

Anomaly-based intrusion detection leveraging optimized firewall log analysis: a real-time machine learning solution

Tran Cong Hung¹, Dam Minh Linh², Han Minh Chau³, Ngo Xuan Thoai², Thai Duc Phuong¹,
Huynh De Thu¹

¹School of Computer Science and Engineering, The Saigon International University, Ho Chi Minh City, Vietnam

²Information Security Technology Lab and Faculty of Information Technology, Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam

³Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received May 26, 2025

Revised Jul 7, 2025

Accepted Jul 12, 2025

Keywords:

Anomaly detection

Cybersecurity

Firewall logs

Intrusion detection

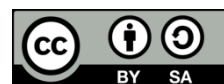
Multi-threading

Real-time systems

ABSTRACT

Firewall logs play a vital role in cybersecurity by recording network traffic and flagging potential threats. This study evaluates five machine learning algorithms—decision tree (DT), random forest (RF), extra trees (ET), CatBoost (CB), and AdaBoost (AB)—on a dataset of 65,532 firewall log entries. Models were assessed using accuracy, precision, recall, training/prediction time, and Pearson correlation for feature selection, across multiple train-test splits. The DT model achieved the best performance, reaching 99.45% test accuracy, 97.457% precision, and 93.389% recall at a 7:3 split, along with the fastest training time (0.20642 s). We propose real-time flow-level intrusion detection (RT-FLID), novel, lightweight, real-time intrusion detection system that leverages multithreaded processing and flow-level analysis to boost detection speed and scalability. Unlike existing approaches that rely heavily on deep packet inspection or computationally intensive processing, RT-FLID requires minimal resources while maintaining high detection accuracy. The architecture efficiently handles large traffic volumes and dynamically identifies anomalies such as distributed denial-of-service (DDoS) and port scans. Validated on real-world logs, the system maintained high accuracy in critical classes like “deny” and “reset-both.” These findings highlight RT-FLID’s novelty and practical advantages, demonstrating its potential for deployment in high-throughput, low-latency network environments.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Dam Minh Linh

Information Security Technology Lab and Faculty of Information Technology, Posts and Telecommunications Institute of Technology

Ho Chi Minh City, Vietnam

Email: linhdm@ptit.edu.vn

1. INTRODUCTION

Firewalls play a crucial role in an organization's network security, acting as the primary barrier against threats. They offer protection against both external and internal attacks, ensuring comprehensive security coverage. Given their significance in safeguarding systems, the logs generated by firewalls provide valuable insights into network traffic patterns and facilitate enhanced monitoring and analysis.

Cisco's 2024 cybersecurity readiness index offered an in-depth assessment of global cybersecurity preparedness within organizations [1], revealing that only 3% of companies are equipped to address current threats, whereas two-thirds of organizations are at the beginner or formative stages of preparedness. Notably,

73% of respondents expected a cybersecurity incident to impact their business within the next 12 to 24 months. The price of unpreparedness can be significant, as 52% of impacted respondents incurred a cost of at least US \$300,000. The Federal Bureau of Investigation (FBI) developed a “Tech Tuesday” in [2] and the Phoenix Field Office alerted the public about criminal actors engaging in phishing and spoofing scams and provided tips to reduce the likelihood of falling victim. The FBI’s internet crime complaint center (IC3) reported that phishing scams were the most prevalent type of cybercrime in 2020, with over 240,000 victims and losses of nearly \$50 million. Although spoofing scams impacted fewer individuals—approximately 28,000—the financial losses exceeded \$215 million. Noting the rise in global security threats, the 2024 Microsoft digital defense report [3] highlighted a sharp increase in identity-related attacks, especially password breaches, which account for over 99% of the 600 million daily identity-focused attacks. However, one key solution inferred from recent cybersecurity reports is to enhance firewall log data analysis to improve real-time threat detection and significantly reduce the risk of successful cyberattacks.

In recent years, the application of artificial intelligence (AI) techniques—particularly machine learning (ML) and deep learning (DL)—has gained increasing attention in cybersecurity, especially in firewall anomaly detection. Traditional firewall systems, which rely on manually configured rule sets to filter network traffic, are prone to misconfigurations and may fail to detect novel threats. To overcome these limitations, advanced ML and DL approaches have been proposed to enhance the detection of suspicious activities in firewall logs [4]. Komadina *et al.* [5] investigated the effectiveness of anomaly detection by injecting synthetic attack data into firewall logs, offering a controlled yet realistic evaluation environment. Building upon this foundation, subsequent research [6] leveraged AI to autonomously optimize and validate firewall rule sets in high-performance network infrastructures, thereby enhancing policy accuracy and reducing manual configuration errors. Extending this line of work, a lightweight and cost-effective *Smart Unified Threat Management System* was proposed in [7], which was deployed on a Raspberry Pi platform to secure home networks against modern cyber threats. The system achieved a detection accuracy of up to 99% while reducing memory consumption by approximately 55% compared to traditional signature-based solutions.

Firewall logs are a vital component of network security, providing visibility into data traffic and enabling the identification of potential network security risks and suspicious behaviors. The critical role of firewall logs in intrusion detection systems (IDS) has spurred the adoption of ML approaches to evaluate network security datasets, particularly for identifying abnormal traffic patterns indicative of targeted attacks on critical servers. Aljabri *et al.* [8] developed a ML-based framework to classify firewall sessions into categories such as “Allow,” “Drop,” “Deny,” and “Reset-both.” Their evaluation utilized a firewall log dataset containing 65,532 records, distributed as follows: Allow (37,640), Deny (14,987), Drop (12,851), and Reset-Both (54). By introducing two novel features, application and category, the study significantly improved classification performance, with the random forest (RF) algorithm achieving up to 99.64% accuracy. However, the current system remains suboptimal in terms of availability and rapid response for real-time network environments. Based on the same firewall log dataset, study [9] proposed a multiclass classification method using support vector machine (SVM) to categorize connection sessions into four actions: allow, deny, drop, and reset-both, through systematic comparison of kernel functions to optimize classification performance. Subsequently, the research in [10] implemented and evaluated multiple classification algorithms, with RF achieving up to 99% accuracy, demonstrating the effectiveness of feature extraction techniques in firewall log analysis. Nevertheless, both research efforts primarily rely on pre-labeled data and provide limited exploration of real-time responsiveness and data imbalance issues.

Li *et al.* [11] introduced a novel intrusion detection model, adversarial environment with soft actor-critic (AE-SAC), which integrates adversarial learning and deep reinforcement learning to address challenges in imbalanced datasets and minority attack detection. Experimental results demonstrate that AE-SAC achieves an accuracy of 84.15%, an F1-score of 83.97% on the NSL-KDD dataset and exceeds 98.9% for both metrics on the AWID dataset. Similarly, Bamber *et al.* [12] emphasized optimizing feature selection through recursive feature elimination combined with a DT classifier and evaluated multiple DL architectures, with the convolutional neural network–long short-term memory model outperforming others by achieving 95% accuracy, 0.89 recall, and a 0.94 F1-score on the NSL-KDD dataset. However, these two studies exhibit certain limitations related to model accuracy, processing time, and varying data split ratios between training and testing phases.

To effectively mitigate targeted attacks on server infrastructures, the systematic collection and analysis of log files play a pivotal role in anomaly detection and tracing attack behavior chains. As demonstrated in the study by Artioli *et al.* [13], advanced Security Information and event management systems can be enhanced by integrating natural language processing-based classifiers trained on synthetic log datasets, such as Siem Ingesting EVernts, which are generated using the semantic augmentation technique known as semantic perturbation and instantiation for content enrichment. The SVM model maintained consistent performance across both synthetic and real-world logs (Macro-F1: 0.9477–0.9636). In contrast,

although the bidirectional encoder representations from transformers (BERT) model—a pre-trained DL model widely used in text classification—achieved high performance on synthetic data (Macro-F1: 0.9528–0.9730), it exhibited limited generalization capability when applied to real-world logs (Macro-F1: 0.8864–0.9182). To further support anomaly detection and classification in IDS systems, a context-aware logging and advanced log analysis framework—Semantic-aware generator—has been proposed in [14]. Meanwhile, the studies [15], [16] emphasize the role of policies and log analysis in distributed firewalls, proposing a data mining and ML approach to detect anomalies from large-scale logs collected in real-world environments. At the same time, Brighenti and Valenza [17] emphasize that optimizing firewall configurations to reduce energy consumption while maintaining cybersecurity is essential for enhancing system sustainability. Similarly, the study in [18] proposes a semi-automated approach for firewall anomaly detection and resolution, which automatically addresses sub-optimizations while involving human intervention for conflict management, thereby reducing the workload of administrators. Subsequently, Park *et al.* [19] developed a visualization tool to assist administrators in monitoring and managing cybersecurity incidents, while also performing anomaly classification within firewall policies. Additionally, the research in [20] addresses effective and standardized alarm rationalization for cybersecurity monitoring.

Sharma *et al.* [21] made a significant contribution by proposing an optimized solution for firewall packet classification using advanced ensemble models, applied to a large dataset consisting of 65,532 log entries with four firewall action labels (accept, drop, reject, TCP reset). The study employed both voting and stacking ensemble models based on five popular ML algorithms, with the stacking model using a RF as the meta-classifier, achieving an accuracy of 99.8% and a precision of 91%. However, this research mainly focused on classification using static log data and did not thoroughly explore aspects such as real-time processing capabilities and latency, which remain important areas for further investigation.

Efeoglu and Tuna [22] evaluated multiple classification algorithms on a firewall log dataset comprising 65,532 entries with 12 attributes, using the action field as the target class. Simple cart and Naive Bayes (NB) tree achieved the highest accuracy (99.84%), while decision stump performed the worst (79.68%). To address class imbalance, the study employed the Matthews correlation coefficient for more robust evaluation. However, the approach lacks real-time processing capabilities. In contrast, Mingze [23] contributed a firewall log visualization system that achieved 98.3% accuracy, 92.1% precision, 97.5% recall, 98.1% F1-score, and 91.2% real-time performance in experimental evaluations. To clearly position our work in relation to existing research, Table 1 summarizes the key approaches, advantages, and limitations of the related studies discussed.

Table 1. Summary of related work on firewall log analysis and intrusion detection

Study/Reference	Dataset used	Algorithms	Best accuracy	Training/testing time (s)	Real-time capability
Aljabri <i>et al.</i> [8]	65,532 firewall logs	RF, k-nearest neighbors (KNN), NB, J48, artificial neural network (ANN)	RF: Accuracy 99.64%	Not reported	Not supported
Rahman <i>et al.</i> [10]	65,532 firewall logs	RF (others not specified)	RF: F1-score 99%	Not reported	Not supported
Efeoglu and Tuna [22]	65,532 firewall logs	Simple cart, NB tree, FT tree, J48, BF tree, Decision stump	Simple cart: Accuracy 99.84%	Not reported	Not supported
Mingze [23]	65,532 firewall logs	IG-based feature selection, visualization-based classification	Accuracy 98.3%, F1-score 98.1%	Not reported	Supported
Li <i>et al.</i> [11]	NSL-KDD	AE-SAC (adversarial Env + SAC RL)	Accuracy: 84.15%	Not reported	Limited performance on NSL-KDD; not tested on firewall logs
Bamber <i>et al.</i> [12]	NSL-KDD	ANN, long short-term memory (LSTM), bidirectional LSTM (BiLSTM), gated recurrent unit (GRU), bidirectional GRU (BiGRU), convolutional neural network (CNN)-LSTM	CNN-LSTM: Accuracy 95%, Recall 89%, F1-score 94%	Not reported	Not applied to real logs; lacks real-time processing
Our study	65,532 firewall logs	DT, RF, ET, CatBoost, AdaBoost; Proposed: RT-FLID	Accuracy (Train/Test): 99.86% / 99.45%; Precision (Train/Test): 99.20% / 97.46%; Recall (Train/Test): 99.25% / 93.39%	Train: 0.20642s Test: 0.05292 s	RT-FLID: lightweight, multithreaded, real-time capable; detects distributed denial-of-service (DDoS)/port scan; effective on real logs

The major contributions of this paper are:

- a. A real-time detection algorithm is proposed to identify abnormal increases in total packet size within server traffic, based on key statistical characteristics extracted from the dataset. Additionally, the study utilizes dataset [8], publicly available via [24], and demonstrates superior accuracy and real-time applicability compared to prior works [21], [22] and [23], particularly in practical server environments.
- b. Five ML algorithms—DT, RF, ET, CB, and AB—were evaluated based on multiple performance metrics, including accuracy, precision, training/testing time, and confusion matrix, across diverse train-test ratios ranging from 1:9 to 9:1. Among them, the DT model demonstrated superior accuracy and consistent performance, leading to its selection as the final prediction model.
- c. The experimental results revealed that a 7:3 train-test split provided the most balanced trade-off between training sufficiency and testing reliability. Furthermore, Pearson correlation coefficient analysis was incorporated to assess the relevance of input features, supporting effective feature selection and enhancing overall model performance.
- d. The proposed experimental system presents a novel multi-threaded network intrusion detection architecture that leverages parallel machine learning-based prediction to achieve real-time and scalable threat detection. By integrating efficient flow aggregation, synchronized multi-threaded processing, and automated resource management, the experimental implementation significantly enhances detection accuracy and operational performance compared to traditional IDS solutions. This architecture is particularly well-suited for dynamic and high-throughput network environments, providing a robust foundation for adaptive and intelligent network security.

These limitations reveal a significant research gap in developing real-time, lightweight intrusion detection systems that can effectively operate in dynamic and high-volume network environments. The motivation for this study stems from the growing volume and complexity of modern network traffic, which presents significant challenges for timely and accurate intrusion detection. Traditional methods often rely on deep packet inspection or computationally intensive techniques, which are not suitable for high-speed, resource-constrained environments. Meanwhile, firewall logs—although readily available—remain underutilized for real-time threat detection. To address these limitations, we propose RT-FLID, a lightweight and scalable intrusion detection system that leverages machine learning and flow-level log analysis to detect anomalies such as DDoS and port scanning efficiently in live network environments.

The structure of the paper is as follows: section 2 outlines the proposed algorithm, including the detection model architecture and the real-time network traffic anomaly detection algorithm. Section 3 describes the evaluation methods and dataset, covering the firewall logs dataset, evaluation metrics and statistical techniques, as well as dataset splitting strategies for model assessment. Section 4 presents the results and discussion, focusing on the performance evaluation of ML-based intrusion detection models using firewall logs, the experimental validation of real-time processing on a live server environment, and an overall discussion of key findings. Finally, section 5 concludes the study and provides acknowledgements.

2. PROPOSED ALGORITHM

The proposed algorithm introduces a real-time, flow-level network intrusion detection approach that leverages machine learning for accurate and adaptive traffic classification. The detection model architecture is organized into seven interconnected phases, encompassing packet acquisition, flow aggregation, thread synchronization, parallel background processing, multi-threaded prediction, result aggregation, and automated flow management. This design enables the system to efficiently process high-volume network traffic, promptly identify both known and unknown threats, and maintain robust performance under dynamic network conditions. The RT-FLID algorithm integrates synchronized feature extraction and parallel ML inference to deliver timely and reliable anomaly detection for modern network environments.

2.1. Detection model architecture

As illustrated in Figure 1, the proposed multi-threaded network intrusion detection system is architected as a sequence of interconnected processing phases, each responsible for a distinct functional aspect of real-time traffic analysis and threat detection. By decomposing the system into well-defined phases, we ensure modularity, scalability, and efficient resource utilization throughout the detection pipeline. The following sections detail each phase of the system, from initial packet acquisition to parallelized machine learning-based prediction and automated flow management.

Phase 1: Packet capture and processing

The system initiates with real-time packet sniffing on the server's network interface (e.g., *Tailscale0*). Each incoming transmission control protocol/user datagram protocol (TCP/UDP) packet is parsed to extract metadata, including source IP, destination port, protocol type, packet size, and TCP flags

(e.g., SYN). Flows are uniquely identified by the tuple (source IP, destination port), enabling granular behavioral tracking.

Phase 2: Flow aggregation and statistics compilation

Extracted flow metadata is aggregated into a structured dictionary, where each entry stores cumulative statistics:

- Total packets: Count of packets per flow.
- Total data volume: Sum of payload sizes (bytes).
- SYN counter (*pktsent*): Number of SYN packets (indicative of connection attempts).
- Last activity timestamp: Time since the last packet in the flow.

This phase ensures continuous monitoring of flow behavior while minimizing redundant computations.

Phase 3: Thread synchronization

A thread lock safeguards concurrent access to the shared aggregation dictionary. This prevents race conditions during parallel updates by the packet processing thread (Phase 1) and background maintenance threads (Phase 4), ensuring data integrity.

Phase 4: Parallel background threads

Three dedicated threads operate concurrently:

- Packet processing thread: Core thread for continuous packet capture and flow updates.
- Summary and prediction thread: At fixed intervals (e.g., 60 seconds), this thread triggers flow analysis, invokes parallel ML predictions via a ThreadPoolExecutor, and generates intrusion reports.
- Flow cleanup thread: Periodically purges inactive flows (e.g., no packets within 60 seconds) to optimize memory usage.

Phase 5: Multi-threaded ML prediction

During each summary cycle, flow features (dport, total data, packet count, pktsent) are normalized using a pre-trained scaler and fed into a machine learning model. Predictions (Allow/Deny) are executed in parallel across a thread pool, leveraging multi-core CPUs to maintain low latency under high traffic loads.

Phase 6: Result consolidation and reporting

Prediction results are aggregated into a human-readable summary, highlighting suspicious flows (e.g., high DENY rates). Alerts are logged or forwarded to security tools for automated mitigation.

Phase 7: Flow aging and cleanup

Inactive flows are automatically evicted from the aggregation dictionary, ensuring the system remains lightweight and responsive to evolving traffic patterns.

The architecture in Figure 1 visually encapsulates these phases, emphasizing the parallelized workflow between packet processing, ML inference, and resource management. The ThreadPoolExecutor's role in enabling concurrent predictions is a critical innovation, distinguishing this system from conventional single-threaded IDS solutions. This phased design not only enhances detection accuracy through ML-driven analysis but also ensures operational efficiency in dynamic network environments.

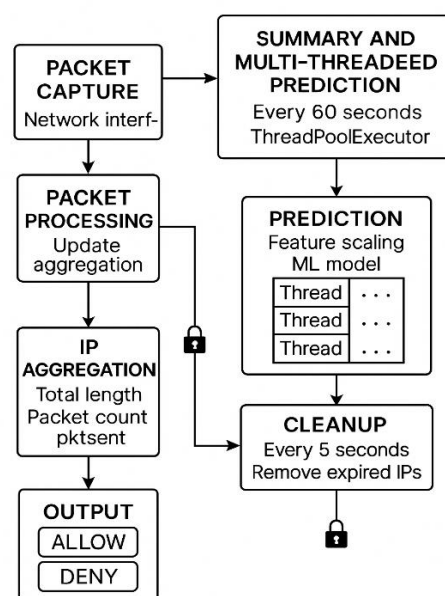


Figure 1. Architecture of a multi-threaded network IDS with parallel machine learning-based prediction

2.2. Proposed real-time network traffic anomaly detection algorithm

To address the challenge of detecting malicious traffic in real-time, we propose a lightweight yet effective algorithm that operates directly at the flow level. The proposed approach leverages a pre-trained machine learning classification model and maintains minimal in-memory statistics to achieve both low latency and high throughput. The algorithm, named real-time flow-level intrusion detection (RT-FLID), is designed to process live packets, extract essential flow-level features, and periodically classify traffic behavior without interrupting the ongoing packet capture process.

Algorithm 1. RT-FLID – real-time flow-level intrusion detection using ML classification

Input: Incoming packets captured on network interface

Output: Traffic label $\in \{\text{ALLOW}, \text{DENY}\}$

```

1: Initialize model ← Load classification model from joblib
2: Initialize scaler ← Load feature scaler from joblib
3: Initialize ip_aggregation as empty dictionary
4: Start background thread to execute Print_Summary_Every_60s()
5: Start background thread to execute Cleanup_Expired_IPs()
6: Set my_ip ← Get IP of monitored interface
7: while True do
8:   pkt ← Capture next incoming packet from network interface
9:   if pkt contains IP and (TCP or UDP) then
10:    src ← Source IP address of pkt
11:    dst ← Destination IP address of pkt
12:    dport ← Destination port from TCP/UDP header
13:    size ← Length of pkt in bytes
14:    timestamp ← Current time
15:    if dst=my_ip then
16:      ip_key ← (src, dport)
17:      Acquire Lock
18:      if ip_key not in ip_aggregation then
19:        ip_aggregation[ip_key] ← {total_length: size, packet_count: 1, pktsent: 0,
last_update: timestamp}
20:      if pkt is TCP and SYN flag is set and ACK is not set then
21:        ip_aggregation[ip_key].pktsent ← 1
22:      end if
23:    else
24:      Update total_length, packet_count, last_update
25:      if pkt is TCP and SYN flag is set and ACK is not set then
26:        ip_aggregation[ip_key].pktsent ← pktsent + 1
27:      end if
28:    end if
29:    Release Lock
30:  end if
31: end if
32: end while

Procedure: Print_Summary_Every_60s()
1: while True do
2:   Sleep 60 seconds
3:   Acquire Lock
4:   for each (ip_key, data) in ip_aggregation do
5:     if packet_count > 0 then
6:       features ← [dport, total_length, packet_count, pktsent]
7:       Scale features using scaler
8:       prediction ← model.predict(features)
9:       label ← "Allow" if prediction=1 else "Deny"
10:      Print prediction with src IP and dport
11:    end if
12:  end for
13:  Release Lock
14: end while

Procedure: Cleanup_Expired_IPs()
1: while True do
2:   Sleep 5 seconds
3:   Get current_time
4:   Acquire Lock
5:   for each (ip_key, data) in ip_aggregation do
6:     if current_time - last_update > 60 then
7:       Remove ip_key
8:       Print timeout message
9:     end if
10:  end for
11:  Release Lock
12: end while

```

RT-FLID is a proposed real-time detection algorithm that leverages ML to classify network traffic based on aggregated flow statistics. The system continuously captures packets from the monitored interface and maintains per-flow statistics based on the tuple (source IP, destination port). Every 60 seconds, these statistics are converted into feature vectors, scaled, and classified using a pretrained ML model to determine whether the traffic should be allowed or denied.

- Initialization phase (Lines 1–6): The system loads the pretrained ML model and corresponding scaler using `joblib`. It initializes a dictionary (*ip_aggregation*) to store flow statistics and launches two background threads: one for periodic prediction and one for garbage collection of stale flow entries.
- Real-time packet processing loop (Lines 7–32): Each incoming packet is analyzed if it includes IP and transport-layer headers (TCP or UDP). If the destination matches the monitored host, the flow is identified by its (source IP, destination port) tuple. The algorithm then updates the total bytes, packet count, and SYN packet count, maintaining a lock to ensure thread safety.
- Prediction procedure – *Print_Summary_Every_60s()* (Lines 1–14): Every 60 seconds, the system iterates over active IP flows, extracts feature vectors [`dport`, `total_length`, `packet_count`, `pktsent`], applies scaling, and performs classification using the trained model. Each flow is then labeled as Allow or Deny based on the prediction result. The outcome is printed for monitoring and logging purposes, supporting real-time intrusion detection and decision auditing.
- Cleanup procedure – *Cleanup_Expired_IPs()* (Lines 1–12): To maintain memory efficiency and prevent resource exhaustion, this background thread periodically (every 5 seconds) checks for inactive IP flows in the *ip_aggregation* structure. If a flow has not been updated for more than 60 seconds, it is removed from memory, and a timeout message is logged. This mechanism ensures optimal performance and scalability of the real-time intrusion detection system.

Overall, RT-FLID achieves a balance between low-latency packet handling and batch-based ML classification, making it suitable for lightweight deployment in real-time intrusion detection on resource-constrained systems.

3. EVALUATION METHODS AND DATASET

In this section, we utilize a firewall log dataset collected from a real-world environment, which contains detailed information about inbound and outbound packets as well as network access behaviors. The performance of the models is evaluated using quantitative metrics, including confusion matrix (CMA), accuracy (Acc), precision (Pre), and recall (Rec), along with training and prediction time. These are combined with statistical analysis techniques to ensure the reliability and significance of the results. The dataset is partitioned using various training/testing ratios to examine the stability and generalization capabilities of the models across different data distribution scenarios. The integration of quantitative evaluation metrics with statistical validation enhances the objectivity and credibility of the abnormal traffic detection assessment.

3.1. Firewall logs dataset

This dataset collected events from the firewall device in the attack detection system [8] [24], and the dataset was named “firewall logs.” The firewall device is used to collect logs of events that occur during the process of preventing attacks such as DDoS, and phishing. The firewall logs dataset contained 65,532 records. The number of corresponding actions for each type is presented in Table 2, in which the Action attribute was classified into four labels: “Allow” (3), “Deny” (2), “Drop” (1), and “Reset-Both” (0). These actions are directly related to granting or denying access to resources from the external network to the internal network of the firewall system. Additionally, the dataset includes 12 different attributes, as described in Table 3.

3.2. Machine learning models

Machine learning algorithms were selected in this study due to their efficiency, interpretability, and strong generalization capabilities across diverse data distributions. These characteristics make ML particularly suitable for processing complex, high-volume traffic logs in real-time network environments. To comprehensively evaluate their applicability to anomaly detection in firewall logs, five widely-used ML algorithms were systematically selected and assessed:

- Decision tree: A simple yet effective tree-based model that splits data based on feature thresholds, offering high interpretability and fast training time.
- Random forest: An ensemble of decision trees that improves classification performance and reduces overfitting through bootstrap aggregation (bagging).
- Extra trees: A randomized ensemble method similar to Random Forest, which introduces additional randomness in feature splits to enhance computational speed and robustness.

- CatBoost: A gradient boosting algorithm optimized for handling categorical features efficiently, offering competitive accuracy with minimal preprocessing.
- AdaBoost: An adaptive boosting technique that iteratively focuses on misclassified instances to construct a strong composite classifier from multiple weak learners.

The five machine learning models were evaluated across nine distinct train–test split scenarios (ranging from 1:9 to 9:1). Their performance was assessed using standard evaluation metrics, including Acc, Pre, Rec, CMA, and training/prediction time (in seconds). Pearson correlation coefficient analysis was also applied to examine feature relevance and support effective feature selection. Among all evaluated models, the decision tree consistently achieved the highest Acc and exhibited the most stable performance across all experimental configurations, leading to its selection as the final prediction model for abnormal traffic detection in server systems.

Table 2. Firewall logs dataset

Action type	No. of actions
Allow	37640
Deny	14987
Drop	12851
Reset-both	54
Total	65532

Table 3. Attributes in the dataset

No.	Attribute	Description
1	Source port	Port to which the packet is sent from the source machine, integer data type [0 to 65535]
2	Destination port	The gateway on the destination device knows which application or service the packet needs to be delivered to, integer data type [0 to 65535] (e.g., requesting access to a website over HTTPS; it is 443)
3	NAT source port	Network address translation (NAT) source port is the conversion from source port (e.g., internal, 22354) of internal IP (e.g., 192.168.1.100) to source port (e.g., After NAT, 56676) of public IP (e.g., 203.0.113.1)
4	NAT destination port	NAT Destination port is changing from destination port (Outside; e.g., port 80) to destination port (Internal; port 8080), and forwarding the packet to the internal server (e.g., 192.168.1.10:8080)
5	Action	The IPS system detects and processes traffic. Allow: Allow normal connections (classified as “3”). Deny: Reject the connection with notification to the sender (“2”). Drop: Block traffic suspected to be malicious (“1”). Reset-Both: Terminate any connection detected as malicious (“0”).
6	Bytes	Sum of bytes sent and bytes received (e.g., 177 bytes=94 bytes + 83 bytes)
7	Bytes sent	Total data (in bytes) sent over the network by a device or application (e.g., 94 bytes)
8	Bytes received	Total data (in bytes) received from the network by the device or application (e.g., 83 bytes)
9	Packets	Sum of packets sent and packets received; Each packet is 1,500 bytes in size (Ethernet standard)
10	Elapsed time (sec)	The firewall will record the elapsed time (in seconds) to track the lifetime of each connection.
11	Pkts_Sent (packets sent)	Total number of packets sent by the device or application over the network.
12	Pkts_Received (packets received)	Total number of packets received by the device or application from the network.

3.3. Evaluation metrics and statistical methods

To assess the classification performance of the proposed ML models, several standard evaluation metrics were employed, including the CMA, Acc, Pre, and Rec, as described in [25]–[27]. These metrics allow for a detailed analysis of both overall performance and error distribution across classes. In particular, the confusion matrix, as presented in (1), captures the correspondence between predicted and ground truth labels in the classification process.

$$CMA = [TN \quad FP \quad FN \quad TP] \quad (1)$$

The confusion matrix provides four essential quantities for binary classification analysis. True positive (*TP*) represents correctly identified malicious instances (e.g., drop, deny, reset-both), while True Negative (*TN*) corresponds to correctly identified benign instances (e.g., allow). In contrast, false positive (*FP*) and false negative (*FN*) represent misclassifications of benign and malicious traffic, respectively, and serve as the foundation for further metric calculations.

$$Acc = \frac{TP+TN}{TP + TN + FP + FN} \quad (2)$$

$$Pre = \frac{TP}{TP + FP} \quad (3)$$

$$Rec = \frac{TP}{TP + FN} \quad (4)$$

Acc measures the overall correctness of predictions. Pre quantifies the proportion of positive identifications that were actually correct, while Rec (or true positive rate) indicates the model's ability to identify positive instances. A high recall is especially important in security contexts, where failing to detect a malicious instance can be costly.

The Pearson correlation coefficient [28] is a statistical measure used to quantify the strength and direction of the linear relationship between two continuous variables. It is frequently used in ML and data analysis for feature selection, where identifying strong correlations can improve model interpretability and performance. For example, Chen *et al.* [29] utilized this coefficient to assess the importance of input variables in reliability analysis and predictive modeling. The coefficient is computed as (5):

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (5)$$

The average values of variables x and y are computed as $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$ and $\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$, respectively. In these formulas, x_i and y_i denote the values of x and y at the i -th observation, and n represents the total number of observations. Furthermore, the Pearson correlation coefficient r quantifies the degree of linear association between the two variables, where $r = 1$ indicates a perfect positive correlation, $r = -1$ indicates a perfect negative correlation, and $r = 0$ signifies no linear correlation.

3.4. Dataset splitting for model evaluation

Equation (6) was used to define multiple training–testing splits in order to evaluate model performance metrics, including Acc, Pre, and Rec, across five ML algorithms: DT, RF, ET, CB, and AB. In each simulation scenario i , the test size refers to the proportion of the dataset reserved for testing, while the training size is the complement of that fraction, as illustrated in Figure 2. This ratio-based splitting approach enabled the identification of an optimal data partitioning scheme for predictive modeling.

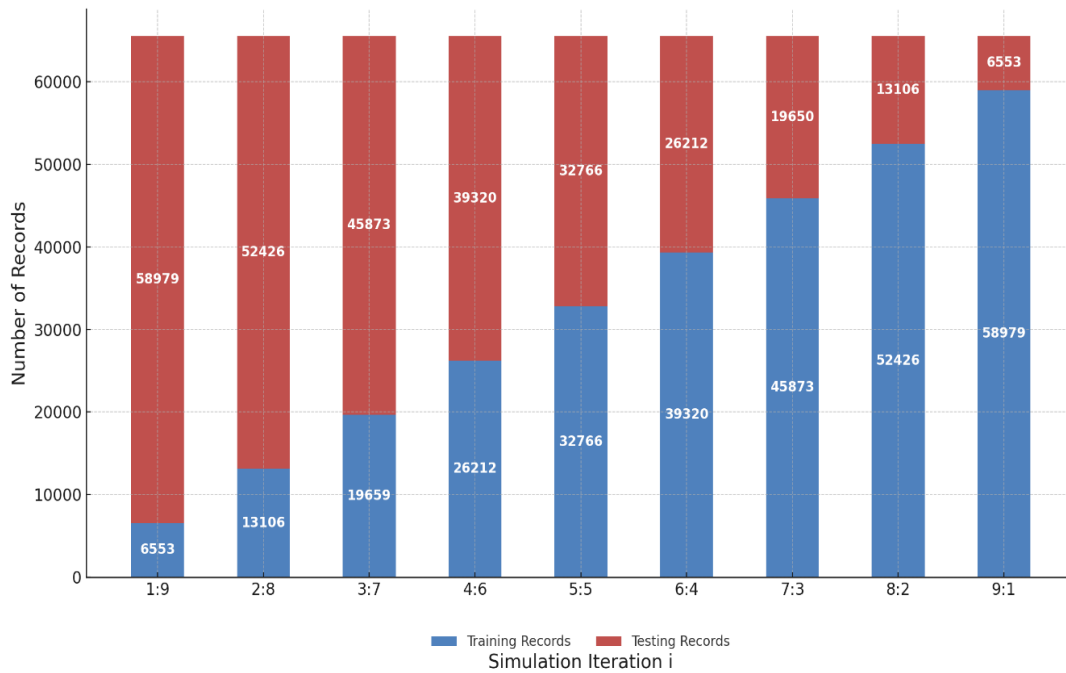


Figure 2. Distribution of training and testing data proportions across nine simulation scenarios using the firewall logs dataset (65,532 records)

$$\text{Testing}(i) = 1 - \text{Training}(i) \quad (6)$$

where $i \in \{1, 2, \dots, 9\}$ represents different simulation configurations. The Firewall logs dataset, consisting of 65,532 records, was divided accordingly into training and testing subsets. As the training portion increases across simulations, the corresponding testing portion decreases, allowing an empirical examination of how model performance varies with data availability. The computation time for training or testing in each scenario was calculated using (7), by taking the difference between the end and start timestamps:

$$\Delta t = t2 - t1 \quad (7)$$

where Δt denotes the elapsed time, $t1$ is the start time, and $t2$ is the end time.

The training and testing data proportions used in this study are illustrated in Figure 2, based on the firewall logs dataset, which comprises 65,532 records. Across nine simulation iterations (from 1 to 9), the proportion of training data progressively increased, while the testing data proportion decreased correspondingly. Each iteration corresponded to a specific train–test ratio, ranging from 1:9 to 9:1. These data partitioning schemes were employed to investigate the effect of different data splits on the performance of classification models.

4. RESULTS AND DISCUSSION

The experimental assessments were conducted on a high-performance computing server equipped with dual Intel Xeon E5-2696 v3 processors (2.30 GHz, 36 cores/72 threads) and 64 GB of DDR4 RAM, powered by an ASUS TUF 1200 W Gold ATX 3.0 power supply to ensure operational stability during intensive workloads. Although the system was equipped with an NVIDIA GeForce RTX 3090 XC3 Ultra Hybrid GPU (24 GB GDDR6X, 10,496 CUDA cores), all training and inference tasks were executed on the CPU, as the study focused exclusively on ML models rather than DL architectures. DL models typically require larger datasets and more computational resources, which may not be feasible or necessary for the problem of detecting abnormal traffic in server systems. Therefore, ML algorithms were preferred for their efficiency and interpretability in this study. Five ML algorithms—DT, RF, ET, CB, and AB—were systematically evaluated across nine train–test ratio scenarios (ranging from 1:9 to 9:1). Model performance was assessed using standard metrics, including *Acc*, *Pre*, *Rec*, and *CMA* analysis, training and testing times (measured in seconds) of the five algorithms, as well as the evaluation of the correlation matrix. Among these models, DT consistently demonstrated the highest accuracy and most stable performance, leading to its selection as the final prediction model in this study on detecting abnormal traffic in server systems.

4.1. Assessing ML-based intrusion detection models using firewall logs

As shown in Table 4, DT consistently achieved the highest and most stable accuracy across all train–test splits, with training accuracy ranging from 99.862% to 99.893% and test accuracy from 98.835% to 99.45%. At the 7:3 split, DT attained its peak test accuracy of 99.45%, outperforming ET (99.389%), CB (99.338%), RF (95.005%), and AB (95.005%). While ET and CB remained competitive (e.g., 99.374% and 99.343% at the 9:1 and 8:2 splits), their performance was consistently lower than that of DT. In contrast, RF and AB yielded noticeably lower test accuracies, particularly under minimal training conditions (e.g., both at 96.108% with the 1:9 split). These results from Table 4 confirm that DT offers superior generalization and robustness across varying data distributions for detecting abnormal traffic in server systems, and Figure 3 further reinforces this finding by comparing the accuracy of the five algorithms at the 7:3 training–testing ratio, where DT demonstrates the best performance.

Table 4. Accuracies of the five algorithms across different train–test splits.

Training– testing ratio	DT		RF		ET		CB		AB	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
1:9	0.99862	0.98835	0.96307	0.96108	0.99862	0.98904	0.99603	0.98986	0.96307	0.96108
2:8	0.99885	0.99078	0.96238	0.96269	0.99885	0.99048	0.99427	0.99113	0.96238	0.96269
3:7	0.99893	0.99217	0.97670	0.97680	0.99893	0.99125	0.99394	0.99193	0.97670	0.97680
4:6	0.99881	0.99282	0.97688	0.97723	0.99881	0.99231	0.99393	0.99249	0.97688	0.97723
5:5	0.99874	0.99328	0.96267	0.96337	0.99874	0.99230	0.99389	0.99252	0.96267	0.96337
6:4	0.99877	0.99355	0.96431	0.96455	0.99877	0.99298	0.99387	0.99248	0.96431	0.96455
7:3	0.99875	0.9945	0.94920	0.95005	0.99875	0.99389	0.99359	0.99338	0.94920	0.95005
8:2	0.9987	0.99427	0.96467	0.96459	0.99870	0.99435	0.99359	0.99343	0.96467	0.96459
9:1	0.99871	0.99359	0.97765	0.97741	0.99871	0.99374	0.99370	0.99267	0.97765	0.97741

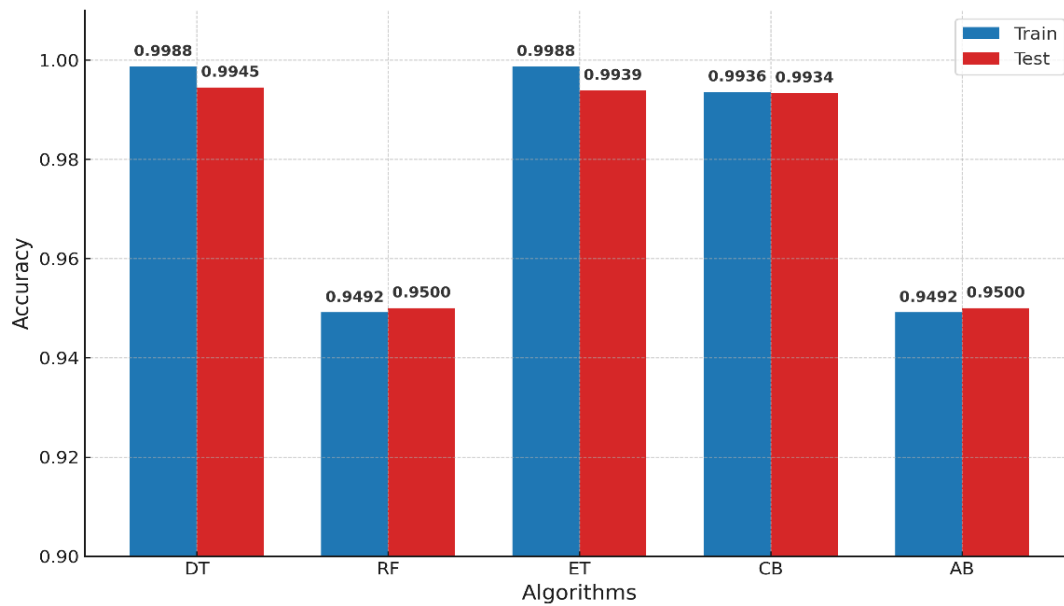


Figure 3. Comparison of the accuracy of the five algorithms at the 7:3 training–testing ratio

According to the results presented in Table 5, DT consistently achieved the highest training precision across all train–test ratios, ranging from 99.2% to 99.875%. Although its test precision initially started lower (74.985% at the 1:9 split), it improved significantly to 99.455% at the 8:2 split, demonstrating strong generalization as more training data was introduced. In comparison, RF and AB exhibited limited improvement, with test precision remaining below 72.556%. ET and CB showed stronger performance gains—CB reached 99.341% and ET achieved 99.472% at the 8:2 split—yet they still trailed DT in overall consistency. These findings from Table 5 further affirm DT’s robustness and adaptability in detecting abnormal traffic within server systems under varying data conditions.

Table 5. Comparison of precision among the five algorithms after training and testing

Training– testing ratio	DT		RF		ET		CB		AB	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
1:9	0.99839	0.74985	0.71130	0.71027	0.99839	0.83987	0.99616	0.95504	0.71130	0.71027
2:8	0.99865	0.85029	0.71102	0.71141	0.99865	0.92880	0.99435	0.94169	0.71102	0.71141
3:7	0.99875	0.92151	0.72522	0.72561	0.99875	0.92941	0.99393	0.97656	0.72522	0.72561
4:6	0.9986	0.87677	0.72515	0.72583	0.99860	0.91811	0.99390	0.97014	0.72515	0.72583
5:5	0.99855	0.88435	0.70974	0.71077	0.99855	0.91005	0.99394	0.96174	0.70974	0.71077
6:4	0.99858	0.91549	0.71185	0.71221	0.99858	0.94385	0.97837	0.96507	0.71185	0.71221
7:3	0.992	0.97547	0.71453	0.71496	0.99200	0.97865	0.97895	0.99341	0.71453	0.71496
8:2	0.99269	0.99455	0.72058	0.72055	0.99269	0.99472	0.95963	0.99343	0.72058	0.72055
9:1	0.99342	0.99393	0.72554	0.72523	0.99342	0.99418	0.98377	0.99287	0.72554	0.72523

As detailed in Table 6, DT consistently achieved the highest recall on the training data, ranging from 98.982% to 99.923%, and demonstrated notable test recall performance, peaking at 93.389% with the 7:3 split. In contrast, RF and AB maintained relatively low recall on test sets across all splits, remaining around 70.000% to 73.000%. ET displayed improved test recall, reaching 98.020% at the 7:3 split, whereas CB demonstrated stable performance, with test recall gradually increasing to 87.492% at the 5:5 split and 85.987% at the 8:2 split. Despite fluctuations, DT consistently outperformed the other methods in both training and test scenarios. These findings reinforce DT’s effectiveness in capturing true abnormal traffic patterns within server systems.

As shown in Table 7, DT consistently exhibited the lowest computational time across all train–test splits, with training times ranging from 0.05843 seconds (1:9) to 0.20642 seconds (7:3) and testing times from 0.05292 seconds (7:3) to 0.13196 seconds (4:6). RF and ET demonstrated moderate training and testing durations; for instance, RF’s training time increased from 0.23983 seconds (1:9) to 0.99801 seconds (9:1), while ET’s training time rose more steeply from 0.45540 to 2.30398 seconds, with its maximum test time reaching 2.03108 seconds at the 9:1 split. In contrast, CB incurred the highest training costs among all

models, starting at 6.43591 seconds (1:9) and climbing to 32.7983 seconds (9:1), although its testing time remained relatively stable, ranging from 0.41535 to 0.47654 seconds. AB maintained relatively low testing times, such as 0.4228 seconds at 7:3 and 0.43434 seconds at 9:1, but its training time still increased steadily from 0.22743 to 1.00326 seconds across the splits. These results highlight the computational efficiency of DT, particularly in scenarios requiring low-latency model deployment, while underscoring the substantial training overhead introduced by ensemble methods like CB and ET.

Table 6. Comparison of recall across the five algorithms

Training– testing ratio	DT		RF		ET		CB		AB	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
1:9	0.999	0.75748	0.72442	0.72411	0.99900	0.76396	0.99726	0.77439	0.72442	0.72411
2:8	0.99915	0.83671	0.72470	0.72532	0.99915	0.84865	0.86039	0.76803	0.72470	0.72532
3:7	0.99923	0.84353	0.73154	0.73178	0.99923	0.84321	0.88727	0.84392	0.73154	0.73178
4:6	0.9991	0.87774	0.73207	0.73223	0.99910	0.90896	0.86026	0.82359	0.73207	0.73223
5:5	0.98982	0.87479	0.72648	0.72712	0.98982	0.89277	0.88538	0.87492	0.72648	0.72712
6:4	0.99128	0.89313	0.72726	0.72816	0.99128	0.92690	0.86362	0.83633	0.72726	0.72816
7:3	0.99251	0.93389	0.70038	0.70127	0.99251	0.98020	0.85164	0.85552	0.70038	0.70127
8:2	0.99324	0.90537	0.71738	0.71724	0.99324	0.97339	0.85675	0.85987	0.71738	0.71724
9:1	0.99393	0.846	0.73299	0.73267	0.99393	0.84597	0.86882	0.79577	0.73299	0.73267

The correlation matrix in Figure 4 was utilized in Formula 5 to evaluate the linear relationships among the 12 attributes listed in Table 3. Pairwise correlations were analyzed, with values ranging from -1 to 1. A strong positive correlation was observed between “Bytes Sent” and “Packets” (correlation ≈ 1), suggesting that the number of bytes sent is closely related to the number of packets transmitted. Multicollinearity was also assessed; if two attributes exhibited a correlation coefficient above 0.8—such as “Pkts_Sent” and “Bytes Sent”—one of them could be excluded to reduce redundancy. Anomalies and potential data skewness were detected, as “Source Port” showed no meaningful correlation with traffic-related attributes (*e.g.*, “Bytes Sent”), indicating inconsistency. Overall, the correlation matrix supported both the identification of key attributes and the reduction of redundant features.

Table 7. Computational time (sec) for training and testing the five algorithms

Training– testing ratio	DT		RF		ET		CB		AB	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
1:9	0.05843	0.06757	0.23983	0.50174	0.45540	1.54212	6.43591	0.45376	0.22743	0.51638
2:8	0.06439	0.08054	0.31961	0.47798	0.63115	1.34037	10.5973	0.42859	0.32101	0.47017
3:7	0.18488	0.11959	0.43802	0.47426	0.68648	1.40431	13.2961	0.43802	0.41556	0.46706
4:6	0.18458	0.13196	0.53620	0.64084	0.94017	1.49161	17.9506	0.44295	0.77490	0.69974
5:5	0.09423	0.07615	0.89977	0.65402	1.12006	1.49527	19.3591	0.44001	0.81980	0.42654
6:4	0.14438	0.07434	0.71156	0.44066	2.02746	1.66075	23.2456	0.41535	0.71063	0.41697
7:3	0.20642	0.05292	0.80928	0.42493	1.54956	1.57220	25.6398	0.44855	0.85290	0.42280
8:2	0.1631	0.09987	0.91445	0.43844	1.77004	1.62620	28.3562	0.47654	0.90184	0.44737
9:1	0.14216	0.06657	0.99801	0.43490	2.30398	2.03108	32.7983	0.41856	1.00326	0.43434

Based on Figure 5, which presents the confusion matrices for the five classification algorithms under a 7:3 training–testing split, clear differences in classification performance are evident, particularly for sensitive classes such as reset-both and deny. The DT model demonstrated highly accurate classification across all classes, with 11,227 out of 11,292 reset-both samples correctly predicted and minimal confusion across other classes. ET performed comparably well, correctly identifying 11,219 reset-both instances and misclassifying only 73. In contrast, both RF and AB struggled significantly with the deny and reset-both categories. For example, RF misclassified 191 deny samples and 626 reset-both samples, indicating poor discrimination capability between those categories. CatBoost, while not as precise as DT or ET, maintained competitive accuracy with 11,206 correct predictions for reset-both and low misclassification rates for other classes. These results suggest that DT and ET offer superior robustness in distinguishing between nuanced traffic control actions, making them more suitable for detecting abnormal behaviors in server systems.

4.2. Experimental evaluation of real-time

Figure 6 presents a comprehensive real-time summary and prediction report generated by the proposed intrusion detection system, highlighting its operational effectiveness in a dynamic network environment. The system continuously aggregates flow-level statistics from multiple source IPs

(100.70.10.1–100.70.10.5), all targeting the server’s HTTP port (80), each exhibiting identical behavioral metrics—namely, 192 packets, 7680 bytes of data transferred, and 96 SYN packets per flow. This uniformity in traffic patterns is indicative of automated or coordinated activities, such as DDoS attacks or systematic port scanning, which are often challenging for traditional signature-based IDS to promptly identify. Leveraging machine learning-based classification, the system assigns a DENY label to all detected flows, as evidenced by the consistent “[Prediction] DENY– Src IP: ... -> Dst Port: 80” output, thereby demonstrating the model’s ability to accurately recognize and respond to anomalous and potentially malicious behaviors in real time. These experimental results underscore the robustness of the RT-FLID algorithm in synthesizing flow features and executing parallel inference, enabling timely and reliable defense against both known and emerging network threats—a capability that aligns with recent advancements in deep learning-based intrusion detection systems.

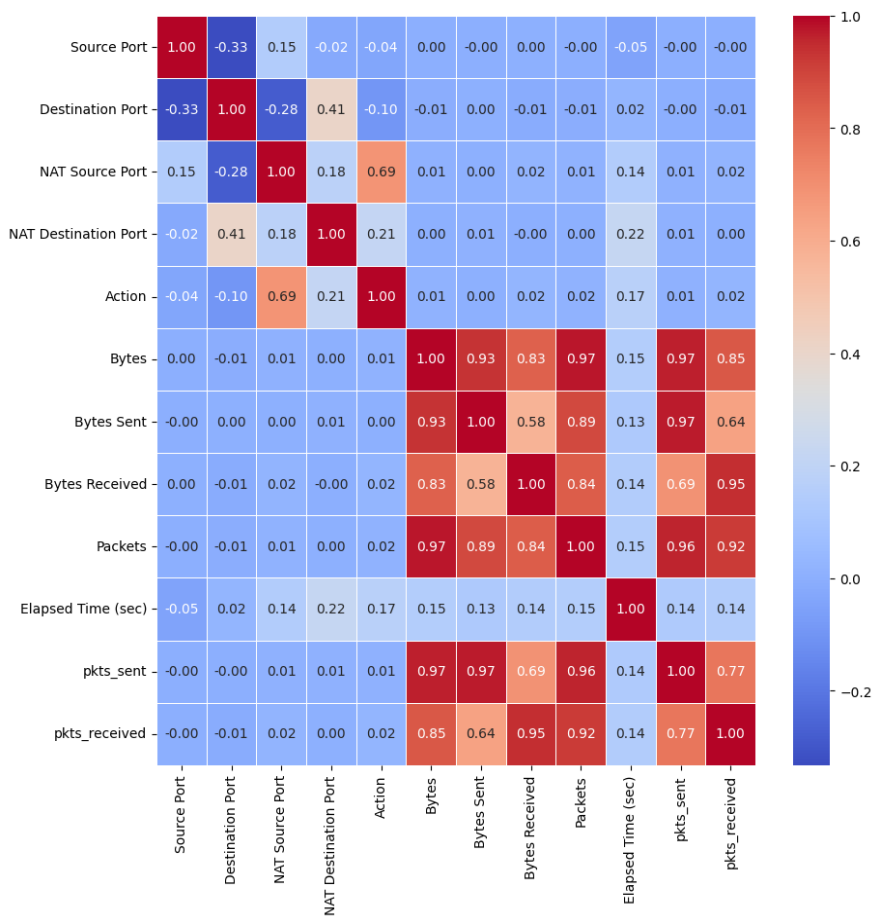


Figure 4. Correlation matrix showing the statistical relationships among 12 attributes in the dataset

Figure 7 presents the system status table containing a list of active nodes in the network, including IP addresses (e.g., 100.70.10.1, 100.70.10.2, 100.70.10.3, 100.70.10.4, 100.70.10.5), along with information on devices, users, operating systems, connection status (active, direct), and transmission statistics (tx/rx). These nodes act as attacking machines, generating abnormal traffic toward the victim machine with IP address 100.92.72.16 by using the HPing3 tool to simulate various attack patterns. The consistency between the IP addresses shown in this table and those labeled as DENY in the traffic summary report, see Figure 6, confirms that the intrusion detection system operates effectively not only on simulated data but also in a real-world network environment with multiple nodes and concurrent connections. Furthermore, the traffic statistics per node provide a solid basis for verifying the model's prediction decisions, thereby enhancing the reliability of the Allow/Deny labels. The combination of evidence from both figures demonstrates that the proposed RT-FLID algorithm is capable of accurately analyzing and classifying traffic in real time, while also proving its practical applicability in monitoring and protecting dynamic, multi-device network environments, including real-world attack scenarios.

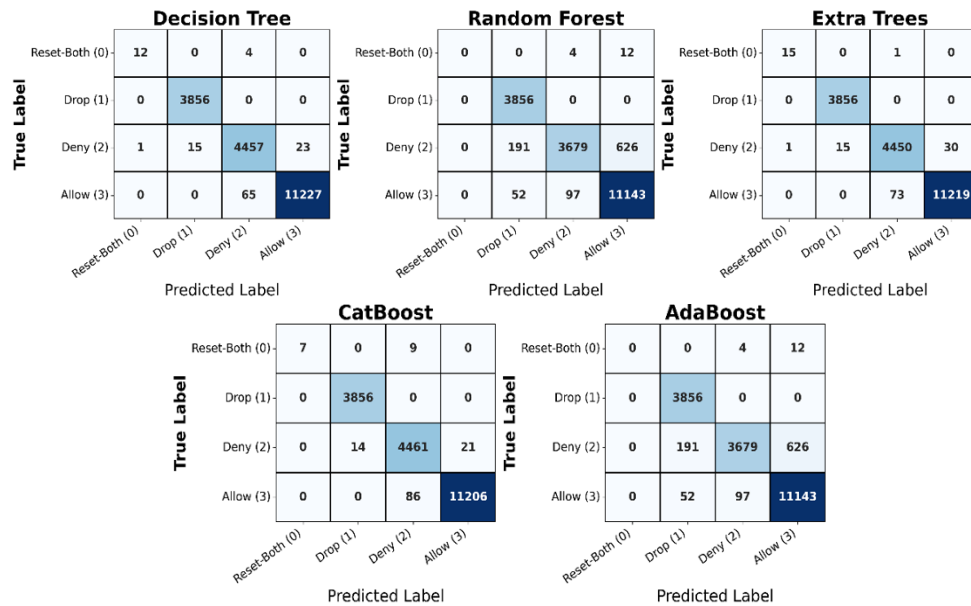


Figure 5. Confusion matrices illustrating the performance of five classification algorithms under a 7:3 training–testing split

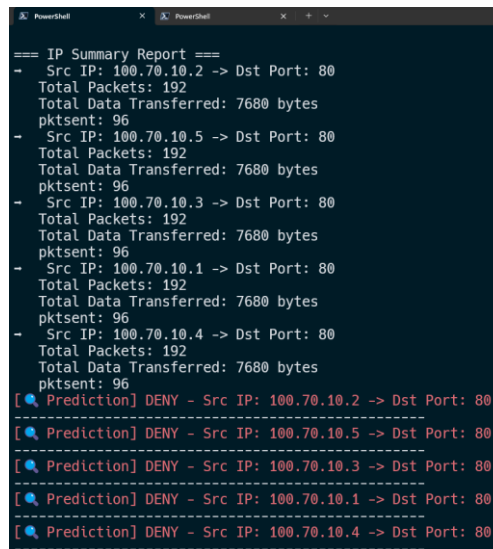


Figure 6. Real-time summary report and ML-based flow-level prediction results of the proposed intrusion detection system

4.3. Discussion

This study conducted a comprehensive evaluation of five machine learning algorithms (DT, RF, ET, CB, AB) across various training/testing data split ratios. The models were assessed using standard performance metrics such as accuracy, precision, recall, confusion matrix, training time, and prediction time. Among these, the DT model demonstrated the highest accuracy and stability, achieving a peak test accuracy of 99.45% at the 7:3 split ratio. Moreover, DT outperformed the others in processing speed, recording the shortest training time (only 0.20642 seconds at the 7:3 ratio) and rapid prediction times. These results highlight the model's strong potential for deployment in low-latency environments and serve as compelling evidence of DT's effectiveness and practicality in detecting abnormal traffic in server systems.

Experimental results indicate that a 7:3 training/testing split offers an optimal balance between providing sufficient training data and ensuring reliable test evaluation. At this ratio, most models—particularly the DT model—achieved the highest accuracy and recall scores, demonstrating strong

generalization capabilities. Specifically, DT reached a precision of up to 99.455% and a recall of 93.389%, significantly outperforming the other algorithms. This analysis provides valuable practical guidance for determining appropriate data split ratios in real-world network monitoring applications.

The proposed system, named RT-FLID, is a real-time intrusion detection architecture that employs a multi-threaded mechanism to concurrently process network traffic flows. The architecture integrates three core components: flow aggregation by session, synchronized multi-threaded processing, and automated resource management. As a result, the system significantly enhances detection speed, accuracy, and scalability, making it suitable for dynamic and high-throughput network environments. This represents a substantial contribution, especially considering the limitations of traditional IDS systems when confronted with distributed and sophisticated cyberattacks.

IP	Node Name	OS	Status
100.74.10.3	gcp04-vps03	thoainxbeta1@ linux	-
100.74.10.4	gcp04-vps04	thoainxbeta1@ linux	-
100.70.10.1	gcp07-vps01	thoainxbeta1@ linux	active; direct 35.189.183.64:16288, tx 26576 rx 45632
100.70.10.2	gcp07-vps02	thoainxbeta1@ linux	active; direct 35.234.39.23:8096, tx 25904 rx 45728
100.70.10.3	gcp07-vps03	thoainxbeta1@ linux	active; direct 35.229.190.161:46976, tx 26864 rx 45760
100.70.10.4	gcp07-vps04	thoainxbeta1@ linux	active; direct 35.194.200.189:7072, tx 28496 rx 49760
100.70.10.5	gcp07-vps05	thoainxbeta1@ linux	active; direct 35.234.40.120:8128, tx 25904 rx 45728
100.70.10.6	gcp07-vps06	thoainxbeta1@ linux	-
100.70.10.7	gcp07-vps07	thoainxbeta1@ linux	-
100.70.10.8	gcp07-vps08	thoainxbeta1@ linux	-
100.70.10.9	gcp07-vps09	thoainxbeta1@ linux	-
100.70.10.10	gcp07-vps10	thoainxbeta1@ linux	-
100.93.112.85	gcp08-vps01	thoainxbeta1@ linux	-
100.68.10.1	gcp6-vps01	thoainxbeta1@ linux	-
100.68.10.2	gcp6-vps02	thoainxbeta1@ linux	-
100.68.10.3	gcp6-vps03	thoainxbeta1@ linux	-
100.115.87.47	kuratajr	thoainxbeta1@ windows	-
100.119.71.21	nezha	thoainxbeta1@ linux	active; direct 38.114.121.162:14855, tx 6730696 rx 2105528
100.66.10.1	nx01-vps01	thoainxbeta1@ linux	-
100.66.10.3	nx01-vps03	thoainxbeta1@ linux	-
100.66.10.4	nx01-vps04	thoainxbeta1@ linux	-
100.66.10.5	nx01-vps05	thoainxbeta1@ linux	-
100.66.10.7	nx01-vps07	thoainxbeta1@ linux	-
100.66.10.8	nx01-vps08	thoainxbeta1@ linux	-
100.92.72.16	victim	thoainxbeta1@ linux	-

Figure 7. Active node and IP status table in the monitored network environment

Another key contribution of this study is the integration of quantitative evaluation with statistical analysis to enhance the reliability and objectivity of the results. In addition to conventional metrics such as accuracy, precision, recall, and training/prediction time, the study employs Pearson correlation analysis to effectively select input features, thereby optimizing model performance and reducing computational cost. More importantly, the system is validated on firewall log data collected from real-world network environments, demonstrating its ability to accurately identify abnormal behaviors such as DDoS and port scanning by correctly labeling suspicious flows (e.g., “DENY”), with the DT model consistently achieving high accuracy in critical classes like “deny” and “reset-both” as evidenced by the confusion matrix.

Although parallel processing can be resource-intensive and time-consuming in certain contexts, the application of ThreadPoolExecutor and multithreading in this study significantly reduces latency compared to sequential processing, while also enhancing the scalability of the system. This design choice represents a meaningful contribution and can be considered a breakthrough in the field of real-time network intrusion detection. Exploring this direction further in subsequent projects offers a promising avenue for developing more efficient and accurate solutions. The proposed approach demonstrates strong potential and is expected to deliver substantial value to the research community in both academic and practical domains.

5. CONCLUSION

This study thoroughly evaluated five machine learning algorithms—DT, RF, ET, CB, and AB—across various training/testing data splits. The DT model demonstrated superior performance, achieving a peak test accuracy of 99.45%, precision of 99.455%, and recall of 93.389% at the optimal 7:3 split ratio. It also outperformed others in efficiency, with the shortest training time of just 0.20642 seconds and rapid prediction speed, highlighting its suitability for low-latency environments. The proposed RT-FLID system, leveraging multi-threaded processing, significantly enhanced detection speed, accuracy, and scalability in real-time intrusion detection. Moreover, integrating Pearson correlation for feature selection optimized model performance and reduced computational costs. Validated on real-world firewall logs, the system accurately identified abnormal activities like DDoS and port scanning, with the DT model maintaining high accuracy in critical classes such as “deny” and “reset-both,” as confirmed by confusion matrix analysis.

While RT-FLID demonstrates strong real-time performance and scalability, several enhancements are planned for future research. These include exploring hybrid deep learning models for improved detection of complex attack patterns, enabling compatibility with encrypted traffic through flow-based behavioral analysis, and integrating external threat intelligence feeds to enhance anomaly context awareness. Additionally, extending the system to analyze logs from multiple sources (*e.g.*, IDS, system logs) could provide a more comprehensive security posture.

ACKNOWLEDGMENTS

The authors sincerely thank the Associate Editor, reviewers, and Editor-in-Chief for their valuable feedback. As a doctoral student, we appreciate the Posts and Telecommunications Institute of Technology, Ministry of Information and Communications of Vietnam, for their partial financial support through an International Scientific Research Grant.

FUNDING INFORMATION

This study was partially supported by the Theme-based Research Grant on International Science [grant number 999/QĐ-HV] from the Ministry of Information and Communications, Vietnam.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Tran Cong Hung	✓	✓			✓					✓		✓	✓	✓
Dam Minh Linh	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Han Minh Chau			✓			✓		✓		✓				✓
Ngo Xuan Thoai			✓					✓		✓				
Thai Duc Phuong										✓				
Huynh De Thu										✓				

- C : Conceptualization
M : Methodology
So : Software
Va : Validation
Fo : Formal analysis
- I : Investigation
R : Resources
D : Data Curation
O : Writing - Original Draft
E : Writing - Review & Editing
- Vi : Visualization
Su : Supervision
P : Project administration
Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study have been previously published and are openly available at GitHub: <https://github.com/MinhLinhEdu/Firewall-logs-dataset>.

REFERENCES

[1] Cisco Systems Inc., “Cisco’s 2024 cybersecurity readiness index,” *Cisco Systems Inc.*, 2024. <https://newsroom.cisco.com/c/t/newsroom/en/us/a/y2024/m03/cybersecurity-readiness-index-2024.html> (accessed Jul. 12, 2024).

[2] “FBI tech tuesday: protecting yourself from spoofing and phishing scams,” *Federal Bureau of Investigation*, 2020. <https://www.fbi.gov/contact-us/field-offices/phoenix/news/press-releases/fbi-tech-tuesday-protecting-yourself-from-spoofing-and-phishing-scams> (accessed Aug. 15, 2023).

[3] Microsoft, “Microsoft digital defense report 2024,” *Microsoft*, 2024. <https://www.microsoft.com/en-us/security/security-insider/intelligence-reports/microsoft-digital-defense-report-2024> (accessed Nov. 09, 2024).




[4] H. Dhrir, M. Charfeddine, N. Tarhouni, and H. M. Kammoun, “Machine learning- and deep learning-based anomaly detection in firewalls: a survey,” *Journal of Supercomputing*, vol. 81, no. 6, p. 761, 2025, doi: 10.1007/s11227-025-07212-y.

[5] A. Komadina, I. Kovačević, B. Stengl, and S. Groš, “Comparative analysis of anomaly detection approaches in firewall logs: Integrating light-weight synthesis of security logs and artificially generated attack detection,” *Sensors*, vol. 24, no. 8, p. 2636, Apr. 2024, doi: 10.3390/s24082636.




- [6] J.-K. Lee, T. Hong, and G. Lee, "AI-based approach to firewall rule refinement on high-performance computing service network," *Applied Sciences*, vol. 14, no. 11, p. 4373, May 2024, doi: 10.3390/app14114373.
- [7] A. Siddiqui, B. P. Rimal, M. Reisslein, D. Ge, and Y. Wang, "SUTMS: Designing a unified threat management system for home networks," *IEEE Access*, vol. 12, pp. 80930–80949, 2024, doi: 10.1109/ACCESS.2024.3410111.
- [8] M. Aljabri, A. A. Alahmadi, R. M. A. Mohammad, M. Aboulmour, D. M. Alomari, and S. H. Almotiri, "Classification of firewall log data using multiclass machine learning models," *Electronics (Switzerland)*, vol. 11, no. 12, p. 1851, 2022, doi: 10.3390/electronics11121851.
- [9] F. Ertam and M. Kaya, "Classification of firewall log files with multiclass support vector machine," in *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, 2018, vol. 2018-Janua, pp. 1–4. doi: 10.1109/ISDFS.2018.8355382.
- [10] M. H. Rahman, T. Islam, M. M. Rana, R. Tasnim, T. R. Moni, and M. M. Sakib, "Machine learning approach on multiclass classification of internet firewall log files," in *Proceedings of International Conference on Computational Intelligence and Sustainable Engineering Solution, CISES 2023*, 2023, pp. 358–364. doi: 10.1109/CISES58720.2023.10183601.
- [11] Z. Li, C. Huang, S. Deng, W. Qiu, and X. Gao, "A soft actor-critic reinforcement learning algorithm for network intrusion detection," *Computers and Security*, vol. 135, 2023, doi: 10.1016/j.cose.2023.103502.
- [12] S. S. Bamber, A. V. R. Katkuri, S. Sharma, and M. Angurala, "A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system," *Computers and Security*, vol. 148, p. 104146, 2025, doi: 10.1016/j.cose.2024.104146.
- [13] P. Artioli, V. Dentamaro, S. Galantucci, A. Magri, G. Pellegrini, and G. Semeraro, "SIEVE: Generating a cybersecurity log dataset collection for SIEM event classification," *Computer Networks*, vol. 266, p. 111330, 2025, doi: 10.1016/j.comnet.2025.111330.
- [14] A. Yichiet, Y. M. J. Khaw, M. L. Gan, and V. Ponnusamy, "A semantic-aware log generation method for network activities," *International Journal of Information Security*, vol. 21, no. 2, pp. 161–177, 2022, doi: 10.1007/s10207-021-00547-6.
- [15] A. Andalib and S. M. Babamir, "Anomaly detection of policies in distributed firewalls using data log analysis," *Journal of Supercomputing*, vol. 79, no. 17, pp. 19473–19514, 2023, doi: 10.1007/s11227-023-05417-7.
- [16] E. Ucar and E. Ozhan, "The analysis of firewall policy through machine learning and data mining," *Wireless Personal Communications*, vol. 96, no. 2, pp. 2891–2909, 2017, doi: 10.1007/s11277-017-4330-0.
- [17] D. Bringhenti and F. Valenza, "GreenShield: Optimizing firewall configuration for sustainable networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 6, pp. 6909–6923, 2024, doi: 10.1109/TNSM.2024.3452150.
- [18] D. Bringhenti, L. Seno, and F. Valenza, "An optimized approach for assisted firewall anomaly resolution," *IEEE Access*, vol. 11, pp. 119693–119710, 2023, doi: 10.1109/ACCESS.2023.3328194.
- [19] J. Park, B. Park, and T. S. Kim, "Development of an anomaly classification model and a decision support tool for firewall policy configuration," *Applied Sciences (Switzerland)*, vol. 15, no. 6, p. 2979, 2025, doi: 10.3390/app15062979.
- [20] S. Alabdulhadi and A. Al-Matouq, "Efficient and standardized alarm rationalization for cybersecurity monitoring," *IEEE Access*, vol. 12, pp. 166936–166944, 2024, doi: 10.1109/ACCESS.2024.3492264.
- [21] D. Sharma, V. Wason, and P. Johri, "Optimized classification of firewall log data using heterogeneous ensemble techniques," in *2021 International Conference on Advance Computing and Innovative Technologies in Engineering, ICACITE 2021*, 2021, pp. 368–372. doi: 10.1109/ICACITE51222.2021.9404732.
- [22] E. Efeoğlu and G. Tuna, "Classification of firewall log files with different algorithms and performance analysis of these algorithms," *Journal of Web Engineering*, vol. 23, no. 4, pp. 561–594, 2024, doi: 10.13052/jwe1540-9589.2344.
- [23] M. Mingze, "Research and application of firewall log and intrusion detection log data visualization system," *IET Software*, vol. 2024, no. 1, p. 7060298, 2024, doi: 10.1049/2024/7060298.
- [24] GitHub, "Firewall-logs-dataset," *GitHub*, 2024. <https://github.com/MinhLinhEdu/Firewall-logs-dataset> (accessed Nov. 23, 2024).
- [25] D. M. Linh, H. D. Hung, H. M. Chau, Q. S. Vu, and T. N. Tran, "Real-time phishing detection using deep learning methods by extensions," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 14, no. 3, pp. 3021–3035, 2024, doi: 10.11591/ijece.v14i3.pp3021-3035.
- [26] Z. Sun, G. Wang, P. Li, H. Wang, M. Zhang, and X. Liang, "An improved random forest based on the classification accuracy and correlation measurement of decision trees," *Expert Systems with Applications*, vol. 237, p. 121549, 2024, doi: 10.1016/j.eswa.2023.121549.
- [27] B. Padmavathi and V. Selvaraj, "Advanced optimization-based weighted features for ensemble deep learning smart occupancy detection network for road traffic parking," *Journal of Network and Computer Applications*, vol. 230, p. 103924, 2024, doi: 10.1016/j.jnca.2024.103924.
- [28] K. Pearson, "Mathematical contributions to the theory of evolution," in *Proceedings of the Royal Society*, 1896, vol. 60, no. 1834, pp. 489–498.
- [29] P. Chen, F. Li, and C. Wu, "Research on intrusion detection method based on Pearson correlation coefficient feature selection algorithm," *Journal of Physics: Conference Series*, vol. 1757, no. 1, p. 012054, 2021, doi: 10.1088/1742-6596/1757/1/012054.

BIOGRAPHIES OF AUTHORS






Tran Cong Hung    was born in Vietnam in 1961. He received his B.E. degree in electronics and telecommunication engineering with first-class honors from Ho Chi Minh University of Technology in 1987. He received his B.E. degree in informatics and computer engineering from Ho Chi Minh University of Technology in 1995. He earned a Master of Engineering degree in Telecommunications Engineering from the Postgraduate Department of Hanoi University of Technology in 1998. He received his Ph.D. degree from Hanoi University of Technology in 2004. His main research areas include B-ISDN performance parameters and measurement methods, QoS in high-speed networks, cyber threat intelligence, deep learning, and MPLS. Currently, he is an associate professor at the Department of Computer Science and Engineering, Saigon International University in Ho Chi Minh City, Vietnam. He can be reached via email at: tranconghung@siu.edu.vn, and conghung@ptithcm.edu.vn. More information: <https://tranconghung.com>.






Dam Minh Linh    was born in 1982 in Tay Ninh Province, Vietnam. He received his engineering degree in information technology in 2010 and his M.Sc. degree in information systems in 2016, both from the Posts and Telecommunications Institute of Technology. He is currently a lecturer at the Faculty of Information Technology and a member of the Information Security Technology Lab at the Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam. His research interests include anti-phishing solutions, cyber threat intelligence, deep learning with transformer models, natural language processing, and computer vision. Since 2024, he has been a Ph.D. student in information systems at the Posts and Telecommunications Institute of Technology, Vietnam. His current research focuses on applying transformer-based deep learning techniques to combinatorial optimization, particularly in cybersecurity and network security domains. He can be reached via email at: linhdm@ptit.edu.vn.






Han Minh Chau    was born in Vietnam in 1980. He is currently the head of the Department of Computer Networks, Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam. He received a bachelor's degree in information technology from Hanoi University of Science and Technology and a master's degree in computer science from the Ho Chi Minh City University of Information Technology, Vietnam National University at Ho Chi Minh City. His main research interests include network security, machine learning, computer vision, cloud computing platforms, and artificial intelligence. He can be reached via email at: hm.chau80@hutech.edu.vn.






Ngo Xuan Thoai    was born in Vietnam in 2001. He received his B.Eng. degree in information security from the Posts and Telecommunications Institute of Technology, Vietnam, in 2024. He is currently employed as a Software Development and Operations Engineer (e.g., DevOps Engineer) at FE Credit, a subsidiary of VPBank. His research interests include machine learning, natural language processing, application development, and computer vision. He can be reached via email at: thoainx01@gmail.com.



Thai Duc Phuong    was born in Vietnam in 1993, has worked in the information security industry as a design, deployment, and management engineer since 2015. He earned a bachelor's degree in computer networking and communications from the University of Information Technology, VNU-HCM, in 2016 and began a master's in computer science at Saigon International University in 2022. He is currently Deputy Technical Director of Services at Vietsunshine Electronic Solution Joint Stock Company. He can be reached via email at: phuong.thai@vietsunshine.com.vn.



Huynh De Thu    received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China in 2018. He is working as a lecturer and researcher in School of Computer Science & Engineering, The Saigon International University, Ho Chi Minh City, Vietnam. His current research interests were in the areas of wireless networks, device-to-device communications, IoT, network security and data science. He can be reached via email at: huynhdethu@siu.edu.vn.