Computers & Security 137 (2024) 103598

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Designing accurate lightweight intrusion detection systems for IoT networks using fine-tuned linear SVM and feature selectors

Jahongir Azimjonov, Taehong Kim *

School of Information and Communication Engineering, Chungbuk National University, Cheongju-si, 28644, Chungcheongbuk-do, South Korea

ARTICLE INFO

Dataset link: https://github.com/JahongirAzimjonov/Lightweight-Intrusion-Detection-Systems-via-Feature-Selection-and-LinearSVM

Keywords:
Determining relevant and efficient feature subsets
Lightweight IDSs
Fine-tuned LSVMs and feature selectors
IoT network security

ABSTRACT

Intrusion detection systems (IDSs) play a crucial role in ensuring the security and integrity of Internet of Things (IoT) networks by blocking unwanted packets and facilitating secure traffic flow. However, traditional IDSs based on data mining, fuzzy logic, heuristics, rough sets, or conventional machine learning (ML) techniques often lack accuracy and are not energy efficient, primarily due to inappropriate feature selection or the use of all features in datasets. To address these challenges, this study proposes a lightweight, accurate, and high-performance IDSs for IoT networks using fine-tuned Linear Support Vector Machines (LSVMs) and feature selection methods. Four feature selectors, including Importance Coefficient-, Forward- and Backward-Sequential-, and Correlation Coefficient-based approaches, were applied to identify the most important and efficient features from three datasets: KDD Cup-1999, BotIoT-2018, and N-BaIoT-2021. The fine-tuned LSVMs algorithm was then trained on subsets of the selected and full features of the datasets to detect various IoT botnet attacks. Evaluation results show that the IDS models trained with subsets of relevant features outperform those trained with the full feature sets of the datasets in terms of training and test performance and accuracy. The study concludes that it is possible to develop lightweight IDSs by training them with a reduced number of features (6) instead of using the full features (40, 15, 115) in KDD Cup-1999, BotIoT-2018, and N-BaIoT-2021, respectively. The findings highlight a potential for significantly improving the efficiency and accuracy of IDSs on IoT networks using the fine-tuned feature selectors and LSVMs.

1. Introduction

IoT has become an integral part of our daily lives, with a wide range of devices connected to the internet, ranging from smart home appliances to industrial control systems. However, the widespread adoption of IoT has also led to an increase in security threats, such as botnet attacks including DoS, DDoS, Reconnaissance, Theft, Mirai, and BASHLITE, which can compromise the security and privacy of IoT devices (Sahoo et al., 2020). In addition, the IoT networks are particularly vulnerable to botnet attacks because they often have weak security protocols, and the devices connected to them may not be properly protected (Roy et al., 2022). Botnet IDSs can help to prevent these attacks by detecting and blocking the communication between compromised devices and the command and control (C&C) servers that control them. These systems can also alert network administrators to the presence of a botnet and provide them with the tools they need to take action (Li et al., 2009). By applying botnet IDSs, IoT networks can better protect themselves from the damaging effects of botnets and other cyber threats (Kumar et al., 2022). Thus, it is essential to develop efficient and effective IDSs that can detect and mitigate such attacks on the low-power IoT devices (Burhan et al., 2018).

IDSs, as a crucial component of IoT systems, help to ensure the integrity and reliability of IoT networks along with security. These systems are designed to detect and block unwanted packets that may pose a threat to the network, such as malware, botnets, and other types of cyber attacks (Lima et al., 2022). However, traditional IDSs, based on data mining, fuzzy logic, heuristics, rough sets, and conventional ML, often lack accuracy and are not energy efficient (Khammassi et al., 2017). This is a significant issue in the context of IoT networks, which are characterized by a large number of interconnected devices and a high demand for power efficiency (Sahu et al., 2021). To address these challenges, there is a need for lightweight, accurate, and high-performance IDSs that are more energy efficient and capable of detecting a wide range of threats (Mashal et al., 2015). One promising approach for developing lightweight IDSs for IoT networks is the use of LSVMs. LSVMs are a type of ML algorithms that can be used for

* Corresponding author.
E-mail addresses: jahongir@chunbuk.ac.kr (J. Azimjonov), taehongkim@cbnu.ac.kr (T. Kim).

https://doi.org/10.1016/j.cose.2023.103598
Received 8 January 2023; Received in revised form 14 August 2023; Accepted 11 November 2023
Available online 17 November 2023

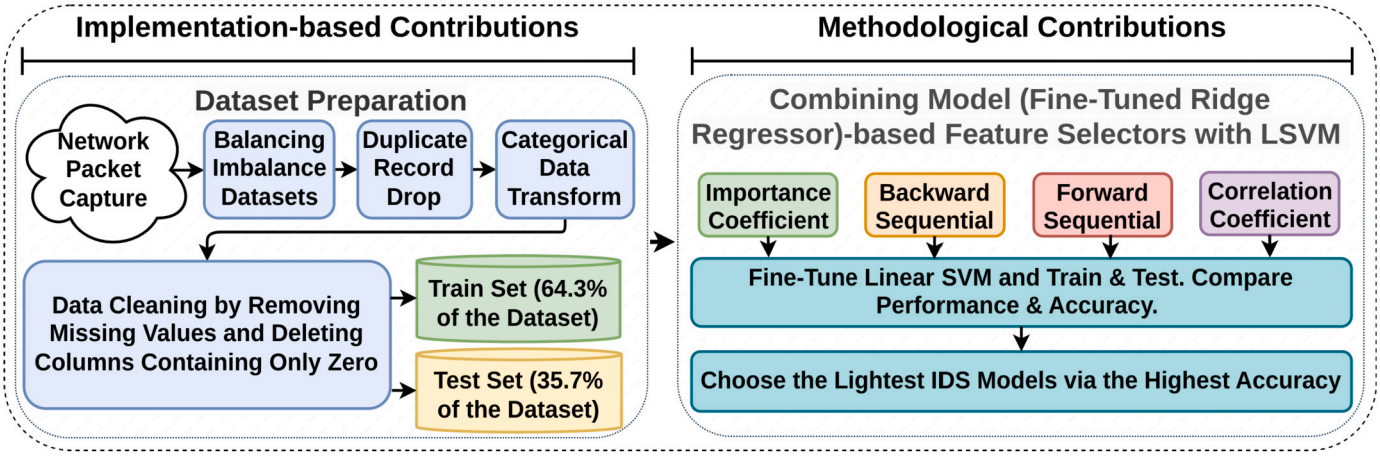


Fig. 1. The development process of the proposed lightweight IDSs.

classification tasks and have been widely used in various applications, including intrusion detection. However, the performance of LSVMs depends on the hyper-parameters that control the learning process, since optimal hyper-parameters contributes to reducing the over-fitting and under-fitting cases of an ML model. Therefore, it is vital to optimize the hyper-parameters in order to achieve the best performance and accuracy (Said et al., 2013).

In this study, we propose a novel, fine-tuned LSVM-based lightweight IDSs for IoT networks, which utilizes the most important and efficient features, selected via four distinct feature selectors. Feature selection is the process of identifying a subset of relevant features from a larger set of features. It is an important step in the ML process as it can improve the performance and reduce the complexity of IDS models. Model-based four feature selection algorithms including Importance- & Correlation-Coefficient-, and Forward- & Backward-Sequential-based algorithms, were employed to determine subsets of the most crucial and efficient features. The fine-tuned ridge regressor was utilized as a kernel model for the feature selectors, to calculate feature importance coefficients. The performance of the proposed approach was evaluated on three datasets: KDD Cup 1999 (Bay et al., 2000), BoT-IoT-2018 (Koroniotis et al., 2019) and N-BaIoT 2021 (Meidan et al., 2018). The datasets contain a variety of IoT botnet attacks, including DoS (Denial of Service), DDoS (Distributed DoS), Reconnaissance, Theft, Mirai, BASHLITE, U2R (User to Root), and R2L (Remote to Local) attacks, and provide a diverse set of features that can be used to train and evaluate the LSVM-based IDSs. Overall, the proposed approach demonstrates the potential to effectively detect and mitigate IoT botnet attacks and has the potential to be widely adopted in the field of IoT security. The proposed system is illustrated in the Fig. 1. **The main contributions of the study are as follows**:

(i) Development of accurate and lightweight IDSs for IoT networks by combining an LSVMs-based classifier and ridge regressor-based feature selectors. The combination of these methods enables efficient and effective feature selection, leading to improved IDSs accuracy.

(ii) Optimization of hyperparameters for the ridge regressor to enhance the accuracy of the feature selectors. By fine-tuning the ridge regressor, we achieve better feature selection, further enhancing the performance of the IDSs.

(iii) Reduction of the number of trainable features to just 6 from the original full features (40, 15, 115) in KDD Cup-1999, BotIoT-2018, and N-BaIoT-2021 datasets, respectively. Despite using a significantly smaller feature set, we maintain the accuracy of intrusion detection, demonstrating the potential for a lightweight IDS design.

(iv) Demonstration of the feasibility of designing high-accuracy and lightweight IDSs using the fine-tuned feature selectors and LSVMs-based classifier, training them with only a few selected features rather than the entire dataset. This not only achieves accurate IDSs but also significantly reduces computational and energy costs on IoT networks.

2. Literature review

Botnets are networks of compromised devices that are used to launch cyber attacks, such as distributed denial of service (DDoS) attacks, spamming, data ex-filtration, and others. These devices, also known as zombies, are controlled by a central C&C server and can be located anywhere in the world. IoT has greatly expanded the potential for botnet attacks, as it has brought an increase in the number of connected devices and the complexity of IoT networks (Sahoo and Puthal, 2020). To address this threat, researchers have developed traditional botnet IDSs and lightweight botnet IDSs for IoT networks (Madakam et al., 2015).

2.1. Traditional botnet IDSs

Traditional botnet IDSs are designed to detect botnet activities by analyzing network traffic or system logs. These IDSs use various techniques, such as signature-, anomaly-, and behavioral-based detection. Signature-based detection involves matching the traffic or log data against a database of known botnet signatures. This method is effective in detecting known botnets, but it can not detect new or unknown botnets. Anomaly-based detection involves identifying abnormal patterns in the traffic or log data (Elrawy et al., 2018; Hindy et al., 2020). This method is able to detect unknown botnets, but it can also produce false positives due to normal network variations. Behavioral-based method detects anomalies by analyzing the behavior of devices in a network (Kheir, 2013). This method detects unknown botnets by identifying deviations from normal device behavior. However, it requires a baseline of normal behavior, which is difficult to establish in IoT networks with a large number of diverse devices (Mitchell et al., 2014). One example of traditional botnet IDSs for IoT networks is BotMiner IDS, which utilizes ML algorithms to detect botnet activities by analyzing network traffic and system logs. This IDS tool has shown good performance in detecting botnets, but it requires a large amount of data for training and may not be suitable for resource-constrained IoT devices (Zarpelão et al., 2017).

2.2. Lightweight botnet IDSs

Lightweight botnet IDSs are designed to be light and efficient, so they can be deployed on resource-constrained IoT devices. These IDSs use various techniques, such as flow- and behavior-based detection, network traffic analysis (NTA). Flow-based detection involves analyzing the flow of traffic between devices in the network. This method can

detect botnets by identifying abnormal traffic patterns, such as a large number of connections from a single device or a sudden increase in the traffic (Hajiheidari et al., 2019). However, it may not detect botnets that use encrypted traffic or use a distributed C&C structure. NTA methods analyze the content of network traffic, such as packet headers, payloads, and protocol fields. These methods detect botnets by identifying suspicious patterns, such as the use of known botnet C&C protocols or the ex-filtration of sensitive data (Yaqoob et al., 2017). But, it requires a deep understanding of the protocols and may not detect botnets that use custom or unknown protocols. Behavior-based detection analyzes the behavior of devices in the network (Kolias et al., 2017). This method can detect botnets by identifying deviations from normal device behavior (Eesa et al., 2015). However, it requires a baseline of normal behavior and may not detect botnets that use stealthy or sophisticated tactics (Fraley et al., 2017). To sum up, it is essential to develop dynamic and lightweight IDSs that can address the aforementioned issues of the traditional and lightweight IDSs, by optimizing training, testing, and packet processing time (B et al., 2019; Gibert et al., 2020). One promising approach for developing such kind of lightweight IDSs for IoT networks is the use of LSVMs (Raff et al., 2017). High-performance, high-accuracy, and efficient IDS models can be developed by fine-tuning the hyperparameters of this algorithm and by training this algorithm via subsets of the most important and efficient features of datasets (Ghiasi et al., 2015).

2.3. Feature selection approaches

Selection of most important and efficient features of datasets that can accurately represent the network traffic and identify potential threats is one of the essential parts of developing lightweight IDSs (Galal et al., 2016). In this subsection, we will discuss various feature selection methods that have been proposed for IDSs on IoT networks. Feature selection algorithms can be divided into three categories: Wrapper- and Filter-based, and Hybrid methods (Ding et al., 2013). *Wrapper-based methods* use a subset of features and evaluate their performance on a given classifier. The subset with the best performance is selected for the final model. One example of a wrapper method is the forward-sequential selection algorithm, which starts with an empty set of features and adds one feature at a time, until the desired number of features is reached. Another example is the backward-sequential algorithm, which starts with a full set of features and removes one feature at a time until the desired number of features is reached (Kasongo et al., 2020). *Filter-based methods* use a pre-defined criterion to select a subset of features, independent of the classifier. One example of a filter method is the Chi-square test, which measures the independence of a feature and the class label. Another example is the Mutual Information, which measures the mutual dependence between a feature and the class label (Salehi et al., 2017). *Hybrid methods* combine the strengths of wrapper and filter methods. One example of a hybrid method is the Fast Correlation-Based Feature Selection (FCBF) algorithm, which uses the symmetrical uncertainty measure to select the most relevant features. Another example is the Sequential Forward Floating Selection (SFFS) algorithm, which uses a wrapper approach to select features based on the accuracy of a classifier (Shafiq et al., 2020). *Evaluation of Feature Selection Methods.* Several studies have evaluated the performance of different feature selection methods on IDS for IoT networks. A study by Zhang et al. (2018) compared the performance of wrapper and filter methods on a dataset of IoT network traffic. They found that the wrapper method outperformed the filter method, with the forward selection algorithm achieving the best performance (Zhang et al., 2022). Another study compared the performance of hybrid methods on an IoT network dataset. They found that the FCBF algorithm outperformed other hybrid methods, with a higher accuracy and lower false alarm rate (Ring et al., 2019). In conclusion, feature selection is an important aspect of IDSs for IoT networks (Emerson, 2015). Wrapper, filter, and hybrid methods have all been proposed and evaluated for this purpose. Wrapper methods tend to perform better than filter methods, while hybrid methods can combine the strengths of both approaches. Further research is needed to identify the most effective feature selection method for different types of IoT network traffic and threats (Wilkinson et al., 2016).

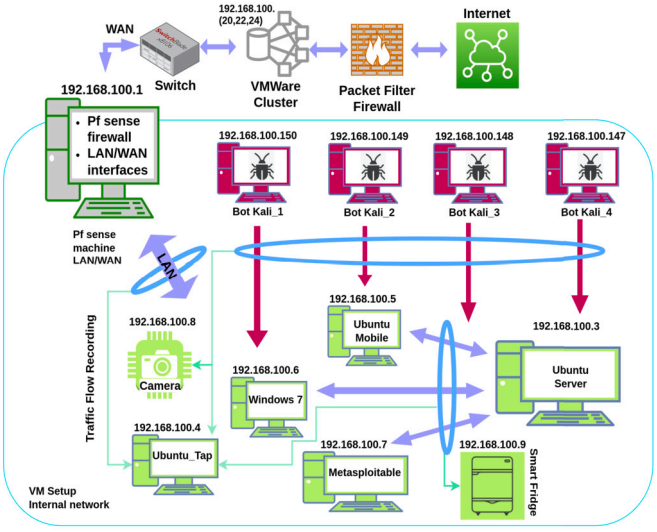


Fig. 2. The simulated BoT-IoT environment that generates benign and malicious packets, (Koroniotis et al., 2019).

2.4. Research gaps and solution

The present research bridges the aforementioned gaps in the current literature by addressing limitations and challenges faced by existing traditional and lightweight IDSs utilized in IoT networks. These prevailing IDSs, which rely on data mining, fuzzy logic, heuristics, rough sets, or conventional ML techniques, often suffer from accuracy issues, notably high false positive rates. Additionally, they encounter energy efficiency problems caused from inadequate feature selection or the incorporation of all available features within datasets. Another shortcoming lies in their demand for an in-depth comprehension of network packet features. In contrast, our research introduces an innovative approach to the creation of precise and lightweight IDSs for IoT networks. This approach capitalizes on fine-tuned LSVMs and feature selection methods. By employing four distinct feature selectors, our study pinpoints the most crucial and effective features from the three datasets. The fine-tuned LSVM-based IDSs, trained on the selected subsets outperformed those trained with full feature sets, demonstrating improved training and test performance. Beyond the improvement of both lightweight design and accuracy of IDSs within IoT networks, this approach concurrently addresses the prevalent challenges of resource constraints and power efficiency that are frequently encountered in IoT environments.

2.5. Datasets

There are several datasets available for the development and evaluation of IDSs for detecting botnet attacks on IoT networks (Garcia et al., 2020). However, in this study, we used three most common datasets: one is real and two are realistic. The realistic ones were created using simulation tools to mimic a genuine IoT environment (as shown in Fig. 2) (Koroniotis et al., 2019). The PCAP files in the BotIoT-2018 dataset were collected using this virtual setup. The normal and attack traffic flow data were extracted and saved in ".csv" files. The datasets contained millions of records with various features (both original and derived from PCAP), including 40, 15, and 115 features. In the BotIoT-2018 dataset, researchers selected 3 million of the most important packets (rows) and ten features using the Correlation Coefficient and Entropy techniques, and tested these with three ML algorithms (SVM,

RNN, and LSTM). However, we use all available features and light structured ML approaches instead of the three large-structured deep learning architectures due to performance concerns. Other two datasets are KDD Cup 1999 and N-BaIoT-2021. The datasets contain different types of network traffic, including normal and abnormal (generated by botnets) traffic packets. They have been used to evaluate the performance of both previous heavyweight and lightweight IDSs based on both ML and rule-based approaches. The KDD Cup 1999 dataset is a widely used for evaluating IDSs on IoT networks as well as other computer networks. It was created in 1999 and is a standard benchmark for evaluating IDSs. It consists of network traffic data from a military network and contains both normal and anomalous behavior. N-BaIoT-2021 is a more recent dataset that was created by collecting traffic data from the real-world IoT devices. It contains network traffic data from a distinct real IoT devices, including smart home devices and industrial control systems. It contains both benign and malicious traffic and is useful for evaluating the performance of IDSs on IoT networks.

3. Methodology: the proposed approach

The proposed approach for developing lightweight IDSs consists of three parts as it is illustrated in Algorithm 1: (i) data preprocessing (balancing imbalanced datasets, dropping duplicate records, categorical data transform and data cleaning), (ii) feature selection with the importance- and correlation-coefficient-, forward- & backward-sequential-based approaches, and (iii) hyper-parameter fine-tuning, training & testing of the LSVMs-based classifier and identifying the most accurate and lightest IDSs based on the models' performance.

3.1. Data preprocessing

The first step of the proposed approach is data preprocessing, which includes four sub-steps: balancing datasets, dropping duplicate records, transforming categorical data, and cleaning the data. In the first sub-stage, the dataset was checked for class label imbalances. If the ratio of classes in the dataset is greater than 20/80, the dataset was considered balanced. If the ratio was less than this value, then the dataset was balanced using resampling techniques and the balanced dataset D^b was obtained. In the second sub-stage, duplication record elimination was performed to remove any redundant, unimportant, or duplicate records from the dataset and D^d was obtained. Such records can negatively impact the training process and may lead to overfitting of the model if not dropped. Next, any cell value containing categorical data (such as characters, strings, non-numeric or text, or other than decimal values) was transformed into numeric data types and D^t was obtained. Finally, the dataset was cleaned by handling missing values and removing columns containing only 0 values and D^c was obtained. By following these steps, the data was properly prepared for use in the feature selection and model training processes. These stages were shown in the first step of Algorithm 1.

3.2. Feature selection approaches

The relevant and efficient features of the three datasets were determined with the feature selection algorithms based on the importance coefficients, backward- & forward-sequential, and correlation coefficients. The fine-tuned ridge regression model (R_{ridge}) was used in the kernel of all feature selectors. The hyper parameters (α, learning rate and others) were optimized using the grid search optimization method. The feature selectors were explained in the following sub-sections. The mathematical formulation and pseudo-code of the feature selectors are illustrated in the second step of Algorithm 1.

3.2.1. Importance coefficient-based feature selection

The importance coefficient-based feature selector identified the most important and efficient features using the fine-tuned ridge regressor (R_{ridge}) in its kernel. The importance coefficient ($\beta \leftarrow (X^T X + \lambda \mathbf{I}_n)^{-1} X^T y$) of each feature was calculated by training the ridge regressor with each feature. Next, the features with an importance coefficient greater than or equal to the average of all the importance coefficients were selected. The most efficient and important six features that passed the average threshold were selected and utilized in the selection model, while the less important features were discarded. The selected subset of features was saved in S_{IC}.

Algorithm 1 Proposed approach for developing lightweight IDSs.

Input: Raw dataset - D, Hyper parameter sets for the ridge regressor and LSVMs classifier: P_{reg} and P_{cls}, respectively.
Output: Accurate and lightweight IDSs.

Step 1: Data Preprocessing.
procedure BALANCE_DATASET(D)
  if D is imbalanced **then**
    if D is KDD-Cup-1999 or BotIoT-2018 **then**
      Down-sample with randomly selected records.
    else
      Down-sample with sequentially selected records.
    end if
  end if
  return D^b
end procedure ▷ Balancing imbalanced datasets.
$D^b \leftarrow Balance\_Dataset(D)$; $D^d \leftarrow Dropping\_Duplicate\_Records(D^b)$
$D^t \leftarrow Transform\_Categorical\_Data(D^d)$; $D^c \leftarrow Clean\_Data(D^t)$

Step 2: Feature Selection with the Ridge Regression Model-based Approaches.
procedure RIDGEREGRESSION(X, y, P_{reg})
  Initialize $n \leftarrow$ number of features in X, $\beta \leftarrow 0$, $\mathrm{I}_n \leftarrow$ identity matrix of size n.
  $\lambda \leftarrow GridSearchFineTune(X, y, P_{reg})$ ▷ Hyper-parameter fine-tuning for ridge regressor.
  $\beta \leftarrow (X^T X + \lambda \mathbf{I}_n)^{-1} X^T y$ ▷ The fine-tuned and trained ridge regressor.
  return β
end procedure
$X, y \leftarrow D^c, R_{ridge} \leftarrow RidgeRegression(X, y, P_{reg})$
procedure SELECTSUBSETSOFFEATURES(X, y, R_{ridge})
  $S_{IC} \leftarrow SelectFeatures\_withIC(X, y, R_{ridge})$ ▷ Importance coefficient-based feature selection.
  $S_{CC} \leftarrow SelectFeatures\_withCC(X, y, R_{ridge})$ ▷ Correlation coefficient-based feature selection.
  $S_{BS} \leftarrow SelectFeatures\_withBS(X, y, R_{ridge})$ ▷ Backward Sequential-based feature selection.
  $S_{FS} \leftarrow SelectFeatures\_withFS(X, y, R_{ridge})$ ▷ Forward Sequential-based feature selection.
  return $[S_{IC}, S_{CC}, S_{BS}, S_{FS}]$ ▷ The selected efficient feature subsets.
end procedure

Step 3: Building IDSs with the Fine-Tuned LSVMs-based Classifier.
$\mathcal{H}_{opt} \leftarrow GridSearchHyperParameterOptimizer(X, y, P_{cls})$
procedure BUILDLSVMCLASSIFIERIDSS(X, Y, S, H_{opt})
  $x, y \leftarrow X[S], Y[S]$ ▷ x and y is a training subset of the selected features from a dataset.
  Training the LSVMs-based classifier (y_{model}) with the subsets of selected features.
  $w, b \leftarrow \min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b))$.
  return $y_{model} = w^T \cdot x + b$ as an IDS
end procedure
$Subsets \leftarrow SelectSubsetsOfFeatures(X, y, R_{ridge})$
$IDSs \leftarrow BuildLSVMClassifierIDSs(X, y, Subsets, H_{opt})$ ▷ The IDSs trained on the selected subsets.
$IDS_{FF} \leftarrow BuildLSVMClassifierIDS(X, y, S_{FF}, H_{opt})$ ▷ IDS trained on the full feature set.

3.2.2. Forward sequential-based feature selection

This approach begins by training the ridge regressor (R_{ridge}) with a subset containing the first feature from the dataset. Subsequently, it trains the regressor in a forward manner, secondly with two features, then three, and so on until all features are included. The model accuracy along with its importance coefficient ($\beta \leftarrow (X^T X + \lambda \mathbf{I}_n)^{-1} X^T y$) is saved to a temporary feature importance vector. The feature importance vector is then sorted in descending order, and the first k features are selected as the most important and efficient ones. In our case, the top six features with the highest importance coefficients were selected and saved in S_{FS}.

3.2.3. Backward sequential-based feature selection

The algorithm implements the backward sequential approach for feature selection using the fine-tuned ridge regression technique (R_{ridge}). This iterative algorithm starts with a set of all features in the input data, X. At each iteration, it removes the feature that results in the lowest evaluation score ($\beta \leftarrow (X^T X + \lambda \mathbf{I}_n)^{-1} X^T y$), as determined by training and evaluating a model on the data without that feature. This process continues until the number of features in the set is reduced to the desired number, k. In our case, the process continued until the six most important and efficient features were left. The selected features were saved in the S_{BS} feature vector.

3.2.4. Correlation coefficient-based feature selection

The approach selects the best features from a full feature set based on correlation coefficients using the fine-tuned ridge regression technique (R_{ridge}, $\beta \leftarrow (X^T X + \lambda \mathbf{I}_n)^{-1} X^T y$) with backward elimination. It begins with a set of all features X and a target variable y. A significance level p is defined, usually set to 0.05. The initial set of features is then set equal to the full set of features X. Next, the algorithm eliminates features one by one in a backward manner until there are no more features whose correlation coefficients have a p-value greater than the defined significance level $p = 0.05$. In the last step, it selects the best k features. In our case, six features were selected as the most important and efficient ones.

3.3. The LSVMs classifier-based lightweight IDSs

The proposed IDSs were developed by training LSVMs classifier on the subsets of the efficient features, selected from the datasets. The reason behind opting for the LSVMs is its recognized lightweight nature compared to other commonly used classifiers in traditional IDSs. This algorithm was fine-tuned to improve its efficiency using a grid search hyperparameter optimization algorithm. Additionally, the number of features was reduced based on their relevance to alleviate the algorithm's training and testing overheads. Datasets used in building IDSs are typically applied directly to a certain classifier without reducing the number of features. This can lead to increased packet processing time and reduced IDS accuracy. To address this challenge, we have shown that the network packet processing time can be greatly reduced without sacrificing detection accuracy by selecting the most relevant and efficient features and by training the classifier with these features. As a result, the classifier becomes lightweight. Therefore, by fine-tuning and training the LSVMs with a reduced number of features, we were able to achieve lightweight IDSs that maintain high accuracy in detecting intrusions. Four feature selectors determined 12 feature subsets from the three datasets. Each dataset has also a full feature set, resulting in a total of 15 (= 5 x 3) feature sets. The fine-tuned classifier was trained on all feature sets, resulting in 15 candidate IDSs. To identify the best lightweight IDSs, all the models trained on the selected subsets and the full feature sets, were compared based on their performance and accuracy. The IDSs with the highest accuracy and the lowest time performance were selected as the final lightweight IDSs. The entire process is illustrated in the third step of Algorithm 1. The estimated optimal hyperparameters for LSVMs: *'C': **7.0**,'loss':'**squared_hinge**','max_iter': **500**,'penalty':'**l2**','random_state': **rS***,'tol': **0.0001*** and for ridge regressor: *'alpha': **1.e-06**,'max_iter': 500,'random_state': **rS***,'tol': **0.001**. (Note: rS* - [67, 11, 101] for the KDD CUP 1999, BotIoT-2018, and N-BaiIoT-2021 datasets, respectively)*.

3.4. Experimental setup

The experiments were conducted on three different datasets. The detailed information of these datasets were not given in this paper because of space constraints, but the datasets info can be obtained from the originally published and publicly available source links: KDD CUP 1999, BotIoT-2018, and N-BaIoT-2021. The current project was also published and publicly available with the detailed experimental information on GitHub: Acces the project on Github. For evaluation metrics, we used the confusion matrix approach as it was shown in Eq. (1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},\ Recall = \frac{TP}{TP + FN},$$
$$Precision = \frac{TP}{TP + FP},\ F_{score} = \frac{2\ Precision\ Recall}{Precision + Recall} \tag{1}$$

4. Results and discussion

The results are presented in three subsections: data preprocessing outcomes, selected subsets of important and efficient features, and the performance of IDSs trained on the selected and full features of the datasets. In the first part, we detail the results of dataset balancing and cleaning, emphasizing the meticulous steps taken to ensure the data's high quality and suitability for analysis. Moving on to the next part, we showcase the relevant and efficient feature subsets that were carefully selected with the importance- & correlation-coefficient-, and backward- & forward-sequential-based feature selectors. The impacts of the feature selectors on the overall performance of IDSs were also discussed. Our analysis and findings shed light on the significance of these techniques in enhancing the efficiency and accuracy of the IDSs. In the final part, the performance of the IDSs trained with the selected feature subsets was compared to those trained with the full features by highlighting the advantages of utilizing the selected feature subsets in terms of improved performance and efficiency. Additionally, we compare the results of our current study with those of similar works, including C-SVM with linear kernel (C-SVML) and Stochastic Gradient Descent-based classifiers (SGDC). This comparative analysis further solidifies the significance and novel contributions of our approach. The detailed results were displayed in Tables 1, 2, 3 and 4.

4.1. Dataset preprocessing outcomes

One of the key challenges that can lead to either overfitting or underfitting in an ML model is the presence of class imbalances in datasets. When there is an imbalance in the distribution of classes, a model that has been trained on this dataset is more likely to be accurate when predicting the majority class, but it tends to struggle when it comes to the minority class. This can result in a model with poor predictive performance, particularly when it comes to the minority class. All three datasets used in this study had the class imbalance issues. In the BotIoT-2018 dataset, there are only 477 normal packets compared to 3668045 malicious (attack) packets. However, all the packets are unique and there are no duplicate records in the dataset. To address this imbalance, we selected a subset of 5000 attack packets along with all of the normal packets. In the KDD CUP 1999 dataset, there are 60592 normal packets and 250436 attack packets, but this dataset includes a large number of duplicate records. We were able to balance this dataset by simply dropping the duplicate records. On the other hand, the N-BaIoT-2021 dataset has also abnormally distributed classes of normal (62154) and attack (766106) packets. This dataset was also balanced using data preprocessing techniques, the detailed results can be seen in Fig. 3.

4.2. The subsets of the selected features

The most efficient and important features of the three datasets were selected using the importance- & correlation-coefficients-, and forward- & backward-sequential-based feature selectors. These techniques were employed to measure the relationship between the independent (input features) and the dependent (output classes) variables using a fine-tuned ridge regressor. Among these methods, the importance coefficient-based algorithm proved to be the quickest in terms of time performance, followed by the forward-sequential- and correlation-coefficient-based algorithms, which also exhibited faster speeds. However, the backward-sequential-based technique showed the worst per-

Table 1
The results of the LSVMs-based IDSs, trained on the feature subsets of the **KDD CUP 1999** dataset, selected via the:

(a) **Backward sequential method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9825	0.9024	0.9407	12146
Attack	0.9425	0.9901	0.9657	19644
—	—	—	—	—
Accuracy			0.9566	31790
Macro AVG	0.9625	0.9462	0.9532	31790
Weighted AVG	0.9578	0.9566	0.9562	31790
—	—	—	—	—
Training Time	557.577	ms	—	—
Test Time-All Packets	0.462	ms	—	—
Test Time-Per Packet	0.000015	ms	—	—

(b) **Correlation coefficients method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9846	0.8268	0.8988	12146
Attack	0.9026	0.992	0.9452	19644
—	—	—	—	—
Accuracy			0.9289	31790
Macro AVG	0.9436	0.9094	0.922	31790
Weighted AVG	0.9339	0.9289	0.9275	31790
—	—	—	—	—
Training Time	1042.354	—	—	—
Test Time-All Packets	0.862	—	—	—
Test Time-Per Packet	0.000027	—	—	—

(c) **Forward sequential method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9739	0.8951	0.9329	12146
Attack	0.9382	0.9852	0.9611	19644
—	—	—	—	—
Accuracy			0.9508	31790
Macro AVG	0.9561	0.9401	0.947	31790
Weighted AVG	0.9519	0.9508	0.9503	31790
—	—	—	—	—
Training Time	760.522	—	—	—
Test Time-All Packets	0.425	—	—	—
Test Time-Per Packet	0.000013	—	—	—

(d) **Importance coefficient method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9853	0.8395	0.9066	12146
Attack	0.9091	0.9923	0.9488	19644
—	—	—	—	—
Accuracy			0.9339	31790
Macro AVG	0.9472	0.9159	0.9277	31790
Weighted AVG	0.9382	0.9339	0.9327	31790
—	—	—	—	—
Training Time	261.460	—	—	—
Test Time-All Packets	0.633	—	—	—
Test Time-Per Packet	0.000020	—	—	—

(e) **Full features.**

	Precision	Re-Call	F1-Score	Support
Normal	0.8917	0.8803	0.886	12146
Attack	0.9266	0.9339	0.9302	19644
—	—	—	—	—
Accuracy			0.9134	31790
Macro AVG	0.9092	0.9071	0.9081	31790
Weighted AVG	0.9133	0.9134	0.9133	31790
—	—	—	—	—
Training Time	1259.983	—	—	—
Test Time-All Packets	1.132	—	—	—
Test Time-Per Packet	0.000036	—	—	—

Table 2
The results of the LSVMs-based IDSs, trained on the feature subsets of the **BotIoT-2018** dataset, selected via the:

(a) **Backward sequential method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9492	1.0	0.9739	112
Attack	1.0	0.9943	0.9971	1046
—	—	—	—	—
Accuracy			0.9948	1158
Macro AVG	0.9746	0.9971	0.9855	1158
Weighted AVG	0.9951	0.9948	0.9949	1158
—	—	—	—	—
Training Time	21.053	—	—	—
Test Time-All Packets	0.157	—	—	—
Test Time-Per Packet	0.000135	—	—	—

(b) **Correlation coefficients method**

	Precision	Re-Call	F1-Score	Support
Normal	0.8667	0.6964	0.7723	112
Attack	0.9682	0.9885	0.9782	1046
—	—	—	—	—
Accuracy			0.9603	1158
Macro AVG	0.9174	0.8425	0.8753	1158
Weighted AVG	0.9583	0.9603	0.9583	1158
—	—	—	—	—
Training Time	41.893	—	—	—
Test Time-All Packets	0.394	—	—	—
Test Time-Per Packet	0.000340	—	—	—

(c) **Forward sequential method**

	Precision	Re-Call	F1-Score	Support
Normal	0.5271	0.6071	0.5643	112
Attack	0.9572	0.9417	0.9494	1046
—	—	—	—	—
Accuracy			0.9093	1158
Macro AVG	0.7422	0.7744	0.7569	1158
Weighted AVG	0.9156	0.9093	0.9122	1158
—	—	—	—	—
Training Time	15.447	—	—	—
Test Time-All Packets	0.155	—	—	—
Test Time-Per Packet	0.000134	—	—	—

(d) **Importance coefficient method**

	Precision	Re-Call	F1-Score	Support
Normal	1.0	0.1339	0.2362	112
Attack	0.9151	1.0	0.9557	1046
—	—	—	—	—
Accuracy			0.9162	1158
Macro AVG	0.9576	0.567	0.596	1158
Weighted AVG	0.9233	0.9162	0.8861	1158
—	—	—	—	—
Training Time	42.372	—	—	—
Test Time-All Packets	0.143	—	—	—
Test Time-Per Packet	0.000123	—	—	—

(e) **Full features.**

	Precision	Re-Call	F1-Score	Support
Normal	0.0919	0.9196	0.1671	112
Attack	0.7568	0.0268	0.0517	1046
—	—	—	—	—
Accuracy			0.1131	1158
Macro AVG	0.4243	0.4732	0.1094	1158
Weighted AVG	0.6925	0.1131	0.0629	1158
—	—	—	—	—
Training Time	27.335	—	—	—
Test Time-All Packets	0.219	—	—	—
Test Time-Per Packet	0.000189	—	—	—

Table 3
The results of the LSVM-based IDSs, trained on the feature subsets of the **N-BaIoT-2021** dataset, selected via the:

(a) **Backward sequential method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9995	0.9976	0.9986	45424
Attack	0.9953	0.9989	0.9971	22650
—	—	—	—	—
Accuracy			0.9981	68074
Macro AVG	0.9974	0.9983	0.9978	68074
Weighted AVG	0.9981	0.9981	0.9981	68074
—	—	—	—	—
Training Time	1004.473	ms	—	—
Test Time-All Packets	0.648	ms	—	—
Test Time-Per Packet	0.000010	ms	—	—

(b) **Correlation coefficients method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9954	0.924	0.9584	45424
Attack	0.8667	0.9913	0.9249	22650
—	—	—	—	—
Accuracy			0.9464	68074
Macro AVG	0.9311	0.9577	0.9416	68074
Weighted AVG	0.9526	0.9464	0.9472	68074
—	—	—	—	—
Training Time	5394.409	ms	—	—
Test Time-All Packets	7.615	ms	—	—
Test Time-Per Packet	0.000112	ms	—	—

(c) **Forward sequential method**

	Precision	Re-Call	F1-Score	Support
Normal	0.9911	0.9939	0.9925	45424
Attack	0.9877	0.9821	0.9849	22650
—	—	—	—	—
Accuracy			0.99	68074
Macro AVG	0.9894	0.988	0.9887	68074
Weighted AVG	0.99	0.99	0.99	68074
—	—	—	—	—
Training Time	1240.104	ms	—	—
Test Time-All Packets	0.804	ms	—	—
Test Time-Per Packet	0.000012	ms	—	—

(d) **Importance coefficient method**

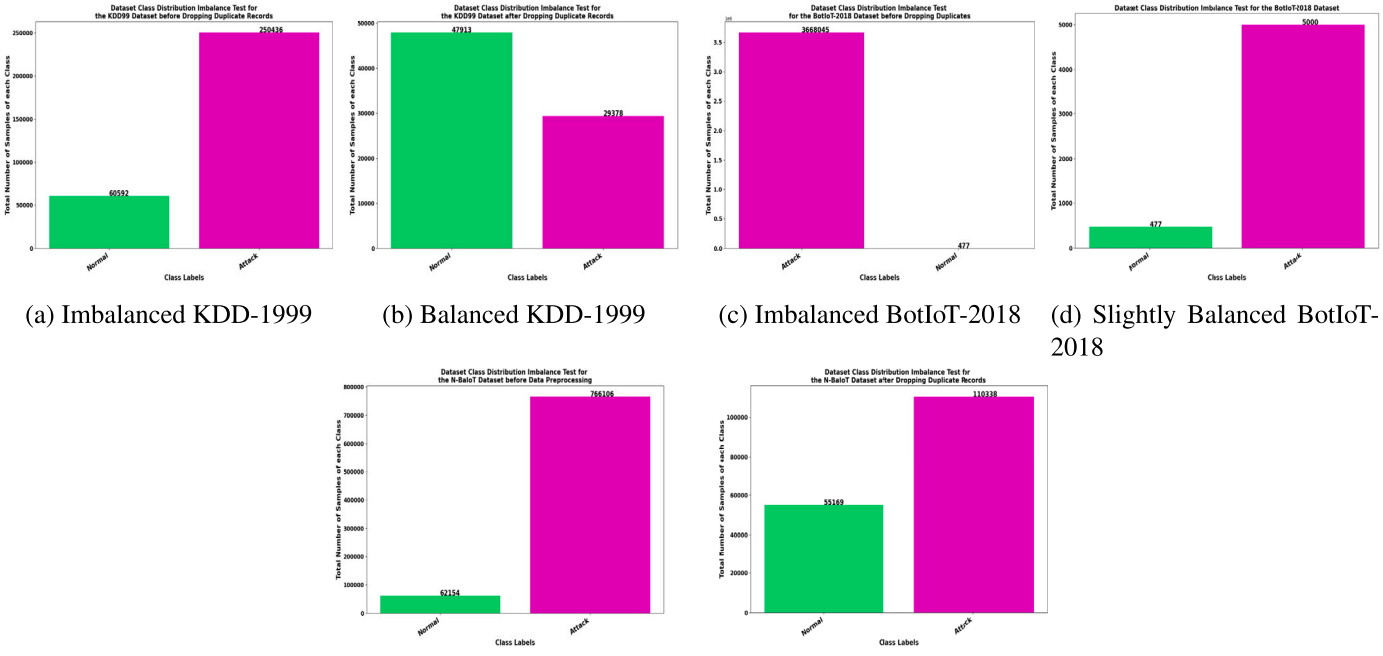
	Precision	Re-Call	F1-Score	Support
Normal	0.927	0.9816	0.9535	45424
Attack	0.9581	0.845	0.898	22650
—	—	—	—	—
Accuracy			0.9361	68074
Macro AVG	0.9426	0.9133	0.9258	68074
Weighted AVG	0.9374	0.9361	0.9351	68074
—	—	—	—	—
Training Time	2405.869	ms	—	—
Test Time-All Packets	0.669	ms	—	—
Test Time-Per Packet	0.000010	ms	—	—

(e) **Full features.**

	Precision	Re-Call	F1-Score	Support
Normal	0.9952	0.7698	0.8681	45424
Attack	0.6826	0.9925	0.8089	22650
—	—	—	—	—
Accuracy			0.8439	68074
Macro AVG	0.8389	0.8812	0.8385	68074
Weighted AVG	0.8912	0.8439	0.8484	68074
—	—	—	—	—
Training Time	11891.864	ms	—	—
Test Time-All Packets	4.820	ms	—	—
Test Time-Per Packet	0.000071	ms	—	—

Table 4
Comparison of the study results with the results of some similar works. FF-full features, IP: importance coefficient, BS and FS: backward- and forward- sequential, and CC: correlation coefficient-based feature selectors.

Methods/Results	Precision (Normal)	Precision (Attack)	Re-Call (Normal)	Re-Call (Attack)	F1-Score (Normal)	F1-Score (Attack)	Accuracy	Training Time	Test Time-All Packets	Test Time-Per Packet
KDD-Cup-1999 Dataset										
C-SVM with linear kernel_FF	0.0211	0.7567	0.0	0.9997	0.0	0.8614	0.7565	4942.586	3068.600	0.007680
SGDC_FF	0.1681	0.7237	0.2113	0.6639	0.1872	0.6925	0.5538	895.712	11.163	0.000028
LSVMs_FF	0.8917	0.9266	0.8803	0.9339	0.886	0.9302	0.9134	1259.983	1.132	0.000036
LSVMs_Ridge Regression_IP	**0.9757**	**0.9504**	**0.9219**	**0.9849**	**0.948**	**0.9673**	**0.9599**	**768.770**	**1.023**	**0.000017**
LSVMs_Ridge Regression_BS	**0.9825**	**0.9425**	**0.9024**	**0.9901**	**0.9407**	**0.9657**	**0.9566**	**557.577**	**0.462**	**0.000015**
LSVMs_Ridge Regression_FS	**0.9739**	**0.9382**	**0.8951**	**0.9852**	**0.9329**	**0.9611**	**0.9508**	**760.522**	**0.425**	**0.000013**
LSVMs_Ridge Regression_CC	**0.9846**	**0.9026**	**0.8268**	**0.992**	**0.8988**	**0.9452**	**0.9289**	**1042.354**	**0.862**	**0.000027**
BotIoT-2018 Dataset										
C-SVM with linear kernel_FF	0.0007	0.9999	0.0231	0.9957	0.0014	0.9978	0.9955	10818.091	5335.290	0.005353
SGDC_FF	0.4615	0.9999	0.0462	1.0	0.0839	0.9999	0.9999	10933.364	13.726	0.000014
LSVMs_FF	0.0919	0.7568	0.9196	0.0268	0.1671	0.0517	0.1131	27.335	0.219	0.000189
LSVMs_Ridge Regression_IP	**1.0**	**0.9151**	**0.1339**	**1.0**	**0.2362**	**0.9557**	**0.9162**	**42.372**	**0.143**	**0.000123**
LSVMs_Ridge Regression_BS	**0.9492**	**1.0**	**1.0**	**0.9943**	**0.9739**	**0.9971**	**0.9948**	**21.053**	**0.157**	**0.000135**
LSVMs_Ridge Regression_FS	**0.5271**	**0.9572**	**0.6071**	**0.9417**	**0.5643**	**0.9494**	**0.9093**	**15.447**	**0.155**	**0.000134**
LSVMs_Ridge Regression_CC	**0.8667**	**0.9682**	**0.6964**	**0.9885**	**0.7723**	**0.9782**	**0.9603**	**41.893**	**0.394**	**0.000340**
N-BaIoT-2021 Dataset										
C-SVM with linear kernel_FF	0.28	0.3319	0.0005	0.9976	0.0009	0.4981	0.3319	1480.917	513.965	0.011429
SGDC_FF	0.6732	0.9275	0.999	0.0257	0.8044	0.05	0.6755	2097.278	3.413	0.000076
LSVMs_FF	**0.9952**	**0.6826**	**0.7698**	**0.9925**	**0.8681**	**0.8089**	**0.8439**	**11891.864**	**4.820**	**0.000071**
LSVMs_Ridge Regression_IP	**0.927**	**0.9581**	**0.9816**	**0.845**	**0.9535**	**0.898**	**0.9361**	**2405.869**	**0.669**	**0.000010**
LSVMs_Ridge Regression_BS	**0.9995**	**0.9953**	**0.9976**	**0.9989**	**0.9986**	**0.9971**	**0.9981**	**1004.473**	**0.648**	**0.000010**
LSVMs_Ridge Regression_FS	**0.9911**	**0.9877**	**0.9939**	**0.9821**	**0.9925**	**0.9849**	**0.99**	**1240.104**	**0.804**	**0.000012**
LSVMs_Ridge Regression_CC	**0.9954**	**0.8667**	**0.924**	**0.9913**	**0.9584**	**0.9249**	**0.9464**	**5394.409**	**7.615**	**0.000112**



(a) Imbalanced KDD-1999 (b) Balanced KDD-1999 (c) Imbalanced BotIoT-2018 (d) Slightly Balanced BotIoT-2018

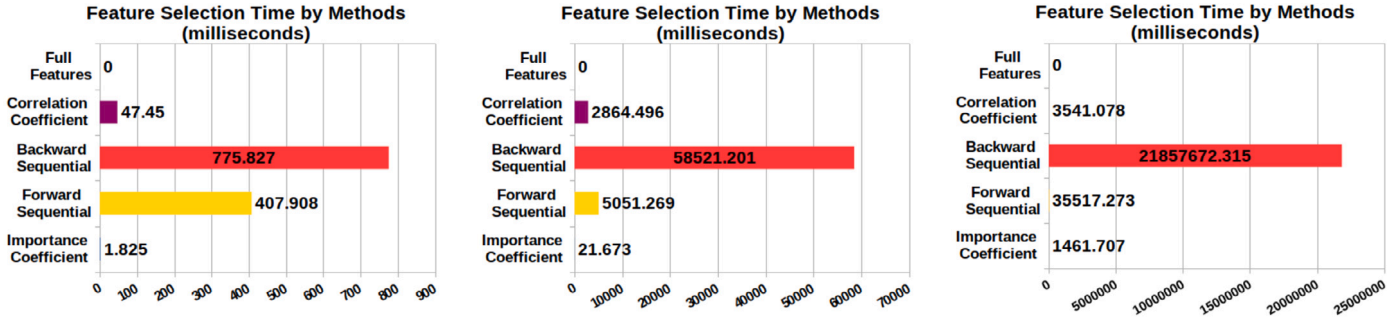
(e) Imalanced N-BaIoT-2021 (f) Balanced N-BaIoT-2021

Fig. 3. Dataset balancing results.

formance in terms of selection time, taking 6.07 hours to select the subset of features on the N-BaIot-2021 dataset, Fig. 4.

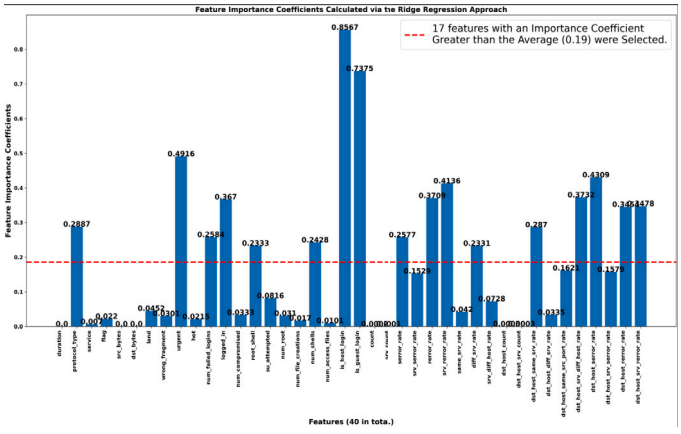
4.2.1. The selected sub-subsets via the importance coefficients

The three sub-subsets were chosen from the feature subsets of the BotIoT-2018, KDD-CUP-1999, and N-BaIoT-2021 datasets. These feature subsets were selected based on the importance coefficients of the features, specifically those that were greater than or equal to the average importance coefficient of all features in the three sets. The 6, 17, and 34 features selected from the original 15, 40, and 115 features in the datasets. From these three feature subsets, the six most important and efficient features were selected as the three sub-subsets. The selected subsets and sub-subsets are depicted in Fig. 5. The algorithm was able to select the sub-subsets with the best performance in terms of feature selection time when compared to other selection methods on all datasets used in this study, see Fig. 4. This is an important consideration because the selection of features in a shorter time can have a significant impact on the overall performance of an ML model in real-time training. By efficiently selecting the most relevant and informative features, the algorithm is able to improve the accuracy and efficiency of

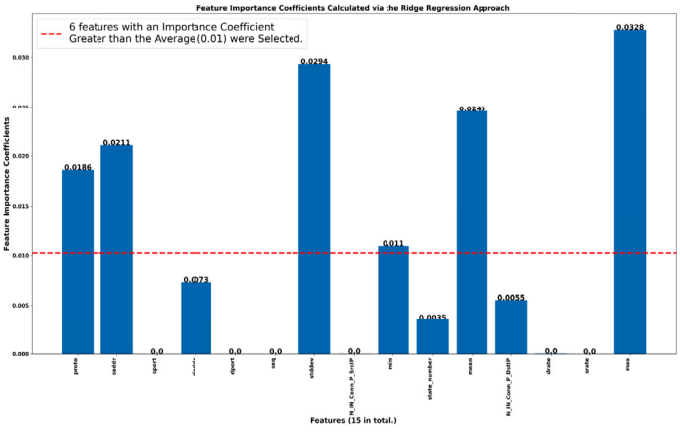


(a) Selection time on BotIoT-2018 (b) Selection time on KDD-Cup-1999 (c) Selection time on NBaIoT-2021

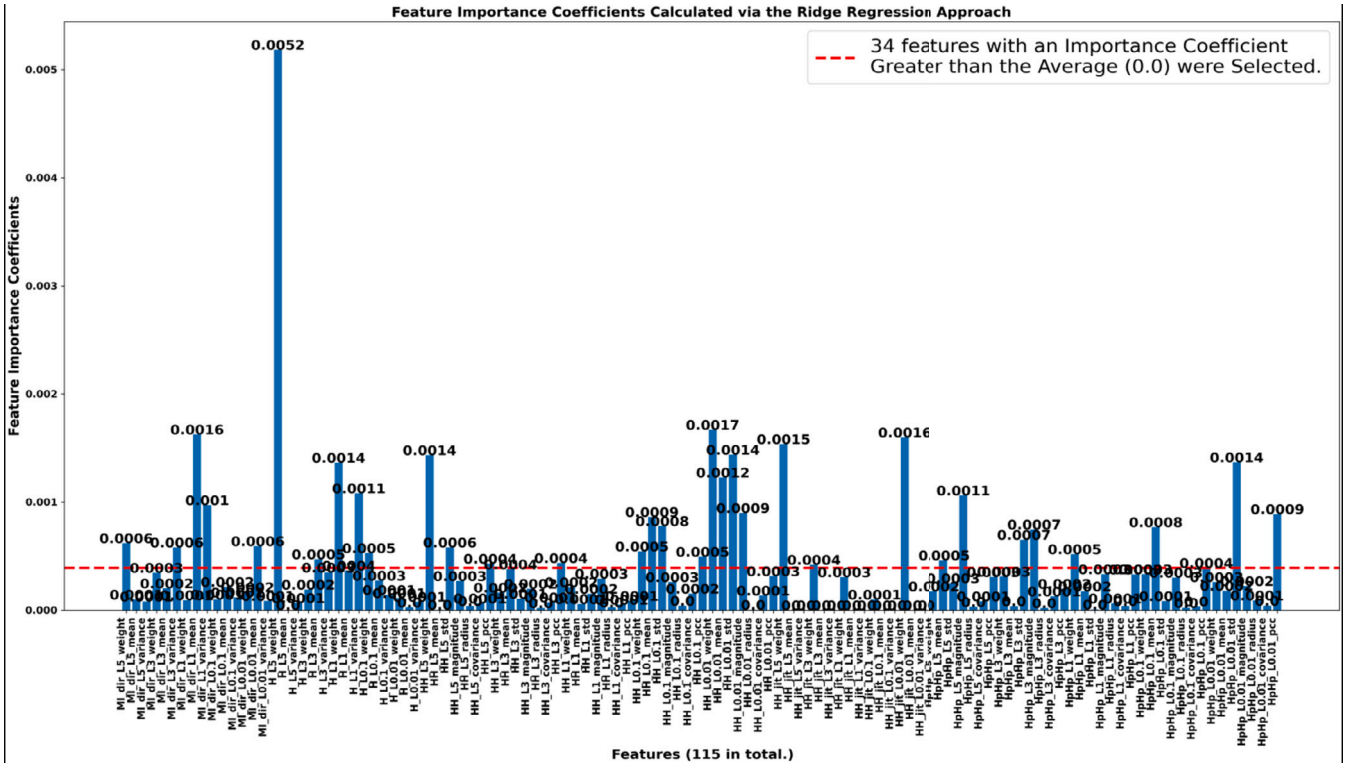
Fig. 4. Time performance of the feature selection algorithms by the datasets.



(a) A subset of 17 selected features of the KDD-1999 dataset.

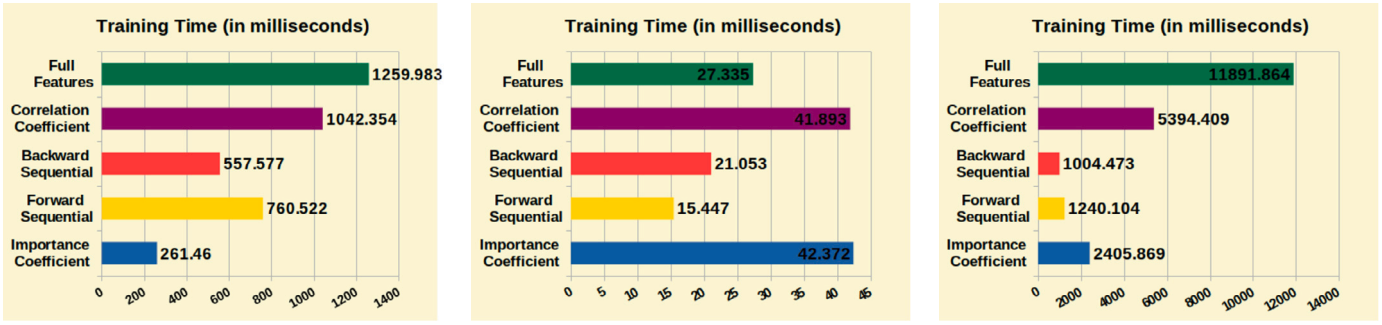


(b) A subset of 6 selected features of the BotIoT-2018 dataset.



(c) A subset of 34 selected features of the NBaIoT-2021 dataset.

Fig. 5. Feature importance coefficient (IP) values of subsets (a, b, c) selected from the original feature sets of the datasets. Features with an IP greater than or equal to the average of all IPs of these features were selected as subsets.



(a) Time on the KDD-1999 dataset. (b) Time on the BotIoT-2018 dataset. (c) Time on the N-BaIoT-2021 dataset.

Fig. 6. Training time performance of the IDSs by the datasets.

the model. In addition, faster feature selection can also reduce the overall time and resources required for training and deploying the model, making it more practical for real-world applications.

4.2.2. *The selected subsets via the forward- and backward-sequential algorithms*

Another six subsets, each containing important and efficient features, were selected from the feature sets of the BotIoT-2018, KDD-CUP-1999, and N-BaIoT-2021 datasets using forward- & backward-sequential methods. Both feature selection methods, the forward- & the backward-sequential approaches, yielded excellent results in terms of the accuracy of the linear models used for intrusion detection when trained on the subsets of features selected by these methods. The forward feature selector was able to choose the feature subsets relatively quickly at 409.91 ms, 5051.27 ms, and 35517.27 ms, while the backward feature selector was significantly slower on all three datasets. In fact, the backward model took 775.8 milliseconds, 58521.2 milliseconds, and 6.07 hours to select the feature subsets from the three datasets (BotIoT-2018, KDD-1999, NBaIoT-2021), respectively. Despite its slower performance, the backward feature selector still provided accurate results for the linear models used in intrusion detection. See Fig. 4 and Tables 1, 2, and 3 for further details.

4.2.3. *The selected subsets via correlation coefficients*

The final three subsets of the selected features that utilized later in training of the LSVM models, were chosen from the larger feature sets of the datasets based on the correlation coefficients between the input features and the output class labels. These selected subsets were determined to be important and efficient, as they resulted in good performance in terms of both intrusion detection accuracy and packet processing times (Tables 1b, 2b, 3b). The performance time of the feature selector on the datasets can be seen in Fig. 4.

4.3. *The IDSs trained via the selected and full features*

The LSVMs classifier-based IDSs trained on the selected feature subsets of the datasets outperformed the same IDS models trained on the full feature sets of the datasets in terms of both training-testing time and accuracy performance (Tables 1, 2, and 3). These results demonstrate the effectiveness of using selected feature subsets in the training of LSVMs-based IDSs. The use of selected feature subsets significantly reduced the training-testing time, while improving the accuracy of the models. This suggests that the inclusion of all features in the training dataset may not always be necessary, and that carefully selecting a subset of relevant features can lead to more efficient models. One possible explanation for these findings is that the full feature sets of the datasets that contained redundant or irrelevant information negatively impacted the performance of the IDSs. By selecting the subsets of relevant features, the training process was able to focus on the most important information, leading to improved performance. Further details were given in the following subsections.

4.3.1. *Training time performance of the IDSs*

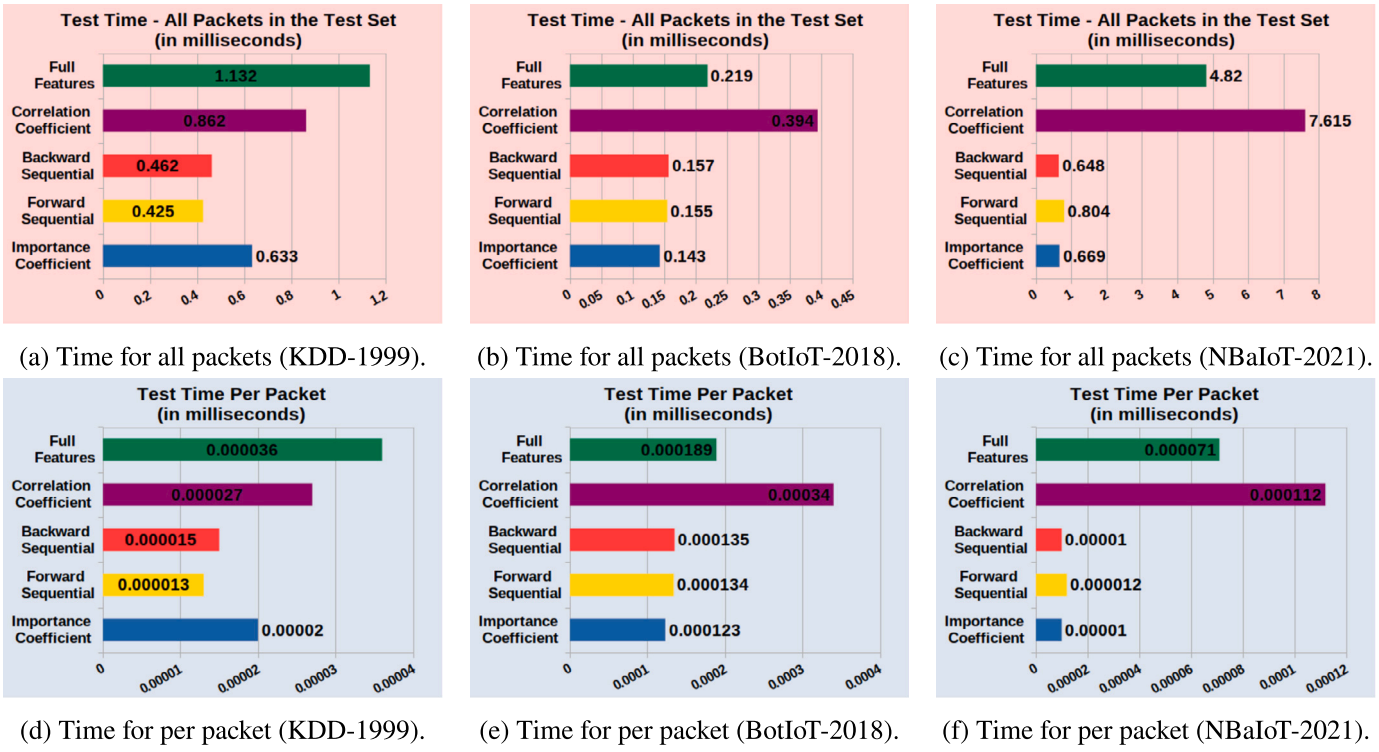
The training process for all 15 subsets of features using the LSVMs model was quite efficient, with training times ranging from the lowest (15.447 milliseconds) for the model trained on the set of 6 features of the BotIoT-2018 dataset (selected using the backward sequential-based feature selector) to the highest (11.89 seconds) for the models trained on the full features of the N-BaIoT-2021 dataset, Fig. 6. All of the models were trained for a total of 500 epochs. One of the reasons for the significantly reduced training times was the fact that the linear models were trained using a smaller number of features from the datasets. Four different feature selection algorithms reduced the number of features in the training subsets: importance-based, backward- and forward-sequential-based, and correlation coefficient-based. These algorithms were able to reduce the number of features from 115 (the full feature set) down to just 6, which greatly reduced the training time for the models. Overall, the training times for the models were significantly lower than they would have been if the full feature sets had been used.

4.3.2. *Testing time performance of the IDSs*

The performance of IDSs trained on the subsets of the selected features from the datasets was measured with the test sets, which contains 31790 (KDD CUP 1999), 1158 (BotIoT-2018), and 68074 (N-BaIoT-2021) packets, respectively. It was found that the test time performance for these models was better than that of the IDSs trained on the full feature sets of the same datasets (Tables 1, 2, and 3). This is because the reduced number of features in the subsets contributed to a decrease in the test time of the models. For example, the packet processing time was 2.23, 1.44, and 6.82 times faster for the models trained on the feature subsets compared to the models trained on the full feature sets. However, there were two exceptions where the test time performance for the models trained on the subsets selected using the correlation coefficient-based feature selector was worse than the test time performance for the same models trained on the full feature sets. These exceptions occurred for the BotIoT-2018 and N-BaIoT-2021 datasets, with test times of 0.394 ms and 7.615 ms for the models trained on the subsets, compared to 0.219 ms and 4.82 ms for the models trained on the full feature sets (Fig. 7). The exact reason for this is unknown, but it is possible that the CPU's workload may have played a role. The increased workload of the computer's CPUs may have resulted in a longer test time for the models trained on the subsets selected using the correlation coefficients.

4.3.3. *Accuracy performance of the IDSs*

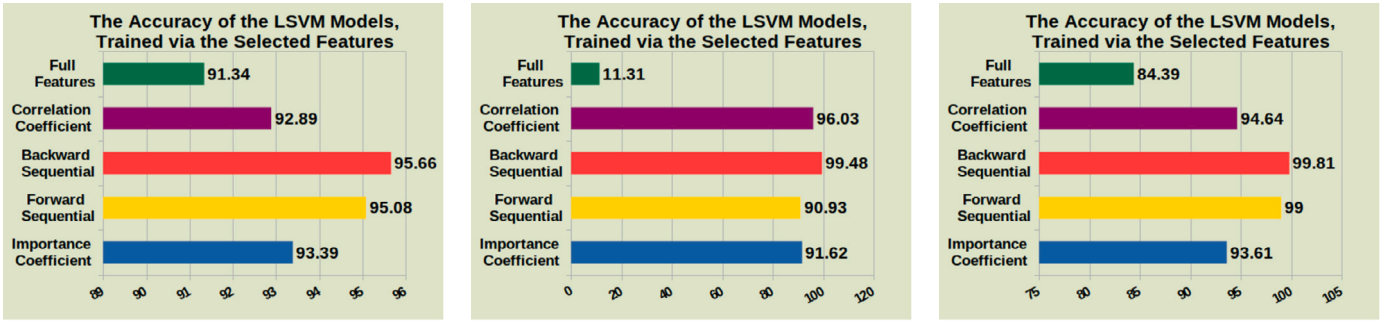
The intrusion detection outcomes showed that the linear models trained on the subsets of the datasets were able to achieve sufficient and promising results (Fig. 8). This was due to the fact that the linear models were trained with the most important and efficient features, and the dropping of unimportant and inefficient features helped to increase



(a) Time for all packets (KDD-1999). (b) Time for all packets (BotIoT-2018). (c) Time for all packets (NBaIoT-2021).

(d) Time for per packet (KDD-1999). (e) Time for per packet (BotIoT-2018). (f) Time for per packet (NBaIoT-2021).

Fig. 7. The test time performance of the IDSs by the datasets.



(a) Accuracy (KDD-Cup-1999). (b) Accuracy (BotIoT-2018). (c) Accuracy (N-BaIoT-2021).

Fig. 8. Accuracy of the IDSs by the datasets.

the accuracy. The accuracy results for the models trained on the full feature sets of the KDD-CUP-1999, BotIoT-2018, and NBaIoT-2021 datasets were 91.34%, 11.31%, and 84.39%, respectively. However, even better accuracy results were obtained with the models trained on the subsets of the features selected with all of the feature selectors. The worst accuracy of 11.31% was achieved with the model trained on the full features of the BotIoT-2018 dataset, which was likely due to the presence of inefficient features in the dataset. The models trained on the subsets of the features selected with the backward sequential feature selector had the best accuracy results, with 95.66% for the KDD-CUP-1999, 99.48% for the BotIoT-2018, and 99.81% for the N-BaIoT-2021 datasets (Tables 1, 2, and 3). While the backward sequential feature selector was able to achieve the highest accuracy results, it also had the worst feature selection time among the other feature selection methods used in the study. This means that it may not be the best choice for situations where feature selection time is a concern. However, it could still be useful for developing lightweight IDSs, especially if the feature selection time can be significantly reduced. Overall, the models trained on the subsets of features selected with all feature selection algorithms performed very well in terms of both intrusion detection time and accuracy. This suggests that any of the feature selection algorithms could be used for the development of lightweight IDSs. It is worth noting that the linear models trained with the backward sequential feature selection technique had the highest detection accuracy, but it may not be the most practical choice in certain situations due to its longer feature selection time.

4.3.4. *Comparison of the study results with similar works*

Table 4 presents a comprehensive comparison of the study with some similar works, specifically focusing on the C-SVM with linear kernel (C-SVML) and Stochastic Gradient Descent Classifier (SGDC) methods in the field. While there are many studies that could be compared, we chose to specifically compare with C-SVML and SGDC methods because they are also linear models of ML, just like LSVMs. The table provides performance metrics for the three methods on three datasets. It was observed that the C-SVML and SGDC classifiers, when trained on the full feature sets of KDD-Cup-1999 and N-BaIoT-2021 datasets, did not perform well in terms of all metrics. However, when the these methods were trained on the BotIoT-2018 dataset, demonstrated accurate detection of attack packets, although it had the worst performance

in terms of training and test time as well as normal packet detection. On the other hand, the IDSs based on the LSVMs classifier, which was trained with the relevant and efficient subsets of selected features, exhibited accurate performance across all metrics. These IDSs were also the lightest in terms of both training and testing time. Overall, the study's findings highlight the effectiveness of the proposed lightweight IDSs, which utilize LSVMs with ridge regression and feature selection methods, in achieving high accuracy and efficiency in intrusion detection on IoT networks. This indicates the potential of these lightweight IDSs to significantly enhance the security and performance of IoT networks.

Limitations of the study and future solutions. Numerous factors can influence the outcomes of the present research. These include datasets, feature selection methodologies, and classification algorithms. The initial factor to consider revolves around datasets. The authenticity and practicality of the dataset (being real or realistic) can significantly shape the simulation environment encompassing internet protocols, IoT devices, and software tools. Moreover, the range of attacks, along with their typologies, within the dataset can wield either positive or negative impacts on the trained IDSs. The subsequent factor lies within the domain of feature selection algorithms and the regression models embedded within the core of these feature selectors. With a pool of 29 feature selection techniques at disposal, the meticulous choice of suitable algorithms and their associated kernel regression models becomes paramount. Failing to exercise prudence in this selection process could result in the identification of inappropriate feature subsets by these algorithms, thereby casting a shadow over the holistic performance of the entire system. Lastly, the efficacy of classifiers occupies a pivotal role in the construction of accurate and lightweight IDSs. The judicious selection of classifiers holds the potential to significantly elevate the IDSs' performance. It is imperative to opt for classification algorithms that harmonize with the inherent characteristics of the selected dataset. By embracing lightweight classification techniques tailored to the dataset's intrinsic nature, the pursuit of high-performance IDSs can be effectively realized.

5. Conclusion

The lightweight IDSs for IoT networks were developed using a fine-tuned LSVMs model and four feature selection methods. The study findings demonstrate that using the selected feature subsets of the datasets led to promising detection results. This was due to the fact that the IDSs were trained with the most important and efficient features, and the removal of unimportant and inefficient features contributed to the increasing accuracy of IDSs. The models trained on the full feature sets of the KDD-CUP-1999, BotIoT-2018, and NBaIoT-2021 datasets achieved IDS accuracy of 91.34%, 11.31%, and 84.39%, respectively. However, the models trained on the feature subsets, selected with the feature selectors performed far better than the models trained with full feature sets of the same datasets. The worst accuracy of 11.31% was achieved with the model trained on the full features of the BotIoT-2018 dataset, which was likely due to the presence of inefficient features within the dataset. On the other hand, the models trained on subsets of the features selected with the backward sequential feature selector had the best accuracy results, with 95.66% for KDD CUP 2018, 99.48% for BotIoT-2018, and 99.81% for N-BaIoT-2021. This indicates that using this feature selector can lead to higher detection accuracy. However, this method had the worst feature selection time compared to other three approaches. Thus, it may not be the best choice when time is a factor while it may be effective in terms of accuracy. Nonetheless, with further optimization and improvements, the selection time of this technique could potentially be used for lightweight IDSs. Overall, the results of the study suggest that using feature subsets, selected with any of the feature selectors can be effective for developing efficient and accurate IDSs. However, the specific technique chosen may depend on the priorities, whether it be accuracy or feature selection time.

Funding

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2022R1I1A3072355).

CRediT authorship contribution statement

Jahongir Azimjonov: Conceptualization, Data curation, Formal analysis, Methodology, Software, Visualization, Writing – original draft. **Taehong Kim:** Data curation, Funding acquisition, Investigation, Project administration, Resources, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The materials and data were shared with this link: https://github.com/JahongirAzimjonov/Lightweight-Intrusion-Detection-Systems-via-Feature-Selection-and-LinearSVM.

References

B, S., et al., 2019. Firefly algorithm based feature selection for network intrusion detection. Comput. Secur. 81, 148–155. https://doi.org/10.1016/j.cose.2018.11.005.
Bay, S.D., et al., 2000. The uci kdd archive of large data sets for data mining research and experimentation. ACM SIGKDD Explor. Newsl. 2, 81–85. https://doi.org/10.1145/380995.381030.
Burhan, M., et al., 2018. IoT elements, layered architectures and security issues: a comprehensive survey. Sensors (Switzerland) 18, 1–37. https://doi.org/10.3390/s18092796.
Ding, Y., et al., 2013. A fast malware detection algorithm based on objective-oriented association mining. Comput. Secur. 39, 315–324. https://doi.org/10.1016/j.cose.2013.08.008.
Eesa, A.S., et al., 2015. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. Expert Syst. Appl. 42, 2670–2679. https://doi.org/10.1016/j.eswa.2014.11.009.
Elrawy, M.F., et al., 2018. Intrusion detection systems for iot-based smart environments: a survey. J. Cloud Comput. 7, 21. https://doi.org/10.1186/s13677-018-0123-6.
Emerson, R.W., 2015. Causation and Pearson's correlation coefficient. J. Vis. Impair. Blind. 109, 242–244. https://doi.org/10.1177/0145482X1510900311.
Fraley, J.B., et al., 2017. The promise of machine learning in cybersecurity. In: Southeast-Con 2017, pp. 1–6.
Galal, H.S., et al., 2016. Behavior-based features model for malware detection. J. Comput. Virol. Hacking Tech. 12, 59–67. https://doi.org/10.1007/s11416-015-0244-0.
Garcia, S., et al., 2020. IoT-23: a labeled dataset with malicious and benign IoT network traffic. https://doi.org/10.5281/zenodo.4743746. More details here https://www.stratosphereips.org/datasets-iot23, 2020.
Ghiasi, M., et al., 2015. Dynamic vsa: a framework for malware detection based on register contents. Eng. Appl. Artif. Intell. 44, 111–122. https://doi.org/10.1016/j.engappai.2015.05.008.
Gibert, D., et al., 2020. The rise of machine learning for detection and classification of malware: research developments, trends and challenges. J. Netw. Comput. Appl. 153, 102526. https://doi.org/10.1016/j.jnca.2019.102526.
Hajiheidari, S., et al., 2019. Intrusion detection systems in the Internet of things: a comprehensive investigation. Comput. Netw. 160, 165–191. https://doi.org/10.1016/j.comnet.2019.05.014.
Hindy, H., et al., 2020. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. IEEE Access 8, 104650–104675. https://doi.org/10.1109/ACCESS.2020.3000179.
Kasongo, S.M., et al., 2020. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Comput. Secur. 92, 101752. https://doi.org/10.1016/j.cose.2020.101752.
Khammassi, C., et al., 2017. A ga-lr wrapper approach for feature selection in network intrusion detection. Comput. Secur. 70, 255–277. https://doi.org/10.1016/j.cose.2017.06.005.
Kheir, N., 2013. Behavioral classification and detection of malware through http user agent anomalies. In: sETOP'2012 and FPS'2012 Special Issue. J. Inf. Secur. Appl. 18, 2–13. https://doi.org/10.1016/j.jisa.2013.07.006.
Kolias, C., et al., 2017. Ddos in the iot: Mirai and other botnets. Computer 50, 80–84.

Koroniotis, N., et al., 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Gener. Comput. Syst. 100, 779–796. https://doi.org/10.1016/j.future.2019.05.041.

Kumar, A., et al., 2022. Machine learning-based early detection of iot botnets using network-edge traffic. Comput. Secur. 117, 102693. https://doi.org/10.1016/j.cose.2022.102693.

Li, Y., et al., 2009. Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. Comput. Secur. 28, 466–475. https://doi.org/10.1016/j.cose.2009.01.001.

Lima, M., et al., 2022. Beholder – a cep-based intrusion detection and prevention systems for iot environments. Comput. Secur. 120, 102824. https://doi.org/10.1016/j.cose.2022.102824.

Madakam, S., et al., 2015. Internet of things (iot): a literature review. J. Comput. Commun. 3, 164–173.

Mashal, I., et al., 2015. Choices for interaction with things on Internet and underlying issues. Ad Hoc Netw. 28, 68–90. https://doi.org/10.1016/j.adhoc.2014.12.006.

Meidan, Y., et al., 2018. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Comput. 17, 12–22. https://doi.org/10.1109/MPRV.2018.03367731.

Mitchell, R., et al., 2014. A survey of intrusion detection in wireless network applications. Comput. Commun. 42, 1–23. https://doi.org/10.1016/j.comcom.2014.01.012.

Raff, E., et al., 2017. Malware detection by eating a whole exe. arXiv:1710.09435.

Ring, M., et al., 2019. A survey of network-based intrusion detection data sets. Comput. Secur. 86, 147–167. https://doi.org/10.1016/j.cose.2019.06.005.

Roy, S., et al., 2022. A lightweight supervised intrusion detection mechanism for iot networks. Future Gener. Comput. Syst. 127, 276–285. https://doi.org/10.1016/j.future.2021.09.027.

Sahoo, K.S., et al., 2020. An evolutionary svm model for ddos attack detection in software defined networks. IEEE Access 8, 132502–132513. https://doi.org/10.1109/ACCESS.2020.3009733.

Sahoo, K.S., Puthal, D., 2020. Sdn-assisted ddos defense framework for the Internet of multimedia things. ACM Trans. Multimed. Comput. Commun. Appl. 16. https://doi.org/10.1145/3394956.

Sahu, S.K., et al., 2021. An ensemble-based scalable approach for intrusion detection using big data framework. Big Data 9, 303–321. https://doi.org/10.1089/big.2020.0201. pMID: 34271836.

Said, O., et al., 2013. Towards Internet of things: survey and future vision. Int. J. Comput. Netw. 5, 1–17.

Salehi, Z., et al., 2017. Maar: robust features to detect malicious activity based on api calls, their arguments and return values. Eng. Appl. Artif. Intell. 59, 93–102. https://doi.org/10.1016/j.engappai.2016.12.016.

Shafiq, M., et al., 2020. Iot malicious traffic identification using wrapper-based feature selection mechanisms. Comput. Secur. 94, 101863. https://doi.org/10.1016/j.cose.2020.101863.

Wilkinson, M.D., et al., 2016. The fair guiding principles for scientific data management and stewardship. Sci. Data 3, 160018. https://doi.org/10.1038/sdata.2016.18.

Yaqoob, I., et al., 2017. Internet of things architecture: recent advances, taxonomy, requirements, and open challenges. IEEE Wirel. Commun. 24, 10–16.

Zarpelão, B.B., et al., 2017. A survey of intrusion detection in Internet of things. J. Netw. Comput. Appl. 84, 25–37. https://doi.org/10.1016/j.jnca.2017.02.009.

Zhang, C., et al., 2022. Comparative research on network intrusion detection methods based on machine learning. Comput. Secur. 121, 102861. https://doi.org/10.1016/j.cose.2022.102861.

Jahongir Azimjonov received a Ph.D. degree in Computer and Information Engineering from Sakarya University, Sakarya, Turkey, in September 2021. He is currently a postdoctoral researcher with the School of Information and Communication Engineering, Chungbuk National University, Cheongju-si, South Korea. He is also serving as a professional reviewer for over 10 well-reputed journals and conferences. His research interests include medical image analysis (breast and prostate cancer detection and diagnosis), cyber security (lightweight intrusion detection systems), and intelligent transportation systems (vehicle detection and tracking). He has registered 1 patent and published over 7 articles in peer-reviewed international journals and conferences in his research areas.

Taehong Kim (Senior Member, IEEE) received the B.S. degree in computer science from Ajou University, South Korea, in 2005, and the M.S. degree in information and communication engineering and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), in 2007 and 2012, respectively. He was a Research Staff Member at the Samsung Advanced Institute of Technology (SAIT) and the Samsung DMC Research and Development Center, from 2012 to 2014. He was a Senior Researcher at the Electronics and Telecommunications Research Institute (ETRI), South Korea, from 2014 to 2016. Since 2016, he has been an Associate Professor with the School of Information and Communication Engineering, Chungbuk National University, South Korea. His research interests include edge computing, container orchestration, the Internet of Things, and federated learning. He has been an Associate Editor of IEEE Access, since 2020.