Collaborative Packet-Level Intrusion Detection System for IoT Environment

Chenqi Tao
Sophia University
Tokyo, Japan
c-tao-4d5@eagle.sophia.ac.jp

Masaki Bandai
Sophia University
Tokyo, Japan
bandai@sophia.ac.jp

***Abstract*—In this paper, a packet-level intrusion detection system for IoT environments is proposed, utilizing a two-node collaborative architecture. In the proposed system, signature-based prescreening at the gateway operates with a random forest model deployed on a high-performance node. This approach aims to reduce computational load on the gateway while maintaining packet-level detection accuracy and processing throughput. In a real-world network deployment, we confirm that the proposed system can reduce the gateway's computational load by 20% and increased processing throughput by 30% compared with a gateway-only baseline system.**

***Index Terms*—IoT security, intrusion detection system (IDS), packet-level analysis, random forest (RF), Zeek**

I. Introduction

The rapid growth of the Internet of Things (IoT) has expanded the potential attack surface beyond traditional enterprise boundaries. IoT devices are heterogeneous, resource-constrained, and difficult to manage or update regularly, making them vulnerable to attacks and large-scale exploitation. As IoT systems increasingly handle safety- and mission-critical tasks, securing the network layer has become a major concern.

Intrusion Detection Systems (IDS) play a key role in this effort by monitoring network traffic and identifying malicious behaviors that other security controls might miss. There are two main types of IDS: signature-based and anomaly-based [1].

Signature-based IDS use predefined patterns to detect known threats with high accuracy and low false positives. However, they are less effective against new attacks and require frequent signature updates, which is especially challenging in IoT environments with diverse protocols and encryption.

Anomaly-based IDS rely on machine learning (ML) models—either supervised or unsupervised—to analyze packet-level or flow-level features. This approach is more effective at identifying unknown or zero-day attacks. However, it requires more computing power and depends heavily on data quality and consistent deployment conditions.

Detection granularity significantly affects latency. Flow-level analytics aggregate packets to capture temporal and relational context, enhancing semantic understanding. However, it introduces latency during flow construction [2]. In contrast, packet-level analytics extract features from individual packets and can operate before complete flows are formed, enabling earlier detection. However, running packet-level learning continuously in all conditions tends to be fragile and resource-intensive in practice. A more practical strategy is event-triggered packet analysis: when suspicious behavior is detected, a compact set of features is rapidly extracted from a small number of packets to support timely and reliable decisions—balancing between latency and robustness.

In this paper, a two-node collaborative IDS architecture for IoT gateways is proposed. The gateway node performs lightweight, signature-based prescreening and, upon detecting suspicious events, enters an elevated-scrutiny window, during which packet-level feature extraction is applied to all traversing traffic. These features are then streamed to a high-performance node for classification. This architecture realizes low latency without requiring full flow reconstruction, efficiently offloads computation from the gateway, and enables a closed-loop workflow—allowing benign traffic to pass through while promptly blocking malicious traffic.

II. Related Work

This section introduces the related work on IDS for IoT environment.

A. Signature-Based IDS

Waleed et al. [3] perform a unified benchmark comparing Snort, Suricata, and Zeek, highlighting how engine design and packet I/O decisions affect loss-free throughput. The analysis emphasizes Zeek's role as a high-fidelity network monitor suitable for IDS, making it ideal for policy scripting and alerting at observation points. This characteristic supports deploying Zeek at the gateway as a lightweight trigger, raising meaningful events from live traffic with minimal overhead to initiate downstream actions without relying on a heavyweight inline engine.

B. IPF-Based IDS

Kostas et al. [2] provide the first systematic evidence that models trained solely on individual-packet features (IPF) often rely on dataset-specific artifacts, leading to inflated performance within the same dataset but poor generalization across sessions or datasets. Despite these limitations, IPF are retained at the gateway for practical reasons. The system consists of two coordinated components: a Zeek-based, signature-driven module for continuous monitoring and event triggering, and an anomaly-based module that performs line-rate, low-overhead IPF extraction when triggered. By combining interpretable event triggers with ultra-low-latency processing, extracted features are sent to a high-performance node for classification, meeting real-time requirements while mitigating some limitation of IPF-only methods.

C. Machine-Learning-Based IDS

Altulaihan et al. [4] show that supervised learning with feature selection achieves strong in-dataset performance for IoT DoS detection, while lightweight models offer efficient training and inference. Based on these results, the ML classifier is deployed on the high-performance node, which returns classification decisions for features uploaded by the gateway. Model details and training procedures are described in Section III.

D. Lightweight and Distributed IDS

Thakkar et al. [5] survey intrusion detection placements across device, edge/gateway, fog, and cloud, highlighting trade-offs in energy, computation, and bandwidth. Based on this, a distributed design is adopted: lightweight screening and event extraction should be near the data source to minimize latency and uplink traffic, while intensive analysis is offloaded to a high-performance node where ML models are centrally managed. This setup ensures fast edge response, reduces signature maintenance on constrained devices, and improves detection accuracy.

III. Proposed System

We propose a two-node collaborative IDS architecture for IoT gateways. As shown in Fig. 1, the proposed system consists of a gateway and a high-performance node. The gateway performs lightweight, signature-based prescreening using Zeek, while the high-performance node classifies suspicious traffic as either malicious or non-malicious using an random forest (RF) classifier.

A. Operation

The operation of the proposed system proceeds as follows:

1) *Pre-screening at the Gateway*:
   All traffic first passes through the gateway, which continuously runs Zeek with signature-based prescreening rules, such as detecting surges of incomplete connections, port scanning, and ARP inconsistencies. Zeek acts as a rule-level trigger to determine whether further inspection is necessary. If Zeek detects a suspicious event, the system enters an elevated scrutiny window for m seconds. During this window, packet-level features are extracted from all incoming traffic. Table I lists the 60 features extracted per packet. These features are streamed in real time as a JSON file to the high-performance node for malicious traffic classification.
2) *Classification by the High-Performance Node*:
   Upon receiving the JSON file, the high-performance node classifies the traffic as either malicious or non-malicious using a RF classifier. If the traffic is classified as malicious, a decision value of 1 is sent to the gateway. Otherwise, a decision value of 0 is transmitted.
3) *Gateway Response to Classification*:
   Upon receiving the classification decision, the gateway updates its local filtering state accordingly.
   - If the traffic is malicious, it is either quarantined or rate-limited.
   - If the traffic is non-malicious, it is forwarded to IoT devices without further intervention.
4) *Window Closure*:
   At the gateway, if no additional prescreening triggers or malicious classification decisions occur during the remainder of the scrutiny window, the window closes automatically. The system then returns to its normal prescreening state.

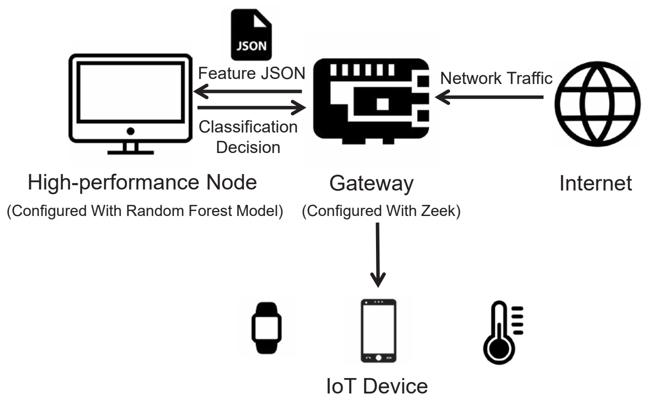

Fig. 1. The proposed two-node collaborative architecture

B. Signature-Based Prescreening at Gateway

We deploy Zeek at the gateway as a signature-based prescreener and lightweight trigger. As discussed in Section II, Zeek is better suited for policy scripting and alerting rather than inline traffic blocking. Therefore, in this paper, we do not use Zeek to classify traffic as malicious or non-malicious. Instead, we leverage Zeek's policy scripts and signature-style event rules to generate triggers. When any of these rules is activated, Zeek opens the elevated scrutiny window. The following rules are applied by Zeek:

- *DoS:* Trigger on a five-second window where the TCP SYN count spikes and the three-way handshake completion ratio is low.
- *Port scanning:* Trigger on a five-second window where one source probes many distinct destination ports on the same host and most attempts fail.
- *Mirai:* Trigger on a five-second window where one source contacts many distinct destination IPs using a small fixed set of service ports, with attempts brief or mostly

TABLE I
PACKET-LEVEL FEATURES

Category	Features
Header	`src_port`, `dst_port`, `proto`, `header_length`, `length`, `ttl`, `tcp_flags`
Size	`payload_len`, `payload_0-31`
TCP Flags	`syn_flag`, `ack_flag`, `fin_flag`, `rst_flag`, `psh_flag`, `urg_flag`, `ece_flag`, `cwr_flag`
Protocol Indicators	`is_tcp`, `is_udp`, `is_icmp`, `is_arp`, `is_dns`, `is_dhcp`, `is_http`, `is_https`, `is_ssh`, `is_smtp`, `is_irc`, `is_telnet`

unsuccessful; complementary cues include ICMP echo sweeps and rapid ARP *who-has* bursts on the local subnet.

- *ARP spoofing:* Trigger on a five-second window where ARP replies conflict with learned IP↔MAC bindings, unsolicited ARP bursts appear, or the observed MAC for a given IP changes rapidly.

C. Classification of Malicious Traffic at the High-Performance Node Using an RF Classifier

As discussed in Section II-C, RF classifiers are commonly used in IDS. In this study, we employ an RF classifier at the high-performance node to classify traffic as either malicious or non-malicious.

1) *Dataset*:
We use the CICIoT2023 dataset [6] to train the RF model. This dataset includes traffic from 105 IoT devices and 33 attack types spanning seven categories: DDoS, DoS, Reconnaissance, Web-based, Brute Force, Spoofing, and Mirai. The attacks are launched by compromised IoT devices targeting other devices in the network. The dataset is provided as raw packet capture (PCAP) files, along with aggregated features computed over fixed-size packet windows.
Due to the dataset's large size, we created a subset for training that includes PCAP traces from four attack families—DoS, Spoofing, Mirai, and Reconnaissance—as well as benign (non-malicious) traffic. Packet-level features were extracted from the traces, then standardized and normalized. The final labeled dataset contains a total of 976,461 packets, comprising 417,389 benign instances and 559,072 attack instances.
2) *Feature extraction and selection*:
We employ the 60 packet-level features listed in Table I.—spanning header and size fields, TCP flags, protocol indicators, and a small payload slice to characterize service targeting, traffic footprint, and handshake dynamics. Of these, 27 are non-payload features and considering that most malicious attacks avoid encrypting payloads to sustain high send rates,, we include the payload length and the first 32 payload bytes as byte-level features.
3) *Model training*:
We conduct two training scenarios for the RF model:
   - *Full features model*: The RF is trained using the complete set of 60 features, as listed in Table I.

TABLE II
PERFORMANCE OF RF MODEL

	RF	RF (Top-20)
Accuracy	0.993	0.992
Precision	0.993	0.992
Recall	0.993	0.992
F1	0.992	0.991

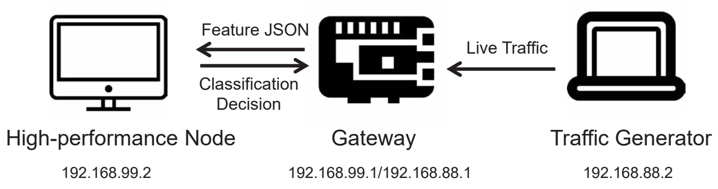

Fig. 2. Evaluation environment for live-traffic experiments.

   - *Reduced features model*: Within each training fold, the top 20 features are selected based on their ranking from the full 60-feature set, and the RF is retrained using this subset.

IV. PERFORMANCE EVALUATION

We evaluate the proposed system in two settings: a dataset study to measure detection accuracy and a deployment in a real network to measure resource utilization.

A. Evaluation using a dataset

For the dataset study, packet-level feature vectors are derived from CICIoT2023 PCAP files. Two RF variants are trained: one using the full feature set and one retrained on the top-20 features ranked by full features model importance.

Table II shows the performance of the RF models. From the table, the RF trained on the full feature set and top-20 features set achieve almost same performance in terms of accuracy, precision, recall, and F1 score. Therefore, to minimize gateway-side feature extraction and serialization time, the top-20 features set is used for all subsequent online experiments in the two-node collaborative architecture.

B. Evaluation in a real network

We implemented the proposed system and conducted live-traffic experiments in a real network. Fig. 2 illustrates the system architecture, which comprises three computers: a traffic generator, a gateway, and a high-performance node. The system components are detailed as follows:

- Traffic generator, running on a Kali Linux host, generates live traffic directed at the gateway. It produces TCP SYN segments using hping3 for 60 seconds at a controlled offered rate, denoted as R.
- Gateway, implemented on a Raspberry Pi 5 running Ubuntu 22.04, runs Zeek to screen for suspicious traffic.
- High-performance node is a desktop PC equipped with an Intel Core i7-11700K CPU and 64 GB of RAM. A RF

TABLE III
RESOURCE USAGE ON GATEWAY

		$R = 15$ kpps	$R = 19.5$ kpps
CPU	Gateway-only	98 %	100 %
	Proposed method	78 %	97 %
Memory	Gateway-only	26 %	26 %
	Proposed method	20 %	20 %

classifier is used on this node to distinguish malicious traffic from non-malicious traffic.

For comparison, we also consider a baseline system in which the gateway itself classifies traffic using the RF model without offloading to the high-performance node. In this configuration, Zeek is not utilized. We refer to this system as Gateway-only.

First, we conducted experiments to determine the maximum traffic rate R that each system can handle without packet loss. The Gateway-only system supports a maximum rate of $R = 15$ kilo packets per second (kpps), beyond which packet losses occur. In contrast, the proposed system supports a higher rate of $R = 19.5$ kpps. These results confirm that the proposed system can handle approximately 30% more traffic than the Gateway-only system, thanks to offloading the RF classification to the high-performance node.

Next, we evaluated the gateway's resource utilization—specifically, the average CPU and memory usage—under two traffic conditions: $R = 15$ kpps and $R = 19.5$ kpps. Table III presents the results. At $R = 15$ kpps, the proposed system reduced CPU and memory usage by approximately 20% and 6%, respectively, compared to the Gateway-only system. Moreover, at $R = 19.5$ kpps, the CPU and memory usage in both systems are nearly identical, although packet loss occurs in the Gateway-only system under this condition. These findings confirm the effectiveness of the proposed system in both performance and resource efficiency.

V. Conclusion

This paper proposed a two-node collaborative IDS for IoT networks, consisting of a Zeek-based prescreener and a RF classifier deployed on a high-performance node to detect malicious traffic. Based on results from a real-world network deployment, we confirmed that the proposed system reduces the gateway's computational load by 20% and increases processing throughput by 30% compared to a gateway-only baseline system. As future work, we plan to extend our evaluation to include additional datasets and attack families, as well as explore multi-gateway deployment scenarios.

References

[1] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (ids)," *International Journal of Information Security*, vol. 22, no. 5, pp. 1125–1162, 2023. [Online]. Available: https://doi.org/10.1007/s10207-023-00682-2

[2] K. Kostas, M. Just, and M. A. Lones, "Individual packet features are a risk to model generalization in ml-based intrusion detection," *IEEE Networking Letters*, vol. 7, no. 1, pp. 66–70, 2025.

[3] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source ids? snort, suricata or zeek," *Computer Networks*, vol. 213, p. 109116, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622002420

[4] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Anomaly detection ids for detecting dos attacks in iot networks based on machine learning algorithms," *Sensors*, vol. 24, no. 2, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/2/713

[5] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, pp. 3211–3243, 2021.

[6] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941