with the softmax (4) and runs only for flows that stage 1 declares malicious, so stage-1 errors cannot be corrected later.

$$\mathcal{L} = \sum_i w_{k(i)}\, \ell\big(y_i, F(\boldsymbol{x}_i)\big) + \sum_m \left(\gamma T_m + \lambda/2 \parallel \boldsymbol{\omega}_m \parallel^2\right) \qquad (2)$$

$$\hat{y} = \begin{cases} 1, & 1/2\left(p_{LGBM}(\boldsymbol{x}) + p_{ET}(\boldsymbol{x})\right) \geq \tau \\ 0, & \text{otherwise} \end{cases} \qquad (3)$$

$$P(c \mid \boldsymbol{x}) = \frac{exp F_c(\boldsymbol{x})}{\sum_{c' \in C} exp F_{c'}(\boldsymbol{x})} \qquad (4)$$

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

B. Class Imbalance

Category-level imbalance is severe (about 1,295 normal flows per U2R flow). Six strategies were compared on a stratified 20% split of KDDTrain+ (Table X). Class weighting with $w_k = N / (K\, n_k)$, where N is the number of samples, K the number of classes and n_k the size of class k, is used in both LightGBM stages and in the Extra Trees model.

C. Isolation Forest

An Isolation Forest [17] with 200 trees is fitted on the 122-dimensional training matrix (normal and attack flows). Its anomaly score for a flow with expected path length $E[h(\mathbf{x})]$ over trees built on sub-samples of size $\psi = 256$ is given by (5), where H is the harmonic number. The original system flagged a flow when its score exceeded the 53.5th percentile of training scores (contamination 0.4654). The improved system flags only flows above the 99.5th percentile, the setting chosen on validation (Section VIII-D).

$$s(\boldsymbol{x}, \psi) = 2^{-E[h(\boldsymbol{x})]/c(\psi)}, \quad c(\psi) = 2H(\psi - 1) - \frac{2(\psi-1)}{\psi} \qquad (5)$$

D. Leave-Attack-Type-Out Validation and Threshold Selection

Random validation splits contain no unseen attack types, so they cannot show how a model behaves on novel traffic. We therefore built five LOATO folds from KDDTrain+ only. The 21 attack types other than neptune (70% of training attacks, always kept in training) were assigned round-robin by size to five groups. In fold k, all flows of group k are removed from training and used as "unseen" validation flows, and a random 20% of the remaining flows serve as "seen" and normal validation flows. Validation accuracy is weighted so that normal, seen-type and unseen-type flows contribute 43%, 40% and 17%, the proportions of KDDTest+, as in (6), where v_i is the weight of flow i. Twenty model configurations (Table VII) and 24 thresholds from 0.005 to 0.95 were compared by mean weighted accuracy over the five folds. Two winners were fixed before the test set was used: the best model overall and, to preserve the original LightGBM design, the best LightGBM-only configuration.

$$Acc_w = \frac{\sum_i v_i \mathbb{1}[\hat{y}_i = y_i]}{\sum_i v_i}, \quad \sum_{i \in \text{normal}} v_i : \sum_{i \in \text{seen}} v_i : \sum_{i \in \text{unseen}} v_i = 0.43 : 0.40 : 0.17 \qquad (6)$$

E. Probability Calibration

The original system took its confidence from a separate LightGBM trained on 80% of the training data and Platt-scaled on the remaining 20%, while decisions came from a different model trained on all the data. The improved system calibrates the detector that makes the decisions. Five candidates were compared: no calibration; Platt scaling (7) [19] and isotonic regression [20] fitted on in-distribution five-fold out-of-fold scores; and Platt scaling and isotonic regression fitted on the LOATO out-of-fold scores. The candidate with the lowest cross-fitted weighted log-loss on the LOATO folds was selected. Calibration is assessed by the Brier score, the log-loss and the expected calibration error (ECE) (8) over B = 10 bins [9].

$$P(y = 1 \mid f) = \frac{1}{1 + exp(Af + B)} \qquad (7)$$

$$ECE = \sum_{b=1}^{B} \frac{|S_b|}{n} |\bar{y}(S_b) - \bar{p}(S_b)| \qquad (8)$$

F. SHAP Explanations

For each alert, SHAP [23] decomposes the LightGBM member's log-odds output into additive feature contributions (9), computed exactly by TreeExplainer [16]; the five largest contributions are attached to the alert. The Extra Trees member is not explained per alert, because exact TreeSHAP over its deep trees is too slow for per-flow use.

$$F(\boldsymbol{x}) = \phi_0 + \sum_{j=1}^{122} \phi_j(\boldsymbol{x}) \qquad (9)$$

G. Severity and Priority Scoring

Severity is assigned by transparent rules kept separate from the ML prediction (Table III). For a detected flow the base level depends on the predicted category and moves up when the calibrated probability of attack $r \geq 0.85$, down when $r < 0.55$, and up again when the anomaly flag corroborates the detection. In the original rules an anomaly-only alert (classifier says normal, anomaly layer says anomalous) was escalated when the classifier was confident that the flow was normal, because its confidence was taken as $1 - r$. On the original pipeline this sent 5,661 anomaly-only alerts to High, of which only 59.2% were attacks, while the 422 left at Medium were 96.7% attacks. The revised rules use only evidence of risk: an anomaly-only alert is Medium, and High when $r \geq 0.20$. Alerts are ranked by the priority score (10), where $w(s) \in \{1, 2, 3, 4\}$ is the weight of severity level s, $e = r$ for detected flows and $e = \max(r, u)$ for anomaly-only flows, u is the

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

percentile rank of the anomaly score among training scores, and m = 1.5 when a detection is corroborated by the anomaly flag (otherwise 1). Flows that are neither detected nor flagged as anomalous receive the label Low and are stored in the incident log, but they raise no alert and are not counted in the alert queue (Table XII). The base levels and thresholds are design judgements; they are not derived from the Common Vulnerability Scoring System (CVSS) or another standard.

$$\pi = w(s) \cdot e \cdot m \quad (10)$$

TABLE III REVISED SEVERITY RULES

Case	Rule	Result
Detected flow	Base level by category: Probe Medium; DoS and R2L High; U2R Critical	Base level
Detected flow	$r \geq 0.85$ / $r < 0.55$	One level up / one level down
Detected and anomalous	Independent corroboration	One further level up (capped at Critical)
Anomaly only	$r < 0.20$ / $r \geq 0.20$	Medium / High
Neither	No alert	Low (logged; no alert raised)

H. MITRE ATT&CK Reference Mapping

Table IV relates the NSL-KDD categories to ATT&CK [25] tactics and techniques to help analysts use current threat-intelligence vocabulary. This mapping is a reference table only; it is not implemented in the software or attached to alerts, and a category-level mapping is necessarily coarse.

TABLE IV CATEGORY-LEVEL REFERENCE MAPPING TO MITRE ATT&CK (NOT IMPLEMENTED IN THE SOFTWARE)

Category	Tactic	Representative technique
DoS	Impact (TA0040)	Network Denial of Service (T1498)
Probe	Reconnaissance (TA0043); Discovery (TA0007)	Active Scanning (T1595); Network Service Discovery (T1046)
R2L	Credential Access (TA0006); Initial Access (TA0001)	Brute Force (T1110); Exploit Public-Facing Application (T1190)
U2R	Privilege Escalation (TA0004)	Exploitation for Privilege Escalation (T1068)

VI. SYSTEM ARCHITECTURE AND IMPLEMENTATION

Fig. 1 shows the architecture of the revised system and Table V its per-flow inference procedure. The revised detector, decision threshold, calibrator, Isolation Forest threshold and severity rules have been integrated into the FastAPI service. A training script rebuilds them from KDDTrain+ (including the LOATO out-of-fold scores used by the calibrator) and reproduced the offline test results exactly. The service loads the improved models by default; an environment variable (NIDS_MODEL_VERSION) switches back to the original models, and each response reports which model version produced it. The system is written in Python with LightGBM 4.7, scikit-learn 1.8, imbalanced-learn 0.14, SHAP 0.51, FastAPI 0.141, SQLAlchemy 2.0 and Streamlit 1.64. All artefacts are serialised with joblib and loaded once when the service starts.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

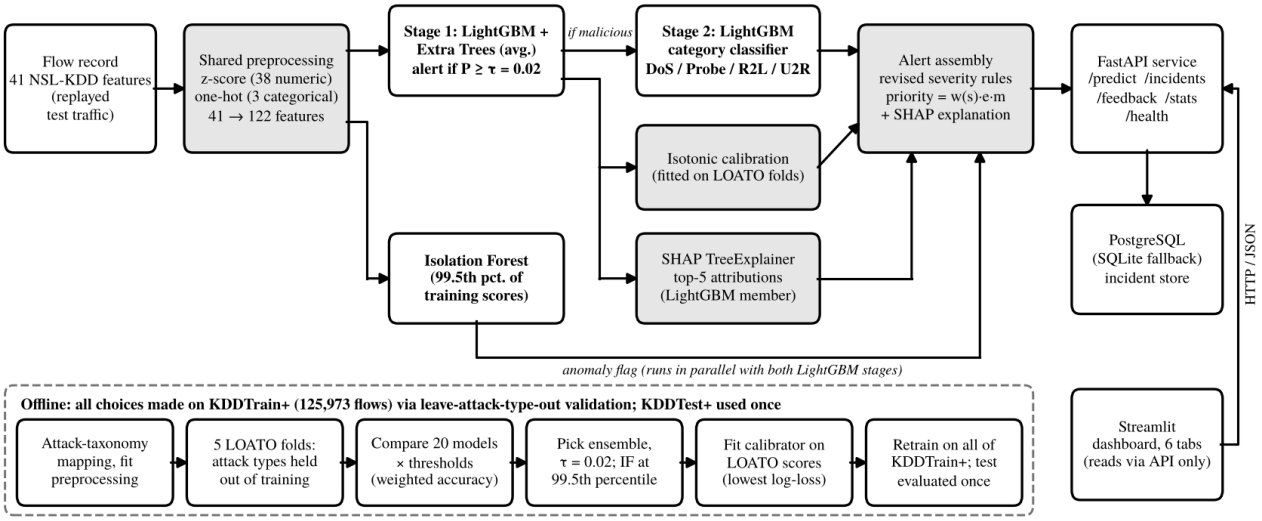


Fig. 1 System architecture. Top: online inference path for one flow. Bottom: offline selection and training. Shaded boxes are post-processing stages that do not change the stage-1 decision.

TABLE V PER-FLOW INFERENCE PROCEDURE (ALGORITHM 1)

Step	Operation
1	Validate the 41-field flow record against the request schema (out-of-range fields are rejected with HTTP 422)
2	x ← preprocessing pipeline fitted on KDDTrain+ only (122 inputs)
3	p ← average of LightGBM and Extra Trees probabilities; $\hat{y}$ ← 1 if p ≥ 0.02
4	r ← isotonic calibration of p; a ← 1 if Isolation Forest score > 99.5th training percentile
5	If $\hat{y}$ = 1: category $\hat{c}$ ← stage-2 LightGBM; otherwise $\hat{c}$ ← normal
6	E ← five largest SHAP contributions of the LightGBM member
7	Severity s ← rules of Table III; priority π ← w(s) · e · m
8	Store ($\hat{y}$, $\hat{c}$, a, r, s, π, E) as an incident and return it in the response

FastAPI exposes /predict, /incidents (filterable by severity and category, ordered by priority), /feedback (an analyst marks an incident as confirmed or rejected), /stats/overview, /stats/model-metrics and /health. All endpoints except /health require a static API key, a minimal access control rather than production authentication. Incidents are stored through SQLAlchemy in PostgreSQL; if PostgreSQL cannot be reached at start-up, the service falls back automatically to a local SQLite file. The incident log examined for this paper (1,110 replayed flows, original pipeline) was stored in the SQLite fallback, and no analyst feedback had yet been recorded. The integration of the improved components was verified in two ways: eight new automated tests check the revised severity rules, run both model versions end to end and confirm that the API serves the improved version by default (all 30 project tests pass); and, as a subsequent integration check, 500 KDDTest+ records were replayed through /predict into a fresh SQLite store after the improved models had been trained and saved. The records were drawn by simple random sampling without replacement from the 22,544 KDDTest+ flows (random seed 42, the default of the replay script) and comprise 199 normal and 301 attack flows, 89 of which belong to eight attack types absent from training. Each response was compared with the record's true label (normal versus any attack): accuracy is the share of correct decisions (460/500), recall the share of attack records flagged (281/301) and the false-positive rate (FPR) the share of normal records flagged (20/199). The service returned 92.0% accuracy, 93.4% recall and a 10.1% FPR, consistent with the offline evaluation. The replay used the already trained and fixed models; its records were not used for training, threshold selection, calibration or tuning, and its figures are a functional check of the API rather than an additional evaluation.

The Streamlit dashboard (Fig. 2) has six tabs (Overview, Live Monitoring, Attack Analytics, Model Analytics, Explainable AI and Incident History) and reads data only through the API. "Live" monitoring replays sampled KDDTest+ records through /predict at a configurable rate; the system does not capture packets. Because the dashboard loads no models of its own, it shows whichever model version the API serves; its Model Analytics tab was extended to report the active version and to compare the original and improved detectors. After the 500-record replay, the dashboard was run against the API serving the improved models and displayed the replayed incidents and the active version; this was a

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

manual check, and the dashboard has no automated tests. The screenshot in Fig. 2 is older: it was taken with the original pipeline, before the revisions described here.

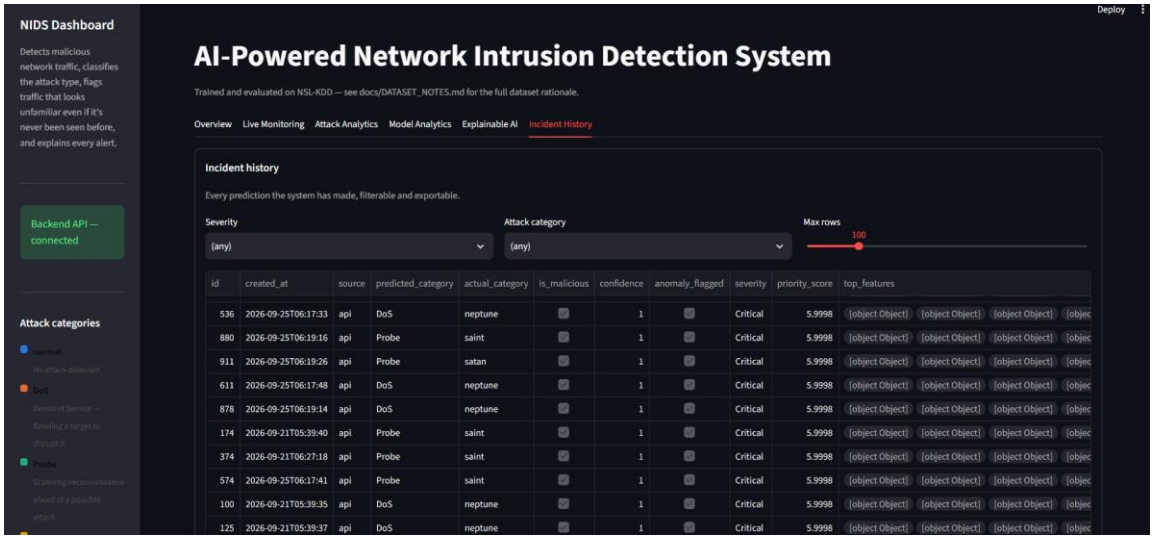


Fig. 2 Incident-history tab of the dashboard: an earlier screenshot taken with the original pipeline, before the improved models were integrated, populated by replayed KDDTest+ flows; each row stores the predicted and true category, confidence, anomaly flag, severity, priority score and SHAP features.

VII. EXPERIMENTAL SETUP

All models were trained on KDDTrain+, and every choice (model, decision threshold, anomaly threshold, calibrator and imbalance strategy) was made on splits of KDDTrain+. The primary held-out evaluation on KDDTest+ (Section VIII) was carried out once, after model selection and threshold tuning on KDDTrain+ (Table VI) had been completed. KDDTest+ records were accessed again afterwards only for the 500-record API integration replay described in Section VI, which verified the integrated API and its prediction pipeline; the replay was not used for model training, calibration, threshold selection or tuning, and its results are not part of the benchmark evaluation. The original pipeline was re-scored from its saved artefacts, and its logged metrics were reproduced exactly. Metrics are accuracy, precision, recall (detection rate), F1, macro- and weighted-F1, FPR, the area under the receiver operating characteristic curve (ROC-AUC) and the area under the precision–recall curve (PR-AUC, average precision), with recall also reported separately for seen and unseen attack types. Ninety-five per cent confidence intervals were obtained with 1,000 bootstrap resamples of the test set [31], and the improved detector was retrained with five random seeds. Latency was measured on a 2-vCPU Intel Xeon (2.1 GHz) Linux machine; the project's 30 unit, integration and API tests pass (22 original and 8 added with the integration).

TABLE VI CONFIGURATIONS COMPARED

Component	Original system	Improved system
Stage 1	LightGBM, 300 trees, 63 leaves, learning rate 0.05, class-weighted; $\tau = 0.5$	Average of the same LightGBM and Extra Trees (200 trees, class-weighted); $\tau = 0.02$
LightGBM-only variant	–	LightGBM, 300 trees, 7 leaves, class-weighted; $\tau = 0.03$
Stage 2	LightGBM, 4 classes, class-weighted	Unchanged
Isolation Forest	200 trees, flag above 53.5th training percentile	200 trees, flag above 99.5th training percentile
Calibration	Platt, separate model (80/20 split)	Isotonic on LOATO out-of-fold scores of the deployed model
Severity	Original rules	Revised rules (Table III)
Selection	Fixed by hand	LOATO validation on KDDTrain+ only

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

VIII. RESULTS AND DISCUSSION

A. Validation Search

Table VII summarises the validation search (eight of the twenty configurations are shown). At the default threshold all models behave alike: weighted accuracy is about 92–94%, the FPR is close to zero, and only 53–71% of held-out attack types are detected. Lowering the threshold raises unseen-type recall much more than it raises false alarms, and the effect is largest for the randomised ensembles (Fig. 3). The LightGBM and Extra Trees average reached the highest weighted accuracy (98.2%); the best LightGBM-only model, a shallow 7-leaf configuration, reached 95.0%. Hyper-parameter changes to LightGBM alone (leaves, sub-sampling, regularisation, top-40 features, no class weights) moved accuracy by less than one percentage point.

TABLE VII LEAVE-ATTACK-TYPE-OUT VALIDATION ON KDDTRAIN+ (MEAN OVER FIVE FOLDS, TEST-LIKE WEIGHTING)

Model	Accuracy at τ = 0.5	Unseen recall at τ = 0.5	Best τ	Accuracy at best τ	FPR at best τ	Unseen recall at best τ
LightGBM + Extra Trees (selected)	93.7%	62.2%	0.02	98.2% ± 1.2	1.26%	92.4%
Extra Trees	92.2%	53.4%	0.03	98.1% ± 1.2	1.60%	92.8%
Random Forest	93.3%	60.1%	0.04	97.7% ± 1.3	1.59%	90.4%
LightGBM, 7 leaves (selected LightGBM)	93.6%	62.4%	0.03	95.0% ± 2.8	2.97%	77.5%
LightGBM, 31 leaves, sub-sampled	93.8%	63.0%	0.005	94.8% ± 4.7	0.81%	70.9%
Logistic regression	93.4%	70.7%	0.2	94.5% ± 2.4	3.83%	79.1%
LightGBM, top-40 features	93.8%	63.1%	0.005	94.4% ± 5.4	1.07%	69.3%
Original LightGBM	93.7%	62.2%	0.005	94.1% ± 5.3	0.16%	65.1%

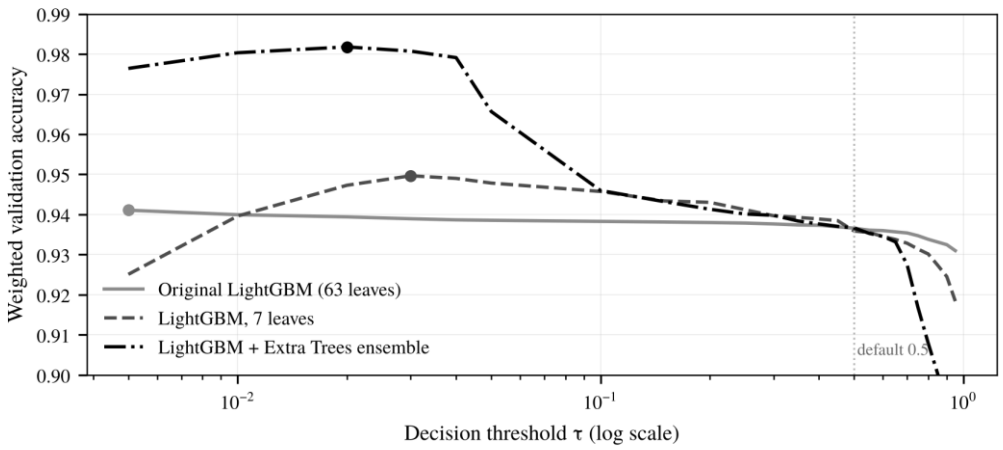


Fig. 3 Weighted LOATO validation accuracy versus decision threshold; dots mark the selected thresholds. No test data were used.

B. Original Versus Improved Detection on KDDTest+

Table VIII compares the original and improved detectors on KDDTest+ (Fig. 4). The improved ensemble raises accuracy from 77.9% to 91.9% and recall from 63.3% to 93.4%. Recall on unseen types rises from 31.1% to 92.7%, and on seen types from 76.6% to 93.8%. The cost is a higher FPR, 2.9% to 10.2% (993 instead of 280 false alarms out of 9,711 normal flows), and precision falls from 96.7% to 92.4%. The LightGBM-only variant gives a milder trade-off: 89.7% accuracy and 85.3% unseen-type recall at a 5.0% FPR. Across five training seeds the ensemble's accuracy was 91.8% ± 0.55 and its FPR 10.3% ± 0.08 (mean ± standard deviation, in percentage points).

The source of the gain is the threshold, not better ranking. The ROC-AUC barely changes (0.963 vs. 0.964), and at τ = 0.5 the ensemble reaches only 78.2% accuracy (last column of Table VIII). Many unseen-type and shifted attacks receive low but non-negligible scores; the default threshold treats them as normal, while LOATO validation reveals this

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

behaviour and selects a lower threshold (Fig. 5). Because the threshold was chosen without the test set, the improvement is a genuine generalisation result. It is still specific to this benchmark and to an operating point that accepts about 10% false alarms.

TABLE VIII STAGE-1 DETECTION ON KDDTEST+ (22,544 FLOWS): ORIGINAL VERSUS IMPROVED

Metric	Original (τ = 0.5)	Improved LightGBM (τ = 0.03)	Improved ensemble (τ = 0.02)	Ensemble at τ = 0.5
Accuracy	77.9%	89.7%	91.9%	78.2%
Precision	96.7%	95.8%	92.4%	96.7%
Recall	63.3%	85.7%	93.4%	63.8%
F1	0.765	0.905	0.929	0.769
Macro-F1	0.778	0.896	0.917	0.781
Weighted-F1	0.776	0.898	0.919	0.779
False-positive rate	2.9%	5.0%	10.2%	2.9%
Recall, seen types	76.6%	85.8%	93.8%	77.2%
Recall, unseen types	31.1%	85.3%	92.7%	31.5%
ROC-AUC	0.963	0.959	0.964	0.964
PR-AUC	0.968	0.963	0.968	0.968
Accuracy, 95% CI	77.3–78.4%	89.3–90.1%	91.5–92.2%	–
FPR, 95% CI	2.5–3.2%	4.5–5.4%	9.6–10.8%	–
Alerts raised	8,402	11,476	12,985	8,468

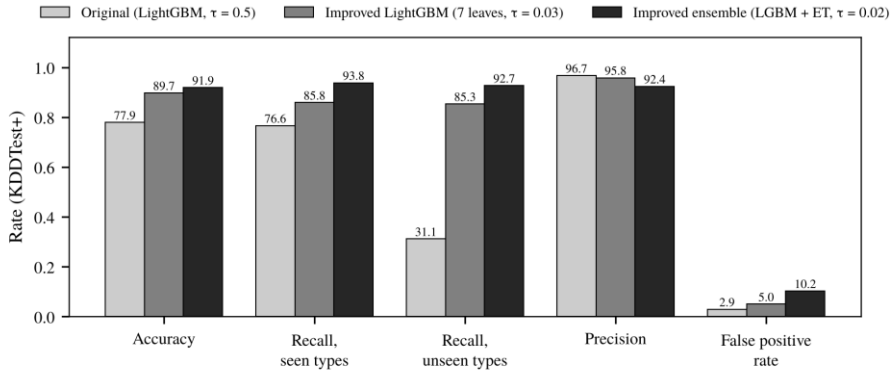


Fig. 4 Accuracy, recall on seen and unseen attack types, precision and FPR of the original and improved stage-1 detectors on KDDTest+ (values in %).

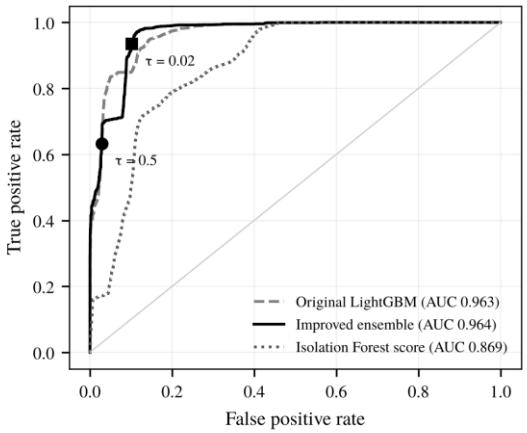


Fig. 5 ROC curves on KDDTest+. Markers show the operating points of the original (τ = 0.5) and improved (τ = 0.02) detectors.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

C. Per-Category Detection and End-to-End Hierarchy

Table IX and Fig. 6 break the results down by category. Stage-1 recall for R2L rises from 6.3% to 74.0% and for U2R from 19.4% to 82.1%. When every malicious flow reaches stage 2 (oracle setting), the category classifier reaches a macro-F1 of 0.704, with category accuracy 95.6% on seen types and 51.6% on unseen types. Behind the real stage-1 decisions, end-to-end five-class accuracy rises from 76.3% to 83.6% and macro-F1 from 0.545 to 0.716. R2L remains the weakest class (end-to-end recall 48.7%), and Probe precision is low (0.556) because stage 2 labels many DoS and R2L flows as Probe.

TABLE IX PER-CATEGORY RECALL AND END-TO-END FIVE-CLASS RESULTS ON KDDTEST+

Category	Stage-1 recall, original	Stage-1 recall, improved	End-to-end recall, original	End-to-end recall, improved	End-to-end F1, improved
DoS	83.9%	98.9%	81.5%	86.3%	0.870
Probe	68.9%	100.0%	62.1%	93.2%	0.697
R2L	6.3%	74.0%	6.1%	48.7%	0.643
U2R	19.4%	82.1%	14.9%	55.2%	0.465
Normal (specificity)	97.1%	89.8%	97.1%	89.8%	0.905
End-to-end accuracy / macro-F1	–	–	76.3% / 0.545	83.6% / 0.716	–

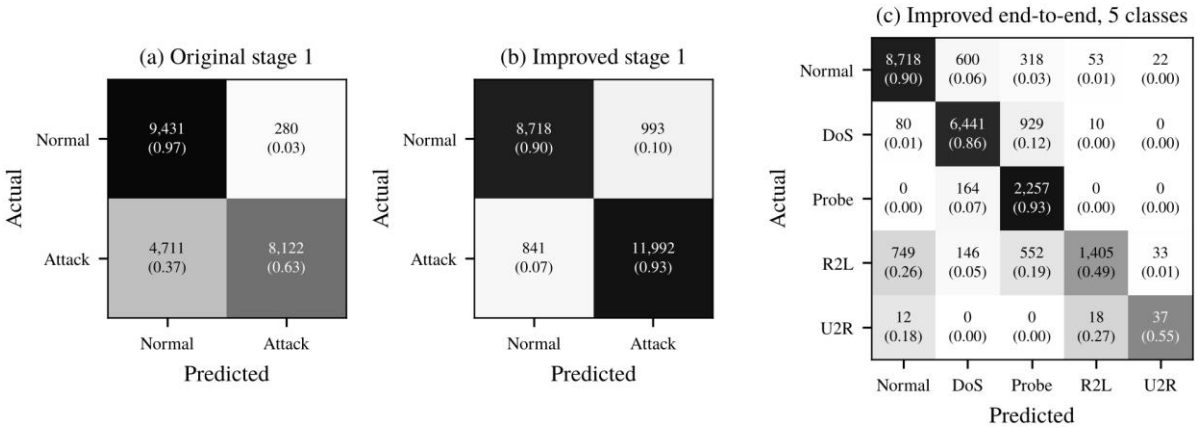


Fig. 6 Confusion matrices on KDDTest+ (counts and row-normalised rates): (a) original stage 1; (b) improved stage 1; (c) improved end-to-end five-class result.

D. Contribution of the Anomaly Layer

With the original threshold, the Isolation Forest flagged 26.8% of normal flows, 77.7% of seen-type attacks and 96.8% of unseen-type attacks, and it flagged 2,465 of the 2,583 unseen-type attacks that the original classifier missed (95.4%). Raising an alert when either signal fired lifted recall to 92.6% but also the FPR to 26.8%; on LOATO validation this union had a 41.2% FPR. Once the stage-1 threshold is chosen on validation, the anomaly layer adds almost nothing: validation preferred the strictest setting (99.5th percentile), at which it flags 0.04% of normal and 2.3% of unseen-type test flows and changes the alert count by only two (12,985 vs. 12,987): of the 119 test flows it flags, 117 are already detected by stage 1, and the two it adds are both normal flows (false alarms). The anomaly layer is therefore retained as a corroborating signal for severity, not as a detector. Its score alone reaches a ROC-AUC of 0.869.

E. Class-Imbalance Ablation

Table X shows that, without imbalance handling, a five-class LightGBM never predicts R2L or U2R on a stratified 20% validation split of KDDTrain+. Class weighting, over-sampling, SMOTE [21] and SMOTE with edited nearest neighbours (SMOTE + ENN) [22] all raise macro-F1 above 0.94 at an FPR of at most 0.0014. Class weighting was adopted; random over-sampling scored marginally higher (0.967 vs. 0.962). U2R recall rests on about ten validation flows and is indicative only.

F. Probability Calibration

Table XI and Fig. 7 compare the calibrators. Calibrators fitted on in-distribution data do not help under the test-set shift; the original separate-model Platt scaling had an ECE of 0.214, and the original confidence came from a different model

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

from the one making decisions (the two agreed on 99.4% of test flows). Fitting the calibrator on LOATO scores, where held-out attack types imitate the shift, gives far better probabilities: isotonic regression, selected by validation log-loss, lowers the test ECE to 0.050 and the Brier score to 0.063. The calibrated probability is still imperfect in the 0.6–0.9 range (Fig. 7).

TABLE X CLASS-IMBALANCE ABLATION ON A STRATIFIED 20% VALIDATION SPLIT OF KDDTRAIN+ (FIVE-CLASS LIGHTGBM)

Strategy	Macro-F1	FPR	Recall, Probe	Recall, R2L	Recall, U2R
None	0.495	0.0419	0.531	0.000	0.000
Class weighting (deployed)	0.962	0.0007	0.998	0.975	1.000
Random under-sampling	0.657	0.0977	0.985	0.945	1.000
Random over-sampling	0.967	0.0007	0.998	0.980	0.900
SMOTE	0.942	0.0007	0.997	0.985	0.800
SMOTE + ENN	0.948	0.0014	0.998	0.990	1.000

TABLE XI CALIBRATION OF THE STAGE-1 PROBABILITY ON KDDTEST+

Method	Brier score	Log-loss	ECE	Validation log-loss
Original: Platt on a separate model	0.2073	1.1933	0.2144	–
Improved detector, uncalibrated	0.1812	0.7229	0.1983	0.2564
Platt, in-distribution 5-fold	0.1830	0.8286	0.1928	0.2870
Isotonic, in-distribution 5-fold	0.1881	0.8125	0.1975	0.2787
Platt, LOATO folds	0.0602	0.2483	0.0543	0.0799
Isotonic, LOATO folds (selected)	0.0631	0.2698	0.0499	0.0733

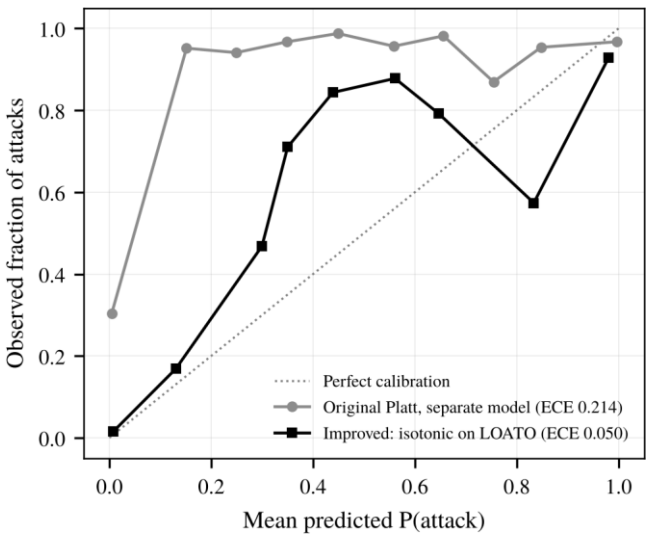


Fig. 7 Reliability diagram on KDDTest+ (10 bins) for the original and improved calibration.

G. Explainability

Fig. 8 shows the global SHAP importance of the original LightGBM detector (mean absolute SHAP value over 1,000 random test flows). src_bytes dominates (4.83), followed by count (2.05), dst_host_srv_count (1.60) and dst_bytes (1.49), so volume and short-window traffic features drive detection, consistent with flooding and scanning behaviour. For a missed Probe flow, src_bytes (+4.63) points towards an attack but is outweighed by count (−2.78) and dst_bytes (−2.24); the explanation shows the analyst why the flow was missed.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2026

DOI: 10.17148/IJIREEICE.2026.141103

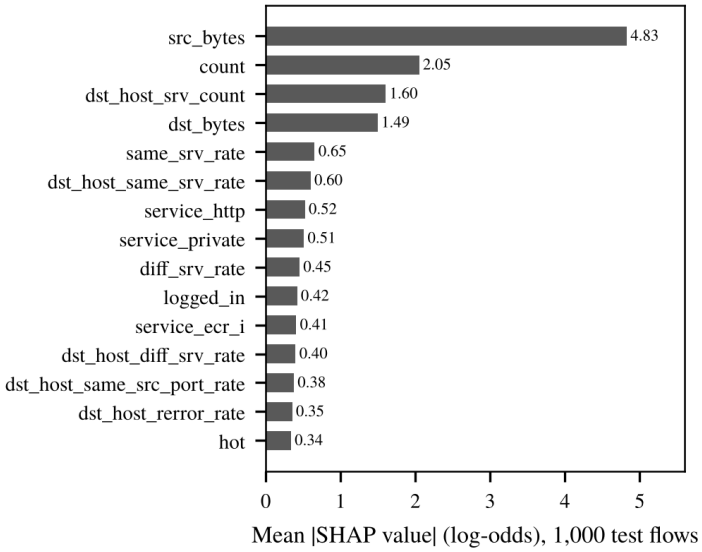


Fig. 8 Global feature importance of the original LightGBM detector (mean |SHAP| over 1,000 random KDDTest+ flows).

H. Alert Queue and Severity

Table XII and Fig. 9 summarise the alert queue. Applying the revised rules to the original pipeline does not change which flows raise alerts, but it reorders them: anomaly-only alerts at High become 96.2% attacks (previously 59.2%), and precision among the top 10,000 alerts rises from 78.5% to 92.7%. Table XII counts all alerts of the full pipeline (stage-1 detections plus anomaly-only alerts), whereas Table VIII counts stage-1 detections only; this is why the original pipeline has 14,485 alerts here but 8,402 in Table VIII. The improved pipeline raises 12,987 alerts (the 12,985 stage-1 detections plus two anomaly-only alerts; 57.6% of test flows) with a queue precision of 92.3%, compared with 14,485 alerts and 82.0% originally. Its top-1,000 precision is 99.5%, slightly below the original 100%, because a few low-threshold detections now rank highly. With the improved detector, 38.3% of all test flows are rated Critical; an operator may wish to tune the base levels to analyst capacity.

TABLE XII ALERT-QUEUE QUALITY ON KDDTEST+

Pipeline and rules	Alerts	Queue precision	Precision, top 1,000	Precision, top 5,000	Precision, top 10,000
Original pipeline, original rules	14,485	82.0%	100.0%	96.7%	78.5%
Original pipeline, revised rules	14,485	82.0%	100.0%	96.7%	92.7%
Improved pipeline, revised rules	12,987	92.3%	99.5%	99.3%	92.2%

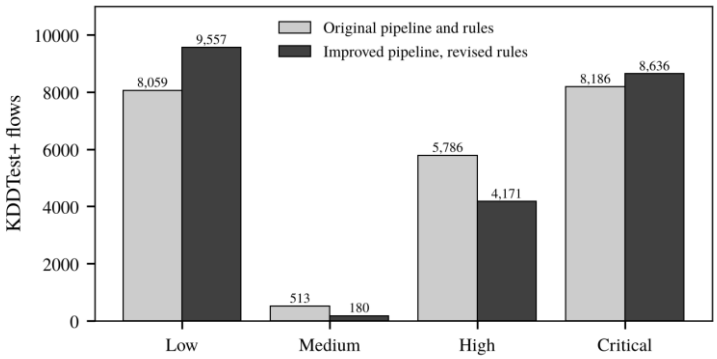


Fig. 9 Severity levels assigned to the 22,544 KDDTest+ flows before and after the revisions.

DOI: 10.17148/IJIREEICE.2026.141103

I. Latency

When flows were scored one at a time through the integrated service pipeline (in-process, without HTTP or database overhead), including preprocessing, the stage-1 model(s), calibration, the Isolation Forest, stage 2, SHAP and severity, the improved pipeline had a median latency of 61.4 ms per flow (95th percentile 80.8 ms), against 23.9 ms (33.1 ms) for the original; both were measured on the same 500 KDDTest+ records in three alternating rounds. In vectorised batch scoring, the improved stage-1 ensemble took 17.9 µs per flow, and the original LightGBM was logged at 5.2 µs per flow. The single-flow figures are therefore in milliseconds and the batch figures in microseconds per flow. Training the ensemble took 15.8 s.

IX. LIMITATIONS AND THREATS TO VALIDITY

1) External Validity: NSL-KDD is derived from traffic simulated in 1998 [10]. It does not represent modern protocols, encrypted traffic or current attack tooling. No cross-data-set evaluation, for example on CIC-IDS2017 [26] or UNSW-NB15 [27], was performed, and the 17 unseen NSL-KDD types are a within-data-set proxy for unknown attacks, not evidence of zero-day detection.

2) Internal Validity: The LOATO protocol imitates unseen attacks but cannot reproduce the full train/test shift of NSL-KDD: validation unseen-type recall at the default threshold (62%) was higher than on the test set (31%). The selected threshold of 0.02 is low and depends on this benchmark. The improved detector accepts a 10.2% FPR, which would produce a heavy alert load in practice. Hyper-parameters were searched over twenty configurations, not exhaustively, and the stage-2 classifier was not re-tuned. Per-alert SHAP explanations cover only the LightGBM member of the ensemble, and the integrated API was checked by automated tests and on 500 replayed test flows, and the dashboard only by manual inspection, not under production traffic.

3) Construct Validity: U2R results rest on 52 training and 67 test flows. The severity rules, escalation thresholds and priority multiplier are design judgements that have not been validated with analysts. The MITRE ATT&CK mapping is a reference table and is not implemented.

4) Operational Limitations: Live monitoring replays test records rather than capturing packets, and NSL-KDD's host-window features would need a flow-feature extractor in a live deployment. API access control is a single static key, and the examined incident log used the SQLite fallback.

X. CONCLUSION AND FUTURE WORK

This paper presented an explainable intrusion detection system that combines two-stage tree-ensemble classification, an Isolation Forest, calibrated probabilities, SHAP explanations and rule-based alert prioritisation in a tested API and dashboard, and evaluated it on the official NSL-KDD split. Selecting the model and decision threshold with leave-attack-type-out validation, without using the test set, raised test accuracy from 77.9% to 91.9% and unseen-type recall from 31.1% to 92.7%, at the cost of an FPR rising from 2.9% to 10.2%. Calibrating on the same folds lowered the ECE from 0.214 to 0.050, and revised severity rules corrected an inversion in the alert queue. The improved components are integrated into the FastAPI service and loaded by default, while the original pipeline remains selectable for comparison; the dashboard, which reads all data through the API, was checked manually with the improved models.

Future work will evaluate the pipeline on CIC-IDS2017 and UNSW-NB15, including cross-data-set transfer; add live flow capture and feature extraction; tune the operating point under explicit FPR budgets agreed with analysts; re-tune the stage-2 classifier for R2L; attach per-attack MITRE ATT&CK techniques to alerts; use stored analyst feedback for retraining; and monitor calibration and concept drift over time.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, for its support.

REFERENCES

[1] M. Roesch, "Snort – Lightweight intrusion detection for networks", in Proc. 13th USENIX Conf. System Administration (LISA), 1999, pp. 229–238.

[2] D. E. Denning, "An intrusion-detection model", IEEE Trans. Softw. Eng., vol. SE-13, no. 2, pp. 222–232, 1987.

DOI: 10.17148/IJIREEICE.2026.141103

[3] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", in Proc. IEEE Symp. Security and Privacy, 2010, pp. 305–316.
[4] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey", ACM Comput. Surv., vol. 41, no. 3, pp. 15:1–15:58, 2009.
[5] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection", IEEE Commun. Surveys Tuts., vol. 18, no. 2, pp. 1153–1176, 2016.
[6] D. Arp et al., "Dos and don'ts of machine learning in computer security", in Proc. 31st USENIX Security Symp., 2022, pp. 3971–3988.
[7] M. Sarhan, S. Layeghy, M. Gallagher and M. Portmann, "From zero-shot machine learning to zero-day attack detection", Int. J. Inf. Secur., vol. 22, no. 4, pp. 947–959, 2023.
[8] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning", in Proc. 22nd Int. Conf. Machine Learning (ICML), 2005, pp. 625–632.
[9] C. Guo, G. Pleiss, Y. Sun and K. Q. Weinberger, "On calibration of modern neural networks", in Proc. 34th Int. Conf. Machine Learning (ICML), 2017, pp. 1321–1330.
[10] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", in Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA), 2009, pp. 1–6.
[11] W. Lee, S. J. Stolfo and K. W. Mok, "A data mining framework for building intrusion detection models", in Proc. IEEE Symp. Security and Privacy, 1999, pp. 120–132.
[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", Ann. Statist., vol. 29, no. 5, pp. 1189–1232, 2001.
[13] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree", in Advances in Neural Information Processing Systems 30 (NIPS), 2017, pp. 3146–3154.
[14] L. Breiman, "Random forests", Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.
[15] P. Geurts, D. Ernst and L. Wehenkel, "Extremely randomized trees", Mach. Learn., vol. 63, no. 1, pp. 3–42, 2006.
[16] S. M. Lundberg et al., "From local explanations to global understanding with explainable AI for trees", Nature Mach. Intell., vol. 2, no. 1, pp. 56–67, 2020.
[17] F. T. Liu, K. M. Ting and Z.-H. Zhou, "Isolation forest", in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422.
[18] Y. Mirsky, T. Doitshman, Y. Elovici and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection", in Proc. Network and Distributed System Security Symp. (NDSS), 2018.
[19] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods", in Advances in Large Margin Classifiers, MIT Press, 1999, pp. 61–74.
[20] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates", in Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2002, pp. 694–699.
[21] 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.
[22] G. E. A. P. A. Batista, R. C. Prati and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data", ACM SIGKDD Explor. Newsl., vol. 6, no. 1, pp. 20–29, 2004.
[23] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", in Advances in Neural Information Processing Systems 30 (NIPS), 2017, pp. 4765–4774.
[24] M. T. Ribeiro, S. Singh and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier", in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[25] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington and C. B. Thomas, "MITRE ATT&CK: Design and philosophy", The MITRE Corporation, Tech. Rep. MP180360, 2018.
[26] 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. Information Systems Security and Privacy (ICISSP), 2018, pp. 108–116.
[27] 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 Conf. (MilCIS), 2015, pp. 1–6.
[28] G. Engelen, V. Rimmer and W. Joosen, "Troubleshooting an intrusion detection dataset: The CICIDS2017 case study", in Proc. IEEE Security and Privacy Workshops (SPW), 2021, pp. 7–12.
[29] M. Ring, S. Wunderlich, D. Scheuring, D. Landes and A. Hotho, "A survey of network-based intrusion detection data sets", Comput. Secur., vol. 86, pp. 147–167, 2019.
[30] F. Pedregosa et al., "Scikit-learn: Machine learning in Python", J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[31] B. Efron and R. J. Tibshirani, An Introduction to the Bootstrap. New York, NY, USA: Chapman & Hall, 1993.