electronics

MDPI

Article

Towards a Lightweight Detection System for Cyber Attacks in the IoT Environment Using Corresponding Features

Yan Naung Soe [1,2], Yaokai Feng [3,*], Paulus Insap Santosa [2,*], Rudy Hartanto [2] and Kouichi Sakurai [1]

[1] Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan; yan.naung.s@mail.ugm.ac.id (Y.N.S.); sakurai@inf.kyushu-u.ac.jp (K.S.)
[2] Department of Electrical and Information Engineering, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia; rudy@ugm.ac.id
[3] Department of Advanced Information Technology, Kyushu University, Fukuoka 819-0395, Japan
* Correspondence: fengyk@ait.kyushu-u.ac.jp (Y.F.); insap@ugm.ac.id (P.I.S.); Tel.: +81-92-802-3640 (Y.F.)

Received: 8 December 2019; Accepted: 8 January 2020; Published: 11 January 2020

Abstract: The application of a large number of Internet of Things (IoT) devices makes our life more convenient and industries more efficient. However, it also makes cyber-attacks much easier to occur because so many IoT devices are deployed and most of them do not have enough resources (i.e., computation and storage capacity) to carry out ordinary intrusion detection systems (IDSs). In this study, a lightweight machine learning-based IDS using a new feature selection algorithm is designed and implemented on Raspberry Pi, and its performance is verified using a public dataset collected from an IoT environment. To make the system lightweight, we propose a new algorithm for feature selection, called the correlated-set thresholding on gain-ratio (CST-GR) algorithm, to select really necessary features. Because the feature selection is conducted on three specific kinds of cyber-attacks, the number of selected features can be significantly reduced, which makes the classifiers very small and fast. Thus, our detection system is lightweight enough to be implemented and carried out in a Raspberry Pi system. More importantly, as the really necessary features corresponding to each kind of attack are exploited, good detection performance can be expected. The performance of our proposal is examined in detail with different machine learning algorithms, in order to learn which of them is the best option for our system. The experiment results indicate that the new feature selection algorithm can select only very few features for each kind of attack. Thus, the detection system is lightweight enough to be implemented in the Raspberry Pi environment with almost no sacrifice on detection performance.

Keywords: IoT; DDoS attack; feature selection; IDS; machine learning; Raspberry Pi

1. Introduction

1.1. Intrusion Detection System (IDS)

A vast number of Internet of Things (IoT) devices have been deployed in many applications as a result of the significant development of related technologies. At the same time, the problem of cyber-attacks has become a challenging issue. This is because most IoT devices have very limited resources (e.g., storage and computation capacity); thus, they cannot carry out complicated intrusion detection systems. It was said that global mobile data traffic increased by about 71% in 2017 over the previous year, and almost all of the mobile traffic data will originate from smart devices by 2022 [1]. The number of IoT malware in 2017 increased about sevenfold compared with that in 2013. The peak

of cyber-attack traffic launched by a huge number of Mirai-infected IoT devices reached 620 Gbps [2]. There were 10,263 botnets hosted in different IoT devices identified in 2018 [3]. Another distributed denial of services (DDoS) attack from compromised IoT devices, called "IoTroop," was discovered by Check Point in 2017 [4]. There were 13,000 IoT devices involved in this powerful attack targeting financial sectors [4].

Figure 1 shows the typical intrusion detection system for the IoT environment. The detection system monitors the IoT components, and it will generate the alert to the users or the response modules to process against the suspected threats as soon as it is decided that the traffic pattern is an attack. The machine learning-based approach needs to train the detection model using a suitable learning classifier. Almost all misuse-based approaches use a pattern matching algorithm to determine whether or not the incoming traffic pattern is an attack. All the above-mentioned systems need to capture the normal traffic patterns or/and the attack signatures patterns for building their detection systems.

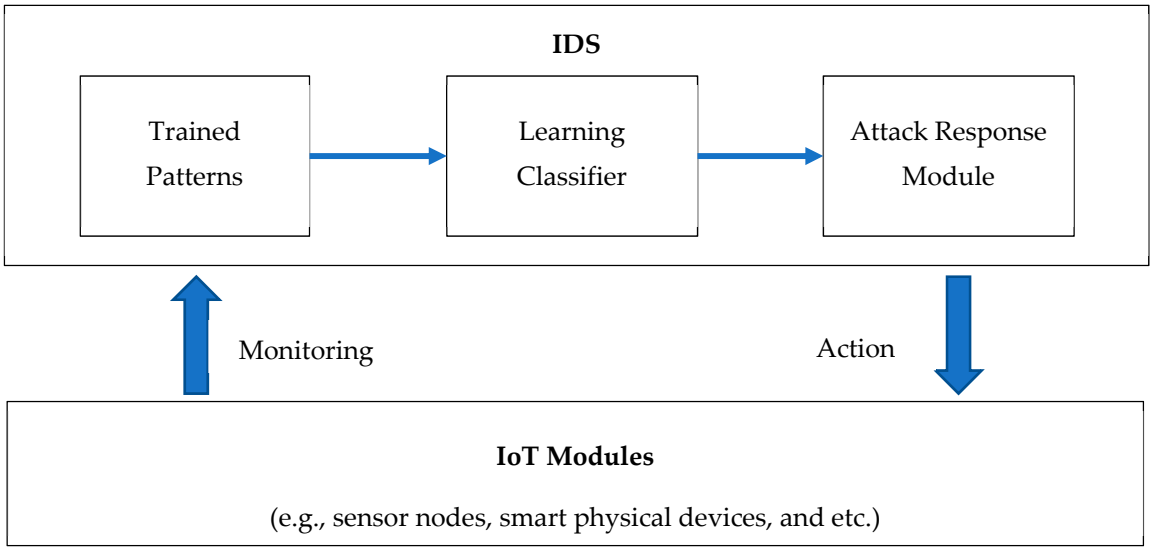


Figure 1. Typical intrusion detection system for IoT environment. IDS, intrusion detection system.

The intrusion detection system can be divided into the host-based detection system (HIDS) and the network-based detection system (NIDS). Formally, the traditional IDSs cannot be hosted on ordinary IoT devices because of limitations in the resources of such devices. Thus, the network-based IDS will be the only possible way to implement the detection system for the IoT environment. Anomaly detection and misuse-based approaches are the primary detection mechanisms in IDSs. Misuse-based detection systems are implemented using predefined attack signatures. Thus, they cannot detect new kinds of attacks and new variants of known attacks [5]. Although anomaly detection systems may be effective for new kinds of attacks, the problem of the high false positive rate is one of the main challenges [6,7]. Moreover, the presence of so many different natures of a wide variety of IoT devices is another challenge for implementing an anomaly detection system. Additionally, the IoT devices-based IDSs must be lightweight because of their computation and storage capacity. Most of the early IDSs were deployed in powerful personal computers or servers. However, there are numerous IoT devices rapidly developing in recent years. These devices are inexpensive, portable, and can be used everywhere. Moreover, these devices only need low power consumption. Therefore, it is not suitable to use highly powerful computers to protect against the cyber-attacks in the IoT environment because they have more power consumption rather than many IoT devices, and the IDS deployment cost can be higher than the setting up of the IoT devices for a small environment. Therefore, the researchers are more interested in deploying the IDS on the resource-constraint devices, like Raspberry Pi. In this study, we implement the machine learning-based lightweight IDS on Raspberry Pi to protect

against the specific attacks on the IoT environment. We also proposed the new feature selection method, named correlated-set thresholding on gain-ratio (CST-GR), to make the system lightweight.

1.2. Common Attacks in the IoT Environment

The most challenging of IoT security is the botnet attacks, like Bashlite, Mirai, Hajime, and so on. The botnet attacks can be classified into DDoS, pushing, identify theft, information leakage, and keylogging [8]. The botmasters conduct port scanning to find the vulnerabilities in them to infect these devices by port scanning, as well as to perform remote mapping networks for gathering the information by operating system fingerprinting (OS fingerprinting). Most of the botnets launched the DDoS attacks to disrupt a service, and to make it unavailable to legitimate users [6]. DDoS attacks can occur in both the network layer and the application layer in the IoT environment [9]. Data theft and privacy issues are the specific security issues in the application layer of the IoT environment [9]. The study [10] addressed the challenges of data disclosure and privacy violations to the users. Thus, we mainly focus on the detection of the following three specific attacks in the IoT environment:

1. Probing attacks (reconnaissance): These are malicious activities to gather information about the targets through remote scanning. They are often categorized by the two subclasses of port scanning and OS fingerprinting.
2. DDoS attacks: These are launched collaboratively by many compromised hosts (called bots). Such attacks try to disrupt the availability of services to legitimate users.
3. Information theft attacks: In these attacks, an adversary seeks to obtain sensitive data. They can be subcategorized into data theft and keylogging.

1.3. Our Contributions

Our main contributions in this paper are as follows.

1. A machine learning (ML)-based lightweight IDS is proposed and implemented on a Raspberry Pi system.
2. To overcome the challenges of the resource constraint problem, a novel feature selection algorithm called correlated-set thresholding on gain-ratio (CST-GR) is proposed for selecting essential features. In our experiment, the number of features is greatly reduced by this algorithm.
3. The essential features are selected for each specific kind of attack. Thus, good detection performance can be expected.
4. The detection performance of our proposal is examined in detail using the botnet dataset, Bot-IoT [6], which is collected in a simulated IoT environment. We observe that the CST-GR algorithm can significantly reduce the processing time with almost no sacrifice on detection accuracy. We also observe that, without the help of the CST-GR algorithm, Raspberry Pi cannot handle the entire dataset used in our experiments.
5. We try several tree-based classifiers—J48, Hoeffding tree (VFDT, very fast decision tree), logistic model tree (LMT), and random forest (RF)—to determine which classifier is the most suitable to the IoT environment in terms of lightweight and detection performance. According to our discussion and experimental results, the J48 algorithm is found to be the most suitable for our detection system.
6. We make sure to what degree the processing time can be decreased for training and testing if the Raspberry Pi device is used in multithreading mode.

1.4. Organization of the Paper

The rest of the paper is organized as follows. After the related works are reviewed briefly in Section 2, the new feature selection algorithm (CST-GR) and the general flow of our entire detection system are explained in Section 3. In the samet, some tree-based learning classifiers that we use in our

system are introduced. In Section 4, the performance of our detection system is verified using a public dataset collected in a simulated IoT environment. Finally, we conclude the paper in Section 5.

2. Related Works

2.1. Public IDS

Most of the existing public IDSs such as Snort, Suricata, and Bro are based on pattern matching and static approaches. Snort, provided by Cisco Systems [11], is a leading network-based IDS. The earlier version of Snort only supported single-threading architecture, but from version 2.9, it can support multi-threading. The Suricata, an open-source IDS/IPS solution [12], is fully supported for the multi-threading architecture. However, it is more suitable for a large-scale network architecture. A study [13] also gives an introduction to the ability of Suricata. Although the Snort is lighter than Suricata, it still has a limitation on the number of rules exploited for attack detection when it is implemented on resource constraint devices, like Raspberry Pi [14].

A research work [7] proposed a machine learning-based detection system to extend Snort. This work also made a testbed performance comparison between Snort and Suricata using different types of malicious traffics, which are collected by the Metasploit framework. After detecting seven types of malicious traffics by the two kinds of public IDSs, the first conclusion was that Suricata required more memory and CPU processing capability than Snort. Another finding was that Snort's rule set could detect six out of seven kinds of attacks, but Suricata could detect only four kinds. Moreover, there were many false alarm rates in both IDS systems: Snort has 55.2% false positive rate (FPR), and Suricata generated up to 74.3% FPR. After adding their machine learning-based detection plug-in to Snort, which was done in parallel with the Snort rule detection engine, this system could reduce the FPR to 16.9% by support vector machine (SVM) and 8.6% FPR by the hybrid approach, which was done by optimized SVM with fuzzy logic. According to their experiments, the hybrid detection system constructed by the signature-based approach and ML-based approach was effective. However, their proposal was complicated by implementation in the resource constraint device, like the Raspberry Pi.

2.2. Machine Learning-Based IDS for the IoT Environment

In recent years, as more and more IoT devices are deployed, IDSs in IoT environments have attracted attention from many researchers and developers. Some studies [15,16] addressed specific types of threats targeting IoT devices. One study [15] proposed a detection system for sinkhole attacks targeting routing devices. The detection rate for sinkhole attacks was up to 92% and 72% on fixed and mobile scenarios, respectively. Another study [16] tried to prevent the three different levels of battery exhaustion attacks on the bluetooth low energy (BLE)-based mesh network.

A novel IDS for the IoT environment was proposed using not only ML methodologies, but also a rule-based architecture [17]. Their effort was for both prediction of the malicious behavior and detection of the malicious IoT nodes on the network from DDoS attacks. They used the naïve Bayes as the classifier, and their performance evaluation was done on the open-source tool Weka. ML-based forensic mechanism for IoT botnets was proposed in a study [18], which also used Weka to examine their detection accuracy.

In many existing studies, the IDSs are implemented for the IoT environment using machine learning techniques. This is considered to outperform signature-based systems because the attackers can circumvent the normal signature-based detection techniques. However, the existing IDSs are mainly aimed at obtaining a high detection accuracy with a low false alarm, which is not enough for IoT environments where lightweight IDS systems are also necessary.

2.3. Raspberry Pi-Based IDS

The IDS studies [5,19–27] also have proposed attack detection based on Raspberry Pi. In the work of [18], the anomaly-based detection method was proposed by capturing the previous traffic patterns

asbenign data for attack detection system construction. The work of [21] has implemented a real-time DDoS detection architecture using the complex event processing (CEP) algorithm, and it allowed for real-time analysis of continuous data-streams. Some other studies [14,20,23] have used public IDSs, such as Snort and Bro, to implement the detection system on Raspberry Pi. The study [14] proposed an IDS solution on Raspberry Pi, but its results showed that the number of rules had to be limited owing to their implementation environment, Raspberry Pi. The study [22] also proposed an IDS solution on an IoT router for verifying the domain name system (DNS) traffic generated from IoT devices. An IDS/IPS solution for radio-frequency identification (RFID) using Raspberry Pi was also proposed in the work of [5].

The cyber-attack detection studies based on Raspberry Pi devices are shown in Table 1. Almost all of the Pi-based detection systems used the public IDS, and these are misuse-based systems that used the pre-defined attacks' signatures. Although some public IDS-based studies have indicated that the detection system could be implemented on Raspberry Pi, such systems are mainly simplified ones (e.g., using decreased rule-base) of IDSs in the traditional network. Therefore, such systems often do not have satisfactory detection performance. How to implement a lightweight and efficient detection system in the IoT environment has become a crucially important issue. The IDS on fog computing [24] was introduced using new modern datasets, namely Australian Defence Force Academy Linux Dataset (ADFA-LD) and ADFA-Windows Dataset (ADFA-WD) [3,25]. This work [24] also used Raspberry Pi to evaluate the performance of attack detecting model. They examined its performance by setting up two workstation computers and the Raspberry Pi devices. They proposed a Raspberry Pi-based IDS for a home network using different Snort-rules sets. However, it did not focus on the DDoS attacks, which was a well-known main security challenge in the IoT environment.

Table 1. Raspberry Pi-based intrusion detection systems. DoS, denial of services; IoT, Internet of Things.

References	Detection Method	Pi Model	Tools	Threats	Environment
Kyaw et al. [20]	Misuse-based	Pi 2-B	Snort, Bro	SYN flood, ARP spoofing, port scanning	Conventional
Coşar et al. [26]	Misuse-based	-	Snort, Suricata	SYN flood, Smurf, UDP flood	Conventional
Tripathi et al. [27]	Misuse-based	Pi 3-B	Snort	ICMP ping, brute-force	Conventional
Sforzin et al. [14]	Misuse-based	Pi 2-B	Snort	-	IoT
Cardoso et al. [21]	Pattern matching	Pi 3-B	Complex Event Processing	SYN flood, UDP flood, ICMP flood, port scanning	IoT
Zitta et al. [5]	Misuse-based	Pi 3	Suricata	Port scanning	IoT
Sperling et al. [22]	Traffic analyzing	Pi 3-B	Python, DPKT	MITM, DoS, DNS cache poisoning	IoT

2.4. Feature Selection

The feature selection process involves selecting the most important features or reducing the irrelevant features from the original feature set. The irrelevant features or redundant information should be eradicated [28,29] because of such features that can not only raise the computation cost, but also deteriorate the detection performance [30]. The feature selection reduces the effects of noise or irrelevant variables. Thus, feature selection is extremely important for good detection performance [31]. The main approaches for feature selection can be categorized into three groups: filter-based, wrapper-based, and embedded-based approaches. Among them, the filter-based approach is suitable to accomplish lightweight systems because it has a separated learning process that can run on a powerful node,

not small IoT devices. However, efficient features may be diverse for different learning algorithms to achieve high detection performance. One simple filter-based approach is the Pearson correlation coefficient, which can only detect linear dependencies between features and the target class [32]. Other approaches, mutual information (MI) [33] and gain-ratio [34], use the information-theoretic ranking criteria based on the conditional entropy. The significant difficulty of these approaches is to decide the threshold value for removing the unimportant features. Correlation-based feature selection (CFS) [35] can overcome this problem because it can extract the best features subset without having a threshold value. This approach automatically extracts the best subset with the highest value of its merit function based on two criteria: correlated with target class and uncorrelated with each other on the features. However, it can make the prediction accuracy lower if the best merit score is not good enough.

The preliminary of our lightweight IDS proposal [36] was based on the selected features by the CFS algorithm. We selected the same features for nine kinds of attacks in a public dataset, UNSW-NB15 [37], for implementing the detection system. We evaluated the performance comparison with two classifiers—J48 and naive Bayes—for learning our system. We found that the detection accuracy of J48 is better than that of naive Bayes when we used the selected features, and the CFS algorithm could significantly decrease the number of irrelevant features. In this study, we introduce a new feature selection algorithm and the features are selected corresponding to each specific kind of attack.

3. Our IDS Proposal

3.1. A New Algorithm for Feature Selection

In this subsection, we introduce our new feature selection algorithm, called CST-GR algorithm, to select the most effective features for detecting each kind of specific attack. The proposed feature selection algorithm is shown in Algorithm 1. The merit function from CFS algorithm [35] is applied to find the best-correlated feature set. However, unlike the CFS algorithm, such features do not necessarily appear in our final feature set. The gain-ratio is also calculated for each of those features to decide which ones are eventually selected. Briefly speaking, the minimum gain-ratio value in the feature set obtained using the merit function is used as the threshold of gain-ratio to select the final features from the original feature set. That is, not only the merit function, but also the gain-ratio are calculated to determine the final feature set.

The feature selection process is to select the really important features and try to reduce the number of irrelevant/unimportant features to the greatest extent. At the same time, the features should not condense the accuracy of a predictive model [38]. Like the CFS algorithm, the merit function is used to evaluate the relations between the input features and output classes. A feature is regarded as redundant if one or more of the other features are highly correlated with it. In Equation (1), k is the number of features in the current subset, $\overline{r_{cf}}$ refers to the average value of all feature-class correlations, and $\overline{r_{ff}}$ is the average value of all feature-feature correlations. Using the merit function (M_{s_k}), the subset of features with the highest merit value can be generated. A detailed discussion and analysis can be found in the work of [35]. In the CFS algorithm, the selection result is very dependent on the merit function, which is a big problem in practice. To overcome this problem, in our CST-GR algorithm, the gain value of each feature is also taken into account.

$$M_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \quad (1)$$

Algorithm 1 Correlated-Set Thresholding on Gain-Ratio (CST-GR)

Input: Feature Set (FI)
Output: Selected Feature Set (FS)

1. Read all the original features to the set FI.

   FI = $\{f_1, f_2, f_3, \ldots \ldots .. , f_n\}$, n = the number of the original features
2. Extract the feature set having the highest correlation based on the merit function: Equation (1), FC ⊂ FI

   FC = $\{f_1, f_2, \ldots , f_c\}$, c = the number of features in the subset having the best merit value
3. Calculate the minimum gain value of the features in FC, f_{min} (FC)
4. Use Equation (2) to calculate the gain-ratio value of each feature of FI

   GR = $\{\{f_1, v_1\}, \{f_2, v_2\}, \ldots , \{f_n, v_n\}\}$, where v_i is the gain-ratio value of f_i $(1 \leq i \leq n)$
5. From GR, select the features whose gain-ratio values are greater than or equal to f_{min} (FC), FS ⊂ FI

   FS = $\{r_1, r_2, r_3, \ldots \ldots .. , r_s\}$, s = the number of finally selected features
6. Output FS

The gain value of each feature can be calculated using Equation (2), which is given as "GainRatioAttributeEval" in the open-source Weka [39] and is also used to overcome the overfitting problem in C4.5. Equation (3) represents the information generated by splitting the dataset (D) into the v partitions, which correspond to the outcomes on attribute A. The Gain (D) can be calculated by Entropy (S) – Entropy (S, D).

$$Gain\ Ratio\ (GR) = \frac{Gain\ (D)}{SplitInfo_A(D)} \tag{2}$$

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|} \tag{3}$$

In the CST-GR algorithm, all possible features are loaded to the feature set (FI) in step 1. Step 2 reveals the best subset (FC) that possesses the highest merit value, and the minimum gain-ratio value (f_{min}) of the features in FC is calculated in step 3. Step 4 calculates the gain-ratio value (GR) of each feature in the original feature set. Finally, all the features in the original feature set whose GRs are greater than or equal to the minimum gain-ratio value, and f_{min} are selected as the final features in step 5. Finally, the best feature set (FS) is generated in step 6. That is, the minimum GR in the feature set obtained using the merit function is used as the threshold of gain-ratio to select the final features from the original feature set. Thus, our feature selection algorithm takes not only the merit values into account, but also the gain-ratio values (GRs).

3.2. The General Flow of the IDS Proposal

The general schemes of our proposal for training and testing are shown in Figures 2 and 3, respectively. There are multiple classifiers (three in Figures 2 and 3 as examples) constructed for the detection of specific attacks on the IoT environments. In the training phase shown in Figure 2, each classifier is trained using the data containing only the corresponding attacks and normal data. The feature set used in each classifier is selected independently by the CST-GR algorithm. In the test phase shown in Figure 3, the necessary features for all the classifiers are extracted from the test data. Then, the classifiers obtain the features they need. The CST-GR algorithm is used to select the most important features for each kind of attack. The experiment results presented in Section 4 indicate that the CST-GR algorithm can greatly decrease the number of features actually used in the classifiers, which makes the entire detection system lighter. More importantly, making the detection system lighter almost does not degrade its detection performance, which means that the CST-GR is able to find the really efficient features for detecting attacks. Moreover, the system can support updating each classifier or extending the additional classifiers for the detection of new kinds of attacks.

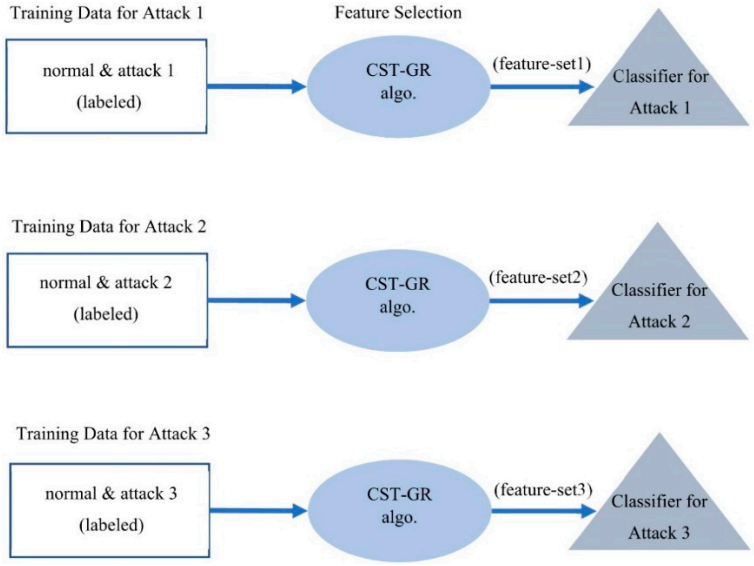


Figure 2. The training phase (the case of three classifiers). CST-GR, correlated-set thresholding on gain-ratio.

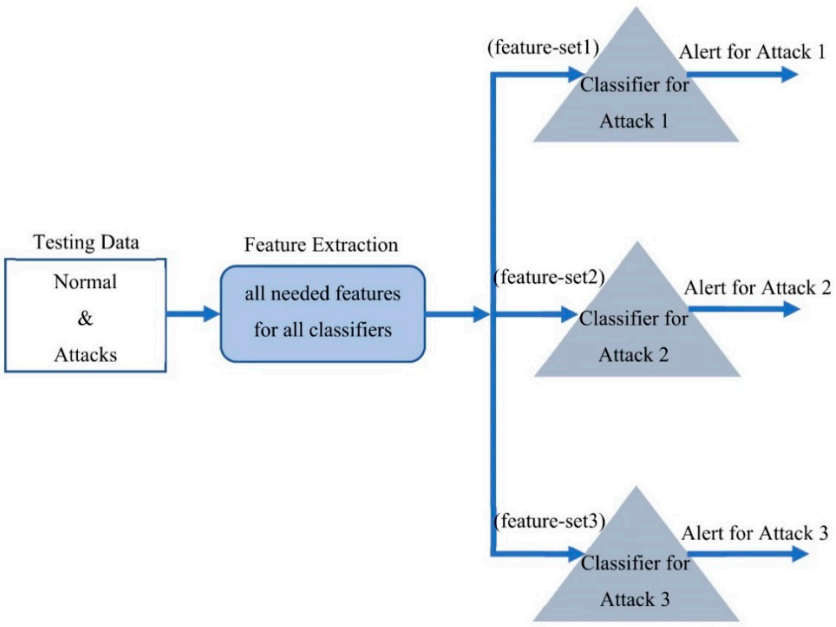


Figure 3. The testing phase (the case of three classifiers).

3.3. Tree-Based Classifiers

We used the well-known tree-based classifiers J48, Hoeffding tree (VFDT), logistic model tree (LMT), and random forest (RF) for implementing our detection system. Tree-based approaches are simple, and most of these classifiers are lightweight processing architecture. These are the type of supervised learning methods for prediction of the learning model by growing the tree(s), and these can be applied for regression and classification problems.

3.3.1. J48

J48 (C4.5) is the descendant of ID3, and it is one of the decision tree generation algorithms developed by Ross Quinlan. It is a tree-like structure that consists of the root node and leaf nodes derived from it [40]. It uses information gain for splitting criteria to construct the tree. The leaf nodes

represent classes or class attributes. It can handle both continuous and discrete features. Using the pruning techniques, the overfitting problem can be solved. It can also be used on training data that have incomplete data and different weighted features.

3.3.2. Hoeffding Tree

The Hoeffding tree is an incremental, anytime decision tree induction algorithm developed by Domingos and Hulten [41]. It is also called VFDT, which means a very fast decision tree. Hoeffding trees exploit the fact that a small sample can be enough to choose the finest splitting attribute. It uses Hoeffding bounds to assure that its output is asymptotically nearly identical to that of a conventional learner. A detailed description can be found in the research [41,42]. The Hoeffding tree can support binary and nominal class, and can also work with mission class values. This algorithm can also work well with nominal and numerical features.

3.3.3. Logistic Model Tree

Logistic model tree (LMT) is the classification of trees with logistic regression functions at the leaves. The algorithm can support not only binary and multi-class target variables, but also numeric, nominal attributes, and missing values. Landwehr et al. [43] proposed the LMT algorithm, and they use a stagewise fitting process to construct the logistic regression models that can select relevant attributes in the data in a natural way. Their method employs the LogitBoost algorithm [44] for building the logistic regression functions at the nodes of a tree and uses the classification and regression tree (CART) algorithm for pruning.

3.3.4. Random Forest

The random forest (RF) is a classifier introduced by Breiman [45] that consists of a collection of decision trees. Each tree is constructed using a random part of the whole feature vector. The final prediction of the RF is obtained by a majority vote, over the predictions of the individual trees. It is also an effective algorithm to overcome the overfitting problem, and the right kind of randomness gives them accurate classification and regression. One hundred trees on the RF were used in this study because it is claimed that the number of trees ranging between 64 and 128 trees was the best option for this algorithm [46].

4. Experiments

Raspberry Pi 3 Model B was used as the experiment platform to implement our lightweight IDS. This device is cheap and small, and thus can be used in many applications. The open-source Weka [39] was used to examine the detection performance of our proposal. Our system tried to capture the following three kinds of specific attacks in the IoT environment: probing attack (reconnaissance), DDoS attack, and information theft attack. The CST-GR algorithm was performed to select the really important features for detecting each specific attack. Tree-based classifiers, J48, VFDT, MLT, and RF, were each tried as the classifier in our detection system to evaluate which was the best one for our system.

4.1. Dataset

The dataset, Bot-IoT, was built and managed in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [6], where the aforementioned three kinds of common attacks in the IoT environment were launched intentionally and the traffic data were collected in a simulated environment based on the IoT botnet scenarios [6]. The node-red tool [47] was used for constructing simulated IoT devices. The Ostinato tool [48] was used to gather benign data. Hping3 [49] and Golden-eye [50] tools were used for DDoS attacks; Nmap [51], hping3 [49], and xprobe2 [52] were used for probing attacks;

qnd Metasploit framework was used to find the vulnerability for data theft [6]. In this dataset, 3.6 million instances were contained, and 40 features were originally extracted.

4.2. Features Selected by the CST-GR Algorithm

Although our system can deal with any specific attacks, the above-mentioned three so-called common attacks in the IoT environment are taken into account in our experiment. The training dataset was divided into three groups according to the three kinds of attacks. That is, each group contained only one specific attack and benign data, to which the CST-GR algorithm was applied. Thus, the selected features are guaranteed to be most suitable for detecting the corresponding attacks.

As the result of the CST-GR algorithm, only two features are selected for the DDoS attacks, five features for the probing attacks (reconnaissance), and three features for the information theft attack. The selected features are shown in Table 2. From Table 2, we can observe that only a very few features are selected for each kind of attack. Thus, each classifier is very small.

Table 2. Selected features by correlated-set thresholding on gain-ratio (CST-GR) for each kind of attack.

Feature Name	Description	Attacks
TnBPDstIP	Total number of bytes per destination IP	DDoS
drate	Destination-to-source packets per second	DDoS
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP	Reconnaissance
AR_P_Proto_P_SrcIP	Average rate per protocol per source IP	Reconnaissance
AR_P_Proto_P_Dport	Average rate per protocol per dport	Reconnaissance, Theft
TnP_PDstIP	Total number of packets per destination IP	Reconnaissance
TnP_PerProto	Total number of packets per protocol	Reconnaissance, Theft
state_number	Numerical representation of feature state	Theft

4.3. Performance Evaluation

The detection performance of our proposal was examined, including processing time and detection accuracy, for each of the three kinds of attacks. The true positive rate (TPR), false positive rate (FPR), precision, and F-measure were examined and presented as detection accuracy. The performance evaluation measurements are shown in Equations (4)–(8), where TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively. The Raspberry Pi device in our experiment could not handle all 3.6 million instances in the original dataset if all 40 features were used. Because the instances of the DDoS attacks and the probing attacks are too many to be handled in our platform (about 1.9 million instances of the DDoS attack and about 83,000 instances of the Probing attack are contained in the original dataset), only a randomly selected part of the instances in the entire datasets of the probing attacks and DDoS attacks was used for the sake of fairness. After selecting features by the CST-GR algorithm, this device could handle the whole dataset without any problem, because the number of selected features is very small.

$$TPR = \frac{TP}{TP + FN} \times 100 \tag{4}$$

$$FPR = \frac{FP}{FP + TN} \times 100 \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \times 100 \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \times 100 \tag{7}$$

$$F - Measure = \left[2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right)\right] \times 100 \tag{8}$$

Specifically, the details of the dataset used in our experiment are shown in Table 3. In order to select features and train a classifier for each of the three kinds of common attacks, the corresponding instances of each kind of attack and the normal instances in the original dataset are taken out to form a sub-dataset for the corresponding attack. For each sub-dataset, around two-thirds of instances are chosen as training data and the remaining parts as testing data. That is, we split 556 instances (367 instances for training and 189 instances for testing) in the information theft attack group, approximately 82,000 instances (54,000 instances for training and 28,000 instances for testing) in the reconnaissance (probing) attack group, and 82,000 instances (54,000 instances for training and 28,000 instances for testing) in DDoS attack group. For the reconnaissance attacks and the DDoS attacks, about 82,000 instances were selected because it was about the most prominent number for our platform to handle in the case of the original 40 features being used. Even though this device could handle more instances in the case of the CST-GR algorithm being used to select features, we must make the performance comparison fair before and after our feature selection algorithm CST-GR was used.

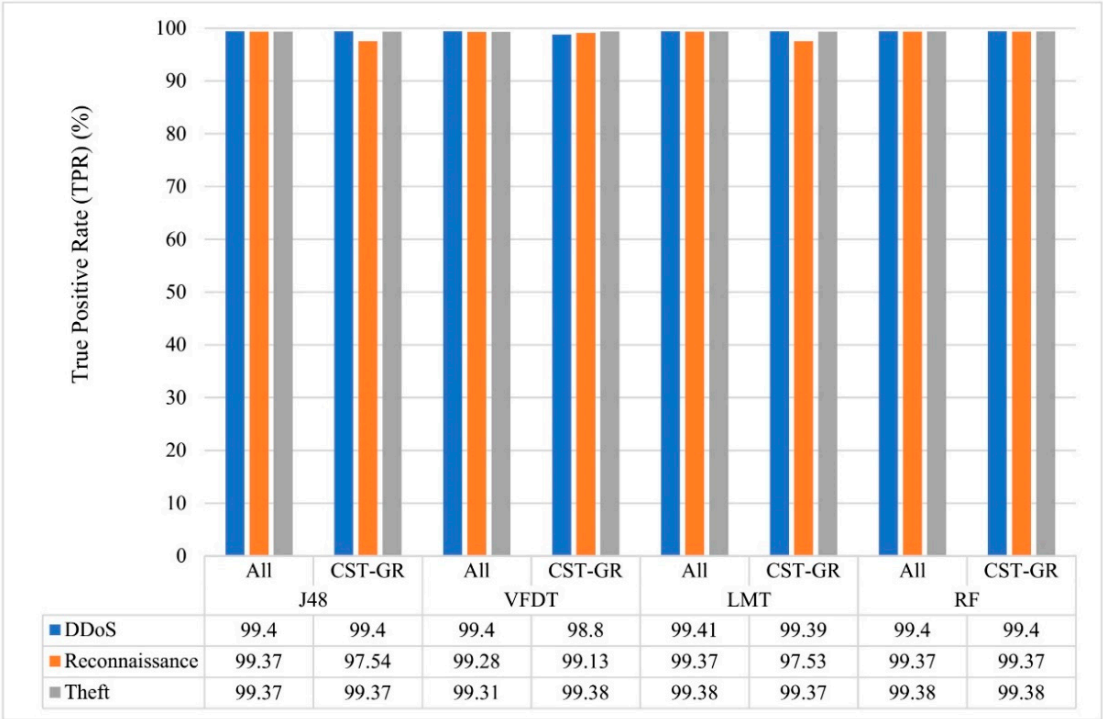
Table 3. The number of instances used in our experiment. DDoS, distributed DoS.

Attack Type	Number of Instances for Training	Number of Instances for Testing	Total Number of Instances
DDoS	54,651	27,326	81,977
Reconnaissance	54,706	27,354	82,060
Theft	367	189	556

4.3.1. Detection Accuracy

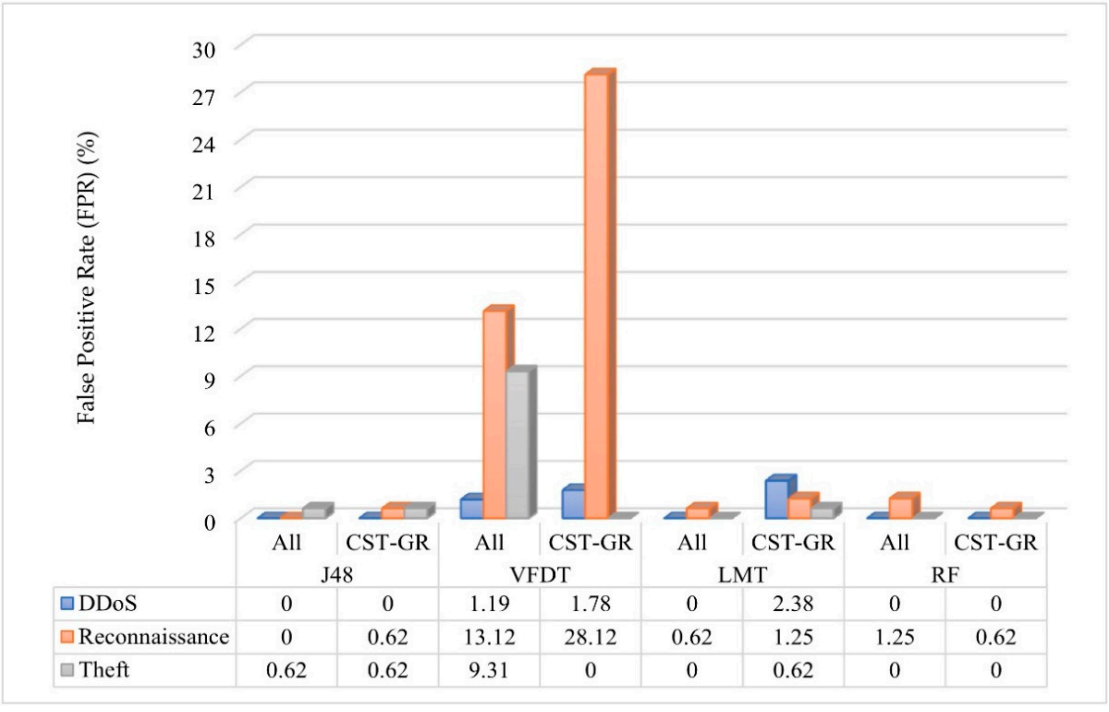
The Raspberry Pi 3 Model B used in our experiment has Quad-Core 1.2 GHz 64 bit-CPU and 1 GB RAM. As mentioned above, a part of the original dataset was selected for our experiment so that the hardware could deal with it even when all the original features were used with the tree-based classifiers. However, the LMT is an exception because we want to select as many instances as possible. Thus, we used a personal computer with i7-8565U and 16 GB RAM to compare the detection performance of the case using LMT with that of other classifiers. By the way, when we applied the features selected by the CST-GR algorithm, the Raspberry Pi 3 Model B can work well for all tree-based algorithms including LMT, which also indicates that our proposed feature selection approach could efficiently guarantee the detection system being lightweight.

To evaluate the detection performance, we compared the TPR, FPR, precision, and F-measure of all the above-mentioned classifiers. Figure 4 shows the comparison of the TPR among the tree-based classifiers in the cases of using all features and using CST-GR selected features for each kind of specific attack (DDoS, reconnaissance, and theft). All these results show that the detection performance is not deteriorated even when using very few features selected by the CST-GR algorithm. Especially in the cases of J48 and RF being used as the classifier and the features being selected by the CST-GR, some TPRs reached up to 99.4% for all three attacks. That is, we can consider J48 and RF to be more suitable for our system if only the TPR is considered. Figure 5 is a comparison of the FPR, which indicates that RF and J48 are the best choices if only the FPR is considered.



	J48		VFDT		LMT		RF	
	All	CST-GR	All	CST-GR	All	CST-GR	All	CST-GR
DDoS	99.4	99.4	99.4	98.8	99.41	99.39	99.4	99.4
Reconnaissance	99.37	97.54	99.28	99.13	99.37	97.53	99.37	99.37
Theft	99.37	99.37	99.31	99.38	99.38	99.37	99.38	99.38

Figure 4. A comparison of TPR of J48, very fast decision tree (VFDT), logistic model tree (LMT), and random forest (RF) between the cases using all features and using CST-GR selected features for each kind of attack.

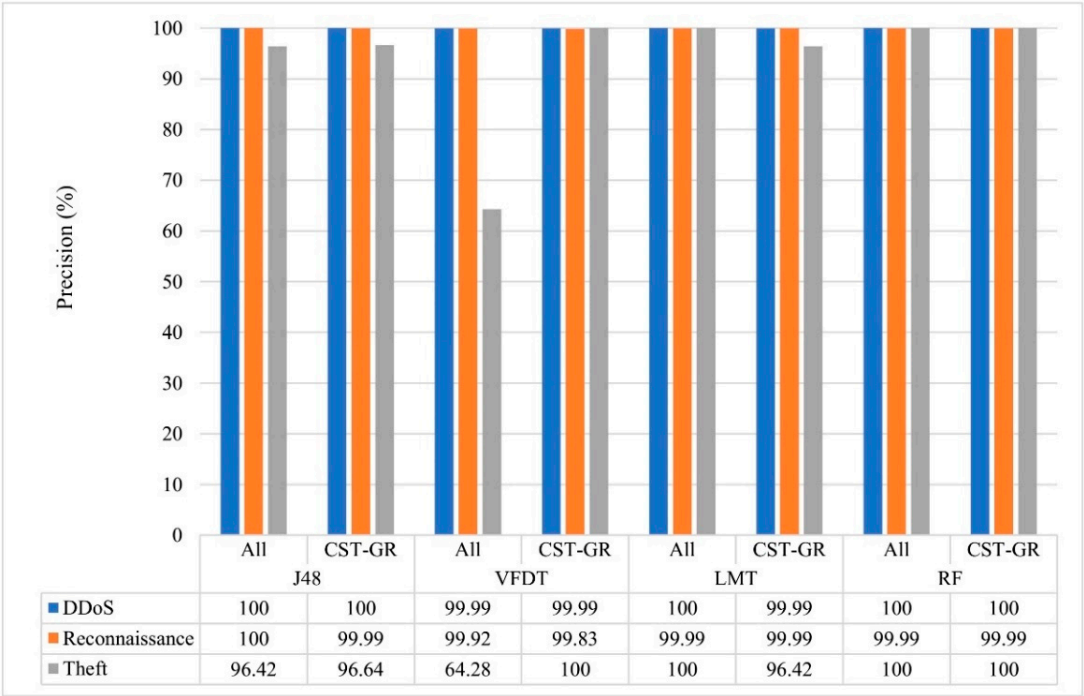


	J48		VFDT		LMT		RF	
	All	CST-GR	All	CST-GR	All	CST-GR	All	CST-GR
DDoS	0	0	1.19	1.78	0	2.38	0	0
Reconnaissance	0	0.62	13.12	28.12	0.62	1.25	1.25	0.62
Theft	0.62	0.62	9.31	0	0	0.62	0	0

Figure 5. A comparison of FPR of J48, VFDT, LMT, and RF between the cases using all features and using CST-GR selected features for each kind of attack.

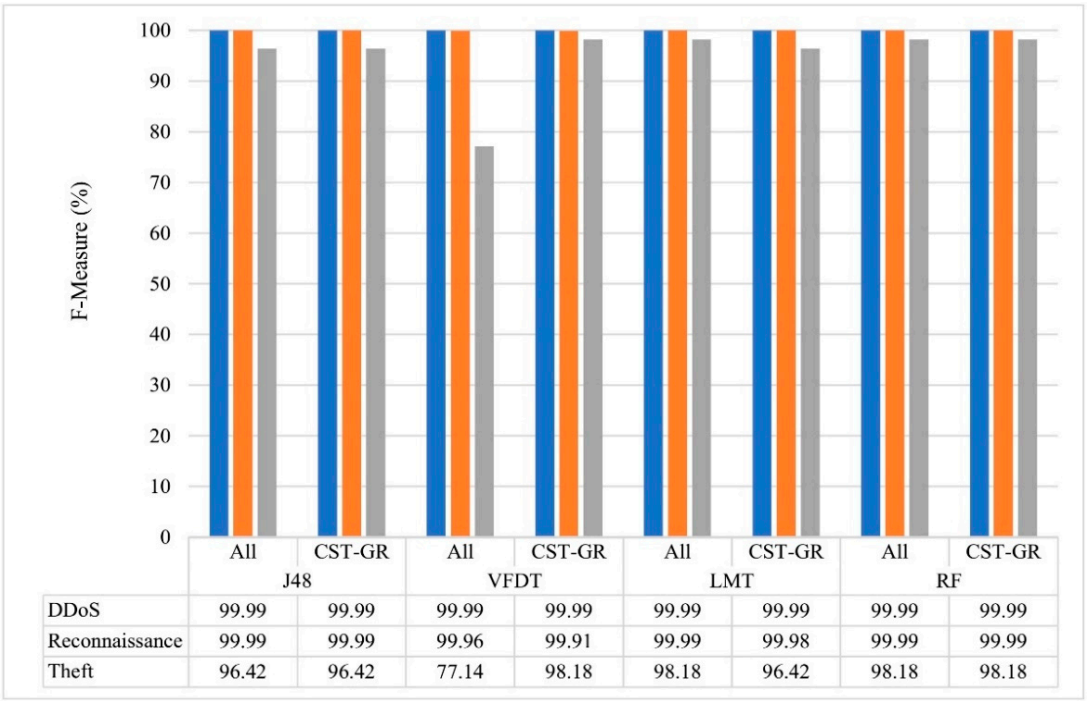
Comparisons on precision and F-measure are shown in Figures 6 and 7, respectively. The same as the results shown in Figure 5, the precision values and F-measures are better when the CST-GR algorithm is used to select features. These results also indicate that CST-GR is significantly good enough

for selecting the most critical features for our lightweight detection system. All of our performance comparisons indicate that the irrelevant or redundant features may not only raise the computational cost, but also deteriorate the detection performance.



	J48 All	J48 CST-GR	VFDT All	VFDT CST-GR	LMT All	LMT CST-GR	RF All	RF CST-GR
DDoS	100	100	99.99	99.99	100	99.99	100	100
Reconnaissance	100	99.99	99.92	99.83	99.99	99.99	99.99	99.99
Theft	96.42	96.64	64.28	100	100	96.42	100	100

Figure 6. A comparison of precision of J48, VFDT, LMT, and RF between the cases using all features and using CST-GR selected features for each kind of attack.



	J48 All	J48 CST-GR	VFDT All	VFDT CST-GR	LMT All	LMT CST-GR	RF All	RF CST-GR
DDoS	99.99	99.99	99.99	99.99	99.99	99.99	99.99	99.99
Reconnaissance	99.99	99.99	99.96	99.91	99.99	99.98	99.99	99.99
Theft	96.42	96.42	77.14	98.18	98.18	96.42	98.18	98.18

Figure 7. A comparison of F-measure of J48, VFDT, LMT, and RF between the cases using all features or using CST-GR selected features for each kind of attack.

4.3.2. Evaluation of Processing Time

The total times for processing all the instances of training data and testing data are presented in Table 4, including all the cases using the above-mentioned classifiers. The processing time for detecting each kind of attack was decreased greatly with the assistance of the CST-GR algorithm. Using the CST-GR algorithm for feature selection clearly makes the detection much faster. Table 5 shows the ranking in descending order of processing time among different classifiers. According to the experiments, the VFDT algorithm is the fastest on training, and the J48 algorithm is the fastest on testing. Although the training time of the RF is faster than the LMT, the testing time of the LMT is faster than RF in the case of using CST-GR features. Note that the training time of the LMT is longer than that of the RF, but the testing time of the LMT is faster than that of the RF when we applied our proposal on a personal computer. Our experiment platform, Raspberry Pi 3, cannot handle the processing of the LMT algorithm when all the original features are used. However, this device could process all the cases using the above-mentioned classifiers if the features selected by the CST-GR algorithm are used. According to the experiment results, the CST-GR algorithm can guarantee the detection system is able to handle all the above-mentioned classifiers on the resource constraint device, Raspberry Pi. Moreover, the J48 could be regarded as the most suitable learning method for our lightweight detection system.

Table 4. Comparison of processing time (seconds). VFDT, very fast decision tree; LMT, logistic model tree; RF, random forest.

Classifier	All Features		CST-GR	
	Training	Testing	Training	Testing
J48	57.65	1.22	8.61	0.81
VFDT	33.72	1.58	4.97	0.92
LMT	N/A	N/A	1198.37	0.99
RF	422.4	11.95	184.22	10.76

Table 5. Ranking of response time.

Rank	All Features		CST-GR Features	
	Training	Testing	Training	Testing
1	VFDT	J48	VFDT	J48
2	J48	VFDT	J48	VFDT
3	RF	RF	RF	LMT
4	N/A	N/A	LMT	RF

4.3.3. Processing Time on the Parallel Mode

Our proposal can also be implemented in a parallel mode to improve the response time. A response time comparison of training time and testing time between a sequential mode and a parallel mode is shown in Table 6. We found that the Raspberry Pi can handle the parallel detection architecture if we selected the corresponding features by the CST-GR. In this architecture, the response times of all classifiers are more than two times faster than in the sequential mode. Moreover, the results show that the response time of the J48 is significantly faster than the RF and slightly faster than the other two, the VFDT and the LMT.

Table 6. Comparison of processing time between sequential mode and parallel mode when using CST-GR features.

Classifiers	Training (seconds)		Testing (seconds)	
	Sequential	Parallel	Sequential	Parallel
J48	8.61	4.47	0.81	0.42
VFDT	4.97	3.03	0.92	0.48
LMT	1198.37	867.49	0.99	0.5
RF	184.22	102.14	10.76	5.92

4.3.4. CPU and Memory Usage on the Parallel Mode

We investigated the CPU and memory usage in the parallel detection architecture using corresponding features that are selected by the CST-GR algorithm. The htop tool [53], an interactive process-viewer of Unix systems, was used to capture the usage of memory and the CPU. Formally, the htop uses around 2% of CPU and 0.3% of the Raspberry Pi 3 Model B's memory. The case using the CST-GR algorithm for feature selection was investigated. The usage of the CPU and memory in the case of the J48 as the classifier is shown in Figures 8 and 9, respectively. Figure 8 indicates that all four cores in the CPU are working in a not-too-busy state. More specifically, around one-third of the memory (around 352 MB) was used. Note that the background processes consumed around 175 MB of memory. The maximum memory usage for four different classifiers was also investigated and results are as follows. The maximum memory usage reached 425 MB with the J48, 352 MB with the VFDT, 663 MB with the LMT, and 538 MB with the RF.

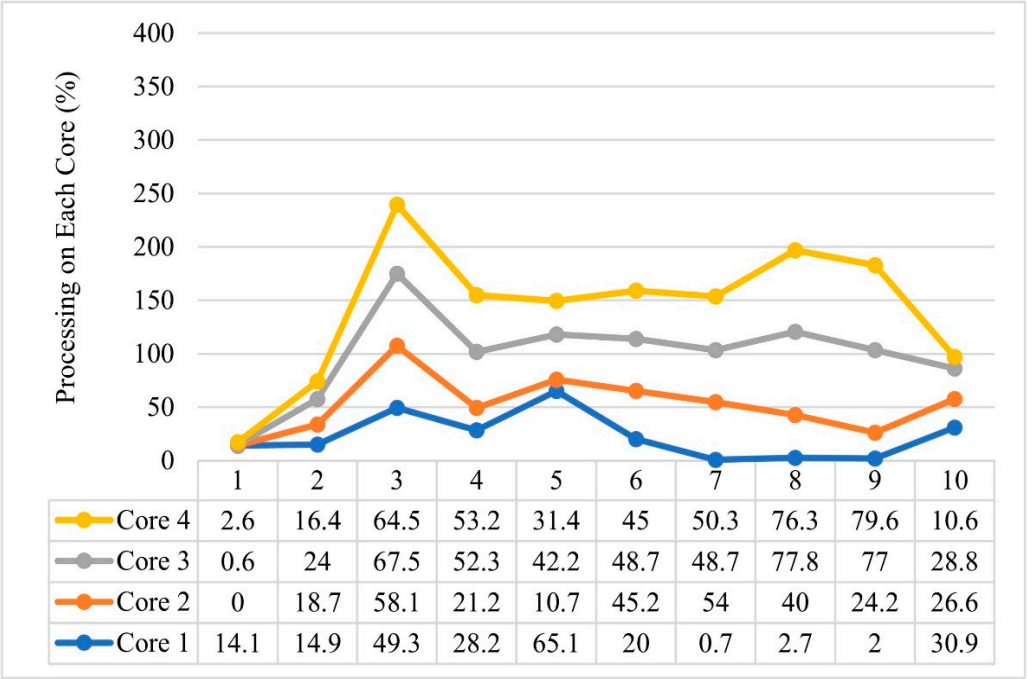


Figure 8. The consumption of each core (400% for four cores) of CPU for the parallel mode with J48 when using CST-GR selected features.

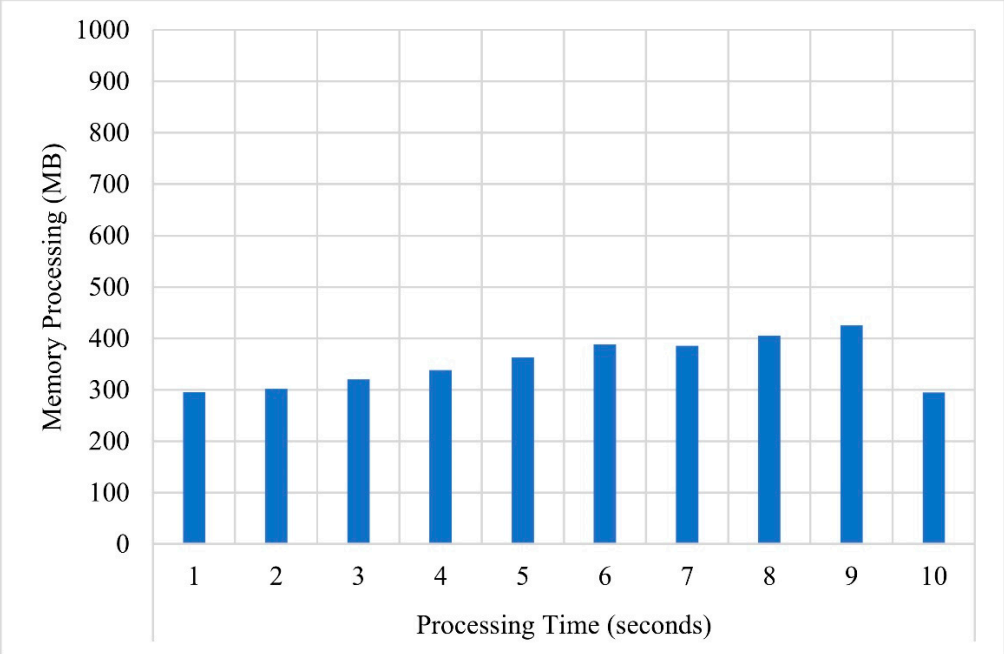


Figure 9. The consumption of device memory for the parallel mode with J48 when using CST-GR selected features.

Also, we investigated that the usage of CPU and memory in the case of all 40 original features was used in the parallel mode without any feature selection. The system halted halfway, which means our platform cannot run such a system in the parallel mode if the features are not selected. From this experiment, we found that the system halted without giving any results when the usage of the CPU core reached 100% and the usage of device memory grew over 800 MB.

4.4. Observations

From the above experiments, we can observe the following:

1. Using our proposed feature selection algorithm (CST-GR) for each kind of attacks, the number of features can be greatly decreased and the detection system can be made much lighter and much faster almost without any sacrifice on detection accuracy (see Tables 2 and 4, Figures 4 and 5). Moreover, the Raspberry Pi device can handle many more instances.
2. When using J48 and RF as the classifier, the detection accuracy (TPR) is still up to 99.4% even when using only very few features selected by the CST-GR algorithm.
3. The detection system can be implemented in the parallel mode in Raspberry Pi. However, it cannot handle all the data in the parallel mode if the original features are used without the help of the CST-GR algorithm.
4. The case of the J48 algorithm being used as the classifier has the shortest response time for detection, although the training time is a little longer than the VFDT (but still faster than the other two) and overall detection accuracy of the J48 is better than that of the VFDT.
5. Although the detection accuracy (TPR and FPR) of the RF is slightly better than that of the J48, the detection time of the J48 is around ten times faster than that of the RF. Therefore, J48 is the best choice for the classifier in our detection system.

5. Conclusions and Future Work

In this paper, we first pointed out the weakness of the existing detection system for IoT environments. After that, a lightweight detection system for the IoT environment using our new

feature selection algorithm (called CST-GR) was designed and implemented on Raspberry Pi. If the CST-GR algorithm is conducted for each of the specific attacks, the detection system needs very few features for detecting the corresponding attacks, which makes the detection system very lightweight and fast. Moreover, this lightweight detection system has almost no sacrifice on detection performance. In addition, several well-known machine learning algorithms—J48, Hoeffding tree (VFDT), logistic model tree (LMT), and random forest (RF)—were tested to be used in our system. The performance of our detection system was examined using the public Bot-IoT dataset, which was collected in a simulated IoT environment. According to our experiment results, the CST-GR feature selection and the J48 classifier could help in the implementation of a lightweight detection system. We also confirmed that our system could be implemented on the Raspberry Pi platform in a parallel mode. Note that, although three common attacks in the current IoT environment were discussed in this paper, other attacks can also be operated in the same way.

In the future, we will extend our system to detect other kinds of attacks on the real environment by implementing the detection system on Raspberry Pi, with smaller storage and weaker CPU. Furthermore, we will extend the current work with an anomaly-detection engine for detecting unknown attacks.

Author Contributions: Conceptualization, Y.N.S., Y.F., P.I.S., R.H., and K.S.; Formal analysis, Y.N.S. and Y.F.; Methodology, Y.N.S and Y.F., Project administration, Y.N.S., Y.F., P.I.S., R.H., and K.S.; Software, Y.N.S. and Y.F.; Supervision, Y.F., P.I.S., R.H., and K.S.; Validation, Y.N.S. and Y.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by AUN/SEED-Net Project (JICA). It is also partially supported by Strategic International Research Cooperative Program, Japan Science and Technology Agency (JST), JSPS KAKENHI Grant Numbers JP17K00187 and JP18K11295.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco. *Cisco Visual Networking Index (VNI) Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper*; Cisco Systems Inc.: San Jose, CA, USA, 2019.
2. Spamhaus Malware Labs. *Spamhaus Botnet Threat Report 2019*; Spamhaus Malware Labs: Geneva, Switzerland, 2018.
3. Haider, W.; Creech, G.; Xie, Y.; Hu, J. Windows based data sets for evaluation of robustness of Host based Intrusion Detection Systems (IDS) to zero-day and stealth attacks. *Future Internet* **2016**, *8*, 29. [CrossRef]
4. CPS Technologies. *Cyber Attack Trends Analysis Report*; CPS Technologies: Norton, MA, USA, 2019; Volume 1.
5. Zitta, T.; Neruda, M.; Vojtech, L. The security of RFID readers with IDS/IPS solution using Raspberry Pi. In Proceedings of the 2017 18th International Carpathian Control Conference, Sinaia, Romania, 28–31 May 2017; pp. 316–320.
6. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. *arXiv* **2018**, arXiv:1811.0070. [CrossRef]
7. Shah, S.A.R.; Issac, B. Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Gener. Comput. Syst.* **2018**, *80*, 157–170. [CrossRef]
8. Amini, P.; Araghizadeh, M.A.; Azmi, R. A survey on Botnet: Classification, detection and defense. In Proceedings of the 2015 International Electronics Symposium (IES), Surabaya, Indonesia, 29–30 September 2016; pp. 233–238.
9. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]
10. Amin, F.; Ahmad, A.; Choi, G.S. Towards Trust and Friendliness Approaches in the Social Internet of Things. *Appl. Sci.* **2019**, *9*, 166. [CrossRef]
11. Baker, A.R.; Esler, J. *Snort IDS, IPS Toolkit*; 30 Corporate Dr.; Elsevier Inc.: Burlington, MA, USA, 2007; ISBN 9783540449119.
12. OISF. *Suricata User Guide*; Open Information Security Foundation: Boston, MA, USA, 2019.

13. Tirumala, S.S.; Sathu, H.; Sarrafzadeh, A. Free and open source intrusion detection systems: A study. In Proceedings of the 2015 International Conference on Machine Learning and Cybernetics (ICMLC), Guangzhou, China, 12–15 July 2015; Volume 1, pp. 205–210.
14. Sforzin, A.; Marmol, F.G.; Conti, M.; Bohli, J.M. RPiDS: Raspberry Pi IDS—A Fruitful Intrusion Detection System for IoT. In Proceedings of the 2016 International IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, France, 18–21 July 2016; pp. 440–448.
15. Cervantes, C.; Poplade, D.; Nogueira, M.; Santos, A. Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 606–611.
16. Guo, Z.; Harris, I.G.; Jiang, Y.; Tsaur, L.F. An efficient approach to prevent battery exhaustion attack on BLE-based mesh networks. In Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC), Santa Clara, CA, USA, 26–29 January 2017; pp. 1–5.
17. Anthi, E.; Williams, L.; Burnap, P. Pulse: An Adaptive Intrusion Detection for the Internet of Things. In Proceedings of the Living in the Internet of Things: Cybersecurity of the IoT-2018, London, UK, 28–29 March 2018; p. 35.
18. Nobakht, M.; Sivaraman, V.; Boreli, R. A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. In Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 147–156.
19. Fu, Y.; Yan, Z.; Cao, J.; Koné, O.; Cao, X. An Automata Based Intrusion Detection Method for Internet of Things. *Mob. Inf. Syst.* **2017**, *2017*, 1750637. [CrossRef]
20. Kyaw, A.K.; Chen, Y.; Joseph, J. Pi-IDS: Evaluation of open-source intrusion detection systems on Raspberry Pi 2. In Proceedings of the 2015 Second International Conference on Information Security and Cyber Forensics (InfoSec), Cape Town, South Africa, 15–17 November 2015; pp. 165–170.
21. Da Silva Cardoso, A.M.; Lopes, R.F.; Teles, A.S.; Magalhaes, F.B.V. Real-time DDoS detection based on complex event processing for IoT. In Proceedings of the Third International Conference on Internet-of-Things Design and Implementation (IoTDI 2018), Orlando, FL, USA, 17–20 April 2018; pp. 273–274.
22. von Sperling, T.L.; de Caldas Filho, F.L.; de Sousa, R.T.; e Martins, L.M.C.; Rocha, R.L. Tracking intruders in IoT networks by means of DNS traffic analysis. In Proceedings of the 2017 Workshop on Communication Networks and Power Systems (WCNPS), Brasília, Brazil, 16–17 November 2017; pp. 1–4.
23. Aspernäs, A.; Simonsson, T. *IDS on Raspberry Pi: A Performance Evaluation*; Linnaeus University: Vaxjo, Sweden, 2015.
24. Khater, B.S.; Wahid, A.; Abdul, B.; Yamani, M.; Bin, I.; Hussain, M.A.; Ibrahim, A.A. A Lightweight Perceptron-Based Intrusion Detection System for Fog Computing. *Appl. Sci.* **2019**, *9*, 178. [CrossRef]
25. Creech, G.; Hu, J. Generation of a new IDS test dataset: Time to retire the KDD collection. In Proceedings of the 2013 IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China, 7–10 April 2013; pp. 4487–4492.
26. Coşar, M.; Kiram, H.E. Performance Comparison of Open Source IDSs via Raspberry Pi. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 16–17 September 2018.
27. Tripathi, S. Raspberry Pi as an Intrusion Detection System, a Honeypot and a Packet Analyzer. In Proceedings of the 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), Belgaum, India, 21–23 December 2018; pp. 80–85.
28. Kohavi, R.; John, G.H. Wrappers for Feature Subset Selection. *Artif. Intell.* **1997**, *97*, 273–324. [CrossRef]
29. Kohavi, R.; Sommerfield, D. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, Montreal, QC, Canada, 20–21 August 1995; pp. 192–197.
30. Feng, Y.; Akiyama, H.; Lu, L.; Sakurai, K. Feature Selection for Machine Learning-Based Early Detection of Distributed Cyber Attacks. In Proceedings of the 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 12–15 August 2018; pp. 173–180.

31. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [CrossRef]
32. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
33. Battiti, R. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Netw.* **1994**, *5*, 537–550. [CrossRef] [PubMed]
34. Karegowda, A.G.; Manjunath, A.S.; Ratio, G.; Evaluation, C.F. Comparative study of Attribute Selection Using Gain Ratio. *Int. J. Inf. Technol. Knowl. Knowl. Manag.* **2010**, *2*, 271–277.
35. Hall, M. Correlation-Based Feature Selection for Machine Learning. Ph.D. Thesis, University of Waikato, Hamilton, New Zealand, 1999.
36. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Implementing Lightweight IoT-IDS on Raspberry Pi Using Correlation-Based Feature Selection and Its Performance Evaluation. In *Advanced Information Networking and Applications, Proceedings of the 33rd International Conference on Advanced Information Networking and Applications AINA-2019; Advances in Intelligent Systems and Computing, Matsue, Japan, 27–29 March 2019*; Springer: Cham, Swiizerland, 2019; Volume 926, pp. 458–469.
37. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015.
38. Kuhn, M.; Johnson, K. An Introduction to Feature Selection. In *Applied Predictive Modeling*; Springer: New York, NY, USA, 2013; pp. 487–519.
39. Witten, I.H.; Frank, E. *Data Mining. Practical Machine Learning Tools and Technicals with Java Implementations*, 2nd ed.; Morgan Kaufmann Series in Data Management Systems; Elsevier Inc.: San Francisco, CA, USA, 2005; ISBN 0080890369.
40. Ashari, A.; Paryudi, I.; Min, A. Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool. *Int. J. Adv. Comput. Sci. Appl.* **2013**, *4*, 33–39. [CrossRef]
41. Domingos, P.; Hulten, G. Mining high-speed data streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2000; pp. 71–80.
42. Hulten, G.; Spencer, L. Mining Time-Chaning Data Streams. *Comput. Sci.* **2001**. [CrossRef]
43. Landwehr, N.; Hall, M.; Frank, E. Logistic model trees. *Lect. Notes Artif. Intell. (Subseries Lect. Notes Comput. Sci.)* **2005**, *2837*, 241–252. [CrossRef]
44. Friedman, J.; Hastie, T.; Tibshirani, R. Additive Logistic Regression: A Statisticl View of Boosting. *Ann. Stat.* **2000**, *28*, 337–407. [CrossRef]
45. Breiman, L. *Random Forests—Random Feature*; University of California: Berkeley, CA, USA, 2001; pp. 1–33.
46. Oshiro, T.M.; Perez, P.S.; Baranauskas, J.A. How Many Trees in a Random Forest? In *Machine Learning and Data Mining in Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 3587, pp. 154–168. ISBN 978-3-540-26923-6.
47. Node-RED. Available online: https://nodered.org/ (accessed on 1 August 2019).
48. Ostinato. Available online: https://ostinato.org/ (accessed on 1 August 2019).
49. Hping. Available online: http://hping.org/ (accessed on 1 August 2019).
50. GoldenEye. Available online: https://github.com/jseidl/GoldenEye (accessed on 1 August 2019).
51. Lyon, G.F. *Nmap Network Scanning: The official Nmap Project Guide to Network Discovery and Security Scanning*; Insecure. Com LLC: Sunnyvale, CA, USA, 1990.
52. Xprobe2. Available online: https://www.aldeid.com/wiki/Xprobe2 (accessed on 7 August 2019).
53. Hisham Htop—An Interactive Process Viewer for Unix. Available online: http://hisham.hm/htop/ (accessed on 15 July 2019).