Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9173291, 15 pages
https://doi.org/10.1155/2022/9173291

Research Article

MidSiot: A Multistage Intrusion Detection System for Internet of Things

Nguyen Dat-Thinh,[1,2] **Ho Xuan-Ninh**,[1,2] **and Le Kim-Hung**[1,2]

[1]*Faculty of Computer Networks and Communications, University of Information Technology, Ho Chi Minh City 70000, Vietnam*
[2]*Vietnam National University, Ho Chi Minh City 70000, Vietnam*

Correspondence should be addressed to Le Kim-Hung; hunglk@uit.edu.vn

Received 20 October 2021; Revised 4 January 2022; Accepted 17 January 2022; Published 21 February 2022

Academic Editor: Hamed Nassar

Internet of Things (IoT) has been thriving in recent years, playing an important role in a multitude of various domains, including industry 4.0, smart transportation, home automation, and healthcare. As a result, a massive number of IoT devices are deployed to collect data from our surrounding environment and transfer these data to other systems over the Internet. This may lead to cybersecurity threats, such as denial of service attacks, brute-force attacks, and unauthorized accesses. Unfortunately, many IoT devices lack solid security mechanisms and hardware security supports because of their limitations in computational capability. In addition, the heterogeneity of devices in IoT networks causes nontrivial challenges in detecting security threats. In this article, we present a collaborative intrusion detection system (IDS), namely, MidSiot, deployed at both Internet gateways and IoT local gateways. Our proposed IDS consists of three stages: (1) classifying the type of each IoT device in the IoT network; (2) differentiating between benign and malicious network traffic; and (3) identifying the type of attacks targeting IoT devices. The last two stages are handled by the Internet gateways, whereas the first stage is on the local gateway to leverage the computational resources from edge devices. The evaluation results on three popular IDS datasets (IoTID20, CIC-IDS-2017, and BOT-IoT) indicate our proposal could detect seven common cyberattacks targeting IoT devices with an average accuracy of 99.68% and outperforms state-of-the-art IDSs. This demonstrates that MidSiot could be an effective and practical IDS to protect IoT networks.

1. Introduction

The number of devices connecting to the Internet has been growing at a breathtaking pace over the past decades. From two billion in 2006, it reached 200 billion in 2020 because of the proliferation of mobile computing and the Internet of Things (IoT) [1]. As a result, these devices play a critical role in primary industries (e.g., healthcare, manufacturing, retailing, security, and transportation) by providing intelligent services, such as tracking inventory, managing machines, monitoring patient health, and detecting abnormality. They are anticipated to boost the total global worth of IoT to 6.2 trillion dollars by 2025, most of which come from manufacturing (2.3 trillion dollars) and healthcare (2.5 trillion dollars) [2]. It is apparent that IoT is the driving force of evolution in every daily aspect.

However, ensuring security and privacy for the IoT devices is a nontrivial challenge due to their limitation in computational capability, which is insufficient for traditional security mechanisms. This makes them susceptible to wide-ranging cyberattacks, such as data leakage, spoofing, and DoS/DDoS. In a report published by Kaspersky, the first half of 2021 witnessed 1.5 billion attacks against smart devices aiming at stealing data, mining cryptocurrency, or building botnets [3]. In September 2016, an infamous attack performed by Mirai malware turned 380,000 devices into botnets that launched DDoS attacks against several services and organizations, including Dyn-a domain registration service provider [4]. Moreover, this malware is capable of mutating [5]. On 12 December 2017, its variant exploited a zero-day flaw in Huawei HG532 routers to speed up its infection. One year later, the number of variants was

increased significantly, such as Okiru, Masuta, OMG, Wicked, Hakai, and Yowai [6].

To eliminate such security threats, an intrusion detection system (IDS) is commonly deployed at network gateways. It constantly monitors network traffics coming from various sources to detect abnormalities, which may be security threats. Following [7], the attack detection approaches of the IDS are categorized into signature-based and anomaly-based. The former approach identifies cyberattacks by comparing a set of signatures (or rules) extracted from known attacks with incoming traffic. We note that the key difference between IoT network traffic and other network traffics is the diversity and volume. The diversity of IoT network traffic comes from the heterogeneity of IoT devices and their communication protocols, resulting in diverse network behaviors. Furthermore, IoT devices exponentially increase and generate massive data traversing the Internet. Due to these characteristics, the rule-based detection mechanism, SNORT is a typical example, is insufficient for IoT networks. In detail, SNORT is ineffective for complex attacks signs of which are various and implicit in network traffic. In addition, SNORT needs to maintain a large rule-set and security experts in the loop to update these rules frequently. Therefore, it is not efficient and scalable enough for IoT scenarios. The latter approach, which is the most popular, makes use of machine learning (ML) to construct a model of normal network traffic patterns. This model is then used to measure the similarity between the incoming traffic and known patterns to detect malicious traffic. Although the anomaly-based approach using machine learning considerably alleviates the weaknesses of the signature-based approach, it still has several limitations.

(i) Neglecting a collaborative edge-cloud architecture: Training and inferring tasks in machine learning are resource-intensive, so they are usually handled by cloud platforms in existing work. This might decrease the detection performance because the network traffic at the cloud level, which is merged from several gateways consisting of various data sources, is intricate. In contrast, edge devices are resource-constraint IoT devices that are insufficient to handle complex machine learning tasks (e.g., detecting abnormal activities, training detection models). Offloading these tasks on edge devices severely affects other services running on these devices. However, we believe that these devices could handle specific tasks to increase the IDS's detection performance regardless of their limitation in computation capability. Therefore, a collaborative edge-cloud architecture for instruction detection systems is necessary to overcome this limitation.

(ii) Lacking IoT device-type identification: Because the IoT device types are various and heterogeneous, their network behaviors are highly diverse. For example, the high UDP packet rate coming from IoT cameras is normal, but the one from temperature sensors is a sign of a security threat. This may lead to false attack detection. Thus, identifying device types and considering them as an input feature of the attack detection model is crucial to increase the detection accuracy of IDSs.

(iii) Detecting a limited set of attack types: Existing works about IDS are extensive but primarily concerned with detecting a limited set of attacks in a general domain, such as DoS and spoofing. Given the rising prevalence of IoT, there is an essential need to address a larger set of attacks targeting IoT networks.

To solve the above limitation, we present MidSiot, a machine learning-based three-stage IDS designed for IoT networks supporting collaboration between local gateways and Internet gateways of Internet Service Provider (ISP) (solving the first limitation). In more detail, to leverage edge computing and enhance the attack detection accuracy, the first stage is operated at local gateways to identify IoT devices based on their behaviors in the network (solving the first limitation), whereas the next two stages powered by a machine learning model are handled by the Internet gateways to not only differentiate between normal and malicious network traffic, but also accurately identify the seven common attack types (solving the first limitation). The evaluation results on existing IDS datasets for the IoT domain (IoTID20, BOT-IoT, and CIC-IDS-2017) show that MidSiot could detect seven popular attacks targeting IoT devices with an average accuracy of 99.68%. Our main contributions presented in this study are as follows:

(1) A collaborative architecture for IoT IDSs to leverage the computational resources of edge gateway to enhance IDS's detection performance.

(2) A lightweight and robust machine learning-based IDS constituting of three stages to accurately detect various cyberattacks pointing at IoT devices.

(3) We intensively evaluate our proposal on popular IDS datasets and examine the resampling techniques to address imbalanced datasets during our experiments.

The remainder of the article is organized as follows. In Section 2, we discuss related work. The MidSiot architecture and its detection method are presented in Section 3. Section 4 reports the evaluation of our method through IDS datasets, and we conclude our work in Section 5.

2. Related Works

In recent years, there has been an increased interest in exploring machine learning for enhancing the detection quality of IDSs [17, 18]. In [9], the authors proposed an anomaly detection mechanism using a single machine learning classifier. The authors of [10] presented a scalable k-NN-based online anomaly detection addressing the lazy-learning problem in wireless sensor networks [10]. The works in [16] also employed anomaly detection techniques for IDSs using binary classification. This means that they cannot identify the type of attack. In [11], the authors proposed an ensemble of autoencoders for online IDS whose

performance is comparable to offline anomaly detectors. Ref. [12] is a novel approach for IDSs in which the authors applied convolutional neural network to predict the attack types. 98% accuracy on the NSL-KDD dataset was achieved in this experiment. There are also hybrid-IDSs where anomaly-based and signature-based approaches are used to develop the IDS. Such a typical system is introduced in [19], in which packet header anomaly detection, network traffic anomaly detection, and SNORT are combined. In [13], the authors leveraged four machine learning algorithms to derive rule-sets used as signatures for their IDS. In [8], a hierarchical architecture including multiple neural networks was proposed to detect malicious packets and identify the attack types hidden inside these packets. The authors in [14] introduced a hierarchical structure for IDS that separates the detection process into different steps. The authors in [20] suggested a distributed architecture for smart home IDSs that offload complex tasks onto the Internet Service Provider (ISP) and deliver simple ones to the smart home gateway [20].

In terms of datasets used for IDSs, the authors in [21] proposed a new dataset called IoTID20, which was also evaluated in their work by implementing several machine learning algorithms (e.g., logistic regression, decision tree, random forest), which results in increasing F1-score for both binary classification and multiclass classification. The authors in [15] developed a new realistic botnet dataset for use in IoT networks, and as a result, it mainly consists of DDoS attacks. Another dataset is CICIDS-2017 including attack traffic generated by their testbed and realistic background traffic created by the B-profile system [22]. All of these 3 datasets were constructed using the CICFlowMeter tool (formerly known as ISCXFlowMeter), thereby having a similar set of features. Regarding resampling methods for network intrusion detection system (NIDS), the literature at [23] compared multiple undersampling techniques for NIDS on CICIDS-2017 and CICIDS-2018 datasets, including random, cluster centroids, and nearmiss algorithms. The authors concluded that these undersampling methods reduced models' training time, and K-nearest neighbor has the most significant improvement. There are also other works that implemented a combination of several resampling techniques, such as oversampling and undersampling [24]. Their experimental results showed that the oversampling method increases the training time, whereas the undersampling method decreases this time. In addition, if the dataset is highly imbalanced, these methods improve the recall score notably. The authors in [25] proposed an algorithm-level class balancing technique that addresses the underlying issue about attack class imbalance in IDS datasets, resulting in identifying various attack categories with better accuracy than the CNN models.

In recent years, many researchers have geared toward blockchain applications in intrusion detection systems thanks to its potential in protecting data integrity and privacy. The authors in [26] investigated the challenges and limitations of blockchain to intrusion detection in addition to their applications, such as the overhead traffic with limited handling capability of intrusion detection and extensive energy and cost usage of blockchain. Despite these difficulties, blockchain still has the potential to mitigate the data sharing and trust management issues in collaborative intrusion detection. As far as collaborative intrusion detection systems are concerned, a series of research [27–29] provided blockchain challenge-based collaborative intrusion detections. In these systems, the authors leveraged the strength of blockchain to investigate the trust mechanism in a network of IDS nodes. Their goals are to enhance the robustness of trust management against attacks as well as to protect the alarm aggregation process from malicious inputs. The works in [30, 31] made some contributions in the same direction, but the authors specifically targeted intrusion detection systems in a software-defined network. The authors in [32] proposed a deep blockchain framework to offer security-based distributed intrusion detection and privacy-based blockchain with smart contracts in IoT networks. Although the experimental results of the intrusion detection system were optimistic, the classification algorithm in use was a bidirectional long short-term memory, which accompanying blockchain might aggregate more computational burden on operating IoT devices.

To summarize the related works, Table 1 presents the state-of-the-art intrusion detection systems and their characteristics, including targeting security threats, attack detection method, evaluation datasets, attack-type and device-type detection, and lightweight. We can see that none of them is lightweight enough to classify the type of attack and its target. In addition, several approaches are evaluated by non-IoT datasets or testbeds having a small number of IoT devices. Thus, previous IDS proposals are insufficient for deploying to practical IoT ecosystem.

3. The MidSiot IDS

3.1. System Overview. In this section, we explain how our proposal works. First of all, Figure 1 illustrates the architecture of the proposed IDS that comprises three stages distributed between local network infrastructures and ISPs. The first stage is operated at the local gateways to identify connected IoT devices through their network behaviors. The next stage, which is conducted at the Internet gateways of ISPs, classifies network traffic of such IoT devices as normality or abnormality. When abnormal traffic is detected, it is transferred to the third stage to identify the attack types. Since the last two stages are done on ISPs which aggregate a huge volume of network traffic, correctly identifying the IoT device types along with their network traffic at the first stage is essential in increasing attack detection performance at following stages, especially for large-scale attacks targeting at multiple networks.

Second, Figures 2 and 3 present the block diagram of main operational phases in MidSiot, including the training and prediction phases, respectively. They also illustrate the connection and interfaces of components of the proposed IDS. In more detail, as shown in Figure 2, the raw network packets are captured by the packet flow inspection component from network traffic and transformed into network flows. These flows are then fed into the feature extraction component, extracting network features and computing network flow

Table 1: Summary of current works on Intrusion Detection Systems for Internet of Things.

Work	Security threat	Detection method	Validation dataset	Attack-type detection	Device-type detection	Lightweight
Zhang et al. [8]	DoS, R2L, U2R, and PROBE	Deep learning	KDD Cup 1999 Data	Yes	No	—
Wang and Stolfo [9]	58 attack types with 1999 DARPA dataset CUCS dataset (Code Red II, Buffer overflow)	1-gram models	1999 DARPA IDS Dataset CUCS Dataset	Yes	No	—
Xie et al. [10]	—	Machine learning	Real WSN data sets	—	No	Yes
Mirsky et al. [11]	Recon., MITM, DoS, Botnet	Autoencoder	Real-testbed	Yes	No	Yes
Ince [12]	DoS, probe, R2L, U2R	Deep learning	NSL-KDD	Yes	No	—
Kumar et al. [13]	Dos, exploit, probe, generic	Hybrid	UNSW-NB15	Yes	No	—
Anthi et al. [14]	Attack reconnaissances, DoS attacks, man-in-the-middle attacks, replay attacks, DNS spoofing	Machine learning	Real-testbed	Yes	Yes	—
Koroniotis et al. [15]	DoS/DDoS attacks, keylogging, data theft	Deep learning	BOT-IoT Dataset	Yes	No	—
Liu et al. [16]	Vulnerability scanners, ARP spoofing, DoS attacks, Mirai Botnet	Machine learning	IOTID-20 Dataset	No	No	Yes
Proposed System	Scanning methods (Host Discovery, Port scanning, OS/Version Detection) ARP Spoofing, SYN Flooding, Host Discovery, Telnet Bruce-force, UDP/ACK/HTTP Flooding	Machine learning	IOTID-20, CICIDS-2017, BOT-IoT Dataset	Yes	Yes	Yes

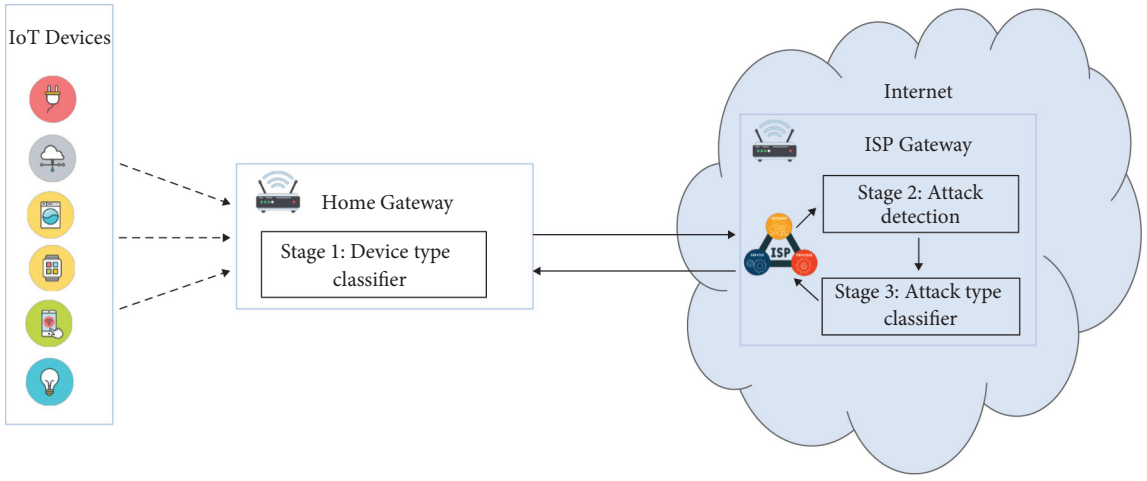

Figure 1: Overview of the architecture of the multistage intrusion detection system.

statistics. In addition, feature selection algorithm is applied to filter inappropriate features from the output features. In the training phase, these features are aggregated into a dataset used to train the models. Once models are trained successfully, the ISPs store these models used for the second and third stages in their local storages, while the models used for the first stage are sent to the local gateways. We note that the second stage employs several models, and each model is responsible for classifying network traffic for a specific device type.

In the prediction phase illustrated in Figure 3, the network features are constructed similarly with the training phase; however, they are then fed to models for detecting malicious traffic. In more detail, the model of the first stage running on the local gateway is loaded to identify the device type of such features. All this information is transferred to ISP's Internet gateways, where a well-trained model corresponding to the device type is used to detect abnormality in these features. If malicious activities are detected, they are forwarded to the third stage to detect attack types by using a universal attack detection model. The detection results, including the IoT device under the attack and the type of attack, are sent to the action manager component to trigger necessary actions (logging attack behaviors, blocking the network traffic of victims, notifying administrators about the attack). Note that, because MidSiot's

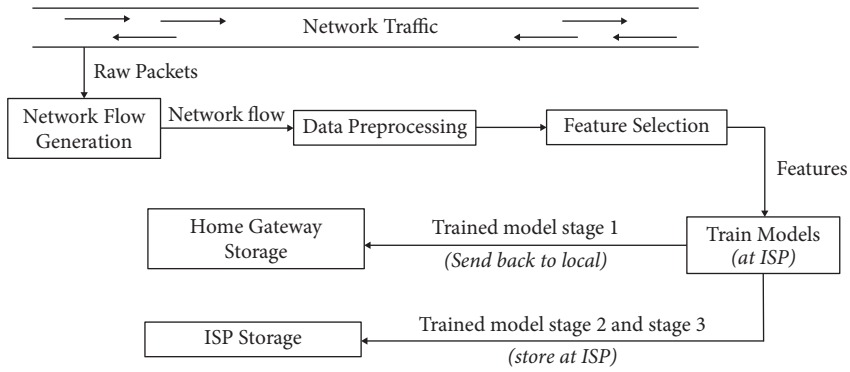

Figure 2: MidSiot's training phase.

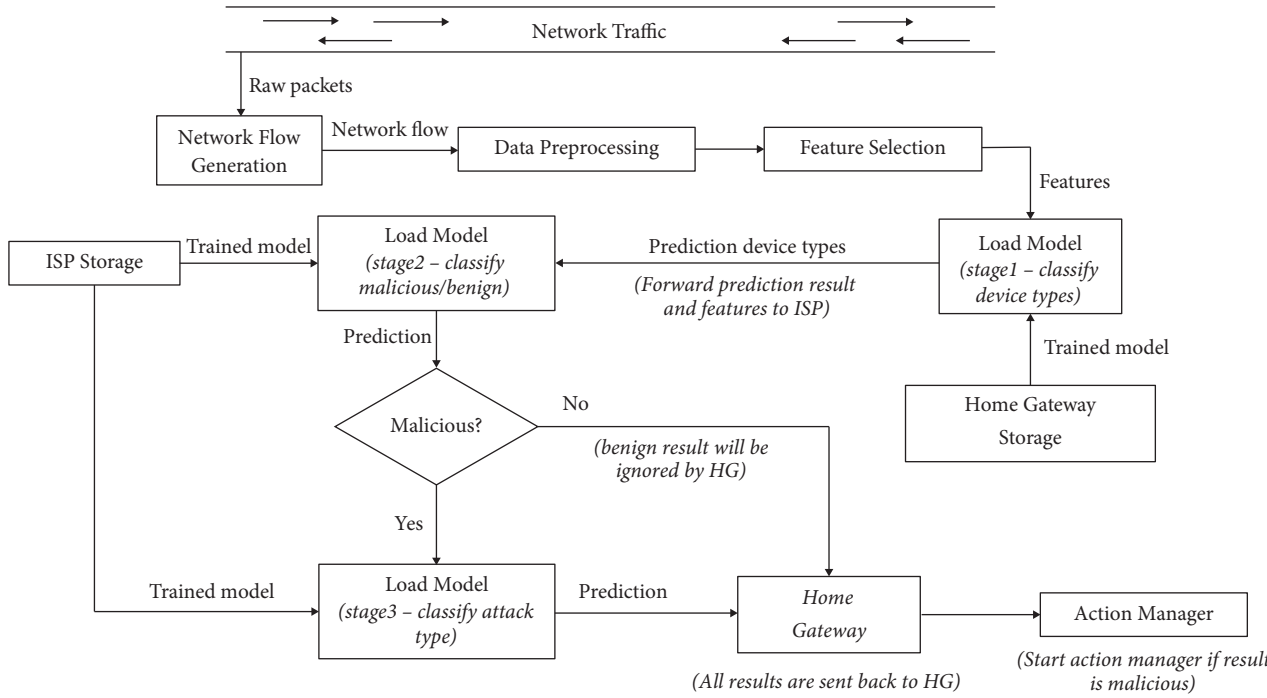

Figure 3: MidSiot's prediction phase.

structure uses linked stages, the errors of one stage might affect not only the following stages but also the overall system's performance. For example, if MidSiot misclassifies the device type, the second-stage results are potentially false. This is because the second-stage model is trained to learn the network patterns associated with a specific device type, and these patterns are different for each device type. Similarly, if the second-stage model misclassifies normal network traffic as abnormal, the final stage result is incorrect and triggers a false alert.

3.2. Network Flow Generation. Network flow generator is used to generate network flows from a batch of raw network packets. In MidSiot, it is powered by the deep packet inspection method that aggregates packets into flows sharing source/destination IP, source/destination port, and protocol and calculates flow features and statistics. In addition, this method supports extracting MAC addresses, making it possible to label devices. Therefore, we could obtain 83 network features (e.g., FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, TimeStamp, and Protocols) listed in Appendix VI (Table 2).

3.3. Data Preprocessing. In IDS datasets, not all features are suitable for machine learning algorithms; some of them may degrade the model training performance, whereas others make models overfit. Therefore, employing a feature selection algorithm is necessary. First, all identity-based features (e.g., *ip_src, ip_dst, flow_id, timestamp*) are dropped to prevent the overfit issues, even features related to MAC addresses after labeling connected devices. We then adopt Pearson's correlation coefficient to identify and remove unimportant features. In more detail, the importance index of each feature is its linear correlation coefficient value

Table 2: Extract network features.

Feature name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of the packet in the forward direction
fw_pkt_l_max	Maximum size of the packet in the forward direction
fw_pkt_l_min	Minimum size of the packet in the forward direction
fw_pkt_l_avg	Average size of the packet in the forward direction
fw_pkt_l_std	Standard deviation size of the packet in the forward direction
Bw_pkt_l_max	Maximum size of the packet in the backward direction
Bw_pkt_l_min	Minimum size of the packet in the backward direction
Bw_pkt_l_avg	Mean size of the packet in the backward direction
Bw_pkt_l_std	Standard deviation size of the packet in the backward direction
fl_byt_s	Flow byte rate that is the number of packets transferred per second
fl_pkt_s	Flow packets rate that is the number of packets transferred per second
fl_iat_avg	Average time between two flows
fl_iat_std	Standard deviation time two flows
fl_iat_max	Maximum time between two flows
fl_iat_min	Minimum time between two flows
fw_iat_tot	Total time between two packets sent in the forward direction
fw_iat_avg	Mean time between two packets sent in the forward direction
fw_iat_std	Standard deviation time between two packets sent in the forward direction
fw_iat_max	Maximum time between two packets sent in the forward direction
fw_iat_min	Minimum time between two packets sent in the forward direction
bw_iat_tot	Total time between two packets sent in the backward direction
bw_iat_avg	Mean time between two packets sent in the backward direction
bw_iat_std	Standard deviation time between two packets sent in the backward direction
bw_iat_max	Maximum time between two packets sent in the backward direction
bw_iat_min	Minimum time between two packets sent in the backward direction
fw_psh_flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
bw_psh_flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
fw_urg_flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
bw_urg_flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
fw_hdr_len	Total bytes used for headers in the forward direction
bw_hdr_len	Total bytes used for headers in the forward direction
fw_pkt_s	Number of forward packets per second
bw_pkt_s	Number of backward packets per second
pkt_len_min	Minimum length of a flow
pkt_len_max	Maximum length of a flow
pkt_len_avg	Mean length of a flow
pkt_len_std	Standard deviation length of a flow
pkt_len_va	Minimum interarrival time of the packet
fin_cnt	Number of packets with FIN
syn_cnt	Number of packets with SYN
rst_cnt	Number of packets with RST
pst_cnt	Number of packets with PUSH
ack_cnt	Number of packets with ACK
urg_cnt	Number of packets with URG
cwe_cnt	Number of packets with CWE
ece_cnt	Number of packets with ECE
down_up_ratio	Download and upload ratio
pkt_size_avg	Average size of packet
fw_seg_avg	Average size observed in the forward direction
bw_seg_avg	Average size observed in the backward direction
fw_byt_blk_avg	Average number of bytes bulk rate in the forward direction
fw_pkt_blk_avg	Average number of packets bulk rate in the forward direction
fw_blk_rate_avg	Average number of bulk rate in the forward direction
bw_byt_blk_avg	Average number of bytes bulk rate in the backward direction
bw_pkt_blk_avg	Average number of packets bulk rate in the backward direction
bw_blk_rate_avg	Average number of bulk rate in the backward direction
subfl_fw_pk	The average number of packets in a subflow in the forward direction

Table 2: Continued.

Feature name	Description
subfl_fw_byt	The average number of bytes in a subflow in the forward direction
subfl_bw_pkt	The average number of packets in a subflow in the backward direction
subfl_bw_byt	The average number of bytes in a subflow in the backward direction
fw_win_byt	Number of bytes sent in initial window in the forward direction
bw_win_byt	# of bytes sent in initial window in the backward direction
Fw_act_pkt	# of packets with at least 1 byte of TCP data payload in the forward direction
fw_seg_min	Minimum segment size observed in the forward direction
atv_avg	Mean time a flow was active before becoming idle
atv_std	Standard deviation time a flow was active before becoming idle
atv_max	Maximum time a flow was active before becoming idle
atv_min	Minimum time a flow was active before becoming idle
idl_avg	Mean time a flow was idle before becoming active
idl_std	Standard deviation time a flow was idle before becoming active
idl_max	Maximum time a flow was idle before becoming active
idl_min	Minimum time a flow was idle before becoming active

varying between −1 and 1. Finally, we remove all features having an importance index lower than 0.8. As a result, the final dataset only comprises 40 features, excluding all labels. We note that In MidSiot, Pearson correlation was only used during the training phase to construct a set of concise and suitable features for machine learning models. This feature set is then saved and loaded to the IDS in the detection phases. This means that Pearson correlation is inactive in the detection phase. Therefore, it has no impact on the detection procedure.

Algorithm 1 Overview. Let $X = [x1, x2, \ldots, xn]$ denote a list of raw packets, and D is the list of processed flows. The major steps of this algorithm are described as follows:

(1) **Network Flow Generation**, in Line 2, receives a list of raw packets X and generates network flows by aggregating packets sharing Source/DestinationIP, Source/DestinationPort, Protocol.

(2) **Label Device Types**, in Line 3, the device type of each flow is deduced via Source/DestinationMAC Address. In addition, this step is performed during the training phase only.

(3) **Drop and Normalize Data**, in Lines 4 and 5, any flows in F having a null value at any field will be dropped. Afterward, the remaining flows are normalized to facilitate the machine learning processes.

(4) **Pearson's Correlation Coefficient**, in Line 6, the Pearson's correlation coefficient is applied on the normalized flows F_2 to select only important features Fts. Finally, the flow list F_2 will have some features dropped and only features from Ft are retained, which results in D.

3.4. Multistage Attack Detection Algorithm

3.4.1. The Overview. The details of the multistage attack detection algorithm is described in Algorithm 2. In more detail, let $X = [x1, x2, \ldots, xn]$ denote a list of raw packets and R is the resulting attack type. The entire detection process consists of the following main steps:

(i) **Processing data,** including generating network flows, dropping unnecessary features, and normalizing data are performed similarly to the Algorithm 1. However, as this is the detection process, device-type labeling and features selecting using Pearson's correlation coefficient are inactive.

(ii) **Classifying device type,** the classification model from the storage of the local home gateway to perform prediction on the processed flow f_2 to deduce the device type.

(iii) **Sending the result to the ISP,** the processed flow f_2 in addition to the prediction results dt is forwarded to the ISP to further perform abnormality and attack-type detection.

(iv) **Detecting the attacks,** the model m is applied on the flow f_2 to deduce whether that flow is normal or abnormal. If it is abnormal, move on to the next step; otherwise, mark this flow as null (which represents benign).

(v) **Classifying attack type,** a universal attack-type detection model is loaded from the storage of the ISP gateway. This model will be then applied on the malicious flow f_2 to deduce the kind of attack a that has happened. At the end of this process, we know the device type of the flow, whether the flow is malicious or benign, and the attack type of the flow if it is malicious.

3.4.2. The First Stage. The primary benefit of the first stage is to enhance the accuracy of the attack detection model in the next steps. In more detail, since the IoT device types are various and heterogeneous, their network behaviors are highly diverse. For example, the high UDP packet rate coming from IoT cameras is normal, but the one from temperature sensors is a sign of a security threat. This may lead to false attack detection. Thus, identifying device types and considering them as an input feature of the attack detection model is crucial to increase the accuracy. Furthermore, this stage should be done on the local gateway for two reasons: (1) local gateways have sufficient

Input: Raw network packets X
Output: Processed data D
(1) **Initialize:** $D =$
(2) **Generate network flows:** $F \leftarrow$NetworkFlowGenerator(X)
(3) **Label device type:** LabelDeviceType(F)
(4) **Drop invalid flows and identity columns:** $F_1 \leftarrow$DropInvalidData(F)
(5) **Normalize data:** $F_2 \leftarrow$Normalize(F_1)
(6) **Select features using Pearson's Correlation Coefficient:**
(a) Fts$\leftarrow$Corr(F_2)
(b) $D \leftarrow$SelectFeatures$(F_2,$ Fts$)$
(7) **Return** D

Algorithm 1: Data preprocessing in training phases.

Input: Raw network packets X
Output: Attack type R
(i) **Local home gateway:**
(1) **Generate network flows:**
$f \leftarrow$GenerateFlows(X)
(2) **Drop unnecessary features:**
$f_1 \leftarrow$DropFeatures(f)
(3) **Normalize data:**
$f_2 \leftarrow$Normalize(f_1)
(4) **Load device type classification model:**
$M_1 \leftarrow$LoadModel
(5) **Classify device type:** Device type
$dt \leftarrow M_1$.predict(f_2)
(6) **Send to ISP:** Send$((dt, f_2))$
(ii) **Internet Service Provider (ISP)**
(1) **Initialize:** $R =$
(2) **Load all abnormality detection models:**
$M_2 \leftarrow$LoadModel2
(3) **Load abnormality detection model:**
$m \leftarrow M_2[dt]$
(4) **Attack detection:**
$y \leftarrow m$.predict(f_2)
(5) **If** y **is not** normal **then**
(a) **Load attack type detection model:**
$M_3 \leftarrow$LoadModel3
(b) **Attack classification:**
$a \leftarrow M_3$.predict(f_2)
(c) $R \leftarrow a$
(6) **Else**
$R \leftarrow$null
(7) **Return** R

Algorithm 2: The overall detection process.

computation power to handle a part of the detection process, which reduces the burden for the cloud; and (2) if device-type classification is done with the other two steps on the cloud, merge a multitude of network packets coming from various IoT networks. This aggregation may make the network data exhibit more generic characteristics than device-specific ones, reducing the device-type classification performance. This directly affects the attack detection accuracy. Therefore, running the device-type classifiers in the local gateways closed to IoT devices could mitigate this issue since only a limited number of device types are considered.

3.4.3. The Second and Third Stages. Take a dataset

$$D = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), \tag{1}$$

where $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^n)^T$, $i = 1, 2, \ldots, N$, x_i is the input instance that represents a network flow. x_i has n features. N indicates the number of features of a network flow contained in the dataset D. $y_i \in [0, 1, 2, \ldots, K-1]$ is the result of each detection record. A decision tree recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together.

Let the data at node m be represented by Q_m. For each split $\theta = (j, t_m)$ consisting of a feature j and a threshold t_m, partition the data into $Q_m^{\text{left}}(\theta)$ and $Q_m^{\text{right}}(\theta)$ subsets:

$$\begin{aligned} Q_m^{\text{left}}(\theta) &= \{(x, y) | x_i <= t_m\}, \\ Q_m^{\text{right}}(\theta) &= Q_m \backslash Q_m^{\text{left}}(\theta). \end{aligned} \tag{2}$$

The quality of a candidate split of node m is then computed using an impurity function or loss function H, the choice of which depends on the task being solved (classification or regression):

$$G(Q_m, \theta) = \frac{N_m^{\text{left}}}{N_m} H\left(Q_m^{\text{left}}(\theta)\right) + \frac{N_m^{\text{right}}}{N_m} H\left(Q_m^{\text{right}}(\theta)\right). \tag{3}$$

Select the parameters to minimize the impurity:

$$\theta^* = \text{argmin}_\theta G(Q_m, \theta). \tag{4}$$

Repeat for subsets $Q_m^{\text{left}}(\theta)$ and $Q_m^{\text{right}}(\theta)$ until the maximum allowable depth is reached $N_m < \min_{\text{sample}}$ or $N_m = 1$.

For the classification of IDS, $y_i \in [0, 1, 2, \ldots, K-1]$ for node m represents a region of R_m with instances of N_m. Assume that p_{mk} is the proportion of class k instance in m and can be obtained by the following formula:

$$p_{mk} = \frac{1}{N_m \sum_{x_j \in R_m} I(y_i = k)}. \tag{5}$$

The common measure of impurity is named Gini and can be obtained by the following formula:

$$H(X_m) = -\sum_k p_{mk}(1 - p_{mk}). \tag{6}$$

Cross-entropy can be obtained by the following formula:

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk}). \tag{7}$$

Misclassification can be obtained by the following formula (not being used in the proposed system):

$$H(X_m) = -\sum_k 1 - \max(p_{mk}). \tag{8}$$

4. Results and Discussion

4.1. Evaluation Metrics. In our experiments, we adopted several evaluation metrics, such as precision (P), recall (R), F-measure (F), and accuracy. Let TP, FP, and FN denote true positives, false positives, and false negatives, respectively, and these evaluation metrics are defined as

$$\begin{aligned} P &= \frac{TP}{TP + FP}, \\ R &= \frac{TP}{TP + FN}, \\ F &= 2.\frac{P.R}{P + R}. \end{aligned} \tag{9}$$

To evaluate the quality of attack detection models, we use the model accuracy as a primary evaluation metric that is directly computed from the confusion matrix based on the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{10}$$

We evaluate the performance of these resampling methods through macroaverage $F1$-score performed independently on each class. Let MAP and MAR denote macroaverage precision and macroaverage recall, and the macroaverage $F1$-score is defined as

$$\text{MacroAvg}F1 = 2 * \frac{\text{MacroAvgPrec} * \text{MacroAvgRec}}{\text{MacroAvgPrec}^{-1} + \text{MacroAvgRec}^{-1}}, \tag{11}$$

where

$$\begin{aligned} \text{MacroAvgPrec} &= \frac{\sum_{k=1}^{K} \text{Precision}_k}{K}, \\ \text{MacroAvgRec} &= \frac{\sum_{k=1}^{K} \text{Recall}_k}{K}, \end{aligned} \tag{12}$$

where K is the total number of classes.

4.2. Dataset and Attack Class Balancing. We assessed MidSiot on three different datasets: IoTID20, CIC-IDS-2017, and BOT-IoT.

(i) The IoTID20 dataset consists of two IoT devices (a smart home device SKT NGU and an EZVIZ Wi-Fi camera) and several non-IoT devices marked as external devices. The cyberattacks on these devices are classified into four attack categories and seven attack subcategories described in detail in Table 3.

(ii) The CIC-IDS-2017 dataset contains the network traffic of six cyberattack types listed in Table 4 targeting 12 different IoT devices, which are labeled according to their operating systems and architectures. It also has several external devices to generate normal traffic.

(iii) The BOT-IoT has five IoT devices and several external devices. The malicious network traffic of these devices is classified into three attack categories and detailed in Table 5.

Through rigorously analyzing the evaluation datasets, we figured out that the number of samples of each attack type is slightly imbalanced; thus, employing resampling methods is necessary. In our experiments, we experimented and evaluated the Random Undersampling algorithm (RU) and its conjunction with Synthetic Minority Oversampling Technique (RU-SMOTE).

(i) Random Undersampling is a random selection process running on overwhelmed attack types to reduce their size. However, randomly selecting data points might accidently ignore critical information, resulting in degraded classification performance.

Table 3: Devices, attack categories, and subcategories of the IoTID20 dataset.

Device	Category	Subcategory
EZVIZ, NUGU, External	Normal, DoS, Mirai, MITM, Scan	Normal, Syn Flooding, Brute Force, HTTP Flooding, UDP Flooding, ARP Spoofing, Host Port, OS

Table 4: Devices, attack categories, and subcategories of the CIC-IDS-2017 dataset.

Device	Category	Subcategory
Web server 16 Public, Ubuntu server 12 Public, Ubuntu 14.4 32bit, Ubuntu 14.4 64bit, Ubuntu 16.4 32bit, Ubuntu 16.4 64bit, Win 7 Pro, Win 8.1 64bit, Win Vista 64bit, Win 10 pro 32bit, win 10 64bit, MAC, External	Benign, Bot, Brute Force, Dos/Ddos, Infiltration, Portscan, Web Attack	Benign, Bot, FTP-Patator, SSH-Patator, DDoS, DoS, GoldenEye, DoS Hulk, DoS Slow, httptest, DoS slowloris, Infiltration, Portscan, Web Attack-Brute Force, Web Attack-Sql Injection, Web Attack-XSS, Heartbleed

Table 5: Devices, attack categories, and subcategories of the BOT-IoT dataset.

Device	Category	Subcategory
Ubuntu Server, Ubuntu Mobile, Metasploitable, Windows 7, Ubuntu Tap, External	Normal, DoS/DDoS, Reconnaissance, Theft	Normal, Service scanning, OS Fingerprinting, DoS/DDoS TCP, DoS/DDoS UDP, DoS/DDoS HTTP, Keylogging, data exfiltration

(ii) Synthetic Minority Oversampling Technique (SMOTE) balances the dataset by synthesizing new samples for the minority class. In more detail, it selects a cluster of samples and draws a line between them; new samples are the points along this line.

To implement these resampling methods, we utilized *imblearn* library [33] supporting multiple resampling techniques along with several running strategies for both binary and multiclass classification. In binary classification, we need to configure the ratio between minority class and majority class after resampling, whereas this configuration is unnecessary in multiclass classification. From the results illustrated in Figures 4-6, it is obvious that these resampling techniques have no significant impact on the performance of models. Furthermore, if the number of samples in majority classes drastically outweighs ones of minority classes, they may decrease the attack detection quality of the smaller classes. Therefore, our models are trained without resampling techniques.

4.3. Results and Discussion. To select classification algorithms for our multistage IDS, we examined the detection time and accuracy of several supervised machine learning algorithms on the IoTID20 dataset. In detail, the ability to classify benign and malicious network traffic (binary attack detection) and identify precisely the types of attacks (multiclass attack detection) are both considered. Since operating on network gateways requires a lightweight attack detection model, experimented algorithms are simple machine learning algorithms, including linear support vector machine, quadratic support vector machine, K-nearest-neighbor, linear discriminant analysis, quadratic discriminant analysis, multilayer perceptron, long short-term memory, autoencoder classifier, and decision tree classifier; their results are presented in Table 6. As shown in the table, the decision tree classifier outperforms other algorithms, and it is considered a lightweight machine learning algorithm [34]. This classifier is thus selected for the attack detection model of MidSiot. Note that empty values in the table (denoted by N/A) imply the long training time exceeding two hours. Moreover, training a decision tree classifier is trivial and possibly performed on IoT devices. We also applied *Classification and Regression Tree* (CART) to boost the detection performance. In more detail, CART splits training data into two subsets based on a specific feature k and a threshold t_k (e.g., "flow duration $\leq$ 100"). This split is repeated on each subset until it reaches the maximum depth or subset size equals to 0. As a result, the computational complexity of the classifier is reduced to $O(\log_2(m))$ with m

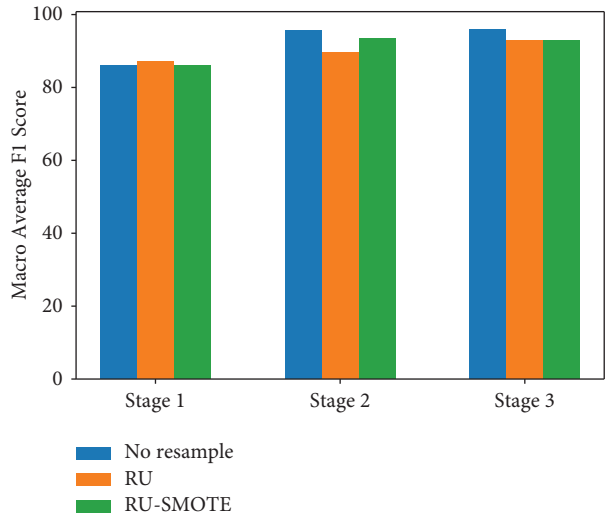

Figure 4: The macroaverage $F1$-score of the IoTID20 dataset.

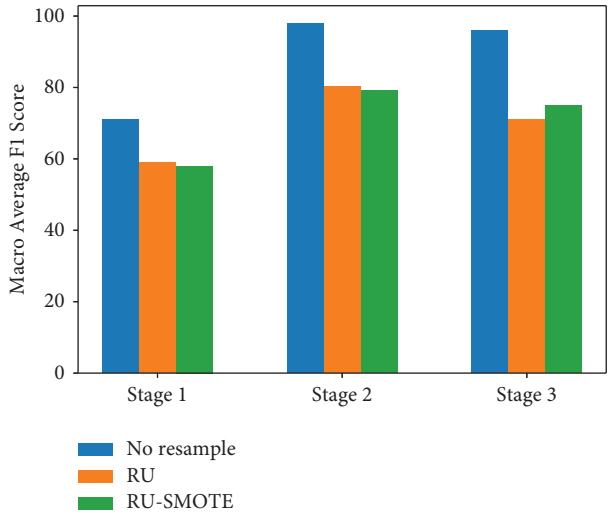

Figure 5: The macroaverage $F1$-score of the CIC-IDS-2017 dataset.

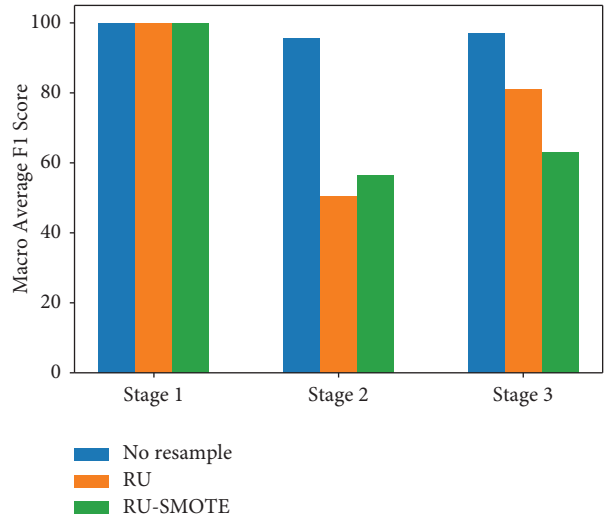

Figure 6: The macroaverage $F1$-score of the BOT-IoT dataset.

Table 6: The results of examined classifiers.

Model	Classification	Accuracy (%)	Training time (μs/flow)	Prediction time (μs/flow)	Adjusted parameters
Linear support vector machine	Binary	98.16	1150.57	204.54	kernel = linear
	Multiclass	N/A	10 298.3	1453.1	gamma = auto
Quadratic support vector machine	Binary	98.25	792.61	311.5	kernel = poly
	Multiclass	N/A	N/A	N/A	gamma = auto
K-Nearest neighbor	Binary	99.79	0.17	2343.72	n_neighbors = 5
	Multiclass	98.61	0.19	2377.81	
Linear discriminant analysis	Binary	95.07	21.6	0.27	All default
	Multiclass	80.73	25.71	4.35	
Quadratic discriminant analysis	Binary	53.6	18.89	12.44	All default
	Multiclass	56.62	16.62	14.74	
Multilayer perceptron	Binary	99.6	2.57	0.65	Input layer and first layer with 50 neurons and activation = relu Output layer with activation = sigmoid
	Multiclass	92.71	4.94	7.96	
Long short-term memory	Binary	96.53	572.44	62.64	input layer and LSTM layer with 50 neurons Output layer with activation = sigmoid
	Multiclass	N/A	N/A	N/A	
Autoencoder classifier	Binary	93.01	11.65	0.62	Encoding layer with 50 neurons and activation = relu Decoding and output layer with activation = softmax
	Multiclass	87.74	13.35	0.81	
Decision tree classifier	Binary	99.94	12.49	0.43	criterion = entropy
	Multiclass	99.69	16.76	0.38	

Table 7: The evaluation results of the multistage IDS.

Dataset	Stage	Accuracy (%)	Training time (μs/flow)	Prediction time (μs/flow)
IoTID20	1	92.57	35.76	0.44
	2	99.95	11.33	0.26
	3	99.15	15.9	0.31
CIC-IDS-2017	1	94.17	110.31	0.89
	2	99.99	25.65	0.25
	3	99.97	78.68	0.27
BOT-IoT	1	99.92	19.27	0.15
	2	99.99	3.24	0.07
	3	99.93	15.05	0.17

being the number of samples in the training set. This significantly increases the training and prediction rate to deal with large datasets.

Attack detection results: Table 7 reports the overall performance of each MidSiot's stage on evaluated datasets. We can see that MidSiot could not only accurately differentiate between normal and malicious traffic in the second stage, but also identify the type of attacks in the third stage. The average accuracy of such stages is about 99.98% and 99.68%, respectively. Regarding classifying IoT devices, our proposal achieved a high classification accuracy reported at 95.55% on average. In detail, device-type classification for BOT-IoT achieves the highest result, at 99.92%, whereas the results of IoTID20 and CIC-IDS-2017 are about 92.57% and 94.17%, respectively. To have a better understanding the attack detection performance of MidSiot, Tables 8–10 illustrate the confusion matrices, which present the comparison between predicted attacks and the actual ones.

Comparing with baseline methods: We compared attack detection quality between our proposal and state-of-the-art IDS and reported the results in Table 11. Overall, MidSiot outperforms its competitors on CICIDS-2017 and BOT-IoT datasets and is comparable with them on IoTID2020 dataset. In more detail, regarding CIC-IDS-2017, the best of our competitors achieves 99.9% in both binary and multiclass classification, whereas our proposed IDS achieves better results recorded about 99.99% and 99.97%, respectively. Similar results are also found in the BOT-IoT dataset, in which our proposal achieves 99.99% accuracy in binary classification and 99.93% in multiclass classification problems. In the IoTID20 dataset, the best competitor detects the attack types with 100% accuracy, and MidSiot also has very competitive results reported at 99.15%. Compared with state-of-the-art IDSs, MidSiot employed more machine learning models to enhance detection accuracy. This demands high computation costs and training datasets to train these models. Indeed, each stage in MidSiot has a different training dataset, which requires a huge effort to label. For example, the first-stage model needs to label the device type of network traffic, whereas the second-stage model demands labeling abnormal traffic. Moreover, deploying the first-stage model from the cloud to IoT local gateway consumes network bandwidth and may trigger delays.

Table 8: Attack detection confusion matrix on the IoTID20 dataset.

		Predicted				
		Mirai	Scan	DoS	Normal	MITM ARP spoofing
Actual	Mirai	412446	2583	0	38	569
	Scan	618	74 473	0	9	155
	DoS	9	2	59356	5	3
	Normal	63	15	6	53	30
	MITM ARP Spoofing	624	200	2	16	34509

Table 9: Attack detection confusion matrix on the CIC-IDS-2017 dataset.

		Predicted				
		Benign	DoS/DDoS	Portscan	Bot	Web attack
Actual	Benign	158303	6	2	7	35
	DoS/DDoS	19	235881	8	0	2
	Portscan	6	6	158911	0	1
	Bot	10	0	0	1929	0
	Web Attack	40	1	2	0	1463

Table 10: Attack detection confusion matrix on the BOT-IoT dataset.

		Predicted			
		DDoS	DoS	Reconnaissance	Normal
Actual	DDoS	1926189	435	0	0
	DoS	1319	830786	0	0
	Reconnaissance	2	6	81818	2
	Normal	1	1	3	2

Table 11: Comparing the proposed system with other one-stage systems.

Dataset	Work	Attack detection model	Binary attack detection (%)	Multiclass attack detection (%)	Prediction time (μs/netflow)
CICIDS-2017	MidSiot	Multistage	99.99	99.97	1.41
	Gamage and Samarabandu [35]	Random forest	N/A	99.86	60.48
	Vinayakumar et al. [36]	Deep neural network	93.10	95.60	N/A
	Elmrabit et al. [37]	Decision tree	99.90	99.90	N/A
	Manimurugan	Deep belief network	99.37	97.73	N/A
IOTID20	MidSiot	Multistage	99.98	99.88	1.01
	Ullah and Mahmoud [21]	Decision tree	99.94	99.69	N/A
	Alkahtani and Aldhyani [38]	Long short-term memory	98.20	N/A	N/A
	Song et al. [39]	Autoencoder	93.76	95.20	N/A
	Hussein et al. [40]	Random forest	99.90	99.90	N/A
	Ullah and Mahmoud [41]	Convolution neural network	99.98	97.76	N/A
	Islam et al. [42]	Decision tree	N/A	100.00	1139.37
BOT-IOT	MidSiot	Multistage	99.99	99.99	0.39
	Ferrag et al. [43]	Rules and decision tree	N/A	97.00	1.54
	Ferrag et al. [44]	Deep autoencoder	N/A	98.39	1916.55
	Dwibedi et al. [45]	Support vector machine	99.99	N/A	N/A
	Pokhrel et al. [46]	K-Nearest neighbor	92.10	N/A	N/A
	Ge et al. [47]	Support vector machine	99.74	99.03	693 040
	Ullah and Mahmoud [41]	Convolution neural network	99.90	99.97	N/A

In conclusion, by using a hierarchical architecture and chaining stages together, MidSiot effectively classifies device types, identifies abnormal network traffic, and differentiates cyberattack types.

5. Conclusion

In this article, we proposed a distributed intrusion detection system for IoT scenarios, in which connected devices are not only resource-constraint but also heterogeneous in hardware specification. To accurately detect various types of cyber-attacks, the proposed IDS consists of three stages: (1) classifying device types; (2) detecting malicious network flows; and (3) identifying attack types. In the experiments on three popular IOT-IDS datasets (IoTID20, CIC-IDS-2017, and BOT-IoT), we demonstrated that our proposal could detect several attacks with an accuracy of 99.68% on average and outperforms state-of-the-art IDSs. In addition, we examined two resampling techniques to balance the datasets and discovered that these techniques slightly reduce the detection rate of minority attack types. In short, MidSiot is beneficial for both the industrial and research communities interested in further developing intrusion detection systems for IoT.

Data Availability

The training data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the fund supporting research activities from the University of Information Technology, Vietnam National University, Ho Chi Minh City.

References

[1] W. a. Kassab and K. A. Darabkh, "A-Z survey of internet of things: architectures, protocols, applications, recent advances, future directions and recommendations," *Journal of Network and Computer Applications*, vol. 163, Article ID 102663, 2020.

[2] A. Menard, "How can we recognize the real power of the internet of things," *Advanced Robotics*, vol. 1, pp. 4-5, 2017.

[3] J. Sengupta, S. Ruj, and S. Das Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot," *Journal of Network and Computer Applications*, vol. 149, Article ID 102481, 2020.

[4] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban ids: an intelligent anomaly-based intrusion detection system for iot edge devices," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6882–6897, 2020.

[5] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *Proceedings of the MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pp. 267–272, IEEE, Baltimore, MD, USA, October 2017.

[6] G. Gallopeni, B. Rodrigues, M. Franco, and B. Stiller, "A practical analysis on mirai botnet traffic," in *Proceedings of the 2020 IFIP Networking Conference (Networking)*, pp. 667-668, IEEE, Espoo, Finland, June 2020.

[7] A. Gangwar and S. Sahu, "A survey on anomaly and signature based intrusion detection system (ids)," *International Journal of Engineering Research and Applications*, vol. 4, no. 4, 2014.

[8] C. Zhang, J. Jiang, and M. Kamel, "Intrusion detection using hierarchical neural networks," *Pattern Recognition Letters*, vol. 26, no. 6, pp. 779–791, 2005.

[9] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," *Lecture Notes in Computer Science*, Springer, in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 203–222, September 2004.

[10] M. Xie, J. Hu, S. Han, and H.-H. Chen, "Scalable hypergrid k-nn-based online anomaly detection in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1661–1670, 2012.

[11] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," 2018. arXiv preprint arXiv:1802.09089.

[12] K. Ince, "A novel approach for intrusion detection systems: V-ids," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 29, no. 4, pp. 1929–1943, 2021.

[13] V. Kumar, A. K. Das, and D. Sinha, "Uids: a unified intrusion detection system for iot environment," *Evolutionary Intelligence*, vol. 14, no. 1, pp. 47–59, 2021.

[14] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[15] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[16] Z. Liu, N. Thapa, A. Shaver, K. Roy, X. Yuan, and S. Khorsandroo, "Anomaly detection on iot network intrusion using machine learning," in *Proceedings of the 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–5, IEEE, KwaZulu Natal, South Africa, August 2020.

[17] H. Kaur, G. Singh, and J. Minhas, "A review of machine learning based anomaly detection techniques," 2013. arXiv preprint arXiv:1307.7286.

[18] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[19] M. A. Aydın, A. H. Zaim, and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," *Computers & Electrical Engineering*, vol. 35, no. 3, pp. 517–526, 2009.

[20] M. Gajewski, J. M. Batalla, G. Mastorakis, and C. X. Mavromoustakis, "A distributed ids architecture model for smart home systems," *Cluster Computing*, vol. 22, no. 1, pp. 1739–1749, 2019.

[21] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Proceedings of the Canadian Conference on AI*, pp. 508–520, Ottawa, Ontario, May 2020.

[22] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108–116, 2018.

[23] B. Silva, R. Silveira, M. Silva Neto, P. Cortez, and D. Gomes, "A comparative analysis of undersampling techniques for network intrusion detection systems design," *Journal of Communication and Information Systems*, vol. 36, no. 1, pp. 31–43, 2021.

[24] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *Journal of Big Data*, vol. 8, no. 1, pp. 1–41, 2021.

[25] P. Bedi, N. Gupta, and V. Jindal, "I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems," *Applied Intelligence*, vol. 51, no. 2, pp. 1133–1151, 2021.

[26] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: a review," *IEEE Access*, vol. 6, pp. 10179–10188, 2018.

[27] W. Li, Y. Wang, J. Li, and M. H. Au, "Towards blockchained challenge-based collaborative intrusion detection," in *Applied Cryptography and Network Security Workshops*, J. Zhou, R. Deng, Z. Li et al., Eds., Springer International Publishing, Cham, Switzerland, pp. 122–139, 2019.

[28] W. Li, Y. Wang, J. Li, and M. H. Au, "Toward a blockchain-based framework for challenge-based collaborative intrusion detection," *International Journal of Information Security*, vol. 20, no. 2, pp. 127–139, 2021.

[29] W. Li, S. Tug, W. Meng, and Y. Wang, "Designing collaborative blockchained signature-based intrusion detection in iot environments," *Future Generation Computer Systems*, vol. 96, pp. 481–489, 2019.

[30] R. M. A. Ujjan, Z. Pervez, and K. Dahal, "Snort based collaborative intrusion detection system using blockchain in sdn," in *Proceedings of the 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 1–8, Ukulhas, Maldives, August 2019.

[31] W. Fan, Y. Park, S. Kumar, P. Ganta, X. Zhou, and S.-Y. Chang, "Blockchain-enabled collaborative intrusion detection in software defined networks," in *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 967–974, Guangzhou, China, November 2020.

[32] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, 2020.

[33] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[34] R. Bikmukhamedov and A. Nadeev, "Lightweight machine learning classifiers of iot traffic flows," in *Proceedings of the 2019 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, pp. 1–5, IEEE, Minsk, Belarus, July 2019.

[35] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: a survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, Article ID 102767, 2020.

[36] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[37] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *Proceedings of the 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, IEEE, Dublin, Ireland, June 2020.

[38] H. Alkahtani and T. H. Aldhyani, "Intrusion detection system to advance internet of things infrastructure-based deep learning algorithms," *Complexity*, vol. 2021, Article ID 5579851, 18 pages, 2021.

[39] Y. Song, S. Hyun, and Y.-G. Cheong, "Analysis of autoencoders for network intrusion detection," *Sensors*, vol. 21, no. 13, p. 4294, 2021.

[40] A. Y. Hussein, P. Falcarin, and A. T. Sadiq, "Enhancement performance of random forest algorithm via one hot encoding for iot ids," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 9, no. 3, pp. 579–591, 2021.

[41] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in iot networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.

[42] N. Islam, F. Farhin, I. Sultana et al., "Towards machine learning based intrusion detection in iot networks," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 1801–1821, 2021.

[43] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "Rdtids: rules and decision tree-based intrusion detection system for internet-of-things networks," *Future Internet*, vol. 12, no. 3, p. 44, 2020.

[44] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Article ID 102419, 2020.

[45] S. Dwibedi, M. Pujari, and W. Sun, "A comparative study on contemporary intrusion detection datasets for machine learning research," in *Proceedings of the 2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 1–6, IEEE, Arlington, VA, USA, November 2020.

[46] S. Pokhrel, R. Abbas, and B. Aryal, "Iot security: botnet detection in iot using machine learning," arXiv preprint arXiv: 2104.02231, 2021.

[47] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for iot networks," in *Proceedings of the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 256–25609, IEEE, Kyoto, Japan, December 2019.