Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.047239

ARTICLE

Suboptimal Feature Selection Techniques for Effective Malicious Traffic Detection on Lightweight Devices

So-Eun Jeon[1], Ye-Sol Oh[1], Yeon-Ji Lee[1] and Il-Gu Lee[1,2,*]

[1]Department of Future Convergence Technology Engineering, Sungshin Women's University, Seoul, 02844, Korea

[2]Department of Convergence Security Engineering, Sungshin Women's University, Seoul, 02844, Korea

*Corresponding Author: Il-Gu Lee. Email: iglee@sungshin.ac.kr

Received: 30 October 2023 Accepted: 23 February 2024 Published: 20 May 2024

ABSTRACT

With the advancement of wireless network technology, vast amounts of traffic have been generated, and malicious traffic attacks that threaten the network environment are becoming increasingly sophisticated. While signature-based detection methods, static analysis, and dynamic analysis techniques have been previously explored for malicious traffic detection, they have limitations in identifying diversified malware traffic patterns. Recent research has been focused on the application of machine learning to detect these patterns. However, applying machine learning to lightweight devices like IoT devices is challenging because of the high computational demands and complexity involved in the learning process. In this study, we examined methods for effectively utilizing machine learning-based malicious traffic detection approaches for lightweight devices. We introduced the suboptimal feature selection model (SFSM), a feature selection technique designed to reduce complexity while maintaining the effectiveness of malicious traffic detection. Detection performance was evaluated on various malicious traffic, benign, exploits, and generic, using the UNSW-NB15 dataset and SFSM sub-optimized hyperparameters for feature selection and narrowed the search scope to encompass all features. SFSM improved learning performance while minimizing complexity by considering feature selection and exhaustive search as two steps, a problem not considered in conventional models. Our experimental results showed that the detection accuracy was improved by approximately 20% compared to the random model, and the reduction in accuracy compared to the greedy model, which performs an exhaustive search on all features, was kept within 6%. Additionally, latency and complexity were reduced by approximately 96% and 99.78%, respectively, compared to the greedy model. This study demonstrates that malicious traffic can be effectively detected even in lightweight device environments. SFSM verified the possibility of detecting various attack traffic on lightweight devices.

KEYWORDS

Feature selection; lightweight device; machine learning; Internet of Things; malicious traffic

Nomenclature

SFSM	Suboptimal feature selection model
TP	True positive
TN	True negative

FP	False positive
FN	False negative
k	Number of exhaustive search target features
r	Number of features used when learning is performed
l	Total number of data points
n	Total number of features
s	Data sampling ratio

1 Introduction

The Internet of Things (IoT) has become an integral part of individuals' daily lives and is expected to have a significant impact on economic aspects in the coming years [1,2]. According to the 2020 Internet of Things report by Business Insider, the continuous growth of the IoT industry will be the driving force behind changes in all industries. The IoT market is projected to grow by over $2.4 trillion annually by 2027 [3]. It is expected that over 41 billion IoT devices will be connected by 2027.

However, existing lightweight IoT technologies have low computing costs and limited storage capacity, leading to security vulnerabilities [4]. Attack methods such as unauthorized access and battery-draining attacks pose significant threats to IoT technologies that operate with low computing power and minimal energy consumption [5,6]. These traditional issues have raised concerns about potential security threats as IoT technology becomes deeply integrated into daily life. Concerns have arisen regarding security threats that might harm personal information protection and the economy [7]. For instance, botnets like Mirai exploit vulnerabilities in IoT devices to take control of devices and rapidly spread, potentially causing adverse effects on the economy. There are threats to personal information, such as phishing, data theft, and spoofing [8]. Security threats leading to malicious traffic or compromising the confidentiality, integrity, and availability of systems through the distribution of malicious code remain a challenge in various fields, including IoT devices [9]. Typical detection methods for malicious attacks include static and dynamic analyses, signature-based detection, and behavior-based detection.

Static analysis involves analyzing and detecting malicious codes without executing them. It can discover potential security vulnerabilities in the code during the runtime environment, but it has limitations, such as difficulty in analyzing obfuscated programs and the challenge of considering all dynamic elements. Additionally, static analysis requires significant time and expertise, and it does not focus on proactive prevention. Dynamic analysis entails executing malicious codes directly and detecting abnormal behaviors through monitoring. It has the advantage of understanding the code's behavior without detailed code analysis, but it is resource-intensive because of the need for establishing an analysis environment. There is also a limitation that malicious codes can infect real computers during the analysis process [10].

The aforementioned conventional countermeasures are post-response methods because they analyze malicious behavior after it occurs, making it impossible to detect in advance. Signature-based detection involves creating databases of the behavioral patterns of malicious code and classifying them as malicious if they match. This is the most widely used method because it provides the best detection accuracy for previously identified malicious behavior. However, it is not effective in responding to transforming variants of malicious code or zero-day attacks because it can only address previously occurring patterns of malicious code [11]. However, conventional signature-based and behavior-based detection techniques are not suitable for IoT devices with limited resources, as they rely on databases with established detection performance. The behavior-based detection technique identifies

malicious behavior when different behavior patterns occur, designed to overcome the limitations of the aforementioned signature-based detection techniques. Unlike signature-based detection methods that rely on established databases, behavior-based detection methods can detect new attacks. However, the category of normal behavior is quite broad, leading to a relatively high false-positive rate compared to signature-based detection techniques. Traditional detection methods face a tradeoff issue between detection performance and energy efficiency. As a result, conventional signature-based and behavior-based detection techniques are not suitable for IoT devices with limited resources, as they rely on databases with established detection performance.

Machine learning-based detection technology is actively being researched and utilized in the detection of malicious activities to address the limitations of various existing detection techniques [12]. Among traditional detection technologies, signature-based detection, which is the most effective, identifies specific patterns of known malicious code. In contrast, machine learning-based detection enables the identification of new types of malware [13]. However, machine learning operates based on probability and statistics, leading to significant computational costs [14]. Therefore, it is not ideal for IoT devices with limited resources. Recent studies have focused on reducing the complexity of machine learning models or reducing the learning time to enable their use in lightweight devices [15]. However, the use of a high-quality machine learning model requires a vast dataset. It is difficult to store and learn extensive log datasets effectively to detect malicious behavior in IoT environments.

This paper proposes an efficient suboptimal feature selection model (SFSM) for detecting malicious behavior in resource-limited IoT environments. It addresses the tradeoff between detection performance and complexity, which conventional feature selection techniques do not consider. The SFSM conducts feature selection based on the importance of features and learning by scrutinizing feature sets. During the feature selection process, the number of features was subject to exhaustive search, and the data sampling ratio parameters were optimized in a suboptimal manner to maintain detection performance while reducing complexity. Previously, feature selection focused on improving learning performance was mainly considered, so if there were restrictions on using lightweight devices. However, SFSM improved learning performance while minimizing complexity by considering feature selection and exhaustive search as two steps. In addition, a model that can be practically used in lightweight devices was proposed by proposing a scheme to optimize parameters used in feature selection to reduce the complexity that increases in the exhaustive search process.

The primary contributions of this paper are as follows:

- We propose a suboptimal method that reduces complexity while maintaining accuracy by optimizing the parameters used in feature selection.
- The performance of the proposed SFSM was evaluated and compared with that of conventional methods using detection accuracy, latency, and complexity evaluation metrics, which are crucial in the IoT environment.
- By introducing a new lightweight detection model that enhances conventional feature selection techniques, we improved the tradeoff between abnormal behavior detection performance and resource complexity in IoT devices.

The remainder of this paper is organized as follows. Section 2 analyzes previous studies on the detection of abnormal behavior in IoT environments. Section 3 explains the proposed SFSM method. Section 4 details the experimental environment for evaluating the performance of the SFSM and analyzes the experimental results, and Section 5 concludes the paper.

2 Related Work

The field of machine learning is being studied in various fields. In addition to the preceding studies mentioned above, a recent study [16] has been conducted to propose a model that improves multi-graph learning technology used for data classification, clustering, and graph abnormality detection in machine learning technology or deep learning (DL)-based damage detection model that enables excellent feature extraction in a noisy environment [17]. In particular, with DenseSPH-VOLOv5, a method for reducing computational complexity while effectively detecting road damage in images by improving feature extraction even in various and noisy environments was proposed. Feature optimization has been actively studied until recently. IoT devices often use the basic authentication value as it is or use it without a password. Hackers can access these devices to execute large-scale attacks [18], and network security must be ensured as they may become vulnerable not only to the corresponding IoT devices but also to other devices connected to the network [19]. To enhance the security of IoT networks, it is crucial to detect and block malicious traffic. In recent years, machine learning technology has been widely adopted to identify and detect malicious attacks accurately in IoT network environments. Machine learning has high detection accuracy but is more complex than other detection methods [20]. In machine learning-based malicious behavior detection technology, duplicate data or features increase computational complexity and reduce detection performance [21]. Therefore, feature selection technology that enables IoT devices to extract and learn appropriate features for effective malicious traffic detection is essential [21,22]. Table 1 provides an overview of the characteristics, contributions, and limitations of previous technologies used for detecting malicious behaviors in IoT environments.

Table 1: Previous studies of malicious traffic detection methods

Category	Main focus	Ref.	Techniques	Contribution	Limitation
Non-machine learning detection model	Static/Dynamic analysis	[23]	Proposed extension of the static analyzer to IoT system	Introduced the expansion direction of the static analyzer through vulnerability analysis	Did not consider resources
		[24]	Used Static/Dynamic analysis to extract features	IoT malicious code spread trends can be analyzed	• Poor performance and less information available than machine learning • Did not consider resources

(Continued)

Table 1 (continued)

Category	Main focus	Ref.	Techniques	Contribution	Limitation
Machine learning detection model	Reducing complexity	[25]	Proposed Tabu Search-Random Forest feature selection	• Reduced computational complexity by reducing feature space and vector • Improved detection accuracy, even for attacks with a few data samples	Because the tabu list must be referenced every time, a reference delay occurs as the number of features increases
		[26]	Estimated the importance of features and select the top-k features	Improved learning performance and shortened learning time	• Insufficient basis for selecting the importance threshold • Did not consider resource
		[27]	• Feature selection based on correlations • Verified learning performance with the deep neural network (DNN) model in multiple scenarios	Improved learning time and calculated amount through feature selection	Some datasets result in poor performance

(Continued)

Table 1 (continued)

Category	Main focus	Ref.	Techniques	Contribution	Limitation
	Improving detection performance	[28]	Lightweight signature generation through multi-level clustering	• Lightweight detection method suitable for IoT environment • Improved malware detection rate through cluster merging	• Unable to detect new malware • Cluster merging process takes a long time
		[29]	• Improved attack detection rate over standard WOA • Reduced search space by adding an intersection operator	• Detected network attacks with high accuracy • Improvement of WOA's local optimization problem	High computational complexity and time delay because of GA operators
		[20]	• Effective feature set filtering with Bijective soft set • Improved accuracy using correlation attribute evaluation (CAE) and accuracy metrics together	• The first study using the CorrACC metric for botnet identification attacks • Performance measurement for various machine learning classifiers	Computational complexity is not considered when reducing the number of features
		[30]	Built a malicious Android application package dataset for accurate feature selection	• Shorter detection time compared to commercial antivirus scanners • High detection rate compared to previous literature	It did not consider limited resources

2.1 Non-Machine Learning Detection Model

Whereas conventional methods are primarily focused on detecting malicious behavior without using machine learning, recent research emphasizes network behavior analysis in large networks [23,24]. Ferrara et al. [23] proposed a method for conducting the Open Web Application Security Project (OWASP) top 10 vulnerability-based static analysis for IoT vulnerability analysis, as static analysis is effective in identifying security vulnerabilities in IoT devices. However, this involves inspecting the Application Programming Interfaces (APIs) of IoT devices post static analysis, requiring expertise and resources. Performance verification and resource considerations have not been adequately addressed. Liu et al. [24] used static and dynamic analyses to extract features such as function calls and traffic patterns from various malicious codes, comparing them to analyze the evolution of malicious codes. Although they analyzed the spread trend of IoT malicious code through large-scale network traffic analysis, they struggled to detect the diverse malicious traffic patterns seen in machine learning techniques. Given IoT's resource limitations, complex and resource-intensive detection methods are not ideal for lightweight protection mechanisms [28].

2.2 Machine Learning Detection Model

To improve the limitations of existing static and dynamic analysis methods, recent malicious traffic detection research has studied machine learning-based detection methods. Nazir et al. [25] attempted to improve the complexity and introduced a Tabu Search-Random Forest (TS-RF) feature selection technique aimed at reducing the dimensionality of high-dimensional data. Tabu search is a high-speed search method based on a metaheuristic optimization algorithm. It calculates the solution fit of neighboring features in the tabu list and moves it to the neighboring node with a higher fit. However, once a predefined number of iterations is reached, no further improvements are possible, or the objective function meets the required threshold, the search process stops. To prevent local optimization problems, recently explored solutions are stored in a tabu list, and each time a solution is moved, the tabu list prevents the next move. Their study [25] may be suitable for IoT devices as it reduces computational complexity by reducing feature space and vectors by over 60% and improving detection accuracy, even in attacks using small data samples. However, referencing the tabu list at every movement may lead to increased reference delays as the number of features increases. Abawajy et al. [26] focused on smartphones using the Android operating system among IoT devices and proposed a general automated classification framework for Android malware detection. The feature subset selection methods of this framework strengthen the learning algorithm's training and enhance classification performance using highly correlated features for classification. Although this method improved learning performance and reduced learning time, resource usage, a critical evaluation metric in the IoT environment, was not adequately considered. Alomari et al. [27] proposed a low-calculated high-performance malware detection system using deep learning. Deep learning is a good malicious code detection method, but there are numerous difficulties when considering the detection mechanism; therefore, we attempted to improve the performance of the training and evaluation process through correlation-based feature selection. Alomari et al. [27] also evaluated the performance of DL models proposed in various learning scenarios, including the presence or absence of feature selection. Although they implemented a lightweight model for IoT devices, reducing the number of features by over half on a Unix/Linux-based platform led to reduced accuracy and similar latency compared to a model without feature selection.

Alhanahnah et al. [28] focused on enhancing detection performance and creating lightweight signatures to overcome the resource constraints of IoT devices. They proposed a multistage clustering technique to cluster IoT malware samples into several families using n-gram string functions. A

combination of coarse- and fine-grained clustering reduced the clustering calculation cost and improved the malware detection rate through cluster merging. However, detecting malware with low similarity to conventional malware requires continuous IoT sample updates, and the cluster merging process is time-consuming. Vijayanand et al. [29] introduced the whale optimization algorithm (WOA), which improved attack detection compared to the standard WOA by adding intersection and mutation operators. The improved WOA designated a randomly selected feature from the initial dataset as the initial location and evaluated the fitness of each feature subset using a support vector machine (SVM). The intrusion detection system (IDS) integrated with the WOA proposed in their study detected network attacks with high accuracy by selecting relevant features and overcoming the local optimization problem of the WOA. Nonetheless, the use of genetic algorithm (GA) operators increased computational complexity and time delay, and the detection rate for classes with a few datasets was low. Shafiq et al. [20] filtered features using a bijective soft set to select an effective feature set for botnet IoT attacks in IoT networks, and accuracy was improved using CAE and accuracy metrics. They selected seven out of 39 features in the dataset and correctly identified traffic using a decision tree and a random forest model. However, it is not suitable for lightweight IoT devices because of the lack of computational complexity in reducing the number of features through feature selection. Mahindru et al. [30] built an effective Android malware detection model, highlighting that correct feature selection impacts malware detection performance. They collected Android application package files and constructed a dataset that extracts permissions and API calls, followed by feature selection and implementing the least-squares SVM (LS-SVM) model. The model reduced the time required for detection compared with conventional systems and showed a 3% higher detection rate compared with models proposed in the literature. However, the study did not thoroughly analyze resource considerations or limitations for IoT devices.

In summary, although static and dynamic analysis research in previous studies has not applied machine learning, it is not suitable for IoT environments with limited resources because of the need for specialized personnel and limited support. By contrast, recent studies that integrate machine learning aim to improve detection performance or computational complexity while considering the IoT's resource-constrained environment, as opposed to non-machine learning detection models. We evaluated prior research from two perspectives. Studies aimed at reducing computational complexity failed to consider resources or exhibited poor performance, while those enhancing detection performance had high computational complexity. Therefore, research addressing this tradeoff is crucial for efficiently detecting malicious network traffic in resource-constrained IoT environments.

3 Proposed Scheme

This section describes the feature search mechanism and the overall operation of the proposed SFSM.

3.1 Search Algorithm for Feature Selection

Feature selection aims to select a minimal representative feature subset from the entire feature set, eliminating redundant and irrelevant features from the dataset [31]. Determining the optimal subset of all features is challenging and is considered a combinatorial nondeterministic polynomial-time (NP)-hard problem [32]. This section outlines the search algorithms employed in the feature selection process.

The conventional exhaustive search method was widely adopted for finding an optimal feature set [25]. An exhaustive search examines all feature subsets to determine the optimal features precisely;

however, the search complexity is extremely high, reaching 2^N [25,33], where "N" represents the number of features. The greater the feature dimensions, the more exponentially complex the search becomes, leading to NP-hard problems [25,33]. The exhaustive search method is unsuitable for the IoT environment because of its high computational complexity, requiring significant computational time and memory to obtain an optimal solution [33]. To address this issue, search algorithms such as random search and meta-heuristics have emerged [25].

A random search continuously generates subsets, thus enhancing the quality of selected features through repeated selection [32]. It also mitigates local optimization problems by introducing randomness into the search procedure [25]. In each step, the next subset is randomly generated using information collected in the previous step [32]. However, in a worst-case scenario, exploring all solutions may be necessary, resulting in issues akin to those encountered in an exhaustive search [32].

The metaheuristic method is a general-purpose optimization technology capable of determining the optimal solution within a reasonable timeframe. Metaheuristic methods are divided into single-solution-based (S-metaheuristics) and population-based (P-metaheuristics) methods [32]. S-metaheuristics construct and iterate a single solution for improvement, while P-metaheuristics generate multiple solutions in each iteration and enhance them by selecting the best solution [25]. Metaheuristic techniques possess the ability to find a reasonable solution without exploring the entire search space [33]. These techniques probabilistically solve optimization problems using the randomness of a random search, engaging in optimization processes that randomly explore and utilize search spaces with specific probabilities, starting from a random solution [34]. Metaheuristic techniques are widely used because of their excellent performance capability, but increasing dataset dimensions can impact their performance [34]. Additionally, a reasonable solution does not guarantee an optimal solution [33].

Conventional search methods are ill-suited for lightweight IoT environments because of their high computational complexities. Moreover, consistently achieving good performance is not guaranteed, as the derived feature set is not necessarily optimal. Therefore, this study suggests the use of a grid search method, which is a lightweight search approach for feature selection while achieving an optimal solution. A grid search entails selecting a combination of feature sets that demonstrate optimal performance by exploring candidate values from the entire set. This method addresses the complexity problem associated with efficient yet highly complex exhaustive search algorithms and also the performance issue of random search, which is efficient but lacks performance optimization.

3.2 Suboptimal Feature Selection Model

In this section, we describe the proposed SFSM. Fig. 1 illustrates its overall operational structure.

The SFSM operates in four phases: data preprocessing, input parameter optimization, feature selection, and attack classification. First, data preprocessing is conducted on the dataset to derive the final features for selection. Input parameter optimization is then performed on the preprocessed dataset. In the initial step of feature selection, the top-k features are selected from the entire feature set using permutation importance [35]. Permutation importance ranks features based on their impact on the classification performance of the dataset. Fig. 2 shows a schematic diagram for selecting top-k features using permutation importance.

As shown in Fig. 2, the error of the original model is first measured, then the feature matrix X^{Perm} is generated by mixing the feature j from the data X, and the error is measured based on the predicted value of the mixed data [36]. Based on the difference between the error value of the predicted and original models, the FI value is calculated by calculating the permutation feature importance

[36]. Lastly, the feature is selected based on its importance by arranging the features in reverse order according to the FI value [36].

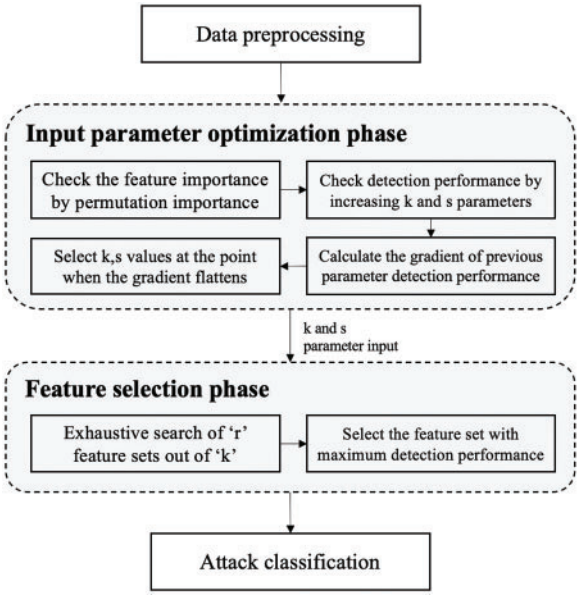


Figure 1: Flowchart of SFSM

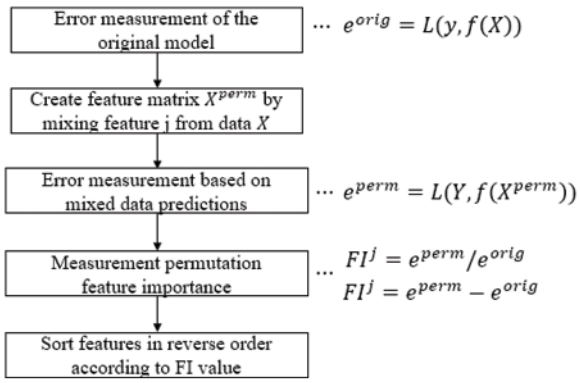


Figure 2: Flowchart of feature selection based on permutation importance

The SFSM employs data sampling to reduce computational complexity while ensuring that performance is not compromised. During this process, parameter optimization is performed to determine the optimal values of k and s. Here, k represents the number of features considered in the exhaustive search, and s represents the data sampling ratio. Initially, k and s values are inputted, and the detection performance is evaluated by incrementally adjusting each parameter. The detection performance results for each iteration are used to calculate the gradient of the rate of increase in performance compared to the previous parameter setting. When the gradient flattens, further parameter adjustments do not have a significant impact on the performance. Therefore, the k and s parameters are selected and feature selection proceeds based on these chosen values. Data are sampled according to the s value, which is the data sampling ratio, and an exhaustive search is performed

to find the optimal r feature set from the k-selected features. This process evaluates the detection accuracy performance for all feature sets nCr, and the feature set with the best accuracy is chosen for subsequent learning. The feature selection we propose significantly reduces computational complexity while showing similar performance to general thorough search methods. ML can obtain accurate performance results with an ML classifier optimized by feature selection. We can improve resources efficiently while increasing ML performance by reducing the computational complexity of feature selection.

Fig. 3 illustrates the system configuration of SFSM.

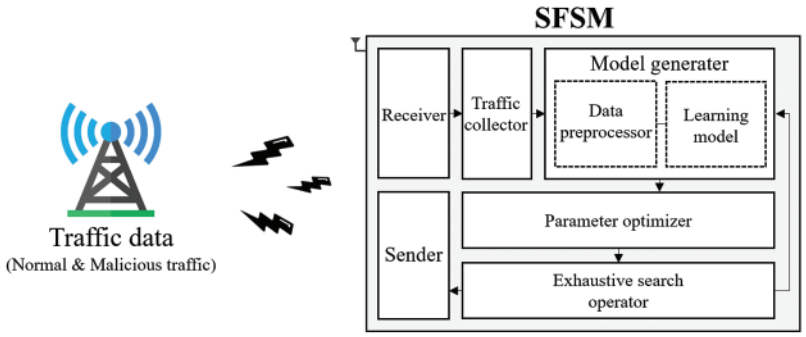


Figure 3: System architecture of SFSM

As depicted in Fig. 3, the SFSM receives mixed traffic comprising both normal and malicious traffic from the receiver and aggregates it using a traffic collector. Subsequently, the collected traffic data are preprocessed using the data preprocessor to generate the learning model and derive feature importance. Next, the optimized parameters are selected using the parameter optimizer. The k and s parameters determined using the parameter optimizer are applied to the dataset for an exhaustive search. The goal is to identify the optimal feature set that exhibits the highest detection performance. This selected feature set serves as input for the learning model responsible for attack classification.

Algorithm 1 provides the pseudo-code of SFSM, which operates in five stages: generate the learning model phase, calculate the feature importance rank phase, optimize parameters that affect the accuracy performance phase, exhaustive search of feature set showing optimal performance phase, and learning with the optimal feature set phase. In Step 1, a learning model to derive feature importance is generated, and in Step 2, feature importance is calculated through the permutation importance library, and the importance ranking is listed. In Step 3, based on the derived feature importance rank, the top features are added individually, points with little change in accuracy performance are derived, and the number of exhaustive search target features (k) and data sampling rate (s) parameters are optimized. In Step 4, the composition of the optimal feature set is derived from an exhaustive search. In Step 5, the process operates by learning with the feature set that showed maximum accuracy.

Algorithm 1: Pseudo-code for SFSM

Input: Malicious traffic dataset, gradient
Output: Detection performance
Step 1: Generate the learning model
data_preprocessed = data_preprocess (dataset)
model = decision_tree (data_preprocessed) ▷ Generate the machine learning model

(Continued)

Algorithm 1 (continued)

```
Step 2: Calculate the feature importance rank
  feature_importance = PermutationImportance (model)        ▷ List the features in order of importance
Step 3: Optimizing parameters that affect accuracy performance
for i in range (len(feature_importance))
    for j in range(i+1):
        new_dataset.append (feature_importance[j])
        acc = model.fit(new_dataset)                        ▷ Input the feature to model in order of feature importance
    if current_acc-before_acc < gradient                    ▷If the gradient of accuracy is below to certain level, loop stop
      n_k = c_k                                             ▷ Determine k (the number of exhaustive search target features) value
      n_s = c_s                                             ▷ Determine s (data sampling ratio) value
      break
Step 4: Exhaustive search of feature set showing optimal performance
  sampled_dataset = data_preprocessed.sample (n = n_s)      ▷ Apply s value to dataset
  for all n_k do
      ES_acc.append (Exhaustive search())                   ▷ Derive the accuracy of each feature set by exhaustive search
      total_set = max (ES_acc)                              ▷ Derive the final set from the feature set with maximum accuracy
Step 5: Learning with optimal feature set
    model.fit (total_set)
```

4 Performance Evaluation

4.1 Experimental Environment

In this section, we describe the experimental environment used to demonstrate the performance of the proposed SFSM. This experiment was conducted using Python3 on an Intel(R) Core(TM) i9-10850K 3.60 GHz CPU with 32.0 GB of RAM. To create an environment where malicious traffic attacks occur, we utilized the UNSW-NB15 dataset [37] and trained it using a decision tree, a machine learning model. All hyper-parameters of the decision tree were set to default values. Max_depth, the maximum depth of the tree, was set to none; min_samples_split, the minimum number of sample data to split a node, was set to 2; and max_features, the maximum number of features to use, was set to none and was set to be divided using all features. In this experiment, our goal was to classify one normal label and two attack labels: benign, exploits, and generic, all of which have sufficient data. The benign label represents normal traffic, exploits denote attacks using vulnerabilities, and the generic label signifies attacks that infiltrate by bypassing the block cipher. Table 2 lists the number of data points for each label and the number of data points after preprocessing.

Table 2: Number of data points by label

Label	Number of data points	
	Before preprocessing	After preprocessing
Benign	37,000	10,000
Exploits	11,132	10,000
Generic	18,871	10,000

The number of data points was balanced at 10,000 for each label. In addition, to preprocess the dataset, we removed ID and time information that were unrelated to the classification among the features. NaN values were preprocessed with zero padding, and string data underwent one-hot encoding.

In this study, we implemented and evaluated greedy and random models as comparative models for the SFSM. The two comparison models were evaluated in the same environment as SFSM and were selected as models applying representative search algorithms. The greedy model [25] performs an exhaustive search targeting all the features. It exhibits the most efficient detection performance as it tests all feature combinations and selects the set with the best performance. However, this model has an extremely large search complexity, 2^N. The random model [22], which performs a random search targeting all features, was implemented by randomly selecting r features from the entire feature set.

We used detection accuracy, latency, and complexity as evaluation metrics to assess the performance of SFSM. Detection accuracy is an evaluation metric that indicates whether malicious traffic has been accurately classified, and it was calculated using Eq. (1).

$$Detection\ accuracy = \frac{Number\ of\ correctely\ predicted\ malicious\ traffic}{Number\ of\ total\ traffic} \tag{1}$$

Latency measures the total time from the dataset preprocessing to the learning process.

Complexity was determined by defining an equation that calculates the amount of computation based on the data size. Complexity considered the complexity consumed in the feature importance calculation and the complexity consumed in the exhaustive search process, as shown in Eq. (2) [38].

$$Complexity = kCr \times l + n \times l \times s \tag{2}$$

where k, r, l, n, and s are the number of exhaustive search target features, the number of features used when learning is performed, the total number of data points, the total number of features, and the data sampling ratio, respectively.

In this experiment, we evaluated the gradient for selecting the k and s parameters of the SFSM was set to 0.005. Eq. (3) shows the objective function of SFSM.

$$Efficiency = \max_{k \in K, s \in S} \left(\frac{Detection\ accuracy\ (k, s)}{Complexity\ (k, s)} \right) \tag{3}$$

K and S in Eq. (3) are sets with a finite number of ranges of {1,2,3, . . . , i} and {1, 2, 3, . . . , j}, respectively, when i and j are integer values. SFSM aims to maximize accuracy performance and minimize complexity. As the k and s parameters increase, accuracy and complexity both increase, and as the k and s parameters decrease, accuracy and complexity both decrease. Therefore, the optimal

balance of the two evaluation metrics must be found to determine the optimal k and s values. In other words, the k and s values that maximize the objective function of Eq. (3) must be derived.

4.2 Evaluation Results and Analysis

In this section, we compare the performance of the proposed SFSM with that of the comparative greedy and random models in terms of detection accuracy, latency, and complexity to demonstrate its feasibility. We confirmed the optimization of the k and s parameters by increasing the number of features, k, and data sampling ratios. The detection performance is demonstrated as the number of features is increased for data sampling ratios of 0.2, 0.5, and 0.8. Fig. 4 shows the workflow of parameter optimization.

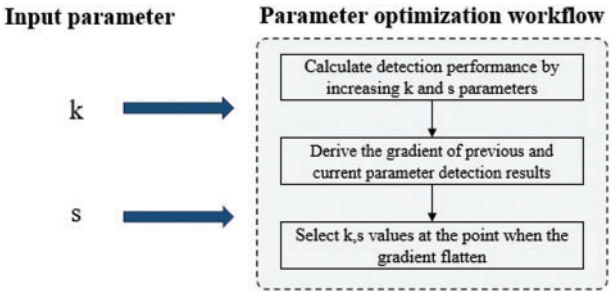
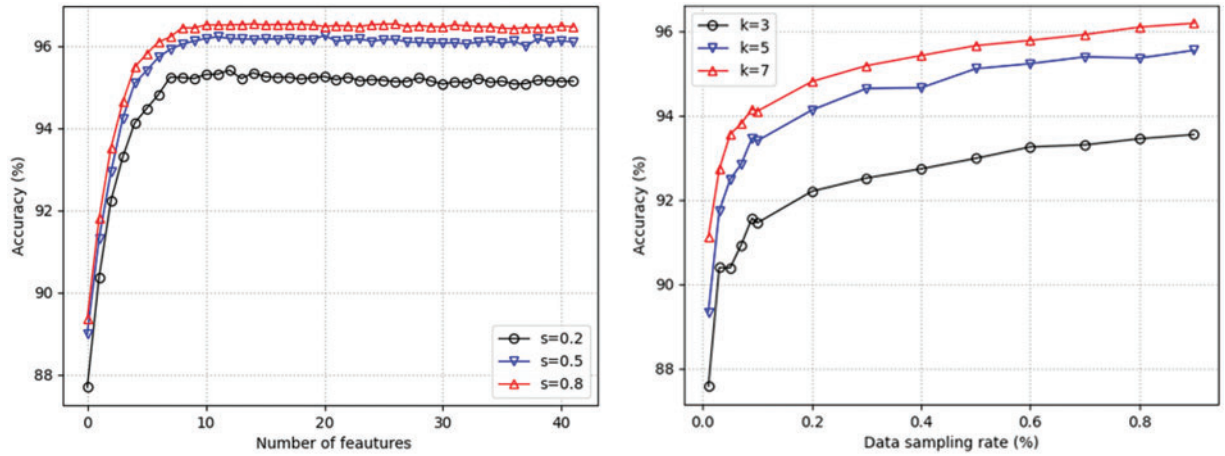


Figure 4: Workflow of input parameter optimization

To select the optimal input parameter, as shown in Fig. 4, SFSM measures the detection accuracy of the model while increasing the k and s parameters. After that, the gradient of the accuracy result of the parameter before increasing and the current parameter is calculated. Then, k and s values are selected when the gradient becomes flattened. Fig. 5 illustrates the evaluation of detection accuracy performance based on the number of features and data sampling ratio.



(a) (b)

Figure 5: Detection accuracy performance by parameters: (a) Number of features; (b) Data sampling ratio

As depicted in Fig. 5, an increase in the number of features and data sampling ratio leads to a plateau in the rate of increase in detection performance at a specific point. Moreover, when the number of features is increased, the rate of increase in detection performance flattens at a point where all three values of k are similar. Similarly, as the data sampling ratio increases, the growth rate of the detection performance slope plateaus at the point where all three values of k are identical. This study determined the number of features and the data sampling ratio based on a slope value of less than 0.005.

Thus, the proposed SFSM optimizes the selection of k and s values when the slope of the detection performance growth rate flattens as the k and s values increase. Fig. 6 presents a comparison of the performances of the SFSM and conventional models.

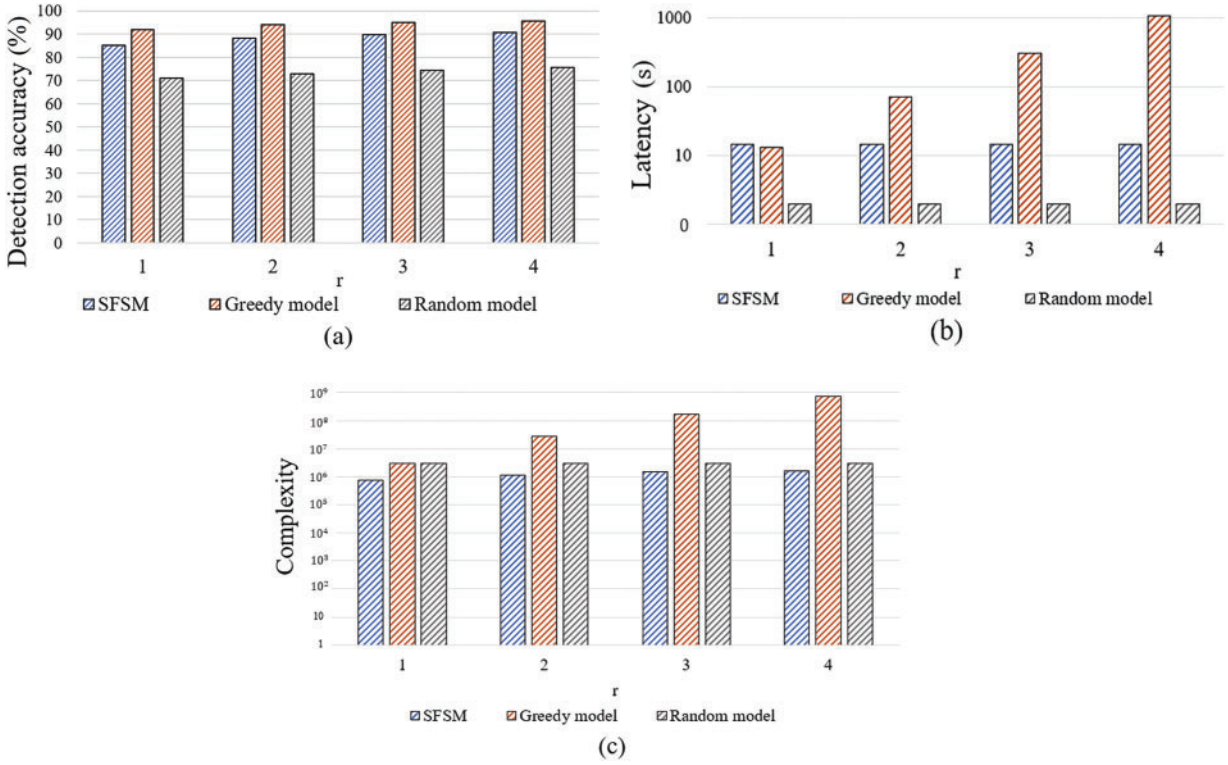


Figure 6: Performance and complexity evaluation results of SFSM compared to conventional models: (a) Detection accuracy; (b) Latency; (c) Complexity

According to Fig. 6, the results indicate that the greedy, SFSM, and random models achieved good detection accuracy in that order. Because of the exhaustive search nature of the greedy model across all features, it achieved the highest accuracy in the range of 90%–95%. Conversely, the random model, which selects features randomly without considering their importance, exhibited the least efficiency in terms of accuracy. Notably, the greedy model showed the lowest efficiency in terms of latency and complexity, two crucial evaluation metrics for IoT environments. Specifically, the latency of the greedy model was approximately 69 times higher than that of the SFSM, while the complexity was approximately 313 times more inefficient than that of SFSM. As a result of calculating the efficiency of Eq. (3) by normalizing complexity according to Fig. 7, the results were high in the order of SFSM, Random model, and Greedy model. SFSM showed the most effective efficiency compared to the greedy model, in which complexity is inefficient, and the random model, in which detection

performance is inefficient, by minimizing the trade-off of the evaluation metrics in terms of detection accuracy and complexity.

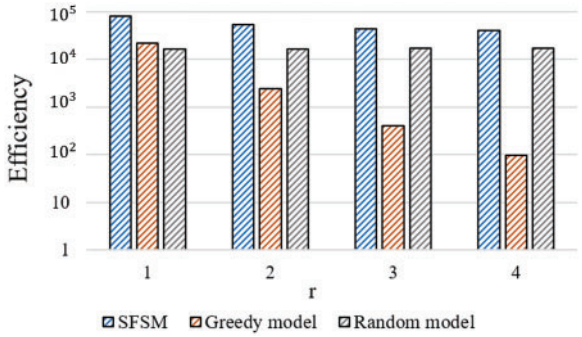


Figure 7: Efficiency of SFSM compared to conventional models

These findings demonstrate SFSM can more efficiently minimize the tradeoff between these two metrics compared to the greedy model. It significantly reduces latency and complexity, essential considerations in the IoT environment, while incurring only a marginal loss in detection performance.

5 Conclusion

Machine learning technology is widely used to identify and detect malicious traffic in IoT network environments. However, irrelevant and overlapping features result in an increase in computational complexity and degrade machine learning performance during the learning process. Therefore, to address the complexity problem and enhance the detection performance of machine learning, a feature selection approach that eliminates less relevant or overlapping features is crucial. In this study, we propose an SFSM that reduces the complexity of the feature selection process for effective machine learning in IoT environments. During this process, the SFSM determines the number of features to be considered and identifies suboptimal sampling rate parameters. It utilizes an optimized feature set for learning through an exhaustive search. The effectiveness of the SFSM in detecting malicious traffic in resource-constrained IoT environments has been demonstrated. However, SFSM has the limitation that the gradient value, which is the standard for selecting parameters, operates at a fixed value, and because it is a suboptimal model, it does not determine the most optimal value. Therefore, in future studies, we plan to study the gradient value optimization method and other methods to find optimal parameters to derive comparison results with the baseline model. The performance of SFSM will be verified with a more diverse range of malicious traffic datasets and machine learning models and compared with the latest advanced conventional model, considering its quality and reliability. Furthermore, we will optimize its capabilities by varying the gradients for selecting the k and s parameters. In addition, the performance of SFSM will be verified by expanding more diverse attack scenarios and malicious traffic types.

Acknowledgement: This paper is a supplementary and extended version of the paper presented at the 7th International Symposium on Mobile Internet Security (MobiSec'23) Conference.

Funding Statement: This work was partly supported by the Korea Institute for Advancement of Technology (KIAT) Grant funded by the Korean Government (MOTIE) (P0008703, The Competency Development Program for Industry Specialists), and MSIT under the ICAN (ICT Challenge and

Advanced Network of HRD) Program (No. IITP-2022-RS-2022-00156310) supervised by the Institute of Information & Communication Technology Planning and Evaluation (IITP).

Author Contributions: The authors confirm contribution to the paper as follows: So-Eun Jeon: Conceptualization, methodology, software, validation, visualization, writing–original draft. Ye-Sol Oh: Resources, formal analysis, writing–reviewing and editing. Yeon-Ji Lee: Resources, validation, writing–review and editing. Il-Gu Lee: Conceptualization, validation, writing–review and editing, supervision, project administration, funding acquisition.

Availability of Data and Materials: The data that support the findings of this study are available from the first and corresponding authors upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Mohammed, M. A., Abdul Wahab, H. B. (2024). Enhancing IoT data security with lightweight blockchain and okamoto uchiyama homomorphic encryption. *Computer Modeling in Engineering & Sciences, 138(2),* 1731–1748. https://doi.org/10.32604/cmes.2023.030528
2. Chen, C., Liu, S., Chaudhry, S., Chen, Y., Khan, M. (2022). A lightweight and robust user authentication protocol with user anonymity for IoT-based healthcare. *Computer Modeling in Engineering & Sciences, 131(1),* 307–329. https://doi.org/10.32604/cmes.2022.018749
3. Insider Inc. (2020). The Internet of Things 2020: Here's what over 400 IoT decision-makers say about the future of enterprise connectivity and how IoT companies can use it to grow revenue. https://www.businessinsider.com/internet-of-things-report (accessed on 12/09/2023).
4. Yun, S. W., Park, N. E., Lee, I. G. (2023). Wake-up security: Effective security improvement mechanism for low power Internet of Things. *Intelligent Automation & Soft Computing, 37(3),* 2897–2917. https://doi.org/10.32604/iasc.2023.039940
5. Mizna, K., Sufian, H., Abdul, Q., Syed, A. S., Dirk, D. (2023). Towards SDN-based smart contract solution for IoT access control. *Computer Communications, 198,* 1–31. https://doi.org/10.1016/j.comcom.2022.11.007
6. Il-Gu, L., Kyungmin, G., Jung-Hoon, L. (2020). Battery draining attack and defense against power saving wireless LAN devices. *Sensors, 20(7),* 2043. https://doi.org/10.3390/s20072043
7. Bagaa, M., Taleb, T., Bernabe, J. B., Skarmeta, A. (2020). A machine learning security framework for IoT systems. *IEEE Access, 8,* 114066–114077. https://doi.org/10.1109/Access.6287639
8. Malak, A., Amal, A. A., Rami, M. A. M., Fahd, A., Menna, A. et al. (2023). Machine learning-based detection for unauthorized access to IoT devices. *Journal of Sensors and Actuator Networks, 12(2),* 27. https://doi.org/10.3390/jsan12020027
9. Mohanta, B. K., Jena, D., Satapathy, U., Patnaik, S. (2020). Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet of Things, 11,* 100227. https://doi.org/10.1016/j.iot.2020.100227
10. Ngo, Q. D., Nguyen, H. T., Le, V. H., Nguyen, D. H. (2020). A survey of IoT malware and detection methods based on static features. *ICT Express, 6,* 280–286. https://doi.org/10.1016/j.icte.2020.04.005
11. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity, 2,* 20. https://doi.org/10.1186/s42400-019-0038-7

12. Al-amri, R., Murugesan, R. K., Man, M., Abdulateef, A. F., Al-Sharaf, M. A. et al. (2021). A review of machine learning and deep learning techniques for anomaly detection in IoT data. *Applied Sciences, 11,* 5320. https://doi.org/10.3390/app11125320
13. Liu, K. J., Xu, S. W., Xu, G. A., Zhang, M., Sun, D. W. et al. (2020). A review of android malware detection approaches based on machine learning. *IEEE Access, 8,* 124579– 124607. https://doi.org/10.1109/Access.6287639
14. Abdur, R. K., Amanullah, Y., Syed, M. U., Saddam, H., Shehzad, K. et al. (2022). Exploring lightweight deep learning solution for malware detection in iot constraint environment. *Electronics, 11(24),* 4147. https://doi.org/10.3390/electronics11244147
15. Mendonça, R. V., Silva, J. C., Rosa, R. L., Saadi, M., Rodriguez, D. Z. et al. (2022). A lightweight intelligent intrusion detection system for Industrial Internet of Things using deep learning algorithms. *Expert Systems, 39(5),* e12917. https://doi.org/10.1111/exsy.v39.5
16. Jiang, B., Chen, S., Wang, B. B., Luo, B. (2022). MGLNN: Semi-supervised learning via multiple graph cooperative learning neural networks. *Neural Networks, 153,* 204–214. https://doi.org/10.1016/j.neunet.2022.05.024
17. Arunabha, M. R., Jayabrata, B. (2023). DenseSPH-YOLOv5: An automated damage detection model based on DenseNet and Swin-Transformer prediction head-enabled YOLOv5 with attention mechanism. *Advanced Engineering Informatics, 56,* 102007. https://doi.org/10.1016/j.aei.2023.102007
18. Ahmad, R., Alsmadi, I. (2021). Machine learning approaches to IoT security: A systematic literature review. *Internet of Things, 14,* 100365. https://doi.org/10.1016/j.iot.2021.100365
19. Mishra, S., Albarakati, A., Sharma, S. K. (2022). Cyber threat intelligence for IoT using machine learning. *Processes, 10(12),* 2673. https://doi.org/10.3390/pr10122673
20. Shafiq, M., Tian, Z., Bashir, A. K., Du, X., Guizani, M. (2020). IoT malicious traffic identification using wrapper-based feature selection mechanisms. *Computers & Security, 94,* 101863. https://doi.org/10.1016/j.cose.2020.101863
21. Sun, G., Li, J., Dai, J., Song, Z., Lang, F. (2018). Feature selection for IoT based on maximal information coefficient. *Future Generation Computer Systems, 89,* 606–616. https://doi.org/10.1016/j.future.2018.05.060
22. Naheed, N., Shaheen, M., Khan, S. A., Alawairdhi, M., Khan, M. A. (2020). Importance of features selection, attributes selection, challenges and future directions for medical imaging data: A review. *Computer Modeling in Engineering & Sciences, 125(1),* 315–344. https://doi.org/10.32604/cmes.2020.011380
23. Ferrara, P., Mandal, A. K., Cortesi, A., Spoto, F. (2020). Static analysis for discovering IoT vulnerabilities. *International Journal on Software Tools for Technology Transfer, 23,* 71–88.
24. Liu, Z., Zhang, L., Ni, Q., Chen, J., Wang, R. et al. (2018). An integrated architecture for IoT malware analysis and detection. IoT as a Service. *4th EAI International Conference (IoTaaS 2018)*, pp. 127–137. Xi'an, China.
25. Nazir, A., Khan, R. A. (2021). A novel combinatorial optimization based feature selection method for network intrusion detection. *Computers & Security, 102,* 102164. https://doi.org/10.1016/j.cose.2020.102164
26. Abawajy, J., Darem, A., Alhashmi, A. A. (2021). Feature subset selection for malware detection in smart IoT platforms. *Sensors, 21(4),* 1374. https://doi.org/10.3390/s21041374
27. Alomari, E. S., Nuiaa, R. R., Alyasseri, Z. A. A., Mohammed, H. J., Sani, N. S. et al. (2023). Malware detection using deep learning and correlation-based feature selection. *Symmetry, 15(1),* 123. https://doi.org/10.3390/sym15010123
28. Alhanahnah, M., Lin, Q., Yan, Q., Zhang, N., Chen, Z. (2018). Efficient signature generation for classifying cross-architecture IoT malware. *2018 IEEE Conference on Communications and Network Security (CNS)*, Beijing, China.

29. Vijayanand, R., Devaraj, D. (2020). A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network. *IEEE Access, 8,* 56847–56854. https://doi.org/10.1109/Access.6287639
30. Mahindru, A., Sangal, A. L. (2021). FSDroid:- A feature selection technique to detect malware from Android using Machine Learning Techniques. *Multimedia Tools and Applications, 80,* 13271–13323. https://doi.org/10.1007/s11042-020-10367-w
31. Ma, T., Zhou, H., Jia, D., Al-Dhelaan, A., Al-Dhelaan, M. et al. (2019). Feature selection with a local search strategy based on the forest optimization algorithm. *Computer Modeling in Engineering & Sciences, 121(2),* 569–592. https://doi.org/10.32604/cmes.2019.07758
32. Taradeh, M., Mafarja, M., Heidari, A. A., Faris, H., Aljarah, I. et al. (2019). An evolutionary gravitational search-based feature selection. *Information Sciences, 497,* 219–239. https://doi.org/10.1016/j.ins.2019.05.038
33. Al-Tashi, Q., Kadir, S. J. A., Rais, H. M., Mirjalili, S., Alhussian, H. (2019). Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access, 7,* 39496–39508. https://doi.org/10.1109/Access.6287639
34. Arora, S., Singh, H., Sharma, M., Sharma, S., Anand, P. (2019). A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection. *IEEE Access, 7,* 26343–26361. https://doi.org/10.1109/ACCESS.2019.2897325
35. Altmann, A., Toloşi, L., Sander, O., Lengauer, T. (2010). Permutation importance: A corrected feature importance measure. *Bioinformatics, 26,* 1340–1347. https://doi.org/10.1093/bioinformatics/btq134
36. Fisher, A., Rudin, C., Dominici, F. (2018). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research, 20(177),* 1–81.
37. Moustafa, N., Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, pp. 1–6.
38. Jeon, S. E., Oh, Y. S., Kil, Y. S., Lee, Y. J., Lee, I. G. (2023). Two-step feature selection technique for secure and lightweight Internet of Things. *The 32nd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6. Hawaii, USA.