Hindawi
Security and Communication Networks
Volume 2022, Article ID 3990386, 20 pages
https://doi.org/10.1155/2022/3990386

Research Article

Bridging the Last-Mile Gap in Network Security via Generating Intrusion-Specific Detection Patterns through Machine Learning

Xibin Sun,[1,2] **Du Zhang**,[1] **Haiou Qin**,[1] **and Jiahua Tang**[1]

[1]*Faculty of Information Technology, Macau University of Science and Technology, Macau, China*
[2]*Guangdong Polytechnic of Science and Technology, Zhuhai, China*

Correspondence should be addressed to Xibin Sun; jacky5555@qq.com

Received 9 September 2021; Accepted 22 December 2021; Published 12 February 2022

Academic Editor: Kuo-Hui Yeh

With successful machine learning applications in many fields, researchers tried to introduce machine learning into intrusion detection systems for building classification models. Although experimental results showed that these classification models could produce higher accuracy in predicting network attacks on the offline datasets, compared with the operational intrusion detection systems, machine learning is rarely deployed in the real intrusion detection environment. This is what we call the last mile problem with the machine learning approach to network intrusion detection, the discrepancy between the strength and requirements of machine learning and network operational semantics. In this paper, we aim to bridge the aforementioned gap. In particular, an LCC-RF-RFEX feature selection approach is proposed to select optimal features of the specific type of attacks from dataset, and then, an intrusion-specific approach is introduced to convert them into detection patterns that can be used by the nonmachine-learning detector for the corresponding specific attack detection in the real-world network environment. To substantiate our approach, we take Snort, KDDCup'99 dataset, and Dos attacks as the experimental subjects to demonstrate how to close the last-mile gap. For the specific type of Dos attacks in the KDDCup'99 dataset, we use the LCC-RF-RFEX method to select optimal feature subset and utilize our intrusion-specific approach to generate new rules in Snort by using them. Comparing performance differences between the existing Snort rule set and our augmented Snort rule set with regard to Dos attacks, the experimental results showed that our approach expanded Snort's detection capability of Dos attacks, on average, reduced up to 25.28% false-positive alerts for Teardrop attacks and Synflood attacks, and decreased up to 98.87% excessive alerts for Mail bomb attacks.

1. Introduction

Intrusion detection systems (IDS) are part of the network security infrastructure designed to provide timely detection of various malicious attacks and take proactive responses to safeguard a network system. Two main approaches exist for the IDS detection process: misuse/signature-based detection [1] and anomaly based detection [2]. Anomaly based detection is based on establishing profiles of normal behaviors and regards any activity deviating from the profiles as abnormal. The anomaly detection method can be used to recognize unknown attacks. However, it often suffers from high false alarms. Signature-based IDS is based on pre-built patterns of malicious activities and regards anything that does not match the pre-built patterns as normal. The signature-based IDS can usually obtain high accuracy in recognizing known attacks of which signatures exist in its predefined pattern repository. Furthermore, it also has the advantage of easy and fast deployment to the real-world network environment. Therefore, the signature-based IDS has an absolute advantage in quantity over anomaly based IDS among the actual operational IDS products [3]. However, the signature-based IDS suffers from detecting unknown attacks or new attacks. The quality and reliability of the signature-based IDS detection results rely on the frequent updating of the signatures repository.

As described above, we can find the challenges in both the traditional signature-based IDS and anomaly based IDS. Therefore, many researchers introduce machine learning (ML) and data mining (DM) technologies to build

classification models for intrusion detection. Reference [4] surveyed the intelligent techniques for feature selection and building classification models for network intrusion detection by using the selected features, such as Decision trees, Neural networks, Fuzzy sets, and so on. References [5–18] used supervised machine learning techniques to build classification models, such as References [6–10] mainly used to support vector machine (SVM) methods, [8–11] employed decision tree (DT) methods, and [15–18] presented deep learning approaches for classifying malicious traffic. In addition, References [19–25] proposed the anomaly based network IDS (ANIDS), which attempts to detect attacks by using semi-supervised or unsupervised methods. Figure 1 shows the general process of building classification models by using ML/DM.

So far, there are extensive academic researches about ML/DM for intrusion detection, and the proposed classification models by using ML/DM can usually get better prediction results of network attacks on the test dataset. However, as described in Reference [3], compared with the operational intrusion detection systems, the ML/DM method is rarely employed in the real-world intrusion detection environment. This is what we call the last mile problem with the machine learning approach to network intrusion detection, the discrepancy between machine learning and network operational semantics in Figure 1.

There are two possible reasons for the last mile problem. First, the ML/DM methods are best suited for offline network environments rather than real-time network environments. For example, the process of building classification models by using ML/DM is a usual offline process, especially for the features based on flow(s)-level granularity in the training dataset. Furthermore, for the input network traffics, before they are fed to the training models for the prediction, they must be converted to the specific records according to the data format of the training dataset.

Second, for classification models obtained through ML/DM for signature-based or anomaly based detection, it is hard to convert them into appropriate operations that can be used by the actual no-machine-learning IDS products for the specific attacks detection in a real-world network environment. That is to say, there is a semantic gap between the ML/DM and network operations. In general, ML methods excel much better at finding current network behavior that is similar to something previously seen in the training dataset, and less good at discovering the semantics of network activities. For example, for signature-based detection by using supervised machine learning (e.g., Support Vector Machine and Deep Learning), although the classification models can classify malicious traffic to a certain extent, they look more like "black boxes," which do not align well with the network semantics. The classification models cannot be converted into feasible operations which can be used by the no-machine-learning detector in the real-world network environment. In addition, the normal behavior profiles by using semi-supervised or unsupervised methods can be used to detect abnormal activities. However, these detected abnormal activities only indicate that they deviate from the normal profiles, and it does not express they are the real attacks. Consequently, compared to ML for signature-based detection, the anomaly based detection by using ML is harder to relate its results into the network operational semantics for attack detection.

At present, many pieces of literature usually focus on improving the performance of the classification models by modifying machine learning algorithms rather than solving the "last mile problem." Therefore, compared to the related literature, this paper mainly focuses on bridging the gap between the strength and requirements of ML/DM and network operational semantics. For most of the operational network IDS, they are usually based on signature-based detection (e.g., Snort [26] and *Suricata* [27]) in the real-world network environment. So our intrusion-specific method for closing the last-mile gap aims to convert the features on which the classification models operate into the corresponding building blocks of the signature-based IDS, and the signature-based IDS can use them to define the detection patterns for specific attacks. The contribution of this paper involves the following aspects:

(1) This paper puts forward the "last mile problem" between the ML/DM and the operational network IDS, and then provided an intrusion-specific approach to bridge the gap between the strength and requirements of ML/DM and network operational semantics.

(2) A hybrid feature selection method (LCC-RF-RFEX) is proposed to extract the key features from the network dataset. Our approach combines the advantages of high efficiency of the filter feature selection method and the ability of the wrapper feature selection method, and absorbs the greedy policy when selecting the feature subset; only the feature subset which can make the classification model get the highest values of accuracy rate are remained. Experimental results showed that our approach can get better performance by comparing other filter and wrap feature selection methods.

(3) Snort's existing detection mechanisms can usually detect well the packet-level attacks and are not good at finding the complex attacks, such as flow(s)-level or connection(s)-level attacks, which span single flow or multiple flows. Therefore, for the flow(s)-level attacks, we added the new rule keywords for detecting them in the Snort.

(4) To substantiate our approach to bridge the gap between the strength and requirements of ML/DM and network operational semantics, we took Snort as our underlying intrusion detection system, KDDCup'99 dataset as the data source, and Dos attacks as the attack type of our study, and the experimental results proved the effectiveness of our intrusion-specific approach for closing the last-mile gap. By mapping the aggregated features into the new rule keywords based on flow(s)-level granularity, the newly generated rules not only expanded the abilities of Snort for detecting Dos attacks but also reduced its false-positive rate and numbers of excessive alarms for the same Dos attack.

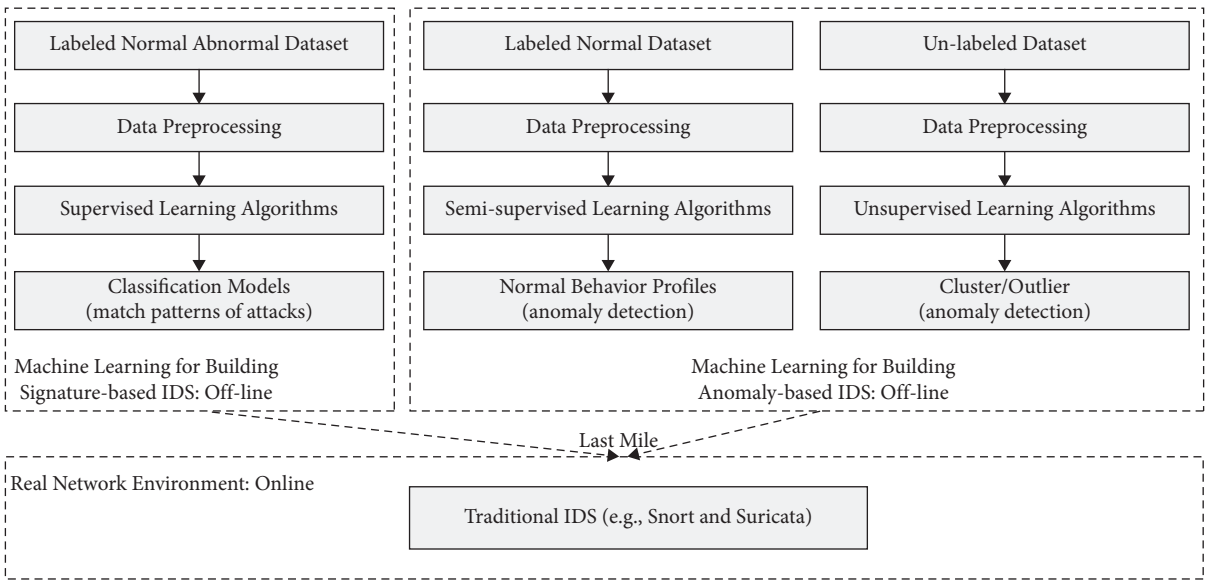

Figure 1: The last mile problem of machine learning for network intrusion detection.

The rest of this paper is organized as follows: Section 2 discusses the related works on improving the performance of the operational network IDS by using ML/DM approaches. Section 3 presents our key features extraction method in detail. Section 4 describes the process of creating refined attack signatures based on the newly generated keywords. Section 5 analyses the experimental results of our proposed approach and compares its performance with those of related works. Finally, Section 6 concludes the paper with comments on future work.

2. Related Works

2.1. Feature Selection by Using ML/DM. The feature selection can reduce the dimension of the dataset by eliminating redundant and irrelevant features, which makes the classification task based on learning algorithms more effective and accurate. The methods for feature selection are classified into three main categories: filter [28–37], wrapper [38–44], and hybrid [45–47] approaches. Filter methods use heuristics based on general characteristics of the data rather than learning algorithms to evaluate the merit of feature subsets, so filter methods are generally much faster than wrapper methods, and more fit for handling high dimension data. Wrapper strategies for feature selection use an induction algorithm to estimate the merit of the feature subset. Although wrapper methods usually can get better results than filter methods, they tend to be much slower than filter methods. Hybrid methods combine wrapper and filter approaches to achieve the best performance.

References [28–37] adopted the filter methods to select features. The filter methods mainly use the heuristic evaluation function to rank the initial features and remove the irrelevant features from the ranked features. References [28–30] used the correlation-based feature selection (CFS) method to select the feature subset. Reference [29] used the Pearson correlation to keep relevant features while removing redundant features, and the final selected features were tested on different classification algorithms, and the results indicated that the J48 classifier got the highest classification accuracy and lowest false-positive rate. References [31–35] used the mutual information method to get relevant features, such as MIMF [31], MIFS-U [32], and MMIMF [33]. Reference [34] suggested the flexible mutual information-based feature selection (FMIFS) that overcame the limitation of setting an appropriate value for β in [31–33]. Reference [35] designed a new mutual information algorithm (RPFMI), which added the redundant penalty between features to select optimal features, and the experiments on KDDCup'99 [48] showed that the RPFMI method got the better results with regard to accuracy rate, detection rate, and false-positive rate metrics by comparing with other feature selection algorithms; especially for Dos attacks, the RPFMI got the highest accuracy rate (99.772%).

In addition, Reference [36] proposed to build the layer-based intrusion detection system (LIDS) which contains four layers corresponding to the four attacks groups (Probe, Dos, R2L, and U2R), and selects features for each layer based upon the type of attacks that the layer is trained to detect. They used domain knowledge along with practical significance to manually select features. Finally, they selected only 5 features for the Probe layer, 9 features for the Dos layer, 14 features for the R2L layer, and 8 features for the U2R layer. Reference [37] proposed an intelligent CRF-based feature selection automatically by extending the existing feature selection method in Reference [36]. They assigned contribution value to each feature in the layer, and used the threshold to extract features for each type of attack based on cumulative contribution values. Compared to Reference [36], Reference [37] selected only five features for Probe attack, five features for Dos attack, eleven features for R2L, and five features for U2R finally.

References [38–44] introduced wrapper methods to select feature subsets. Reference [40] proposed two feature selection methods: Random Forest Forward Selection

Ranking (RF-FSR) and Random Forest Backward Elimination Ranking (RF-BER), and the final selected features using the two methods proposed were tested on the NSL-KDD dataset, with a detection rate of 99.80% and a false-positive rate of 0.1% separately. Reference [44] proposed a dynamic recursive feature selection algorithm that begins with initialization using an empty set of features and continues with the addition of features recursively by applying correlation coefficient values. In this way, the optimal number of features is obtained when the correlation between the features varies beyond a TH of 0.75. The experimental analysis was carried out using the KDDCup'99 dataset and proved that the false-positive rate, energy consumption, and delay were reduced in the proposed work.

References [45–47] proposed a hybrid feature selection approach that combined the advantages of both filter and wrapper methods. Reference [45] designed a combination of feature grouping based on linear correlation coefficient (FGLCC) algorithm and cuttlefish algorithm (CFA) to eliminate irrelevant and redundant features from the original dataset, and the experiment results verified a high accuracy (95.03%) and detection rate (95.23%) with a low-false positive rate (1.65%). Reference [46] proposed to adopt the principle of mutual information and applied Least Square Support Vector Machine (LSSVM) to build feature selection algorithm and got a detection rate of 98.90% with a false-positive rate of 0.521%. Reference [47] designed the conditional random field and linear correlation coefficient-based feature selection algorithm (CRF-LCFS) to select the most contributed features, and final selected features are fed to the convolutional neural network to build a classification model, which finally achieves 98.88% as the overall detection accuracy.

2.2. Improve Performance of Traditional IDS by Using ML/DM. There were already pieces of literature focusing on improving the performance of the traditional IDS (e.g., Snort) by using ML/DM or other approaches. Summarily, they were divided into two types: generating Snort rules and constructing the Snort-based hybrid intrusion detection system.

References [49–54] attempted to directly generate Snort rules, which extended the Snort's ability to detect attacks. References [49–52] mainly used ML/DM technologies to characterize attacks, and then extracted their signatures to generate Snort rules for fast and accurate intrusion detection. For example, References [49, 50] proposed a hybrid intrusion detection system that used DM algorithms to excavate the frequent episode rules from normal traffic, and used them to find abnormal traffic, and then abstracted the signatures from the detected anomalous behavior by using a weighted frequent item set mining scheme. Finally, these signatures were used to generate Snort rules for future detection of similar attacks. Yet, Reference [49] only finished mapping the single connection or flow-level features into Snort rule keywords, which means that the generated Snort rules can only detect simple attacks based on a single connection, such as attacks of R2L and U2R type. For the attributes based on multiple connections, which can be used to characterize patterns of the complex attacks, such as Dos and Probe attacks, it did not give the implementation for mapping them into Snort rule keywords.

Reference [51] proposed a procedure for generating Snort rules based on the data mining technique for detecting network probe attacks. Reference [52] used the classification model based on the k-SVM algorithm to detect the network attacks at first, and then used the WEKA tool to extract the features for the individual attack. Finally, the extracted features were mapped into the corresponding rule keywords of Snort and used these mapped keywords to generate Snort rules. However, References [51, 52] did not explain how the final selected features were mapped into the rule keywords of Snort, especially for the features which cannot be directly mapped into the existing keywords of Snort.

References [53, 54] designed the approaches to derive new rules from the existing rules in Snort, such as Reference [53], which introduced the generalization and specialization methods to relax or vary the parameter conditions of the existing Snort rules, which can generate new rules for extending the ability of Snort to identify novel attacks. Reference [54] proposed a probabilistic abductive reasoning approach that augmented Snort to detect attacks. Nevertheless, References [53, 54] only derived new rules from the existing Snort rules and did not extract new patterns from network traffic to generate new rules.

Furthermore, References [55–60] proposed to build the hybrid intrusion detection system (HIDS) by combining the advantages of the low false-positive rate of signature-based method and the ability of the anomaly based method to detect unknown attacks, which finally realized the performance improvement of signature-based IDS. Reference [55] identified that the misuse recognition framework was customized by embedding the anomaly detection engines (e.g., PHAD, ALAD, LERAD), which made the signature-based IDS own the ability for detecting novel attacks. References [56, 57] introduced how to take the statistical packet anomaly detection engine (SPADE) as a preprocessor plugin to embed into the Snort, which finally extended the functionality of Snort.

References [59, 60] suggested building the hybrid IDS, which was composed of the anomaly based detector and the signature-based detector in parallel. The hybrid IDS finally combined the outcomes of both detectors to enhance the overall detection accuracy. For instance, Reference [59] showed that the parallel HIDS could get a more accurate detection rate of Dos attacks compared with no-hybrid IDS. Although HIDS can expand the scope of detecting attacks of signature-based IDS, it is difficult to synchronize the anomaly detection engine and the misuse detection engine to form the final detection results in the real-time network environment. Unlike the misuse detection methods working in the real-time network environment, the anomaly detection methods preferred the offline working mode.

Finally, Reference [61] used a clustering approach based on decision trees to optimize the rules-to-input comparison process of the Snort detection engine and the experimental results showed that the speed of the detection process was significantly improved by comparing with the initial

detection engine of Snort. However, Reference [61] only improved the speed performance of the detection engine of Snort, rather than generating new rules to expand the functionality of detecting attacks of Snort.

In this paper, we focus on the "last mile problem" between ML/DM and the operational network IDS and propose an intrusion-specific approach to bridge the semantic gap between ML/DM and network operations. The intrusion-specific method for closing the last-mile gap mainly uses ML/DM technologies to abstract the optimal features for the specific attack from the dataset and converts them into the building blocks that can be used by the signature-based IDS to characterize signatures of the specific attack. Figure 2 shows the details of our intrusion-specific approach. Furthermore, as we know, for the traditional signature-based IDS, such as Snort, its existing mechanisms can be used to detect well the packet-level attacks, but usually suffers from the sophistication of flow(s)-level attacks that involve single flow and multiple flows [62]. Therefore, in this paper, for the extracted key attributes from the dataset which express the patterns of the flow(s)-level attacks, we added the corresponding new building blocks in the traditional signature-based IDS. Finally, we take Snort, KDDCup'99 dataset, and Dos attacks as the experimental subjects to prove the effectiveness of our method. The new rule keywords, belonging to the flow(s)-level granularity keywords, are added into Snort, and the newly generated rules not only expand the abilities of Snort for detecting Dos attacks but also reduce its false-positive rate and numbers of excessive alarms for the same Dos attack.

3. Proposed Feature Selection Method

The hybrid feature selection method, which can combine the advantages of the high efficiency of filter feature selection method and the ability of the wrapper feature selection method, can extract more optimal features; therefore, we proposed a hybrid feature selection method, which is used to select features of the specific attack from the network dataset.

3.1. Correlation-Based Feature Selection Method. The correlation-based feature selection method [63], as a kind of filter method, mainly uses a correlation-based heuristic evaluation function to rank the initial features and remove the irrelevant features. At present, Linear Correlation Coefficient (LCC) [64] and Mutual Information (MI) [31] are two popular heuristic evaluation functions for evaluating the relationship between two random variables. Considering the fast-computing speed of LCC, especially when processing large-scale data, it can show higher efficiency. Therefore, the LCC is taken to evaluate the correlations between two features in this paper.

$$\operatorname{corr}(X;Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}, \quad (1)$$

where $x(\_)$ and $y(\_)$ are expected values of the feature X and feature Y. x_i and y_i are the ith values of X and Y separately. Corr (X; Y) is equal to +1 if X and Y are linearly dependent and zero if they are completely independent.

In our feature selection approach, in order to reduce the burden of the wrapper method for selecting features, we use LCC to remove the irrelevant features and redundant features from the original feature space at first.

3.2. Random Forest-Recursive Feature Elimination (RF-RFE) Method. The random forest (RF) algorithm is the most popular bagging ensemble classifier [65]. Random forest consists of many decision trees. The output of random forest is decided by the votes given by all individual trees. Each decision tree is built by classifying the bootstrap samples of the input data using a tree algorithm. Then, every tree will be used to classify testing data. Each tree has a decision to label any testing data. This label is called a vote. Finally, the forest decides the classification result of the testing data after collecting the most votes among trees.

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by their importance in the model and by recursively eliminating a small number of features per loop. Random Forest-Recursive Feature Elimination (RF-RFE) expresses that the RF algorithm is introduced in the RFE to perform feature selection by iteratively training a model, ranking features, and then removing the lowest ranking features.

3.3. Proposed Feature Selection Method. We propose a hybrid feature selection method which calls the filter method (LCC) to remove the irrelevant and redundant features from the original feature space at first, and then the remaining features are fed to the wrapper method (RF-RFE) for further feature selection. For the RF-RFE feature selection method, there is a need to specify the number of final selected features in advance, yet there is no empirical value for this number. Therefore, we modify the RF-REF method and select all possible feature sets by using the RF-RFE method and use machine learning classification algorithms (e.g., Decision Tree) to evaluate the performance of each of them. The feature set which makes the classification model get the best performance will be retained as the final selected features. We call the modified RF-RFE method as RF-RFEX, and the workflow of the LCC-RF-RFEX approach is shown in Figure 3.

Specifically, we use LCC as a heuristic evaluation function and relevance threshold α to select feature subset from the original features of the dataset in the first step of our approach. The final selected feature subset has such characteristics that the linear correlation value between any feature in the subset and the label feature is greater than 0. At the same time, the linear correlation value of any two features in the subset keeps below α. The pseudo-code of the LCC method is shown in Algorithm 1.

In the second step, the filtered features after the first step are fed to the wrapper method RF-RFEX to select the

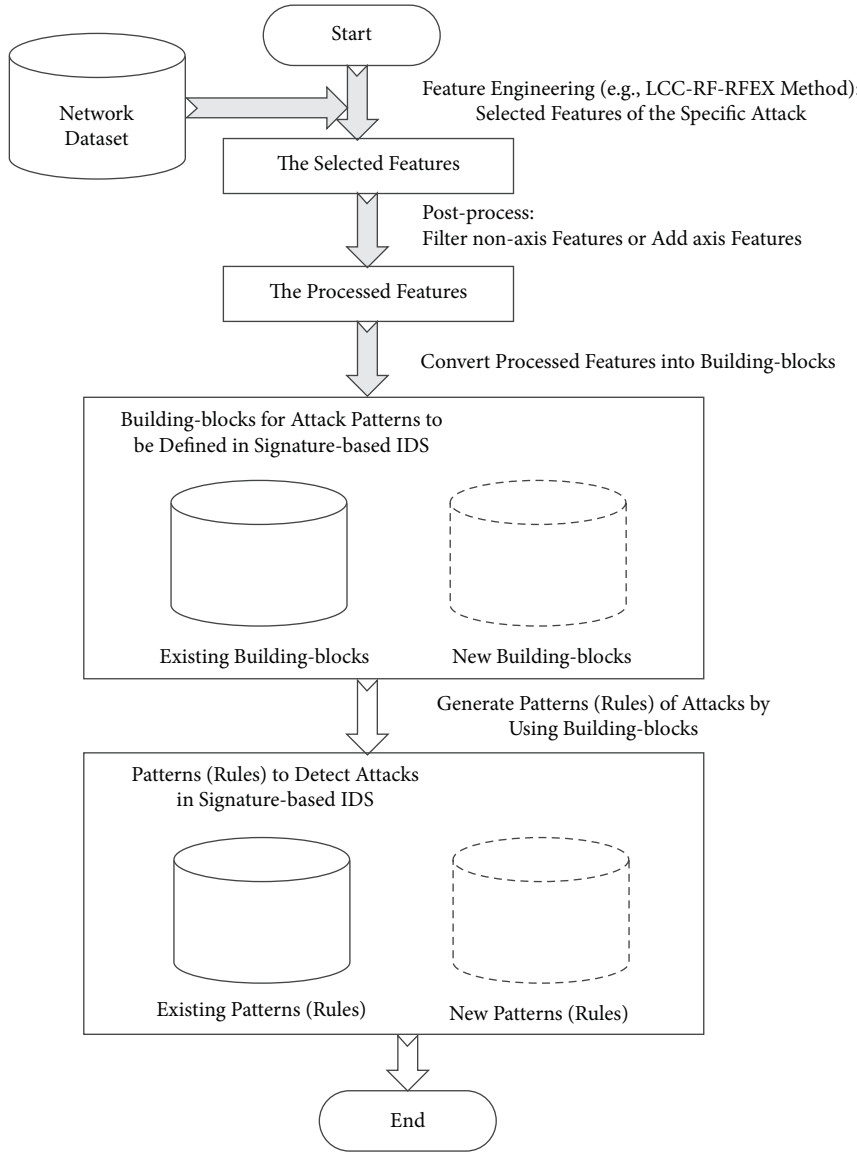

Figure 2: The flow chart of the intrusion-specific approach.

final feature subset. In each iteration of the RF-RFEX method, it calls the RF-RFE method to select a specific feature set, and counts the Accuracy Rate (AR) based on the currently selected feature set and Decision Tree algorithm, then, compares the current AR with the maximum AR of the last iteration and saves the maximum of AR and the corresponding feature set. At the end of the loop, the feature set which owns a maximum of AR will retain as the final selected features. The pseudo-code of the LCC-RF-RFEX method is shown in Algorithm 2.

4. Generating the Rule Keywords and Rules

In this section, we showed how to use the intrusion-specific method to generate the Snort rules. For each specific type of attack, the intrusion-specific method calls the LCC-RF-RFEX method to select the optimal features at first. After the post-process on the selected features, we convert each of the selected features into the corresponding rule keyword of Snort. If there are no existing rule keywords that can map to the selected features, we add new corresponding rule keywords in Snort. Finally, we use these converted rule keywords to generate rules for detecting the specific type of attacks. Figure 2 shows the workflow of our intrusion-specific approach.

4.1. KDDCups Section. KDDCup'99 dataset [48] is the public dataset for intrusion detection evaluation. It contains 39 attack types, with 22 attack types in the training dataset and 17 attack types in the test dataset. Attacks fall into four categories as Dos, U2R, R2L, and Probing. KDDCup'99 training dataset contains about 4,900,000 connection records and the test dataset contains 2 million connection records, and each connection record contains 41 features and is labeled as either normal or an attack. The 41 features

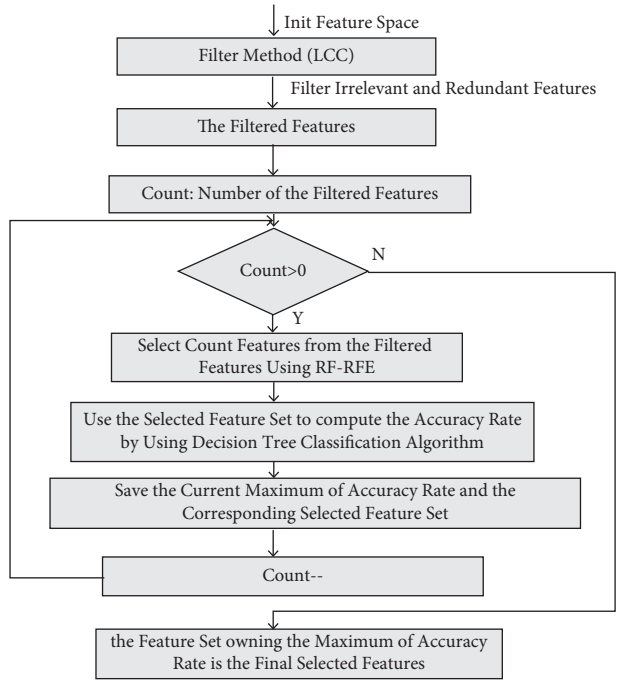

Figure 3: The workflow of the LCC-RF-RFEX method.

Input:,
F: Feature set $F = \{f_i \mid i = 1,...,n\}$
α: Threshold of relevance
Output: S - the filtered feature subset
01: S1: the filtered feature subsetorit Ø
02: Calculate abs of corr (C; f_i): |corr(C; f_i)|, for each feature f_i, $i = 1,..,n$, C notes the label feature
03: Sort |corr(C; f_i)| and Store |corr (C; f_i)| to Array: B in descending order, $i = 1,...,n$
04: For each |corr (C; f_i)| in B do
05: If |corr (C; f_i)|>0 and fi ∉ S and fi ∉ R then
06: S ∪ { f_i }
07: F ⟵ F \ { f_i }
08: For each f_j in F do
09: If |corr(f_j; f_i)|>α then
10: R ⟵ R ∪ { f_j}
11: F F ⟵ $_j$}
12: end if
13: end for
14: end if
15: end for
16: return S

Algorithm 1: The LCC feature selection algorithm.

in the KDDCup'99 dataset are divided into four groups: Basic Connection Features (No.1-No.9), Content Features (No.10–No.22), Statistical Features Based on Time (No.23–No.30), and Statistical Features Based on Connection (No.31-No.41). The Basic Connection Features and Content Features belong to flow-level features, which can be extracted from a TCP/UDP/ICMP connection, and the Statistical Features Based on Time and Statistical Features Based on Connection belong to flows-level features, which aggregate information over multiple connections based on two seconds time window intervals or based on prior 100 connections window intervals separately.

4.2. Post-Processing on the Selected Feature Set. In general, due to the incompleteness and imperfection of the network intrusion detection data set, the selected features on which are operated by the classification models cannot always have an insight into patterns of the malicious traffic. For example, Table 1 shows the final selected features of Dos

```
Input:,
   T: Training Data
   S: Test Data
   F: Feature set F = {f_i |i = 1,...,n}
   α: Threshold of relevance
Output:
   FS: the Final Select Feature Subset
01: FS←1
02: X←2 : Final Select Feα)
03: count←length(X)
04: If count>0 then
05:    T←T [X]
06:    S←S [X]
07: end if
08: Initialization Map: maxMap(key, value)=(0,X)
09: while count>0 do
10:    Calculate Accurate Rate (AR) metric based on T[X] and S[X] by using Decision Tree Classification Algorithm
11: If (key in maxMap) T[X] and then
12:    maxMap(key; value)← (RValue; X)
13:       end if
14:       count←
15:       X← Call RF-RFE Method(T, S, count)[1] to select feature subset
16: end while
17: FS← value in maxMap
18: return FS
```

Algorithm 2: The LCC-RF-RFEX feature selection algorithm. Note: [1]The RF-RFE Method is the Recursive Feature Elimination (RFE) method with the Random Forest classifier in the scikit-learn library, and its URL is https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html.

Attacks by using different Feature Selection Methods, and it is easy to count the times at which each feature has occurred in the eight feature selection methods from Table 1. For the limited space, the features placed in the top 2 are listed in Table 2. Table 2 shows that five ML methods agree uniformly that the *service*, *count*, *src_bytes*, *dst_bytes,* and *protocol_type* features could express the patterns of Dos attacks. However, with the help of network security domain knowledge, we found that the results were not entirely correct. For the two features: *service* and *protocol_type* can describe the basic connection characteristics of Dos attacks, and the *count* feature can reveal the high intensity or frequency characteristics of Dos attacks. Therefore, we called them as axis features [66]. For the *src_bytes* and *dst_bytes* features, they do not always give the best description of Dos attacks, and we called them nonaxis features [66]. After all, hackers can easily evade detection by tampering with the values of the nonaxis features. Therefore, before converting the selected features into the building blocks of the operational network IDS, the nonaxis features should be filtered from the selected features, which makes the converted building blocks express the patterns of malicious traffic more accurately.

Furthermore, for some attacks, we find that the final extracted features do not contain the two axis features: *service* and *protocol_type* features. However, when we generate the rules of Snort by using the final extracted features, the two axis features are required; therefore, when the final extracted features do not contain the two axis features, the two axis features should be added to them.

4.3. Map Features into Snort Rule Keywords. After completing the post-process on the selected features, the final processed features can be mapped into the building blocks of the operational network IDS, which can be used to express the attack patterns. In Snort, the building blocks are Snort rule keywords. Therefore, in this section, we illustrated how to map the selected features from the KDDCup'99 dataset into Snort rule keywords. As we know, Snort rules are divided into two logical sections: the rule header and the rule options. The rule header contains the rule's action, protocol, source, destination IP addresses, subnet mask, and the source and destination ports information. The rule option section provides four major categories of rule keywords, which are general, payload, nonpayload, and post-detection. These existing rule keywords in Snort usually describe the characteristics of packet-level traffics, and the KDDCup'99 dataset mainly contains the aggregated features that describe the characteristics of the flow(s)-level traffic.

Therefore, for a few features in the KDDCup'99 dataset describing the information of the basic connection, the existing rule keywords in Snort can correspond to them. For example, protocol_type, service, and land can be directly mapped into Snort's existing rule keywords which are protocol, destination port, and same ip, respectively. However, for the most aggregated features in the KDDCup'99 dataset, no existing rule keywords correspond to them in Snort. Therefore, we can add new rule keywords to establish mapping relations with them by changing the Snort source code. So far, we have completed mapping the aggregated features (No.1–No.9, No.23–No.41) of

Table 1: The selected features of Dos Attacks by using different feature selection methods.

Method	Count	Select features
MMIFS ($\beta = 0.5$) [33]	8	5, 23, 6, 2, 24, 41, 36, 3
FFSA [33]	3	5, 38, 3
LCFS [33]	36	32, 27, 23, 38, 41, 24, 13, 2, 40, 22, 30, 25, 28, 35, 26, 37,12, 36, 39, 1, 10, 14, 11, 17, 33, 16, 19, 18, 9, 5, 34, 31, 6, 3,29, 3
MIFA [41]	10	41, 40, 13, 10, 5, 6, 23, 28, 24, 27
RPFMI [35]	23	7, 2, 13, 4, 19, 15, 16, 17, 18, 14, 28, 20, 23, 31, 29, 11, 26, 27, 40, 41, 3, 38, 1
FMIFS [34]	12	23, 5, 3, 6, 32, 24, 12, 2, 37, 36, 8, 31
Wrapper (C4.5) [41]	10	2, 3, 5, 6, 11, 12, 23, 24, 27, 41
RF-RFE-LCC ($\alpha = 0.9$)	11	2, 23, 8, 37, 40, 5, 4, 7, 39, 6, 1
MMIFS ($\beta = 0.5$) [43]	8	5, 23, 6, 2, 24, 41, 36, 3

Table 2: The statistics times of features.

Feature	Times	Method
Service	7	MMIFS, FFSA, LCFS, RPFMI, FMIFS, wrapper, RF-RFE-LCC
Count	7	MMIFS, LCFS, MMIFA, RPFMI, FMIFS, wrapper, RF-RFE-LCC
src_bytes	7	MMIFS, FFSA, LCFS, MMIFA, FMIFS, wrapper, RF-RFE-LCC
dst_bytes	6	MMIFS, LCFS, MMIFA, FMIFS, wrapper, RF-RFE-LCC
protocol_type	6	MMIFS, LCFS, RPFMI, FMIFS, wrapper, RF-RFE-LCC

KDDCup'99 into Snort's rule keywords, and the final modified Snort source code has already been uploaded in GitHub (the GitHub URL of the modified Snort source code is https://github.com/jacky-sunxibin/modified-snort). The steps to add new keywords to Snort are described as follows:

Step 1. Enable the preprocessor stream5_global plugin that can track TCP, UDP, and ICMP sessions in snort.conf file.

Step 2. Add new variables to save connection statistical values of the previous 100 connections window and the past 2 seconds window in SessionControlBlock struct (session_common.h)

Step 3. Modify session preprocessor plugin(spp_session.c) source code and use the added variables in Step2 to track statistics of network connections in 2 seconds time window and 100 connections window.

Step 4. For each new keyword, add a new plugin(sp_xxx.h and sp_xxx.c) in detection plugins of Snort source code to complete the addition of the new keyword.

For example, to add a new rule keyword, duration in Snort, at first, the sp_duration.h and sp_duration.c files are created under the detection-plugins directory of Snort source code. And then, the code is added to count the connection duration according to each incoming packet in the sp_duration. Table 3 shows the mapping relations between features in KDDCup'99 and Snort keywords in detail.

4.4. Generating Snort Rules. In this section, we show how to generate Snort rules by using the newly generated keywords, and the process of generating Snort rules is described as follows.

Step 1. Sort the final selected feature subset in descending order according to the linear correlation values between each feature and target/class feature.

Step 2. Map the sorted feature subset into the corresponding rule keywords of Snort according to Table 3.

Step 3. Assign values to the mapped keywords respectively. For each rule keyword, its values come from the corresponding feature. If the feature belongs to the discrete or symbolic category, its final values are the combination of its discrete values in labeling the corresponding attack records of the dataset. Otherwise, it belongs to the continuous category. We can use statistical methods to count its thresholds from the corresponding attack records in the dataset, such as min, avg, and max.

Step 4. Generate the Snort rule according to the mapped keywords. To improve the matching performance of each newly generated rule in Snort, for the rule keyword belonging to the rule options, its order in the Snort rule is the same as its order in the mapped keywords.

Step 5. Evaluate the newly generated Snort rules.

For example, for the Mailbomb attack in the KDDCup'99 dataset, as shown in Table 4, the final selected features are *service, protocol_type,* and *srv_count* separately. According to Table 3, the final mapped rule keywords are *dst port, protocol*, and *twoSecondsSameDstServiceSessions* separately. Furthermore, for the *protocol_type* and *service* features belonging to the symbolic category, their values can directly come from the records of labeling Mailbomb attack in the KDDCup'99 dataset, such as *protocol_type* = tcp and *service* = 25. For *srv_count* belongs to the continuous

Table 3: Map Features (No. 1–No. 9, No. 23–No. 41) into snort rule keywords.

Feature name	Snort rule keywords	New keyword	Granularity
Duration	Duration	Yes	Flow level
protocol_type	Protocol	No	Flow level
Service	dst port	No	Flow level
Flag	sessionFlags	Yes	Flow level
src_bytes	src_bytes	Yes	Flow level
dst_bytes	dst_bytes	Yes	Flow level
Land	Sameip	No	Flow level
wrong_fragment	Overlapfragment	Yes	Flow level
Urgent	urgentCount	Yes	Flow level
Count	twoSecondsSameDstHostSessions	Yes	Flow level
srv_count	twoSecondsSameDstServiceSessions	Yes	Flow level
serror_rate	twoSecondsSameDstHostSynErrorRate	Yes	Flow level
srv_serror_rate	twoSecondsSameDstServiceSynErrorRate	Yes	Flow level
rerror_rate	twoSecondsSameDstHostRejErrorRate	Yes	Flow level
srv_rerror_rate	twoSecondsSameDstServiceRejErrorRate	Yes	Flow level
same_srv_rate	twoSecondsSameDstHostServiceRate	Yes	Flow level
diff_srv_rate	twoSecondsSameDstHostDiffServiceRate	Yes	Flow level
srv_diff_host_rate	twoSecondsSameDstServiceDiffDstHostRate	Yes	Flow level
dst_host_count	before100SameDstHostSessions	Yes	Flow level
dst_host_srv_count	before100SameDstHostServiceSessions	Yes	Flow level
dst_host_same_srv_rate	before100SameDstHostServiceRate	Yes	Flow level
dst_host_diff_srv_rate	before100SameDstHostDiffServiceRate	Yes	Flow level
dst_host_same_src_port_rate	before100SameDstHostSourcePortRate	Yes	Flow level
dst_host_srv_diff_host_rate	before100SameDstHostServiceDiffSIPRate	Yes	Flow level
dst_host_serror_rate	before100SameDstHostSynErrorRate	Yes	Flow level
dst_host_srv_serror_rate	before100SameDstHostServiceSynErrorRate	Yes	Flow level
dst_host_rerror_rate	before100SameDstHostRejErrorRate	Yes	Flow level
dst_host_srv_rerror_rate	before100SameDstHostServiceRejErrorRate	Yes	Flow level

category, its thresholds are counted from the records of labeling Mailbomb attacks by using statistical methods, such as min = 2, avg = 240, and max = 247. if we select ≥ max as the threshold range of *srv_ count*, the mapped rule keywords are *service* = 25, *protocol* = tcp, and *twoSecondsSameDstServiceSessions* ≥ 247. Finally, one of the Snort rules for detecting Mailbomb attacks was generated as follows:

Alert tcp $EXTERNAL_NET any -> $SMTP_SERVER 25 (msg: "mailbomb dos attack"; twoSecondsSameDstServiceSessions:≥247; classtype:attempted-dos; sid:80018; rev:1).

5. Experiments and Results

5.1. Data Preprocessing. In the KDDCup'99 dataset, the *protocol_type*, *service,* and *flag* are symbolic features. Our feature selection method needs input records in the format of real number vectors. Therefore, they should be transformed into the numeric feature in advance. LabelEncoder, a utility class of scikit-learn library, can be used to complete data preprocessing.

5.2. Performance Metrics. These metrics are used to evaluate the performances of feature selection method and newly generated Snort rules, and they are accuracy rate (AR), detection rate (DR), false-positive rate (FPR), precision, false-negative rate (FNR), and F_β-score.

AR (accuracy rate) is formally defined by

$$\mathrm{AR} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}. \tag{2}$$

Precision is formally defined by

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{3}$$

DR (detection rate) is formally defined by

$$\mathrm{DR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{4}$$

FPR (false-positive rate) is formally defined by

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{5}$$

FNR (false-negative rate) is formally defined by

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}, \tag{6}$$

where TP is the number of correct detection attacks, FN is the number of undetected attacks, FP is the number of mistake detection attacks, and TN is the number of correct detection nonattacks. F_β-score is the harmonic mean of precision and DR, which is formally defined by

$$F_\beta - \mathrm{score} = \frac{\left(1 + \beta^2\right) * \mathrm{precision} * \mathrm{DR}}{\beta^2 * \mathrm{precision} + \mathrm{DR}}, \tag{7}$$

where β is used to set the weights of DR and precision. If β is greater than 1, it means DR has a higher weight than precision. Otherwise, precision has a higher weight value.

Table 4: The final processed features of the specific Dos attacks.

Attack	The selected feature
Apache2	srv_serror_rate, dst_host_srv_serror_rate, duration, protocol_type, service
Land	Land, service, protocol_type
Teardrop	wrong_fragment, protocol_type, service
Processtable	dst_host_srv_serror_rate, duration, service, protocol_type
Smurf	Count, dst_host_same_src_port_rate, protocol_type, service
Neptune	same_srv_rate, flag, count, service, diff_srv_rate, protocol_type
Mailbomb	Service, protocol_type, srv_count

5.3. Evaluation of the Proposed Feature Selection Method. In the KDDCup'99 dataset [48], all kinds of attacks are not equally distributed, which may affect the performance of our feature selection method. Therefore, to avoid an impact on unbalanced data distribution, we form the training data and test data, which are shown in Table 5.

We used Python language to fulfill our proposed feature selection method by calling scikit-learn library. In our proposed feature selection method, we must provide the value of relevance threshold (α) at first, and α is a parameter which is determined empirically. If $\alpha = 0$, it means that the algorithm only focuses on the linear correlation value of any feature in the selected feature subset and the label feature, and the dependency between features is not considered. Experimental results showed that the selected features could get the best performance when α is set 0.9, which indicates that our algorithm places more emphasis on the dependency between input features by comparing with the relation between input features and the label feature.

All experiments were performed on a Windows platform having configuration i5 core 4 CPU 2.3 GHz, 8 GB RAM. Table 6 shows the selected features by using our LCC-RF-RFEX method for four types of attacks. Table 7 shows the comparison results of four kinds of attacks between the selected features and full features (41) separately. The results indicated that the classification models based on the selected features can get better results on DR, AR, and FPR metrics. At the same time, as shown in Table 8, the consuming training time and consuming testing time of selected features are less than those of full features. This is mainly because the irrelevant and redundant features, which can degrade the performance of classification models, are removed by using our approach in advance. In addition, from Table 7, we can find that the classification models based on selected features show the poor performance of detecting R2L and U2R attacks compared to Dos and Probe attacks. This may be because the features in the KDDCup'99 dataset are in favor to describe the statistics of a single connection or multiple connections, such as No.1–No.9 features and No.23–No.41 features. The functions of No.10-No.22 features describing the payload of packet or connection are usually weak, which directly degrades the performance of detecting R2L and U2R attacks. Therefore, new features which are used to describe the payload of packet or connection in detail are required in the future to find more R2L and U2R attacks.

Table 9 shows the comparison results of our approach and the existing Filter methods: *linear correlation-based feature selection (LCFS)* [33], *mutual information feature selection (MIFS)* [31], and the wrapper method: *the Modified Random Forest-Recursive Feature Elimination (RF-RFEX)*. Experimental results illustrated that our proposed method can make the classification model get the best performance. This is due to absorbing the advantages of the filter method and wrapper method in our feature selection process. Furthermore, Table 10 shows the consuming time of selecting the features of Dos attacks by using different methods. From Table 10, we can find that our proposed method consumed more time than LCFS and MIFS

Table 5: Datasets 1–4 for evaluating feature selection methods.

Attack category	Types	Training size	Testing size	Dataset
Dos	Normal	20000	20000	Dataset-1
	Smurf	10000	10000	
	Neptune	5000	5000	
	Mailbomb	1500	1500	
	Back	500	500	
	Land	15	15	
	Teardrop	400	400	
	Processtable	350	350	
	Pod	100	100	
	Apache2	250	250	
	Subtotal	38115	38115	
Probe	Normal	10000	10000	Dataset-2
	Ipsweep	1247	306	
	Mscan	600	400	
	Nmap	130	100	
	Portsweep	540	500	
	Saint	400	300	
	Satan	800	600	
	Subtotal	13717	12206	
U2R	Normal	10000	10000	Dataset-3
	buffer_overflow	30	22	
	Httptunnel	158	158	
	Loadmodule	9	2	
	Perl	3	2	
	Rootkit	10	13	
	Subtotal	10210	10197	
R2L	Normal	20000	20000	Dataset-4
	ftp_write	8	3	
	guess_passwd	53	4367	
	Imap	12	1	
	Multihop	7	18	
	Phf	4	2	
	Warezclient	1020	1020	
	Warezmaster	20	1602	
	Subtotal	21124	27013	

Table 6: Selected features of four types of attacks by using the LCC-RF-RFE method.

Attack category	Selected features (No. 1–No. 41)
Dos	2, 23, 8, 37, 40, 5, 4, 7, 39, 6, 1
Probe	3, 5, 6, 33, 37
R2L	2, 22, 12, 3, 11, 1, 33
U2R	3, 33, 5, 14, 16, 10, 12, 15, 18, 19, 1, 17, 34, 40

Table 7: Performance of decision tree and logistic regression classifier with selected features and 41 features.

Attack category	Methods	Accuracy rate (AR)		Detection rate (DR)		False-positive rate (FPR)	
		Selected features (%)	Full 41 features (%)	Selected features (%)	Full 41 features (%)	Selected features (%)	Full 41 features (%)
Dos	Decision tree	99.61	99.61	99.62	99.61	0.39	0.41
	Logistic regression	93.73	92.25	89.37	86.33	2.31	2.38
Probe	Decision tree	98.25	95.92	94.38	81.96	0.89	0.99
	Logistic regression	98.30	96.93	98.83	98.41	1.37	3.39
R2L	Decision tree	83.11	81.14	37.49	27.56	0.90	0.95
	Logistic regression	78.65	75.61	19.29	6.13	0.20	0.53
U2R	Decision tree	99.37	98.21	78.17	10.66	0.21	0.6
	Logistic regression	98.77	98.12	43.65	35.55	0.10	0.14

Table 8: Building time and testing time of decision tree and logistic regression classifier with selected features and 41 features.

Attack category	Methods	Building time (second)		Testing time (second)	
		Selected features	Full 41 features	Selected features	Full 41 features
Dos	Decision tree	0.0626	0.1875	0.1875	0.2188
	Logistic regression	0.5937	0.8212	0.125	0.1875
Probe	Decision tree	0.0234	0.0625	0.0312	0.0938
	Logistic regression	0.1719	0.3281	0.0312	0.0938
R2L	Decision tree	0.0132	0.0938	0.0625	0.1406
	Logistic regression	0.3593	0.6563	0.125	0.1929
U2R	Decision tree	0.0312	0.0625	0.0313	0.0625
	Logistic regression	0.2187	0.375	0.0312	0.0623

Table 9: Accuracy rate of decision tree classifier with existing feature selection methods and proposed method.

Feature selection method	Dos (%)	Probe (%)	U2R (%)	R2L (%)
LCFS ($\beta = 0.5$)	98.35	97.67	98.21	73.82
MIFS ($\beta = 0.5$)	98.61	97.21	98.23	79.51
RF-RFEX	99.32	98.16	98.22	81.51
LCC-RF-RFEX ($\beta = 0.9$)	99.61	98.25	99.37	83.11

methods but less time than the RF-RFEX method. This is because our LCC-RF-RFEX method contains the feature selection processes of LCFS and RF-RFEX methods simultaneously. The experimental results showed that the consuming time of the RF-RFEX method accounts for most of the total consuming time in the LCC-RF-RFEX method. In our LCC-RF-RFEX method, we use LCC to remove the irrelevant features and redundant features from the original feature space in advance, which reduces the calculation amount of the RF-RFEX method; therefore, our method consumes less time than the RF-RFEX method. Finally, we also find that the Linear Correlation Coefficient method has a higher computational efficiency than Mutual Information method, and this is why we choose the LCC method and not the MI method to remove the irrelevant features and redundant features from the original feature space in our proposed method.

Finally, we compared our approach with the recent feature selection methods, and the experimental results are shown in Table 11. As can be seen in Table 11, our proposed approach outperforms the KH, FAFS, and RPFMI methods. The reason is that the proposed LCC-RF-RFEX method combines filter (LCC) and wrapper (RF-RFEX) approaches to achieve the best performance. In addition, another reason

Table 10: Consuming time of feature selection of Dos attacks with existing feature selection methods and the proposed method.

Method	Consuming time (seconds)
LCFS ($\beta = 0.5$)	3.15
MIFS ($\beta = 0.5$)	21.09
RF-RFE	219.92
LCC-RF-RFE ($\beta = 0.9$)	128.76

Table 11: Accuracy rate of decision tree classifier with recent feature selection methods and the proposed method.

Method	Dos (%)	Probe (%)	U2R (%)	R2L (%)
LCC-RF-RFEX ($\alpha = 0.9$)	**99.61**	**98.25**	**99.37**	**83.11**
KH [67]	91.75	95.69	98.03	64.35
FAFS [41]	98.72	95.91	97.54	79.75
RPFMI [35]	97.69	97.65	98.37	77.29

is that we adopt the greedy policy when selecting the feature subset, and only the feature subset which can make the classification model get the highest values of AR remain as the final selected features.

5.4. Generation of Snort Rules by Using the Selected Features. So far, most of the pieces of literature have focused on extracting optimal features of four kinds of attacks in the KDDCup'99 dataset, and they rarely focus on extracting critical features for the specific attack. Therefore, we selected seven kinds of Dos attacks from KDDCup'99 as experimental subjects to substantiate our approach. For each specific Dos attack, the attack records and normal records from the training dataset and test dataset of KDDCup'99 dataset were separately selected out to generate the corresponding datasets 5–11, and the details are shown in Table 12.

For each of the specific Dos attacks, our proposed feature selection method is used on the corresponding dataset 5–11 to select features separately. Table 13 showed the selected features of specific Dos attacks. As we discussed in the front section, we should impose a Post-Processing on the Selected Feature Set before using them to generate Snort rules. Table 4 showed the final processed feature set.

As shown in Table 4, the final processed feature subsets contain many aggregated features that own the continuous values, and we can count their thresholds by using statistical methods before mapping them into Snort rule keywords. As we know, their thresholds usually express the frequency characteristics of Dos attacks. Therefore, the setting of their threshold ranges will directly affect the performance of Snort rules. In our experiment, we separately calculated the min threshold value, the max threshold value, and the avg threshold value of each aggregated feature from the records of labeling corresponding Dos attack in the KDDCup'99 dataset. Finally, we mapped the final processed feature subset into the corresponding rule keywords of Snort according to Table 3 and used them to generate Snort rules. For each specific Dos attack, we can generate multiple Snort rules by combining the threshold ranges of the different aggregated features. So far, we have already generated 61 Snort rules for detecting Dos attacks in the KDDCup'99 dataset, and due to the limited space, we only showed partly generated rules in Table 14.

Table 12: Datasets 5–11 for selecting features of the Dos attacks.

Name	Training dataset		Test dataset	
	Attacks	Normals	Attacks	Normals
Dataset-5:Apache2[1]	794	97278	794	60593
Dataset-6:Land	21	97278	9	60593
Dataset-7:Mailbomb[1]	5000	97278	5000	60593
Dataset-8:Neptune	107201	97278	58001	60593
Dataset-9:Processtable[1]	759	97278	759	60593
Dataset-10:Smurf	280790	97278	164091	60593
Dataset-11:Teardrop	979	97278	12	60593

Note. [1]For Apache2, Mailbomb and Processtable attacks do not exist in the training dataset in KDDCup'99. Therefore, we copied their records from the test dataset of KDDCup'99 into the training dataset of dataset-2, dataset-4, and dataset-6 separately.

5.5. Evaluation of Newly Generated Rules. To evaluate the newly generated rules in Snort, we selected Dos attacks raw packets(.pcap) from 1999 DARPA Intrusion Detection Evaluation Dataset [68] to generate dataset12–18, which can be fed to Snort to evaluate the performance of the newly generated rules, and the details of dataset12–18 are shown in Table 15.

In order to clearly show the correlation trend between the threshold ranges setting and the performance of rules, six kinds of threshold ranges (>min, >20%, >40%, >60%, >80%, >max) were selected to evaluate the performance of newly generated rules. Figure 4 shows the false-positive rate, which presented the downward trend, and Figure 5 shows the false-negative rate, which presented the upward trend, as the values of the threshold range were increasing.

In addition, considering the case in which many network security administrators often complain that the alarms triggered by rules usually contain much more false alarms, the $F_{0.5}$−score metric was selected to evaluate the comprehensive performance of the newly generated rules. Figure 6 shows the values of $F_{0.5}$−score metric on different threshold ranges, and the experimental results showed that $F_{0.5}$−score got the maximum between >60% and >80%, that is to say, the false alarm rate dropped the fastest place.

Furthermore, although the newly generated rules can be used to detect Dos attacks, how to make them get the best performance for detecting Dos attacks is a difficult point. One consideration is the incorporation of local network security demands; in general, if you are more concerned about reducing the false alarm rate, you are suggested to select a larger threshold range of statistical keywords, e.g., select the >max threshold range. Otherwise, you are suggested to select a smaller threshold range for getting the lower false-negative rate. In addition, experimental results also indicated the point at which the false alarm rate drops the fastest often obtains the best result of the F0.5–score metric.

Table 13: The selected features of the specific Dos attacks.

Attack	The selected feature
Apache2	srv_serror_rate, dst_host_srv_serror_rate, duration, src_bytes
Land	Land
Teardrop	wrong_fragment
Processtable	dst_host_srv_serror_rate, duration, src_bytes
Smurf	Count, dst_host_same_src_port_rate, protocol_type
Neptune	same_srv_rate, flag, count, diff_srv_rate
Mailbomb	srv_count, src_bytes

Table 14: The part of generated snort rules for detecting Dos attacks.

Attack name	Rule keyword[1]	Generated snort rule
Teardrop	overlapfragment = 1, protocol = udp, dst port = any	*Alert ip[2] $EXTERNAL_NET any -> $HOME_NET any (msg: "teardrop dos attack"; overlapfragment; classtype:attempted-dos; sid: 80000; rev:1)*
Smurf	twoSecondsSameDstHostSessions (min = 2,max = 511, avg = 480), before100SameDstHostSourcePortRate (min = 1, max = 1, avg = 1), protocol = icmp, dst port = any	*Alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"smurf dos attack";twoSecondsSameDstHostSessions:≥511; before100SameDstHostSourcePortRate:≥1; itype:0;classtype: attempted-dos; sid:80006; rev:1)*
Neptune	twoSecondsSameDstHostServiceRate (min = 0, max = 1, avg = 0.07),	*Alert tcp $EXTERNAL_NET any <> $HOME_NET any (msg: "Neptune same server dos atack"; twoSecondsSameDstHostServiceRate:≥1; sessionFlags:S0,REJ,RST0; twoSecondsSameDstHostSessions:≥302;classtype:attempted-dos; sid: 80009; rev:1)*
	sessionFlags=(S0, REJ, RST0), twoSecondsSameDstHostSessions(min = 1, max = 302, avg = 184), dst port = 0 : 1023[3], twoSecondsSameDstHostDiffServiceRate (min = 0, max = 1, avg = 0.07), protocol = tcp	*Alert tcp $EXTERNAL_NET any <> $HOME_NET any (msg: "Neptune diff server dos atack"; sessionFlags:S0,REJ,RST0; twoSecondsSameDstHostSessions:≥302; twoSecondsSameDstHostDiffServiceRate:≥1; classtype:attempted-dos; sid:80007; rev:1)*
Apache2	twoSecondsSameDstServiceSynErrorRate (min = 0; max = 1.0; avg = 0.32), before100SameDstHostServiceSynErrorRate (min = 0; max = 0.84; avg = 0.12), Duration (min = 0; max = 2100, avg = 785), protocol = tcp, dst port = 80	*Alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"apache2 attack dos attack "; twoSecondsSameDstServiceSynErrorRate:≥1; before100SameDstHostServiceSynErrorRate:≥0.84;duration:≥2100; classtype:attempted-dos; sid:80015; rev:1)*
Mailbomb	dst port = 25, twoSecondsSameDstServiceSessions (min = 2, max = 247, avg = 240), protocol = tcp,	*Alert tcp $EXTERNAL_NET any -> $SMTP_SERVER 25 (msg:" mailbomb dos attack"; twoSecondsSameDstServiceSessions: ≥247; classtype:attempted-dos; sid:80018; rev:1)*
Processtable	before100SameDstHostServiceSynErrorRate (min = 0, max = 0.84, avg = 0.44), Duration (min = 0; max = 8233; avg = 5121), dst port = 23, protocol = tcp	*Alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg: "processtable dos attack ";before100SameDstHostServiceSynErrorRate:≥0.84;duration:≥8233; classtype:attempted-dos; sid:80021; rev:1)*
Land	Sameip, dst port = (79, 23), protocol = tcp	*Alert tcp $EXTERNAL_NET any -> $HOME_NET* [79, 23]*(msg:"land dos attack";* sameip; *classtype:attempted-dos; sid:80003; rev:1;)*

Note: [1]For each Dos attack, this column mainly showed its rule keywords in descending order of priority. For keywords belonging to category rule options in Snort, the keyword with higher priority is placed in the further front position of the generated rule. [2]For UDP protocol, it does not support fragmentation, and when the size of the resulting UDP datagram exceeds the link's MTU, it will be fragmented into multiple IP packets in the network layer. Therefore, when generating the rule for detecting the teardrop attack, the IP is used to replace the UDP, which aims to find if there are overlap fragments among the ip fragments of the UDP datagram. [3]For the Neptune attacks, the service feature in the KDDCup'99 dataset contains 54 distinct values. In order to avoid the generated Snort rule too long, the rule keyword, DST port is set well-known ports (0–1023).

Table 15: Datasets 12–18 for evaluating newly generated Snort rules.

Name	Connections		Total packets
	Attacks	Normals	
Dataset-12: Smurf(.pcap)	1000	200	1400
Dataset-13: Processtable(.pcap)	1000	200	9600
Dataset-14: Apache2(.pcap)	1000	200	7000
Dataset-15: Neptune(.pcap)	1000	200	3000
Dataset-16: Mailbomb(.pcap)	1000	200	24000
Dataset-17: Land(.pcap)	1000	200	3000
Dataset-18: Teardrop(.pcap)	1000	200	2600

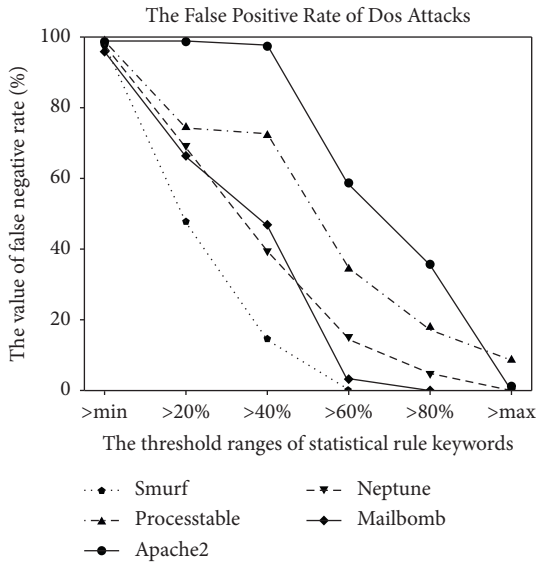

Figure 4: The false-positive rate of Dos attacks.

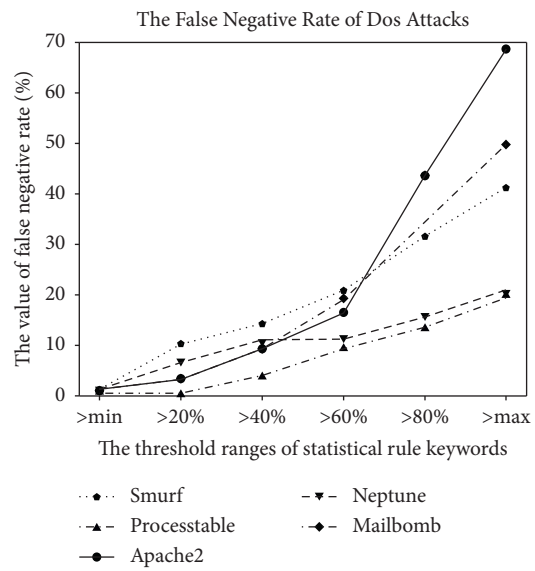

Figure 5: The false-negative rate of Dos attacks.

5.6. Performance Comparison between Our Newly Generated Rules and Third Party Rules. At last, we evaluated the performance of our newly generated rules compared with other rules for detecting Dos/DDos attacks in [69, 70]. In Reference [69], because the rules for detecting Dos attacks mainly used the existing keywords which belong to the type of Nonpayload Detection Rule Options in Snort, we called them the "Nonpayload Option Rules," denoted as NP. In Reference [70], the rules mainly used the existing keyword, *detection_filter,* which belongs to the type of Post-Detection Rule Options in Snort, to describe the high frequency characteristics of Dos/DDos attacks. Therefore, we called them the "Post-Detection Option Rules," denoted as PD. In this paper, we mainly used the newly added keywords which belong to the type of aggregated keywords based on flow(s)-level traffic to gain insights into the semantics of Dos/DDos attacks. Therefore, we called our newly generated rules the "Aggregated Flow Level Rules" (denoted as AF) in which the continuous type keywords used the ≥ max threshold range.

As shown in Figures 7 and 8, the "Aggregated Flow Level Rules" got better performance of FPR and AR metrics than "Non-payload Option Rules." The "Aggregated Flow Level Rules" got 1.5% FPR for detecting teardrop attack and 24.5% FPR for detecting synflood attack separately, which were lower than the corresponding 28.57% FPR and 48% FPR by the "Nonpayload Option Rules," and there are two reasons for them. One is the aggregated keywords deriving from the flow-level traffics used in "Aggregated Flow Level Rules," and these flow-level features can more accurately describe the signatures of Dos attacks by comparing to the packet-level keywords used in "Nonpayload Option Rules." For example, the rule for detecting teardrop attack in "Non-payload Option Rules" (dos.rules file) is shown as follows:

Alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"dos teardrop attack";fragbits:M;id:242; reference: bugtraq,124; reference:cve,1999-0015; reference:nessus,10279) (NP1).

This rule mainly detects the teardrop attack by simply using the packet-level keyword, *fragbits*, to judge whether the MF frag bit of the single packet header is set, and does not consider the correlations among the packets in the same flow. Obviously, this rule can inevitably cause false alarms for normal fragments of the UDP datagram with the MF flag bit setting. For the rule for detecting teardrop attack in "Aggregated Flow Level Rules," the newly generated rule keyword, *overlapfragment*, which belonged to the flow-level keyword was used to check offset overlaps among the IP fragments of the same flow. Therefore, compared to the *fragbits* keyword, it can better describe the characteristics of the teardrop attack. The rule for detecting the teardrop attack in "Aggregated Flow Level Rules" is shown as follows.

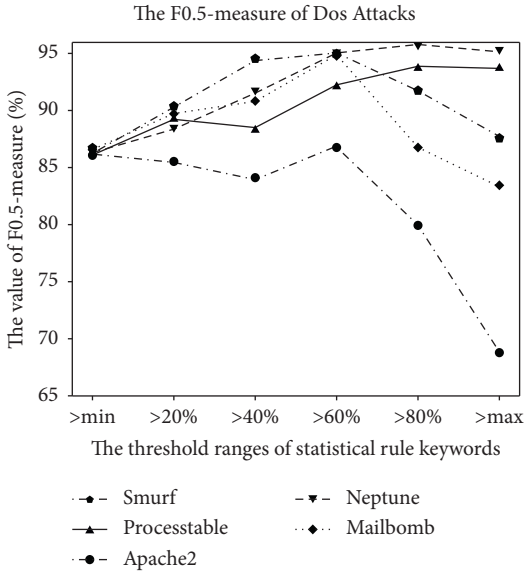

Figure 6: The $F_{0.5}$-score of Dos attacks.

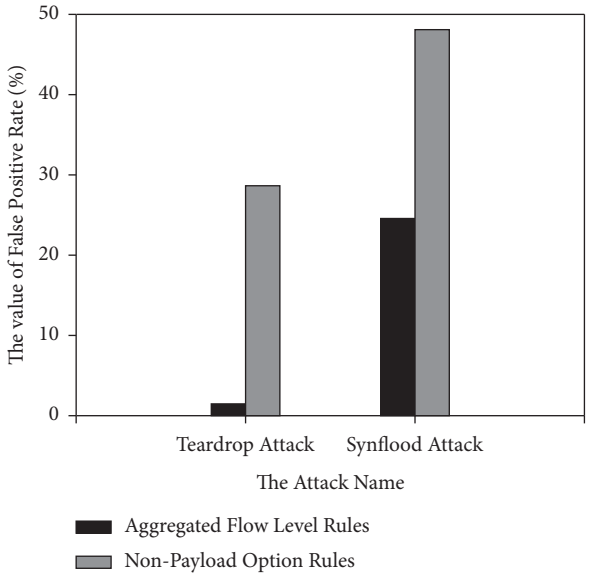

Figure 7: The false-positive rates of "Nonpayload Option Rules" and "Aggregated Flow Level Rules".

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"teardrop dos attack";overlapfragment;classtype: attempted-dos; sid:80000; rev:1) (AF1).

The other reason is that the high intensity and high frequency of network traffic based on specific time intervals are often the main characteristics of Dos/DDos attacks. However, they are not involved in "Nonpayload Option Rules." For instance, the rule to detect synflood attack in "Nonpayload Option Rules" (ddos.rules file).

alert tcp $EXTERNAL NET any <> $HOME NET any(msg:"DDOS shaft synflood";flow:stateless;flags:S,12; seq:674711609;reference:arachnids,253;reference:cve,2000-0138; classtype:attempted-dos; sid:241; rev:10;) (NP2).

This rule has some faults. At first, the synflood attacks can easily escape detection by changing the value of the TCP

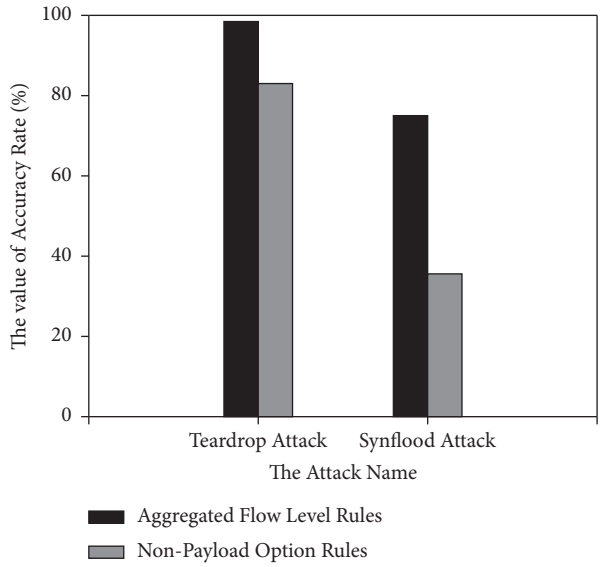

Figure 8: The accuracy rates of "Nonpayload Option Rules" and "Aggregated Flow Level Rules".

sequence number. Furthermore, this rule mainly used the *flags* keyword to check whether the syn flag bit of the TCP datagram header is set, and the high frequency characteristics of Dos/DDos attacks were not considered. As we know, the normal syn TCP datagrams usually set the syn flag, which can be mistaken as synflood attacks by using this rule. In our "Aggregated Flow Level Rules," the aggregated keywords based on flows-level, *twoSecondsSameDstHostServiceRate*, and *twoSecondsSameDstHostSessions* were used to reveal the high-frequency patterns of the synflood attack, which greatly reduced the FPR of Snort. One of our generated rules for detecting the synflood attack was shown as follows.

alert tcp $EXTERNAL_NET any <> $HOME_NET 0: 1023 (msg: "Neptune same server dos atack";twoSecondsSameDstHostServiceRate:≥1;sessionFlags: S0,REJ,RST0;twoSecondsSameDstHostSessions:≥302;classtype:attempted-dos; sid:80009; rev:1) (AF2).

Finally, we also evaluated the performance of our "Aggregated Flow Level Rules" and "Post-Detection Option Rules" in Reference [70]. Compared to the "Nonpayload Option Rules" in Reference [69], the "Post-Detection Option Rules" used the aggregated keyword, *detection_filter*, to describe the high frequency characteristics of Dos/DDos attacks, which led to the decrease of FPR. However, the *detection_filter* keyword was based on packet-level traffics, not based on flow(s)-level traffics, and it usually generates many packet-level alarms for the complex attacks of which activities span multiple flows. For example, the rule for detecting mail bomb attacks in "Post-Detection Option Rules" is shown as follows.

Alert tcp $EXTERNAL NET any -> $SMTP SERVER 25 (msg:"Possible Mail Bomb attack";flags:A+;flow:established; detection_filter:track by dst, count 2000, seconds 2;sid: 10003) (PD1).

Experimental results showed that the above rule generated about 89 packet-level alarms for each e-mail bomb

attack after the number of packets reaches the upper limit of the detection_filter keyword. However, the rules in "Aggregated Flow Level Rules" used the aggregated keywords based on flow(s)-level to describe the semantics of Dos attacks. Therefore, they can be usually triggered to generate alerts by the flow-level record, not by each packet, which greatly reduced the number of alarms by comparing to the rules based on packet-level keywords, such as the rule for detecting mail bomb attacks in "Aggregated Flow Level Rules," which is shown as follows.

Alert tcp $EXTERNAL NET any -> $SMTP SERVER 25 (msg:"Possible Mail Bomb attack"; twoSecondsSameDstServiceSessions:≥247;classtype:attempted-dos;sid: 80018;rev:001) (AF3).

As shown in Figures 9 and 10, we found that our "Aggregated Flow Level Rules" can generate fewer alarms and consume less detection time when compared with the "Post-Detection Option Rules." For the mail bomb attacks in 1998 DARPA Intrusion Detection Evaluation Dataset [68], experimental results showed that our "Aggregated Flow Level Rules" only generated one alarm for each e-mail bomb attack after the number of flows reaches the upper limit of *twoSecondsSameDstServiceSessions* keyword, and the ratio of alarm numbers generated by "Aggregated Flow Level Rules" and "Post-Detection Option Rules" is about 1 : 89. In addition, Figure 10 shows that our newly generated rules can consume less detection time compared with rules in "Post-Detection Option Rules." This is mainly because the flow(s)-level keywords which are used in our newly generated rules effectively reduce the false alarm rate and the number of excessive alerts, and then improve the time efficiency of the detection engine of Snort.

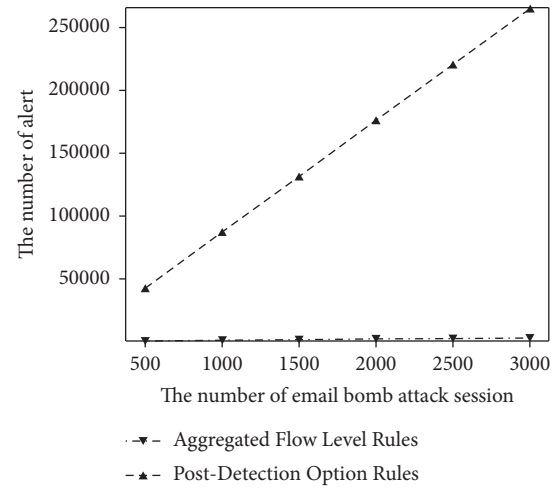

Figure 9: The Number of Alerts generated by "Aggregated Flow Level Rules" and "Post-Detection Option Rules".

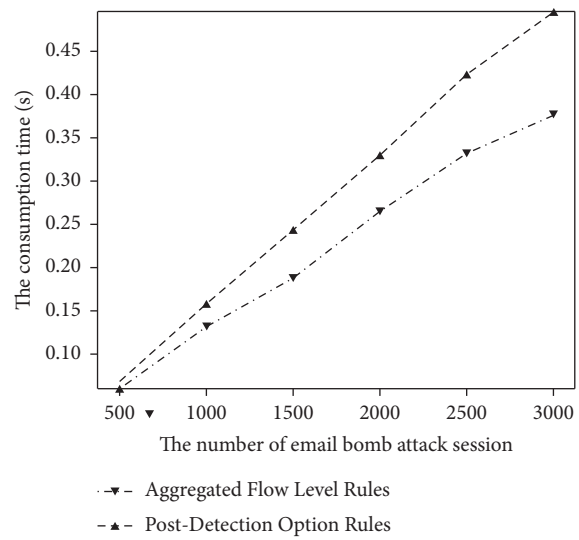

Figure 10: The processing time of "Aggregated Flow Level Rules" and "Post-Detection Option Rules".

6. Additional Points and Future Work

This paper analyzed the reasons why the ML/DM approaches are rarely deployed in the real-world network intrusion detection environment and pointed out that there is a gap between the strength and requirements of machine learning and network operational semantics and properties. Compared with the traditional ML/DM methods aiming to build more accurate classification models, this paper focuses on how to close the last-mile gap. We proposed the intrusion-specific approach to convert the selected features by using the LCC-RF-RFEX method in the network attack-specific building blocks and then generating the detection patterns by using them in the signature-based IDS. Finally, to substantiate our approach, we took Snort, KDDCup'99 dataset, and Dos attacks as experimental subjects to demonstrate the feasibility of our method. Compared to the full features, the Decision Tree models based on the selected features by using our proposed LCC-RF-RFEX method achieved better performance in terms of detection accuracy (average 1.37% increases), false-positive rate (average 0.14% decreases), training time (average 0.07s decreases), and testing time (average 0.05s decreases). In addition, the proposed LCC-RF-RFEX approach improved the average accuracy rate of Dos attacks by 3.56%, Probe attacks by 1.83%, U2R attacks by 1.39%, and R2L attacks by 9.31% when it is compared with the recent feature selection methods.

Furthermore, we also provided the intrusion-specific approach to add the new rule flow(s)-level keywords(building blocks) and generated new rules for detecting Dos attacks in Snort. The experimental results indicated that the newly generated rules can expand the abilities of Snort for detecting Dos attacks. Furthermore, compared with "Nonpayload Option Rules" and "Post-Detection Option Rules," our newly generated rules effectively reduced the false-positive rate and numbers of excessive alarms of the several types of Dos attacks. On average, a reduction of up to 25.28% false-positive alerts for Teardrop attacks and Synflood attacks, and a reduction of up to 98.87% excessive alerts for Mailbomb attacks were achieved in the experiments.

At present, we only finished extracting key features from the labeled dataset and converting them into building blocks

of an operational network IDS. However, for unlabeled network traffic, how to use semi-supervised learning or unsupervised learning technology to extract patterns of the novel or unknown attacks, and then convert them into building blocks to augment the function of the signature-based IDS, is a problem to be solved. Moreover, how to design an agent which can help signature-based IDS continuously expand its ability to detect attacks by mining attack signatures from the real-time network traffics is also a problem to be solved in future work.

Data Availability

The download URL of all the datasets (Dataset1–18) in this manuscript (id:3990386, title: Bridging the Last-Mile Gap in Network Security via Generating Intrusion-Specific Detection Patterns through Machine Learning) is https://github.com /jacky-sunxibin/modified-snort/tree/master/dataset. Among the Dataset1–18, Dataset1–4 for evaluating feature selection methods and Dataset5–11 For selecting features of the Dos attacks mainly come from the KDDCup'99 dataset (UR L: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html), and Dataset12–18 for evaluating the newly generated rules come from the 1998 DARPA Intrusion Detection Evaluation Dataset (URL: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the grant of 045/2016/A2 and by the grant of 0025/2019/AKP from Macau Science and Technology Foundation.

References

[1] V. Kumar and D. O. P. Sangwan, "Signature-based intrusion detection system using SNORT," *Int.J.Comput.p-pl.Inf.Technol*, vol. 1, no. 3, pp. 35–41, 2012.

[2] M. Mishin, *Anomaly Detection Algorithms and Techniques for Network Intrusion Detection Systems*, Thesis Submitted for Examination for the Degree of Master of Aalto University School of Science, Espoo Finland, 2020.

[3] R. Sommer and V. Paxson, "Outside the closed world: on using machine learning for network intrusion detection," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 305–316, Oakland, CA, USA, May 2010.

[4] S. Ganapathy, K. Kulothungan, S. Muthuraj, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: a survey," *EURASIP Journal on Wireless Communications and Networking*, vol. 271, no. 1, pp. 1–16, 2013.

[5] R. Srivastava and V. Richhariya, "Survey of current network intrusion detection techniques," *Journal of Information Engineering and Applications*, vol. 3, no. 6, pp. 27–33, 2013.

[6] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowledge-Based Systems*, vol. 136, pp. 130–139, Nov.2017.

[7] MG. Raman, N. Somu, S. Jagarapu et al., "An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm," *Artificial Intelligence Review*, vol. 52, pp. 3255–3286, 2019.

[8] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," in *Proceedings of the IEEE Int. Conf. Comput. Sci. Eng./IEEE Int. Conf. Embedded Ubiquitous Computing*, pp. 635–638, Guangzhou, China, July. 2017.

[9] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on ga and SVM," *IEE Access*, vol. 6, pp. 13624–13631, 2018.

[10] M. Al-Qatf, M. Alhabib, and K. Al-Sabahi, "Deep learning approach combining sparse autoen-coder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[11] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proceedings of the Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 2*, S. C. Satapathy and A. Joshi, Eds., , Ahmedabad, India, March 2017.

[12] K. Bajaj and A. Arora, "Improving the intrusion detection using discriminative machine learning approach and improve the time complexity by data mining feature selection methods," *International Journal of Computer Applications*, vol. 76, no. 1, pp. 5–11, 2013.

[13] A. Ahmim, M. A. Ferrag, L. Derdour, and H. Janicke, "A detailed analysis of using supervised machine learning for intrusion detection," in *Strategic Innovative Marketing and Tourism*, pp. 629–639, 2020.

[14] R. Panigrahi and S. Borah, "Rank allocation to J48 group of decision tree classifiers using binary and multiclass intrusion detection datasets," *Procedia Computer Science*, vol. 132, pp. 323–332, 2018.

[15] R. H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTMbased deep learning approach for classifying malicious traffic at the packet level," *Applied Sciences*, vol. 9, no. 16, p. 3414, Aug. 2019.

[16] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural networks for representation learning," in *Proceedings of the 31st International Conference on Information Networking*, pp. 712–717, Da Nang, Vietnam, January 2017.

[17] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[18] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," *Neural Computing & Applications*, vol. 21, no. 6, pp. 1185–1190, 2012.

[19] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[20] S. Eltanbouly and M. Bashendy, "Machine learning techniques for network anomaly detection: a survey," in *Proceedings of the IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, Doha, Qatar, February 2020.

[21] M. Mahoney and P. K. Chan, "PHAD: packet header anomaly detection for identifying hostile network traffic," *Florida Tech*, technical report CS-2001-04, 2001.

[22] M. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," *Florida Tech*, technical report CS-2001-06, 2002.

[23] S. Staniford, J. Hoagland, and J. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, no. 1, 2002.

[24] M. Zhang, B. Xu, and J. Gong, "An anomaly detection model based on one-class svm to detect network intrusions," in *Proceedings of the 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 102–107, IEEE, Shenzhen, China, December 2015.

[25] M. Ahmed, M. Mahfouz, and A. Abuhussein, "Network intrusion detection model using one-class support vector machine," in *Advances in Machine Learning and Computational Intelligence*, pp. 79–86, 2020.

[26] M. Roesch, "Snort-light weight intrusion detection for networks," in *Proceedings of the LISA'99:13th Systems Administration Conference*, Seattle, Washington, USA, November 1999.

[27] D. Day and B. Burns, "A performance analysis of Snort and Suricata network intrusion detection and prevention engines," in *Proceedings of the 5th Int. Conf. Digit. Society*, Derby, UK, 1875.

[28] A. Gul and E. Adali, "A feature selection algorithm for IDS," in *Proceedings of the (UBMK17) 2nd International Conference on Computer Science and Engineering*, pp. 816–819, Antalya, Turkey, October 2017.

[29] M. B. Shahbaz and X. Wang, "On efficiency enhancement of the correlation-based feature selection for intrusion detection systems," in *Proceedings of the IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference(IEMCON)*, Vancouver, BC, Canada, October 2016.

[30] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: an application to KDD Cup 99 dataset," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5947–5957, 2011.

[31] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[32] N. Kwak and C.-H. Chong-Ho Choi, "Input feature selection for classification problems," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 143–159, 2002.

[33] F. Amiri, M. Rezaei Yousefi, C. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184–1199, 2011.

[34] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[35] F. Zhao, J. Zhao, X. Niu, and S. Luo, "A filter feature selection algorithm based on mutual information for intrusion detection," *Applied Sciences*, vol. 8, no. 9, pp. 15–35, 2018.

[36] K. K. Gupta, B. Nath, and R. Kotagiri, "Layered approach using conditional random fields for intrusion detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 1, pp. 35–49, 2010.

[37] S. Ganapathy, P. Vijayakumar, P. Yogesh, and K. Arputharaj, "An intelligent CRF based feature selection for effective intrusion detection," *The International Arab Journal of Information Technology*, vol. 13, no. 1, pp. 46–50, 2016.

[38] B. Kumar and T. Swarnkar, "Filter versus wrapper feature subset selection in large dimensionality microarray: a review," *International Journal of Computer Science and Information Technologies*, vol. 2, no. 3, pp. 1048–1053, 2011.

[39] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature selection for intrusion detection using random forest," *Journal of Information Security*, vol. 07, no. 03, pp. 129–140, 2016.

[40] O. Y. Al-Jarrah, A. Siddiqui, M. Elsalamouny, P. D. Yoo, S. Muhaidat, and K. Kim, "Machine-learning-based feature selection techniques for large-scale network intrusion detection," in *Proceedings of the International Conference on Distributed Computing Systems Workshops*, pp. 177–181, Madrid, Spain, July 2014.

[41] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Computers & Security*, vol. 83, pp. 148–155, 2018.

[42] F. H. Almasoudy, W. L. Al-Yaseen, and A. K. Idrees, "Differential evolution wrapper feature selection for intrusion detection system," *Procedia Computer Science*, vol. 167, pp. 1230–1239, 2020.

[43] H. Alazzam, A. Sharieh, and K. Eddin Sabri, "A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer," *Expert Systems with Applications*, vol. 148, Article ID 113249, 2020.

[44] B. Riyaz and S. Ganapathy, "Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks," *IET Communications*, vol. 14, no. 5, pp. 888–895, 2020.

[45] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour, "Cyber intrusion detection by combined feature selection algorithm," *Journal of Information Security and Applications*, vol. 44, pp. 80–88, 2019.

[46] A. Mohammed and X. He, "A novel feature selection approach for intrusion detection data classification," in *Proceedings of the IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, Beijing, China, September 2014.

[47] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Computing*, vol. 24, no. 22, pp. 17265–17278.

[48] A. Gharib, I. Sharafaldiny, A. Habibi Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Proceedings of the Int. Conf. Inf. Sci. Secur. (ICISS)*, pp. 1–6, Pattaya, Thailand, December 2016.

[49] K. Wang, M. Cai, and Y. Chen, "Hybrid intrusion detection with weighted signature generation over anomalous internet episodes," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 1, pp. 41–55, 2007.

[50] K. J. Sahana Devi and M. Bharathi, "Hybrid intrusion detection with weighted signature generation," *International Journal of Computer Applications in Engineering Sciences*, vol. 1, no. IV, 2011.

[51] N. Khamphakdee, N. Benjamas, N. Benjamas, and S. Saiyod, "Improving intrusion detection system based on snort rules for network probe attacks detection with association rules technique of data mining," *Journal of ICT Research and Applications*, vol. 8, no. 3, pp. 234–250, 2015.

[52] M. Naga Surya Lakshmi and Y. Radhika, "Detection and analysis of network intrusions using data mining approaches," *International Journal of Applied Engineering Research, ISSN 0973-4562*, vol. 13, no. 6, pp. 4059–4066, 2018.

[53] U. Aickelin, J. Twycross, and T. H. Roberts, "Rule generalisation in intrusion detection systems using SNORT," *International Journal of Electronic Security and Digital Forensics*, vol. 1, no. 1, pp. 101–116, 2007.

[54] A. Ganesan, P. Parameshwarappa, A. Peshave, Z. Chen, and T. Oates, "Extending signature-based intrusion detection systems with Bayesian abductive reasoning," *Learn. Intell. Cyber Secur. (DYNAMICS) Workshop*, vol. 10, pp. 1–10, 2018, https://arxiv.org/abs/1903.12101.

[55] A. Garg and P. Maheshwari, "A hybrid intrusion detection system: a review," in *Proceedings of the 10th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1–5, Coimbatore, India, January 2016.

[56] S. Biles, *Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)*, Computer Security Online Ltd., 2003, https://www.computersecurityonline.com/spade/SPADE.pdf.

[57] J. Gomez, C. Gil, N. Padilla, R. Banos, and C. Jiménez, "Design of a snort-based hybrid intrusion detection system," in *Proceedings of the 10th International Work-Conference on Artificial Neural Networks, IWANN 2009 Workshops*, vol. 5518, p. 515pu22, Salamanca, Spain, June 2009.

[58] D. Ocampo, T. M. del Castillo, and M. A. Gomez, "Automated signature creator for a signature based intrusion detection system (pancakes)," in *Proceedings of the 2nd International Conference on Cyber Security Peace Fare and Digital Forensic*, pp. 198–205, Kuala Lumpur, Malaysia, March 2013.

[59] Ö Cepheli, S. Büyükçorak, and G. K. Kurt, "Hybrid intrusion detection system for DDoS attacks," *Journal of Electrical and Computer Engineering*, vol. 2016, Article ID 1075648, 8 pages, 2016.

[60] L. Zhang, J. Zhang, Y. Chen, and S. Lao, "Hybrid intrusion detection based on data mining," in *Proceedings of the 11th International Conference on Intelligent Computation Technology and Automation*, Changsha, China, September 2018.

[61] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," *Lecture Notes in Computer Science*, in *Proceedings of the 6th Int. Workshop Recent Adv. Intrusion Detect*, pp. 173–191, Pittsburgh, PA, USA, September 2003.

[62] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013.

[63] M. A. Hall, *Correlation-based Feature Selection for Machine Learning*, Ph.D. thesis, University of Waikato, Hamilton, New Zealand, 1999.

[64] R. Taylor, "Interpretation of the correlation coefficient: a basic review," *The Journal of Defence Modelling and Simulation*, vol. 6, no. 1, pp. 35–39.

[65] N. Farnaaz and MA. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[66] W. Lee, S. J. Stolfo, and K. W. Mok, "Adaptive intrusion detection: a data mining approach," *Artificial Intelligence Review*, vol. 14, no. 6, pp. 533–567, 2000.

[67] L. Xin, Y. Peng, J. Yiming, and L. Tian, "LNNLS-KH: a feature selection method for network intrusion detection," *Security and Communication Networks*, vol. 2021, Article ID 8830431, 22 pages, 2021.

[68] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[69] Eldon, *The GitHub URL of the Dos Attacks*, https://github.com/eldondev/Snort.

[70] A. Gupta and L. S. Sharma, "Mitigation of dos and port scan attacks using snort," *International Journal of Computer Science and Engineering*, vol. 7, no. 4, pp. 248–258, 2019.