iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79633

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

AI-Driven NIDS: A Flow-Based Approach to Threat Detection

Ch. Nagaraju, Mohammed Mazher Ul Hasan, Hashim Ali Abid, Muqtadir Mehmood
Methodist College of Engineering and Technology, Hyderabad, India

Abstract: ***Intrusion detection systems based on signatures face issues with encrypted data and zero-day threats. In this paper, we introduce a novel approach of a hybrid NIDS, combining flow-level analysis with a confidence-gated cascade of classifiers: a weighted average of XGBoost and 1D-CNN (60:40), alongside Isolation Forest invoked upon falling ensemble confidence below a calculated threshold. Training on merged CIC-IDS 2017/2018 datasets (3.49 million flows, 5 classes, 23 behavioral features) with SMOTE-Tomek balancing results in a model with accuracy of 99.83%, FP rate of 0.20% and only 2 false positives among zero-days, outperforming other cascade models (IF-first, AE-first). Testing the same architecture with validation flows (1.83 million honeypot flows) highlights reduced performance for both supervised and unsupervised components due to hypervisorinduced bias, caused by lack of realistic network latency and routing in VM-to-host communication. Results at 10% level of flow injection show that while overall performance is still good (81%), F1-score is notably lower for attacks with specific timing characteristics. Per-flow explainability based on SHAP values is incorporated into our design, and we propose two deployment modes – supervised-only and cascaded for zero-day experiments.***

Index Terms—Network intrusion detection, flow-based classification, ensemble methods, XGBoost, CNN, anomaly detection, Isolation Forest, explainable AI, honeypots.

I. INTRODUCTION

The focus of this paper is on the inability of the current IDS based on signatures to detect zero-day exploits, although they perform well against known exploits [2], [10]. With the increased usage of encryption technologies, deep packet inspections become ineffective because of the inability to read encrypted payloads [2]. Flow-based Network Intrusion Detection Systems (NIDSs) circumvent this problem through connection analysis rather than payload analysis: statistics such as flow duration, packet counts, byte rate, inter-arrivals and TCP flags can be utilized to reveal attack patterns irrespective of the payload being encrypted [3].

Machine Learning (ML) models have certain weaknesses associated with them: the supervised learning models do not detect zero-day attacks, whereas anomaly detectors produce many false alarms [2]. In this research, we propose the following contributions toward designing a flow-based hybrid NIDS: 1) A three-tier, confidence-based decision cascade, incorporating XGBoost [4], 1D-CNN [5] and Isolation Forest [6], with the anomaly detection module working only on low-confidence predictions. 2) A comparative performance analysis among three different architectures that reveals weighted probability averaging with anomaly activation as the best solution in terms of the accuracy, FPR, and sensitivity. 3) Validation using 1.83 million honeypot-derived flows, indicating that the distribution shift becomes the primary impediment to employing anomaly detection in practice. 4) The robustness evaluation of different noise models, revealing the per-class deterioration even in case of constant aggregate performance. 5) A real-time dashboard with SHAP-based flow-level explanations [7].

II. RELATED WORK

A. Supervised Learning for Intrusion Detection

The use of ensemble methods leads to consistent accuracy in benchmark networks like CIC-IDS-2017 [3], [4], [9]. However, the boundaries created by these learning algorithms depend on the behavior of the traffic found in the training environment. As a result, using the method on a new network where the timing and protocols are different makes the boundaries meaningless.

B. Unsupervised and Anomaly-Based Detection

The Isolation Forest is able to isolate the outliers in highdimensional traffic data without the need for any labeled instances (Liu et al. [6]). This algorithm works through the random division of the feature space whereby outliers, which are few, take lesser number of partitions before isolation. However, all abnormal traffic regardless of whether it is legitimate or not is flagged, hence making the management of false positives the main hurdle to implementation [2].

C. Deep Learning for Flow Classification

1D-CNNs treat the ordered list of flow features as a sequence and slide convolutional filters across it, detecting patterns between neighboring features (e.g., the relationship between forward and backward packet lengths) that tree-based models evaluate independently [5]. Combining CNNs with classical ML in hybrid ensembles captures both local feature interactions and global tabular patterns [9], [10].

D. Honeypots and Traffic Collection

The 1D-CNN model analyzes the sequence of flow characteristics as a temporal series, using convolutional filters that traverse the temporal series to find patterns between consecutive characteristics (like the relation between the lengths of packets traveling forward and backward). In the case where CNNs are combined with classic machine learning techniques in hybrid ensembles, the CNN model incorporates both local and global patterns found in tabular data.

E. Explainability and Imbalance Handling

SHAP assigns each feature a contribution score for every prediction, showing *why* a flow was classified as an attack— critical for analyst trust in automated alerts [7]. SMOTETomek addresses class imbalance by synthetically generating minority-class samples (SMOTE) and then removing ambiguous boundary samples (Tomek links), producing a cleaner training set [8]. Adversarial robustness—whether attackers can craft flows that evade detection—remains an active concern [12].

III. PROPOSED SYSTEM ARCHITECTURE

A. Overview

The presented NIDS architecture consists of a three-level confidence-gated decision cascade (Fig. 1). An incoming network flow is represented as a 23-dimensional feature vector extracted using CICFlowMeter.

1) Level 1 — XGBoost: Produces a probability vector over five classes.
2) Level 2 — 1D-CNN: Generates an independent probability vector over the same five classes.
3) Ensemble: $\mathbf{p}_{ens} = 0.6 \cdot \mathbf{p}_{xgb} + 0.4 \cdot \mathbf{p}_{cnn}$.
4) Confidence gating: If confidence $\geq \tau$, the decision is BLOCKED (attack) or PASS (normal); otherwise, it is forwarded for further analysis.
5) Level 3 — Isolation Forest: Activated when confidence $< \tau$, producing an anomaly-based ALERT.

The threshold $\tau = 0.50$ is determined using binning analysis (Section V-D). Consequently, each instance results in one of three outcomes: BLOCKED (high-confidence attack), PASS (high-confidence normal), or ALERT (low-confidence instance flagged by the Isolation Forest as a potential zero-day attack).

B. Layer 1: XGBoost

A scalable ensemble of gradient boosted trees [4] selected due to its solid performance with tabular data, fast inference (in the range of milliseconds), and support for SHAP [7]. While tree-based methods are theoretically invariant to feature scaling, the use of StandardScaler normalization ensures that the input data fed into both XGBoost and CNN have undergone the same preprocessing procedure, ensuring that the resulting weighted probability blending compares outputs from the same feature space.

C. Layer 2: 1D Convolutional Neural Network

Processes the 23 normalized features as a one-channel sequence to capture interactions between local features that could go unnoticed by splitting using trees, such as interaction between neighboring inter-arrival time features or even between forward/backward packet features, which have been noted by Kim [5]. Yields a softmax probability vector. Feature ordering should remain the same both during training and inference because the CNN filters are order-sensitive; changing the order will result in a change in the learned features.

D. Ensemble: Weighted Probability Averaging

Instead of using a simple majority voting scheme, which ignores prediction confidence, the model employs a weighted soft voting approach to combine class probability outputs (refer to Equation 1):

$$\mathbf{p}ens = 0.6 \cdot \mathbf{p}xgb + 0.4 \cdot \mathbf{p}cnn \quad (1)$$

Optimization of ensemble weights. The weights assigned to XGBoost and the 1D-CNN (i.e., 0.6 and 0.4, respectively) were determined empirically via grid search on the validation set. The weight space [0,1] was explored in increments of 0.1, and performance was evaluated using Macro-F_1 score and calibration metrics.

The results indicate that the 1D-CNN effectively captures local spatial dependencies among neighboring features, while XGBoost provides superior performance on structured tabular patterns inherent in the CIC-IDS feature space. The weighted combination leverages the complementary strengths of both models, yielding improved Macro-F_1 performance and bettercalibrated probability estimates.

Consequently, $\mathbf{p}_{ens}$ serves as a reliable and mathematically consistent confidence measure before being passed through the thresholding stage with $\tau = 0.50$.

E. Layer 3: Isolation Forest (Anomaly Guard)

With traffic data trained for the purpose of defining the statistics associated with regular flows [6], the algorithm is used only when the confidence level of the supervised models becomes less than the threshold τ. This algorithm will generate too many false positives in case of its application to all the flows because even slight deviation from the training baseline by any particular flow will be considered an anomaly by the algorithm.

IV. FEATURE ENGINEERING AND PREPROCESSING

A. Feature Selection

Out of all the 27 behavioral features used in CICFlowMeter [3], we choose 23, discarding the Destination Port feature, even though it is one of the most important according to SHAP value. However, Destination Port shows strong overfitting towards specific values used for the dataset (port 80 for HTTP attacks). An actual attacker might use malware on other ports as well. Thus, an over-reliance on port numbers poses the risk of missing out on attacks executed using other ports. All the features are listed in Table I. From these 23 features, 18 are log(1 + x) transformed to address the issue of heavy-tailed distribution (e.g., Flow Bytes/s can vary

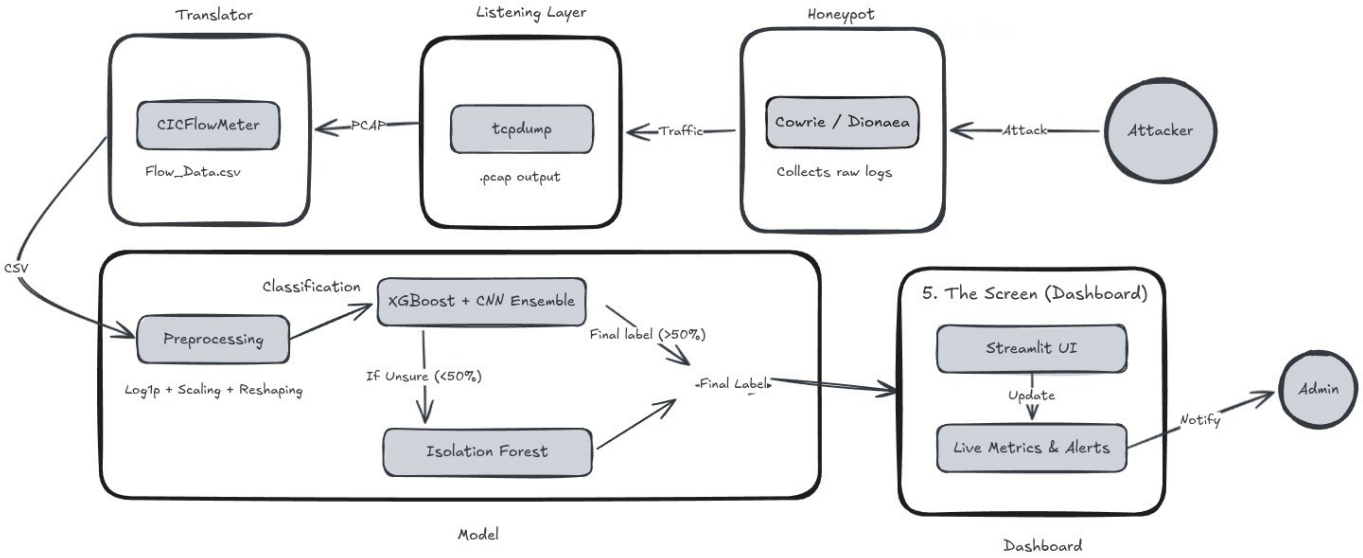


Fig. 1. Three-level confidence-gated decision cascade. The Isolation Forest is activated only when the ensemble confidence falls below $\tau = 0.50$.

between 0 to several millions).

TABLE I
SELECTED 23 BEHAVIORAL FLOW FEATURES

#	Feature	Log
1	Flow Duration	✓
2	Flow Bytes/s	✓

3	Flow Packets/s	✓
4	Total Fwd Packets	✓
5	Total Backward Packets	✓
6	Total Length of Fwd Packets	✓
7	Total Length of Bwd Packets	✓
8	Fwd Packet Length Mean	✓
9	Fwd Packet Length Max	—
10	Fwd Packet Length Std	✓
11	Bwd Packet Length Mean	✓
12	Bwd Packet Length Std	✓
13	Packet Length Std	✓
14	Flow IAT Mean	✓
15	Flow IAT Max	✓
16	Fwd IAT Total	✓
17	Bwd IAT Total	✓
18	Fwd Header Length	✓
19	Init Win bytes forward	—
20	Init Win bytes backward	—
21	SYN Flag Count	—
22	PSH Flag Count	—
23	act data pktfwd	✓

Log = log(1+x). Grouped: flow-level (1–3), counts (4–7), packet length (8–13), IAT (14–17), TCP (18–22), activity (23)

B. Preprocessing Pipeline

This process was employed in both the training and prediction procedures, and involved the following steps: (1) strip extra white space from the columns, (2) rename columns based on a prescribed mapping, (3) add zero-padding for missing attributes, (4) ensure that all data is numeric, (5) replace ±∞ with NaN, (6) fill NaNs with 0s, (7) clamp all values to be non-negative by setting them to zero if necessary, (8) transform the log(1 + x) function on 18 attributes, (9) scale using StandardScaler, and (10) reshape to (N,23,1).

V. EXPERIMENTAL SETUP

A. Datasets

The training set is created using CIC-IDS 2017 [3] and CIC-IDS 2018 datasets, including 3.49 million flows. The finegrained classes (DoS GoldenEye, DoS Hulk, DoS Slowloris) are aggregated into five coarse-grained classes: Normal, DoS/DDoS, Brute-Force, PortScan, and Bot because finegrained attacks share similar flow characteristics, and aggregation promotes better class balancing. Class Heartbleed and Infiltration are excluded as they do not have enough examples to learn meaningful patterns (less than 100 samples each). Class counts are presented in Table II.

TABLE II
DATASET CLASS DISTRIBUTION (COMBINED CIC-IDS 2017+2018)

Class	Raw Count	%	After Balancing
Normal	2,900,000	83.1	Undersampled (200K)
DoS/DDoS	380,000	10.9	Balanced
PortScan	159,000	4.6	Balanced

Brute-Force	27,000	0.8	SMOTE oversampled
Bot	24,000	0.7	SMOTE oversampled
Total	3,490,000	100	Balanced

SMOTE-Tomek [8] applied after undersampling Normal to 200K. Split: 80/20 stratified.

Class Normal is downsampled to 200k samples so that the model would not achieve good performance by always predicting the dominant class (83% of data). SMOTE-Tomek [8] algorithm is used to balance minority classes: SMOTE creates synthetic samples by interpolating between samples from the minority class, and Tomek links eliminate borderline samples located near class boundaries. The CIC-IDS 2018 adds more samples to the Bot and Brute-Force classes that were underrepresented in 2017 only. Two additional features are selected based on the SHAP analysis (Bwd Packet Length Std, Fwd Header Length), so that there will be 23 features in total.

B. Honeypot-Based Live Traffic Collection

To create validation outside of benchmark CSVs, we created a controlled honeypot testbed inside our local network [11]. The Linux Mint laptop (IP 192.168.1.7) provided Dockerbased deployments of Cowrie (SSH honeypot) and Dionaea (multi-protocol honeypot) that acted as victims. The Kali
Linux virtual machine (IP 192.168.1.8) used in VirtualBox bridged mode played the role of an attacker running such scans as Nmap (Port Scan), hping3/ slowloris (DoS/DDoS), Hydra (Brute Force) and our own customized Command & Control Server with Python bot (C2-Bot) built on Flask framework. Raw PCAP files captured with tcpdump from wireless interface of the victim computer were converted to flow level CSVs via CICFlowMeter using 78 features scheme identical to CIC-IDS 2017 benchmark. 1,832,873 flows were generated during this experiment per different attacks.

C. Noise Injection for Robustness Testing

The real-world networks suffer from problems such as congestion, packet drops, and fluctuation in bandwidths which have an effect on the statistics of flows. In order to simulate these scenarios, physically motivated noise [12] is added to a collection of 1,400 flows where the noise is applied with a severity of 10%. The noise models considered for this purpose affect the statistics differently where Pareto noise (α = 3.0) applied on time statistics represents burst congestion in the router causing unpredictable delays in traffic only increasing the time statistics; fractional dropout applied to the count statistic simulates packet drops causing a reduction in packet counts since certain packets do not arrive at all; and Gaussian jitter (μ = 1.0, σ = severity) applied to rate statistics represents fluctuation in speeds that occur symmetrically. Three architectural variants are evaluated under noise to isolate the contribution of each component: Model A (IF → XGB → CNN), Model B (AE → XGB → CNN), and Model C (Weighted Ensemble + IF).

D. Threshold Derivation

For determining when the confidence level of the model reaches an acceptable state, all predictions of the validation set are categorized into 20 bins according to their confidence levels ranging from 0 to 5%, 5 to 10%, etc., and up to 95 to 100%. The value τ = 0.50 is selected as the confidence threshold such that any prediction made with a confidence higher than or equal to τ has a precision of 0.95 or more. Any prediction with confidence less than τ is considered invalid, so the flow is assigned to the Isolation Forest algorithm.

VI. RESULTS AND ANALYSIS

A. Architectural Variant Comparison

Table III compares variants on the clean testset(~550,866 flows).

TABLE III
CLEAN-DATA PERFORMANCE OF THREE ARCHITECTURAL VARIANTS

Metric	A: IF+XGB+CNNB:	AE+XGB+CNNC:	Ens.+IF (Ours)
Accuracy	99.67%	99.67%	99.83%

F1 (macro) 99.44% 99.44% 99.68% ROC-AUC 1.0 1.0 1.0

A/B: sequential cascade. C: weighted ensemble (0.6×XGB+0.4×CNN) with confidence-gated IF.

The accuracy of sequential cascades (A, B) is 99.67%. The accuracy of Model C is 99.83% since it enables both supervised models to inspect all flows and considers the probability scores before making a decision. In the case of Models A and B, the anomaly detection model acts as a gatekeeper by deciding whether or not the supervised models have access to a particular flow. Consequently, any error in this step will be carried forward. In the case of Model C, the Isolation Forest acts only when the ensemble model makes a mistake in a very small number of flows [9].

B. Live Traffic Evaluation

The approach was tested on a dataset consisting of 1,832,873 flows obtained using the honeypot testbed. Supervised and unsupervised layers showed considerable degradation in performance due to hypervisor bias, which occurs when the attacking VM and the victim host are running on the same physical computer. The flows do not have any of the typical features of real WAN data, including no delay, router jitter, low fragmentation, and few flow statistics. As a result, the CICFlowMeter distributions differ significantly from those of the CIC-IDS 2017/2018 datasets acquired in a multi-hop campus network scenario [1].

Isolation Forest performed exceptionally badly on benign flows, generating a huge amount of false positives in excess of 99%. The supervised layer could not generalize well to the range of compressed features of the same-host traffic. Such behavior indicates an essential criterion for any flow-based IDS—namely, having access to large amounts of networklevel data, such as timestamps and routing statistics. Samehost configuration cannot create sufficient statistical diversity for classification.

Production implementation suggestions:

1) Production environments: deploy the supervised ensemble on hardware that allows establishing network connections (such as a span port or network tap).

2) Zero-day research scenarios: utilize the complete three-layer cascade on distributed datasets only.

C. Noise Robustness

Table IV shows results at 10% severity.

TABLE IV NOISE ROBUSTNESS EVALUATION AT 10% SEVERITY

Architecture	Accuracy	F1 (macro)
A: IF → XGB → CNN	80.57%	0.4550
B: AE → XGB → CNN	80.57%	0.4550
C: Weighted Ens. + IF	80.71%	0.4756

1,400-flow dataset with physically-motivated noise.

Accuracy remains ∼81% (since the Normal class forms the majority of the sample, accurate recognition of Normal samples contributes positively to accuracy). However, macro F_1 drops to ∼46–48%, demonstrating the actual decrease in efficiency. Bot recognition fails ($F_1 \approx 0.00$) because it relies heavily on exact TCP window sizes and inter-arrival intervals—Pareto jitter affects this parameter set directly, making Bot flows indistinguishable from regular traffic. Detection of DoS/DDoS degrades ($F_1 \approx 0.36$) because distortion of very high byte and packet rates (critical for detection) occurs. Detection of PortScan remains unaffected ($F_1 \approx 0.98$) since it is based on packet *count* patterns.

VII. EXPLAINABILITY

SHAP TreeExplainer [7] provides per-flow explanations for XGBoost by measuring each feature's contribution to the final prediction. For example, the feature *Init Win bytes forward* positively impacts the probability of Bot, which can be attributed to the preference of bots to use unconventional TCP window sizes. Global analysis across the test set shows that *Init Win bytes forward* and *Init Win bytes backward* are the most distinctive features for Bot. *Flow Duration* and *Flow IAT Max* are primarily responsible for identifying DoS/DDoS(long-duration flooding attacks versus short bursts). *Fwd Packet Length* statistics mainly govern PortScan (many small probe packets) and Brute-Force (repetitive login attempts with consistent packet sizes).

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

VIII. DISCUSSION AND FUTURE WORK

A. *Key Findings*

Three major lessons can be gleaned from this experiment. First, Confidence-Gated Cascading outperforms standard front-gating: by letting supervised models do classification initially and invoking the anomaly detector solely in cases where confidence is low (<0.50), we obtain a large decrease in false positives, a characteristic issue of an unsupervised NIDS approach. Second, Weighted Probability Averaging leads to a more robust decision boundary compared to the models individually and ensures proper function of the gating mechanism in distinguishing between confident predictions and uncertain cases. Third, in evaluating robustness to noise, it became clear that Timing-Dependent Attack Classes (Botnets) exhibit great fragility. Being based on the time features which are disturbed by noise, Botnets suffer greatly in terms of supervised recall, while Anomaly Layer maintains the defense.

B. *Limitations*

1) Static thresholding. The system employs fixed thresholds, with confidence $\tau = 0.50$ and anomaly threshold 0.2, which remain constant across all operating conditions. However, real-world network traffic exhibits stochastic variability. During low-activity periods, the threshold may be overly strict, flagging benign fluctuations as anomalies, whereas during peak traffic, it may become too permissive, missing subtle intrusions. Consequently, the model cannot adapt to dynamic baseline shifts, leading to increased false negatives and reduced robustness.

2) Temporal fragility. Detection of timing-dependent attacks such as Botnets relies heavily on precise interarrival time patterns. Experimental observations with injected noise (∼10%) show that even moderate jitter distorts these temporal signatures, making them indistinguishable from normal traffic. This limitation arises because flows are treated as independent snapshots, and the temporal continuity of attack behavior is not explicitly modeled.

3) Dataset dependency. The expert models (XGBoost and CNN) are trained on CIC-IDS-2017/2018 datasets, which were collected in controlled academic environments. These datasets lack the protocol diversity and complexity of modern production networks, including IoT traffic, 5G communication, and encrypted tunnels. As a result, the model may fail to generalize to emerging attack patterns not represented in the training data.

4) Zero-day attack limitation. Although the Isolation Forest layer can detect anomalous flows, it functions as a generic anomaly detector without semantic understanding. It can identify deviations from normal behavior but cannot distinguish between novel attacks and benign but unusual network configurations, limiting its effectiveness against zero-day exploits.

C. *Future Directions*

1) Dynamic probability gating. Future work will explore a dynamic probability gate within an adaptive thresholding mechanism. This approach continuously monitors the average confidence of legitimate flows and adjusts the threshold parameter τ in real time to maintain a stable false positive rate under varying network conditions.

2) Reinforcement learning for continuous adaptation. To address the need for richer and evolving data distributions, a continuous learning pipeline leveraging reinforcement learning (RL) will be investigated. An RL agent can utilize feedback from prediction outcomes to update model parameters, enabling adaptation to emerging attack patterns without requiring full retraining.

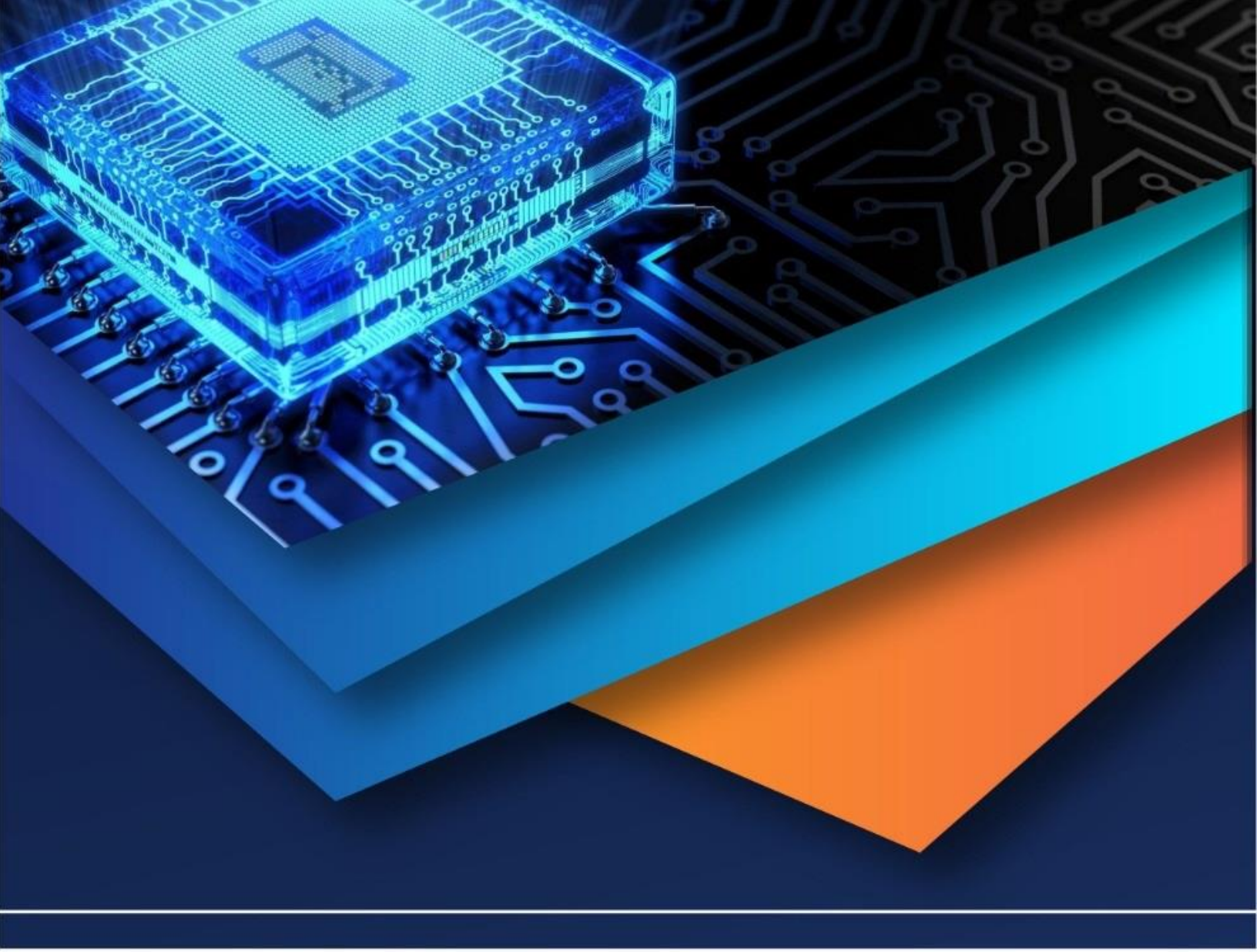
3) Recurrent models for temporal modeling. To mitigate temporal fragility, future work may replace the 1D-CNN with recurrent architectures such as Long Short-Term Memory (LSTM) networks. LSTMs maintain memory of previous states, allowing effective modeling of sequential dependencies and improving detection of attacks spanning multiple packet sequences.

IX. CONCLUSION

The proposed approach implements a confidence gate, which triggers the anomaly detection only when the prediction is made with very low confidence. Consequently, the model achieves 99.83% precision and 0.20% FPR, outperforming approaches that perform anomaly detection on all flows. Validation with honeypots shows that flow-based models need actual traffic on the network level to work well, while flows generated using VM to Host configurations are too few from a statistical point of view to ensure a correct classification. A study on noise reveals that, even though the model performs relatively well, attacks that depend on timing attributes do not work in presence of jitter. Integration with SHAP provides insight into the reasoning of each individual flow.

REFERENCES

[1] P. Laskovet al., "Learning intrusion detection: Supervised or unsupervised?" Proc. 13th Int. Conf. Image Analysis and Processing, pp. 50–57, 2005.
[2] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," IEEE S&P, pp. 305–316, 2010.
[3] I. Sharafaldinet al., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," ICISSP, pp. 108–116, 2018.
[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," ACM SIGKDD, pp. 785–794, 2016.
[5] J. Kim et al., "A 1D-CNN based deep learning approach for intrusion detection system," IEEE BigComp, pp. 605–608, 2018.
[6] F. T. Liu et al., "Isolation Forest," IEEE ICDM, pp. 413–422, 2008.
[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," NeurIPS, vol. 30, 2017.
[8] G. E. Batista et al., "A study of the behavior of several methods for balancing ML training data," ACM SIGKDD Explorations, vol. 6, no. 1, pp. 20–29, 2004.
[9] M. A. Ferraget al., "Deep learning for cyber security intrusion detection: A survey," J. Inf. Sec. Appl., vol. 52, 2020.
[10] M. Z. Alom et al., "A deep learning approach for network intrusion detection system," IEEE CCWC, pp. 248–253, 2019.
[11] J. Franco et al., "A survey of honeypots and honeynets for IoT," IEEE COMST, vol. 23, no. 4, pp. 2351–2383, 2021.
[12] N. Martins et al., "Adversarial ML applied to intrusion and malware scenarios," IEEE Access, vol. 8, pp. 35403–35419, 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY