IET Networks

Research Article

Machine learning-based IDS for software-defined 5G network

ISSN 2047-4954
Received on 23rd September 2017
Accepted on 25th November 2017
doi: 10.1049/iet-net.2017.0212
www.ietdl.org

Jiaqi Li[1], *Zhifeng Zhao*[1] ✉, *Rongpeng Li*[1]
[1]*College of Information Science & Electronic Engineering, Zhejiang University, Zheda Road 38, Hangzhou 310027, Zhejiang Province, People's Republic of China*
✉ *E-mail: zhaozf@zju.edu.cn*

Abstract: As an inevitable trend of future fifth generation (5G) networks, software-defined architecture has many advantages in providing centralised control and flexible resource management. However, it is also confronted with various security challenges and potential threats with emerging services and technologies. As the focus of network security, intrusion detection systems (IDSs) are usually deployed separately without collaboration. They are also unable to detect novel attacks with limited intelligent abilities, which are hard to meet the needs of software-defined 5G. In this study, the authors propose an intelligent IDS taking the advances in software-defined technology and artificial intelligence based on software-defined 5G architecture. It flexibly integrates security function modules which are adaptively invoked under centralised management and control with a global view. It can also deal with unknown intrusions by using machine learning algorithms. Evaluation results prove that the intelligent IDS achieves better performance with lower overhead. It is also verified that the selected machine learning algorithms show better accuracy and reduced false alarm rate in flow-based classification.

1 Introduction

Software-defined fifth generation (5G) architecture will be a crucial tendency in the development of future 5G networks [1]. It takes the advantage of software-defined network (SDN) [2] and network functions virtualisation (NFV) [3] through centralised management and dynamic resource allocation to meet the demands of 5G networks. Besides, the separation of the control and execution planes also facilitates the supervision of network status and the collection of information. With the uprising of novel attacks, it will also be faced with various challenges and security threats. As a result, new network security architecture and systems are desperately needed to enhance the security of software-defined 5G networks [4].

As an essential technology in network security, intrusion detection systems (IDSs) have received more and more concerns in efficiently detecting malicious attacks. Existing IDSs are separately deployed within restricted areas and hard to cooperate with each other. Moreover, they are usually signature based by matching behaviours of incoming intrusions with historical knowledge and pre-defined rules, which are unable to detect novel attacks intelligently.

To overcome the limitation of traditional IDS, artificial intelligence (AI) has been employed for intelligent detection. They classify abnormal traffic using machine learning techniques with a self-learning ability [5]. At present, there have been a few researches combining IDS and AI. However, they are still inadequate for coordinated detection considering the evolution and development of network systems.

In this paper, we propose an intelligent IDS based on software-defined 5G architecture. Benefit from the software-defined technology, it integrates relevant security function modules into a unified platform which are dynamically invoked under centralised management and control. Besides, it applies machine learning to intelligently learn rules from huge quantities of data and detects unknown attacks based on flow classification. It uses random forest (RF) for feature selection and combines *k*-means++ with adaptive boosting (AdaBoost) for flow classification. The proposed system enhances the strength of security protection for future 5G networks.

The remainder of this paper is organised as follows. Section 2 discusses the related researches in the field of intrusion detection by machine learning. Section 3 provides an overview of the architecture and describes each module in details. Section 4 introduces the proposed machine learning algorithms for feature selection and traffic classification. Section 5 presents the performance evaluation and results. Section 6 summarises this paper and proposes potential future work.

2 Related work

As SDN dynamically manages network configurations and controls packet processing in a centralised manner, it has well satisfied the evolution demand of cellular networks in the 5G era, which aims to provide flexible service provisioning mechanisms [6]. Therefore, the combination of 5G and SDN has attracted a lot of research interest. In [7], a new paradigm called SoftAir toward next generation (5G) wireless networks is introduced. The novel ideas of NF cloudification and network virtualisation are exploited to provide a scalable, flexible and resilient network architecture. However, there are still challenges in realising SoftAir. An architecture of 5G software-defined vehicular network integrated with SDN, cloud computing and fog computing technologies is proposed in [8]. A minimum transmission delay can be achieved by avoiding frequent handover between vehicles to meet requirements of intelligent transportation systems. However, there are still challenges to flexibly combine different types of network architectures in 5G vehicular networks.

However, with the fast development of software-defined 5G networks, the emergence of unknown attacks also poses severe security challenges. The advance of SDN has promoted the evolution of wireless network security systems through supporting the supervision of network status and the collection of flow statistics. It also provides network-layer security services such as packet routing, identity authentication and automated security management in a global view which facilitates the detection and prevention of attacks [9]. Some previous studies have investigated the ability of SDN and introduces several mechanisms to defend specific types of attacks. In [10], the framework adaptively classifies the network traffic into different classes according to the quality of service requirements for SDNs. It can be realised by exploiting the superior computation capacity, the global visibility and the inherent programmability of the network controller. Bawany *et al.* [11] present a comprehensive survey of existing SDN-based distributed denial of service (DDoS) attack detection solutions and present an SDN-based proactive DDoS defence

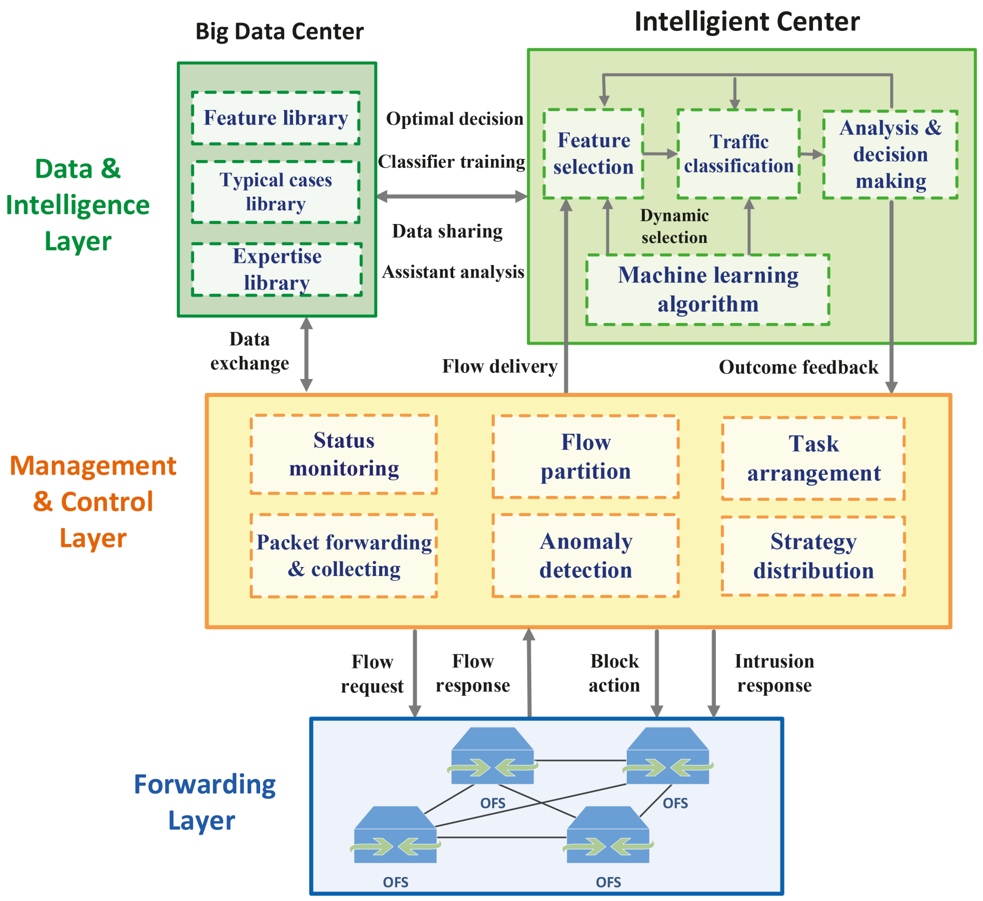

Fig. 1 *Machine learning-based IDS for software-defined 5G network*

framework (ProDefense). The system is capable of meeting application-specific DDoS attack mitigation requirements. Huang *et al.* [12] study the security-enforced packet processing problem in deep packet inspection-enabled SDN and designs a two-phase algorithm that can quickly select proxy and find routing paths for incoming flows. He *et al.* [13] propose a software-defined virtual traffic classification framework (vTC), a design of virtual NFs to flexibly select and apply the best suitable flow features and most effective machine learning classifiers with the help of an NFV controller which brings substantial flexibility and scalability. These proposed schemes are applicable to solve specific types of security problems and can perform well under certain scenarios.

As the focus of network security, network intrusion detection has gained intensive discussion and wide investigations in recent years [14]. Most of them depend on a set of manual rules, which are still unable to identify emerging diverse attacks intelligently. In terms of this, various machine learning algorithms have been adopted in traffic-based classification in solving such problems [15]. Le *et al.* [16] propose a network-based intrusion detection and prevention system relying on SDN approach which might reduce the cost and decrease the latency of the whole system. It performs C4.5 algorithm to build the decision tree for classifying the traffic. The proposed work in [17] focuses on identifying network anomalies efficiently under SDN environment using bat algorithm metaheuristic method for feature extraction and entropy-based decision tree for traffic classification. In [18], Kim *et al.* propose a three-layer recurrent neural network (NN) acting as a neural classifier for misuse detection. A deep NN model which is capable of automatically finding correlations between flow records and detecting network intrusions is built in [19]. The optimal hyper-parameter for the model including the number of hidden layers and hidden neurons need to be adjusted. However, current IDS systems monitor the network status through a single point isolated and cannot coordinate with each other. Besides, they also lack the combination of flow collection, classification and attack response along the whole process of intrusion detection and prevention. Therefore, we propose an intelligent IDS which flexibly integrates and coordinates security function modules to detect network intrusions adaptively and dynamically.

3 Architecture

The architecture of our proposed intelligent IDS is illustrated in Fig. 1. There are three layers: forwarding layer, management and control layer and data and intelligence layer. Forwarding layer consisting of open flow-controlled entities in 5G (OFs) is responsible for traffic monitoring and capturing. It can collect and upload network flows to the control layer, and block malicious flows according to the instructions of the controller. Management and control layer identifies suspicious flows and detects anomalies preliminarily using uploaded flow information. It also generates protection strategies according to decisions made by the intelligent layer and instructs forwarding layer. In data and intelligence layer, intelligent centre makes further analysis and judgement through feature selection and flow classification using adaptive machine learning algorithms.

3.1 Forwarding layer

This layer is in charge of forwarding packets between OFs. It provides management and control layer with real-time network status through collecting and uploading anomaly information from distributed OFSs. Besides, intrusions can also be blocked by OFSs through dropping malicious packets under the command of upper layers.

3.2 Management and control layer

3.2.1 Packet collecting and flow partitioning: This layer provides a more global view of the entire 5G network. The status monitoring module supervises network status and periodically requests packets from OFs. It collects packets uploaded by OFs for further analysis. Flow partition module processes and parses traffic statistics, clusters packets into flows and generates six-tuple flow IDs as follows:

```
Input: Flow features $Z_i$, $i = 1,2 \ldots M$
Output: The importance of each feature $D_i$
Procedure:
Begin
    For each feature $Z_i$ of instances in the training data set:
        For each tree $T_j$, $j = 1,2 \ldots N$ in the forest:
            a. apply the $T_j$ to classify OOB data and the number of
               of correctly classified data is marked as $C_1$;
            b. randomly disturb the OOB data set by permuting the value
               of $Z_i$ and count the votes of right class again marked as $C_2$;
            c. repeat steps a.b for all the trees and the average importance
               of each feature i is calculated using formula: $D_i = \frac{1}{N}\sum_{j=1}^{N}(C_1 - C_2)$
End
```

Fig. 2 *Importance measurement of flow features using RF*

$$\{srcip, dstip, srcport, dstport, duration, protocol\} \tag{1}$$

The flow IDs are used to define and label different flow records representing specific network connections and activities. The packet collection and inspection is performed at a regular time interval. The time interval is delicately selected in order to avoid undesirable delays for anomaly identification and scale packet overhead to an acceptable level.

3.2.2 Anomaly detection: Before exploiting elaborate intrusion detection in the intelligence layer, some basic flow statistics are used to roughly recognise abnormal behaviours and potential anomalies. The module applies entropy analysis [20] based on Shannon theory to detect distribution variations of packet sequences. The entropy of a random variable x is

$$H(x) = -\sum_{i=1}^{N} p(x_i)\log(p(x_i)) \tag{2}$$

where x_i is the value of x ranging from 1 to N. $p(x_i)$ calculates the possibility of x being x_i observed among all the feasible values. Here, we consider four basic characteristics: source address, source port, destination address, destination port as variables in the above (2) which are sketched from packets in every consecutive duration. Within the given period of time, we compute a new entropy $H(x)$ of each characteristic to detect anomalies in a subsequent way. If E stands for the mean entropy and S represents the corresponding standard deviation, there will be possible anomalies if $H(x)$ falls outside the duration between $(E - S)$ and $(E + S)$. The suspicious groups of flows are delivered to the intelligent layer for further analysis [15].

3.2.3 Task arrangement and strategy distribution: The modules respond to detected intrusions through managing and organising specific actions for defending attacks. They develop optimal strategies, arrange objective tasks and distribute them to OFs in forwarding layer. OFs are instructed to drop packets of malicious flows and protect the 5G network from being a further attack.

3.3 Data and intelligence layer

3.3.1 Feature selection: The feature selection module is designed to extract concerned features of sceptical flows and find an optimal subset of preferable features. They are used to precisely describe and discriminate a flow. The module can process high-dimensional data efficiently and remove irrelevant data, which improves the learning efficiency and predictive accuracy of flow classification. The selected features are considered optimal if they are closely correlated to the correct classification result while not redundant. In our system, various algorithms can be selected to measure the relevance and redundancy of features.

3.3.2 Traffic classification: The module classifies network flows by marking whether it belongs to specific types of attacks or benign traffic. The output of the classifier labels each flow as a certain class. Combination algorithms can be used to increase the accuracy of machine learning classification.

3.3.3 Analysis and decision making: According to the classification results, the module makes a comprehensive analysis of the real-time network status and determines whether the network is under attack or not. The analysis results can be a feedback to the former two modules and assist them in selecting algorithms adaptively. More importantly, they are delivered to the controller for tactics arrangement and defence.

3.3.4 Big data centre: As an auxiliary module in data and intelligence layer, big data centre maintains various bases of historical records and knowledge of intrusions to facilitate classifier training and decision making. It is comprised of libraries of typical flow features, user activity models and expertise advice. The data is persistently revised and updated.

4 Intelligent intrusion detection process

In our system, we employ selected machine learning methods in two critical steps of intrusion detection. First, we use RF to select an optimal subset of flow features through measuring variable importance. Afterwards, a hybrid clustering-based AdaBoost will classify traffic into different classes of attacks with selected features as input.

4.1 Random forest

RF [21] is a collection of uncorrelated structured decision trees deemed as forest. Those trees make classifying judgements independently and the final result will be the one gaining the majority votes. The concept of 'random' typically manifests in two aspects as follows:

- If the number of input training data is N, we take N samples randomly with replacement from the original data. The selected samples are likely to be repeated and will be used for growing the tree. Meanwhile, those data which has not been selected to build a tree is known as out-of-bag (OOB) data. The data is utilised to measure the classification accuracy of the forest.
- For each tree, we choose m ($m < M$, usually $m = \sqrt{M}$) features out of M-attribute entire set randomly as input variables without replacement. The value of m remains unchanged during the whole process of forest construction. While determining the best feature at each splitting node, we calculate Gini ratio [22] of m attributes and choose the one with the highest value to split the node. We will stop growing the tree when the selected attribute is the same as its father node.

The algorithm is a valid ensemble machine learning algorithm used for classification and regression. There is no need to prune each tree as it grows in case of over-fitting under those two restrictions. In particular, it can also be used to select features by ranking the significance of different features. The measuring of variable importance is based on classification accuracy of OOB data. The main steps are showed in Fig. 2.

We obtain the variable importance of the M features and select those with higher values in accordance with the pre-established number of features or retaining ratio. The importance of each feature is measured by the influences it exerts on the result of classification. A specific feature of higher importance usually degrades the OOB accuracy to a great extent.

4.2 Hybrid clustering-based AdaBoost

There are two steps in traffic classification. For the first stage, we make a preliminary judgement by adopting k-means++ to divide the traffic into two clusters which most probably represent the normal and abnormal instances. Later, we further partition the

```
Input: Data set with $N$ instances and each has $n$ dimensions, labeled as $x_{ik}$
    $i = 1,2...N$  $k = 1,2...n$
Output: K clusters with different instances after clustering the whole data set
Procedure:
Begin
    Step1: Choose a sample point $z$ randomly in the data sets as the first
        Initialization of clustering center.
    Step2: Calculate the shortest distance $D(x_i) = \sqrt{\sum_{k=1}^{n}(x_{ik} - z)^2}$ between the
        recently selected first cluster center and every other points in the data set.
    Step3: Select the next new center: the sample with a larger $D(x_i)$ gains larger
        probability to be selected.
    Step4: Repeat step2 to 3 until all the K centers have been chosen, afterwards we
        can perform the standard K means clustering algorithm.
End
```

Fig. 3 *Main steps of k-means++ algorithm*

Table 1 Distribution of data used in our evaluation

Class	Training dataset		Testing dataset	
	Number of samples	Percentage, %	Number of samples	Percentage, %
normal	97,278	19.69	60,593	19.48
probe	4107	0.83	4166	1.34
DoS	391,458	79.24	229,853	73.9
U2R	52	0.01	228	0.07
R2L	1126	0.23	16,189	5.2

anomaly clusters into four main classes of attacks using the ensemble algorithm AdaBoost.

4.2.1 k-Means++: As an unsupervised learning algorithm, the dominating object of applying *k*-means clustering method is to separate and group unlabelled traffic into normal and attack classes for coarse-grained classification. However, it is worth noting that the number of centroids *k* requires to be pre-determined and the random selection of initial centroids may result in locally optimal clustering. Hence, we introduce an improved *k*-means++ technique [23] aiming at choosing the optimal clustering centres. The fundamental principle we comply with is to make the distances between any initial clustering centres as far as possible. The process of initialling centroids is described in Fig. 3.

For step 3, the key point is that how to reflect the relationship between the distance variable $D(x_i)$ of one sample and its possibility of being selected. We aggregate the distance $D(x_i)$ of each point into a whole assemblage in sequence and compute the sum of them as $\text{Sum}[D(x_i)] = D(x_1) + D(x_2) + \cdots D(x_N)$. Then a random value λ is selected which is located in the range of 0 to $\text{Sum}[D(x_i)]$. We update the value of λ by computing $\lambda = \lambda - D(x_i)$ from $i = 1$ to N until it falls below zero. At this time, the section which λ drops in with the length $D(x_i)$ indicates the corresponding sample point *i* to be the next centroid. Now that the value of λ is stochastic, there is more chance that it drops in the section of a larger $D(x_i)$. In this paper, we set $K = 3$, considering that two of the four kinds of attacks [user-to-root (U2R), remote-to-local (R2L)] are easily confused with the normal flows.

4.2.2 Adaptive boosting: AdaBoost is a strong ensemble classifier linearly composed of different weak classifiers after trained by the same set of data [24].

The weight of each sample is same while initialising the training data. At each iteration, the data distribution is adaptively altered through changing the weights of samples. The weight of each sample which is mis-classified in the former basic classifier will be improved in the next round of training. In contrast, it will be reduced if it is correctly classified. In this way, more attention is laid on the samples hard to be properly classified to promote the overall performance.

All the weak classifiers after training are combined assigned with different weights of contribution and form a strong classifier. Thus, the final result is determined by votes of each basic classifiers with the distinct right of speech α. α is inversely proportional to the classification error rate *e*, which indicates that those weaker classifiers gaining a higher classification accuracy contribute more to the final result. Each weak classifier is regulated by factor α and the linear formation of them achieves a better result.

In most cases, each weak classifier is constructed by a one-layer decision stump, which splits once solely based on a single feature. It is worth noting that the feature used at each decision tree for decision making in classification is optimally chosen from *N* features. The feature used at each classifier is totally independent and can be reused again.

5 Experiment result

In this section, we conduct several experiments to evaluate our proposed system and selected algorithms.

5.1 Dataset

The KDD Cup 1999 dataset has been widely used to evaluate the performance of intrusion detection methodologies in recent years [19]. It contains ∼5,000,000 network connections in the training set and nearly 2,000,000 instances in the testing set. Each single connection vector consists of 41 features sorted into three classes: basic connection-based feature, content-based feature and traffic-based feature. Each traffic sample is labelled as either a normal flow or a malicious intrusion which exactly falls into four different categories in accordance with their own characteristics: DoS, R2L, U2R and probe. Since the amount of the original dataset is huge, we perform a five-class flow classification emulation using 10% of the whole KDD99 intrusion detection raw dataset. The distribution of both training and testing data marked by their attack type is summarised in Table 1.

Since there are redundant records in the original KDD99 data, it causes the learning algorithms to be biased toward some frequent records. To solve the issue, NSL-KDD [25], as an improvement, removes duplicate records and maintains a more balanced distribution of the dataset.

5.2 Evaluation metrics

Generally, the performance of the IDS is evaluated in the light of precision (P), recall (R), F-score (F), accuracy (AC) and false positive rate (FPR) calculated in the formulas below. We desire a system with higher detection rate as well as lower false rate.

Precision (P): The percentage of intrusion predicted that it truly existed

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

Recall (R): The number of correctly predicted intrusions versus all the presenting intrusions

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

F-score (F): Makes a tradeoff between the precision (P) and recall (R) to reach a better measurement of classification accuracy

$$F = \frac{2}{(1/P) + (1/R)} \tag{5}$$

Accuracy (AC): Manifests the flows exactly classified over the entire traffic traces

Table 2 Cost matrix

Class	Normal	Probe	DoS	U2R	R2L
normal	0	1	2	2	2
probe	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Table 3 Performance comparison using different number of features

Number of features	Precision, %	Recall, %	F-score, %	FPR, %	Time, s	Cost
23	94.48	92.62	91.02	0.54	110	0.241
41	93.60	92.06	90.03	0.54	149	0.257

Table 4 Performance comparison using different combinations of algorithms

Combination of algorithm		Number of features	Precision, %	Recall, %	F-score, %	FPR, %
RF	KA	23	94.48	92.62	91.02	0.54
RF	Gradient Boosting Decision Tree (GBDT)	23	93.09	91.21	89.37	2.84
RF	Decision Tree (DT)	23	92.65	91.78	90.01	3.31
RF	Support Vector Machine (SVM)	23	90.14	91.46	89.44	1.47
Tree	KA	23	93.34	91.90	89.99	0.64
Fisher	KA	10	93.25	91.72	89.79	1.91
ReliefF	KA	8	91.55	90.96	89.07	8.35

$$\mathrm{AC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{6}$$

FPR: Indicates the percentage of normal traffic which is misclassified as attacks

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{7}$$

where TP is the number of attacks precisely detected; TN is the number of normal traffic precisely classified; FP is the number of normal traffic incorrectly classified; and FN is the number of attacks unsuccessfully detected.

To our common sense, various intrusions generate different levels of consequences to the entire network. Hence, another comparative metric is defined to measure the cost damage of misclassification for different attacks per sample calculated as below:

$$\mathrm{cost} = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij} \times C_{ij} \tag{8}$$

where M_{ij} indicates the number of instances of class i misclassified in class j. C_{ij} is the cost value representing the penalty for each sample obtained from cost matrix [26] employed for KDD99 as shown in Table 2. Let N be the total number of samples for testing.

5.3 Performance analysis

We intend to evaluate the proposed system from three aspects. In Section 5.3.1, we assess the selected algorithms for feature selection and flow classification and the implementation of their combination. In Section 5.3.2, we verify the optimality and convergence of our system and make comparisons with existing solutions. In Section 5.3.3, we estimate the detection ability for different classes of flows when trained with various datasets.

5.3.1 Evaluations on selected algorithms: In Table 3, we verify the advance of RF algorithm for feature selection through the performance of detection using the sub-feature dataset in contrast with a full-feature dataset. It is encouragingly illustrated that the selected features contribute more to differentiate attack traffic giving rise to higher accuracy and lower false rate. As noted from the table, the selected features achieve a slight reduction of operating time as well as lessening the computational cost without degrading the overall performance. It is proved that the selected algorithm for optimal feature selection performs well.

We evaluate the combination of our selected algorithms (i.e. RF and a hybrid clustering-based AdaBoost) comparing with several groups of traditional feature selection and machine learning algorithms in Table 4. Since we know that the selection of algorithms for feature selection and traffic classification have a mutual influence on each other, we care more about the performance of the combination of them. As we can see, it is obvious that the selected methods generate a better performance among all the combination alternatives in every metric.

5.3.2 Evaluations on the performance of the proposed system: We measure the overhead that the proposed system occurs using the above-mentioned concept cost. We compare our system with [20] in Fig. 4. The metric indicates that the more misclassified flows there are when making a classification, the more overhead the system generates for misclassification. As the number of flows grows, both systems become more well-trained to make classification with fewer faults. It can be seen that our system produces more overhead initially than [20]. However, as the training data increases, our system produces less overhead than its comparison later. We can see that our system can produce sufficiently low overhead when trained with enough amounts of data.

There have been several intelligent architectures proposed to detect and prevent network intrusions under SDN environment [27, 28]. We make a comparison with previous researchers in terms of classification accuracy and error rate as shown in Table 5. The processing time of each system using a portion of flows in the dataset is also illustrated in Fig. 5 to evaluate the efficiency of systems. It can be noted that the proposed system improves the classification accuracy without sacrificing much time complexity. This is attributed to the preliminary clustering technique helping to group data into normal and attack classes in preparation for fine-grained classification, which consumes more time than a single algorithm. Also, the need of building pre-defined NTs in the ensemble algorithm AdaBoost also increases the overall processing time.

Except for the above evaluations on the performance of the system, the convergence of the detection process using clustering-

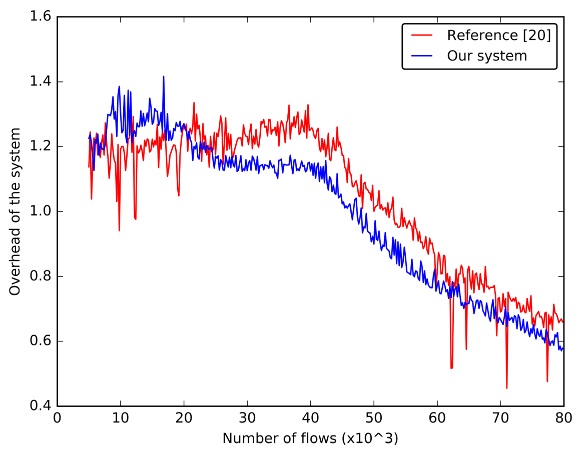

Fig. 4 *Overhead produced by different systems with different numbers of flows*

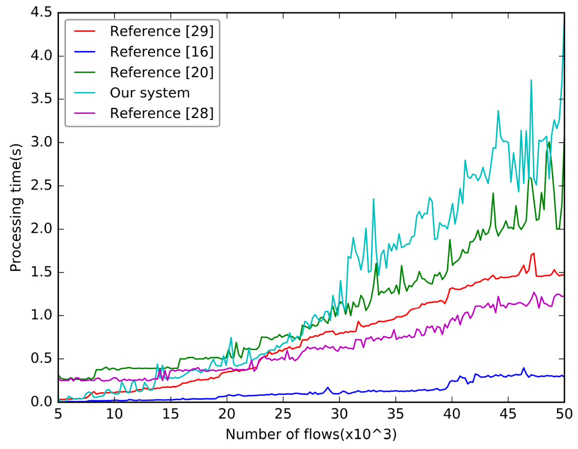

Fig. 5 *Processing times of different systems*

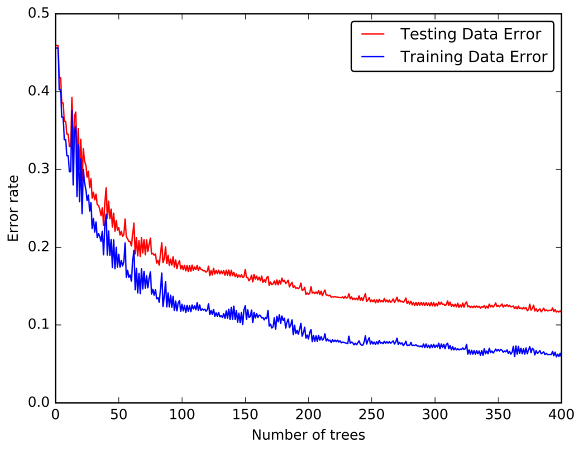

Fig. 6 *Error rate with the different NTs*

based AdaBoost for flow classification is another aspect that we consider. We define the convergence of the algorithm as the effect of the increase in iterations on the detection accuracy. In the process of traffic classification in our system, we use a hybrid clustering-based AdaBoost by combining the *k*-means++ and AdaBoost to detect anomaly network flows. Since the implementation of *k*-means++ only makes a preliminary preparation for the process of classification, we mainly consider the convergence of AdaBoost in our paper. In [29], it has been proved that AdaBoost iteratively converges to the minimum of the exponential loss. It shows upper and lower bounds as well as their conjectured tighter bounds for convergence rates to a given target loss, which are achieved by some finite combination of the weak hypotheses. So, we have been no longer discussing the convergence problem with a limit to the length of our paper.

Table 5 Performance comparison of different systems

Proposed system and related works	Accuracy, %	FPR, %
our system	92.62	0.54
[20]	92.16	0.52
[28]	91.68	0.71
[27]	90.82	0.82
[16]	88.64	1.45

Table 6 Accuracy (%) on the NSL-KDD dataset compared with the original KDD99 dataset

Classes of flows	KDD99 dataset	NSL-KDD dataset
normal	99.44	98.89
probe	91.79	89.61
Dos	99.95	92.56
U2R	64.29	80.21
R2L	80.68	86.43

Since the final result of AdaBoost is the weighted combination of each basic classifier, it is crucial to figure out the effect of the number of basic classifiers on the classification accuracy. We try to prune the basic parameter of AdaBoost by building the ensemble classifier with different numbers of basic classifiers. It means the number of trees (NTs) built as basic classifiers. NTs and the error rate of classification using the training and testing dataset, respectively, are plotted in Fig. 6. It is shown that the error rate decreases as NTs become larger. It indicates that the larger number of weak classifiers can compensate for each other and achieve a better total performance. After NTs increase to a certain limit, the error rate remains almost unchanged. As we know that the more numbers of trees it needs to build, the longer of time the algorithm needs to classify flows. We strike a balance between the detection performance and time consumption and choose NT = 200 as the desirable value in the following evaluation experiments.

5.3.3 Evaluations on the detection ability of the proposed system using different datasets: In Table 6, we compare the evolution performance of our system by applying the testing dataset of the KDD99 dataset and NSL-KDD dataset. We can see from the table that the classification accuracy for rare classes (U2R and R2L attacks) of NSL-KDD are significantly higher than the original KDD99, whereas slightly lower in the majority classes (DoS and probe attacks). We summarise that detection capability of the system can be evaluated in a more comprehensive way when using NSL-KDD dataset without biased to redundant records.

We observe the detection performance of our system using different distributions of datasets and the result is demonstrated in Fig. 7. In a real network, some intrusions generate more connections than others which lead to an extremely unbalanced dataset for classification. Thus, we resolve the problem through downsampling the majority of intrusions (normal and DoS) as well as oversampling the minority intrusions (U2R and R2L). In Table 7, it gives an overview of percentages of samples classified into five classes and the distributions show extreme variations between the two datasets. We make a comparison of detection rate between the ordinary dataset and the balanced dataset using the same default parameters by splitting 40% of the complete set as testing data. It is apparent that the sampling technique improving the detection accuracy of minority intrusions dramatically while maintains a reasonable detection rate of the majority ones. The result indicates that pre-processing the input data into uniform distribution upgrades the detection of minority intrusions which elevates the overall performance of our system.

Finally, we evaluate the detection ability for unknown attacks through cross-validation (CV) [30] by splitting 90% of the dataset for training while the rest 10% for testing in Table 8. It is clearly depicted that classifications with CV behave well by significantly boosting the accuracy of detecting U2R and R2L attacks. The rates of other categories are gently promoted as well. To our knowledge, the four main types of attacks mentioned above are subdivided into

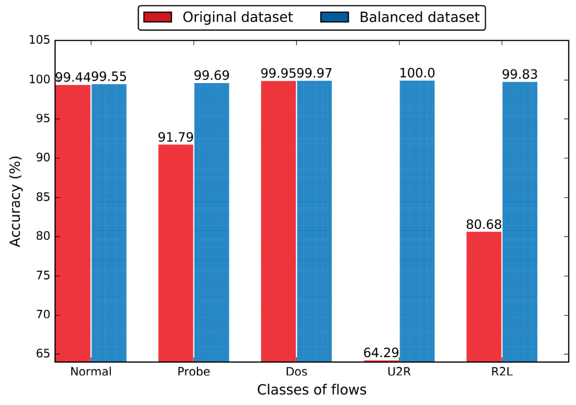

Fig. 7 *Accuracy on the balanced dataset compared with the original dataset*

Table 7 Classification accuracy (%) comparison between normal process and processes with CV in each type of attack

	Normal	Probe	DoS	U2R	R2L
normal process without CV	99.46	73.89	97.36	0.88	5.8
training data with CV	99.9	97.04	97.04	12.5	88.46
testing data with CV	98.54	97.96	99.97	68	65.5

Table 8 Percentage (%) of flows of each class in different datasets

Classes of flows	Original dataset	Balanced dataset
normal	19.35	19.79
probe	0.81	20.29
Dos	79.51	20.96
U2R	0.04	19.85
R2L	0.28	19.08

39 small classes, 17 of which only appear in the model we come up with testing data. So, it shows that our proposed system lacks a generalisation ability to detect various attacks without previous training.

6 Conclusion

This paper presents an intelligent IDS based on software-defined 5G architecture using machine learning algorithms. It is implemented under software-defined environment which facilitates status monitoring as well as traffic capturing under a global view. It integrates and coordinates security function modules under centralised management and applies machine learning algorithms to detect intrusions intelligently. We use RF to select a subset of typical traffic features and classify network flows by combining *k*-means++ and AdaBoost algorithms. Evaluation results validate the optimality of our proposed system in achieving higher accuracy and lower overhead without much time consumption. The experiments also reveal that the system improves its detection ability under training well with the balanced dataset. The combination of selected algorithms is also proved to be effective compared with existing solutions.

In the future, we intend to find the intrinsic relations between features as well as classifiers and adaptively choose the best combination of learning approaches. Besides, we will spare no efforts to further find the suitable way to select the regular time interval for packet collection and inspection.

7 Acknowledgments

This work was supported in part by the Program for Zhejiang Leading Team of Science and Technology Innovation (no. 2013TD20), the National Natural Science Foundation of China (nos. 61731002 and 61701439), the Zhejiang Key Research and Development Plan (no. 2018C03056), the National Postdoctoral Program for Innovative Talents of China (no. BX201600133) and the Project funded by China Postdoctoral Science Foundation (no. 2017M610369).

8 References

[1] Casoni, M., Grazia, C.A., Klapez, M.: 'A software-defined 5G cellular network with links virtually pooled for public safety operators', *Trans. Emerging Telecommun. Technol.*, 2016, **28**, (3)

[2] Kim, H., Feamster, N.: 'Improving network management with software defined networking', *IEEE Commun. Mag.*, 2013, **51**, pp. 114–119

[3] Mijumbi, R., Serrat, J., Gorricho, J.-L.: 'Network function virtualization: state-of-the-art and research challenges', *IEEE Commun. Surv. Tutor.*, 2015, **18**, pp. 236–262

[4] Li, R., Zhao, Z., Zhou, X.: 'Intelligent 5G: when cellular networks meet artificial intelligence', *IEEE Wirel. Commun.*, 2017, **PP**, pp. 2–10

[5] Buczak, A.L., Guven, E.: 'A survey of data mining and machine learning methods for cyber security intrusion detection', *IEEE Commun. Surv. Tutor.*, 2015, **18**, pp. 1153–1176

[6] Zhang, Y., Chen, M., Lai, R.X.: 'Cloudified and software defined 5G networks: architecture, solutions, and emerging applications', *Mob. Netw. Appl.*, 2016, **21**, pp. 727–728

[7] Akyildiz, I.F., Wang, P., Lin, S.C.: 'SoftAir: a software defined networking architecture for 5G wireless systems', *Comput. Netw.*, 2015, **85**, pp. 1–18

[8] Ge, X., Li, Z., Li, S.: '5G software defined vehicular networks', *Comput. Netw.*, 2017, **55**, pp. 87–93

[9] Rawat, D.B., Reddy, S.R.: 'Software defined networking architecture, security and energy efficiency: a survey', *IEEE Commun. Surv. Tutor.*, 2017, **19**, pp. 325–346

[10] Wang, P., Lin, S.C., Luo, M.: 'A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs'. IEEE Int. Conf. Services Computing, 2016, pp. 760–765

[11] Bawany, N.Z., Shamsi, J.A., Salah, K.: 'DDoS attack detection and mitigation using SDN: methods, practices, and solutions', *Arab. J. Sci. Eng.*, 2017, **42**, pp. 1–17

[12] Huang, H., Li, P., Guo, S.: 'Traffic scheduling for deep packet inspection in software-defined networks', *Concurrency Comput. Pract. Exp.*, 2016, **29**, pp. 16

[13] He, L., Xu, C., Luo, Y.: 'vTC: machine learning based traffic classification as a virtual network function'. ACM Int. Workshop on Security in Software Defined Networks and Network Function Virtualization, 2016, pp. 53–56

[14] Farah, N., Avishek, M., Muhammad, F.: 'Application of machine learning approaches in intrusion detection system: a survey', *Int. J. Adv. Res. Artif. Intell.*, 2015, **4**, (3)

[15] da Silva, A.S., Wickboldt, J.A., Granville, L.Z.: 'ATLANTIC: a framework for anomaly traffic detection, classification, and mitigation in SDN'. Network Operations and Management Symp., 2015, pp. 27–35

[16] Le, A., Dinh, P., Le, H., *et al.*: 'Flexible network-based intrusion detection and prevention system on software-defined networks'. Int. Conf. Advanced Computing and Applications, 2017, vol. **19**, pp. 325–346

[17] Sathya, R., Thangarajan, R.: ‘Efficient anomaly detection and mitigation in software defined networking environment’. Int. Conf. Electronics and Communication Systems, 2015, pp. 479–484

[18] Kim, J., Kim, J., Thu, H.L.T.: ‘Long short term memory recurrent neural network classifier for intrusion detection’. Int. Conf. Platform Technology and Service, 2016, pp. 1–5

[19] Tang, T.A., Mhamdi, L., Mclernon, D., *et al.*: ‘Deep learning approach for network intrusion detection in software defined networking’. The Int. Conf. Wireless Networks and Mobile Communications, 2016

[20] Nychis, G., Sekar, V., Andersen, D.G.: ‘An empirical evaluation of entropy-based traffic anomaly detection’. Internet Measurement Conf., 2008, pp. 151–156

[21] Singh, K., Guntuku, S.C., Thakur, A.: ‘Big data analytics framework for peer-to-peer botnet detection using random forest’, *Inf. Sci.*, 2014, **278**, pp. 488–497

[22] Rutkowski, L., Jaworski, M., Pietruczuk, L.: ‘The CART decision tree for mining data streams’, *Inf. Sci. Int. J.*, 2014, **266**, pp. 1–15

[23] Arthur, D., Vassilvitskii, S.: ‘*k*-Means++: the advantages of careful seeding’. 18th ACN-SIAM Symp. Discrete Algorithms, 2007, pp. 1027–1035

[24] Zhu, J., Zou, H., Rosset, S.: ‘Multi-class AdaBoost’, *Stat. Interface*, 2009, **2**, pp. 349–360

[25] Revathi, S., Malathi, A.: ‘A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection’, *Int. J. Eng. Res. Technol.*, 2013

[26] Elkan, C.: ‘Results of the KDD'99 classifier learning’, *ACM Sigkdd Explor. Newsl.*, 2000, **1**, pp. 63–64

[27] Wang, P., Chao, K.M., Lin, H.C., *et al.*: ‘An efficient flow control approach for SDN-based network threat detection and migration using support vector machine’. IEEE Int. Conf. E-Business Engineering, 2017, pp. 56–63

[28] Ye, X., Chen, X., Wang, H., *et al.*: ‘An anomalous behavior detection model in cloud computing’, *Tsinghua Sci. Technol.*, 2016, **21**, pp. 322–332

[29] Mukherjee, I., Rudin, C., Schapire, R.E.: ‘The rate of convergence of AdaBoost’, *J. Mach. Learn. Res.*, 2011, **14**, pp. 2315–2347

[30] Kohavi, R.: ‘A study of cross-validation and bootstrap for accuracy estimation and model selection’. Int. Joint Conf. Artificial Intelligence, 1995, pp. 1137–1143