WebAD^2: A Cascading Model Based on Machine Learning for Web Attacks Detection

Ying Lin(✉) and Bo Li

School of Computer Science and Engineering,
Beihang University, Beijing, China
{linying,libo}@act.buaa.edu.cn

Abstract. Anomalies in network are complicated and fast-changing, which pose serious threats to network security. In an intrusion detection system (IDS), achieving high detection rate and low false alarm rate is an essential requirement. Furthermore, faced with the explosive growth of network data, rapid recognition counts for as much as accuracy. In this paper, we propose a two-stage cascading model, named WebAD^2, for detecting web attacks. WebAD^2 applies machine learning techniques to detect anomalous behaviors. However, unlike traditional approaches, WebAD^2 divided machine learning process into two stages. In the first stage, partial but key features are selected for training and detecting to accelerate the detection speed. The intermediate results are passed to the second stage and all features are applied to refine the detection results, therefore improve the accuracy of the model. We conduct comprehensive experiments to evaluate the effectiveness and efficiency of WebAD^2. The results show that WebAD^2 could significantly improve the model efficiency without sacrificing the detection accuracy. The processing speed is reduced up to more than 70% on average, with an accuracy decrease less than 1%. What's more, the performance results on NSL-KDD also verify that WebAD^2 could be universal to detect network flow traffics.

Keywords: Web attack · Anomaly detection · Machine learning
Cascading model · URI analysis

1 Introduction

With the prevalence of web applications, web attacks increase with a tremendous speed and has become the top threat of Internet [1,2]. To counteract web attacks, Intrusion Detection Systems equipped with web analyzing functions emerge and play a more and more significant role in network security [3–6]. Intrusion detection systems search for malicious activities or policy violations by monitoring network traffic or host activities. In general, intrusion detection techniques could be divided into two types: misuse detection and anomaly detection. Misuse detection requires keeping the dictionary of attacks up to date, and are totally blind to zero day attacks. Anomaly detection could identify unknown attacks, however suffer from higher false positives rate compared with misuse detection techniques [7,8]. Recently, with the development and maturity of machine learning approaches, anomaly detection attracts great attention from academy and gains more and more application in industry.

With the explosion of network data, nowadays intrusion detection systems are facing with huger challenge than five years ago. An effective IDS should be capable of detecting anomaly with high accuracy and keep up with the continuously changing of network traffic. Real-time monitoring is of essential importance for IDS [9]. Simple anomaly might snowball into substantial harm because of the complex structure of network. Most available researches in anomaly detection are based on previous offline benchmark datasets which are old and out of date. Many approaches put great emphasis on algorithm optimization to achieve higher accuracy regardless of the expenses of space and time complexities. The reality is that these approaches could not be directly put into use because of bad performance especially when dealing with massive network traffic. One should keep in mind that detecting malicious behaviors efficiently and precisely is an essential function for Intrusion Detection Systems.

Another problem for anomaly detection is the lack of a representative accessible network traffic dataset due to the variety of networks, traffic profiles and attack types. Many researches still use the darpa'98 and kdd'99 cup as the benchmark datasets, which suffer from the problems discussed by McHugh [10]. And they may not be a perfect representative of existing real networks.

To address the above issues, we propose a two-stage cascading model, WebAD^2, to detect anomalies in web logs. WebAD^2 relies on machine-learning based classifiers to differentiate anomaly web traffics from normal traffics. Through our previous work on URI analysis [11] we found that it is possible to identify the majority of anomaly behaviors just using several high-quality features. Inspired by this observation, we design a cascading model to balance the efficiency and accuracy in the detection phase. We used two benchmarking datasets to evaluate WebAD^2. One is 440 GB Web Log data from CNCERT, which contains more than 11 attack types and 2,000,000 samples. Another is NSL-KDD, a dataset suggested to solve some of the inherent problems of kdd'99. Our main contributions are summarized as follow.

1. A two-stage cascading model is designed to balance the detection accuracy and efficiency especially when facing with massive network traffic. Partial features are selected in the first stage to reduce detection time. The intermediate results are passed to the second stage and all features are used to refine the detection results and improve the accuracy of our model. We choose the confidence score as a indicator to determine whether the web traffic data handled in the first stage should be passed to the second stage for further inspection or not.
2. These features using in first stage should be self-tuned to the actual conditions of different dataset. We also present a formula to compute the balance score between time and accuracy and select the best combination.

3. Experiments are conducted to evaluate how detection accuracy and time consumption changes with the varying of feature counts. The results convince that it is reasonable to detect most anomaly traffics with several features in a short time. These high cost-efficient features are selected as the seed features in the first stage of our model.
4. We evaluate WebAD^2 through comprehensive experiments on two real-life datasets: CNCERT web logs and NSL-KDD. By comparing the results of several classifiers, we found that our two-stage model can improve the modeling efficiency without sacrificing the accuracy.

The remainder paper is organized as follows. Section 2 presents the related work in the domain of anomaly detection and URL analysis. In Sect. 3, we explain the features processing approaches and algorithms. Section 4 describes the details of detection model and the approach to select high cost-efficient features. Then the experiment results of model in CNCERT web logs and NSL-KDD are illustrated in Sect. 5, and contrasted with the results of basic model. Section 6 concludes this paper and outlines future work.

2 Related Work

A Internet security report in Q1 2017 from [12] Akamai Technologies[1] reveals that SQLI, LFI (Local File Include), and XSS accounted for 93% of observed web application attacks. And Web application attacks increased nearly 35% compared with Q1 2016. That indicates the malicious attackers will not budge, and the number of Web attack will continue to grow.

Misuse detection and anomaly detection are two widely-used techniques in nowadays intrusion detection systems [3,4]. The core technique in misuse detection is pattern matching, which identifies "bad guys" by matching current records with known attack patterns. However, misuse detection suffers from the invisibility of unknown attacks such as 0 day vulnerabilities. Anomaly detection approaches is intrinsically different from misuse detection. A normal behavior model is established by learning upon benign network traffic. In the detection phase, the normal model is used as a baseline to identify outliers. With the renaissance of artificial intelligence and machine learning, anomaly detection became one of the hottest research area in network security. Generally speaking, anomaly detection could be classified into three categories: statistics-based methods, data mining-based methods and machine learning-based methods [13,14].

Statistics-based methods aim at discovering distribution information through statistical techniques. Some research works focus on the statistical analysis in URI, G.V. [15,16]. Gaussian distribution and Markov model are applied to analyze attribute length, attribute character distribution, structural inference, token finder, attribute presence or absence and attribute order. However, the computation only depending on the weight will lead to low detection rate. Study in

[1] An American content delivery network (CDN) and cloud services provider.

[4] is based on the analysis of URL entropy. Split URL strings into tokens with delimiters and calculate its entropy. Entropy can be viewed as a summary of a string. The key is that normal URL is much simpler than abnormal, which can find some complicated anomalies but also may neglect some anomaly whose entropy is small.

Data mining-based methods have various kinds. Research [17] achieves high detection rate relying on a combination of association rule and fuzzy set theory. [18] recognizes traffics using anomalous entropy to reduce false alarm rate with random forest; [19] applies KNN, Bayes Network and Random Forest to classify traffic, and clustering was employed to recognize the unknown applications. Methods in [20,21] aggregate the traffic flow by density-based clustering and sub-space clustering, and detect anomaly by ranking.

Machine learning-based methods mainly consist of SVM, Markov model, PCA, etc. Study in [22] relies on SVM and random forest to improve classification accuracy and reduce the runtime. [23] achieves low false alarm rate with the kernel function of Gaussian Radial Basis Function. Fan [24] proposes an ensemble approach which can effectively identify web-based attacks using hidden Markov models with different parameters. [25] applies PCA for traffic anomaly detection to improve precision, which is sensitive to noise though.

Feature selection technique is also the important research field of anomaly detection. Iglesias and Zseby [26] propose a multi-stage feature selection method using filters and stepwise for network traffic based anomaly detection. [27] has focused on many existing feature selection techniques to remove irrelevant features from NSL-KDD dataset to develop a robust classifier that will be computationally efficient and effective. Feature selection aims to increase the efficiency of machine learning. WebAD^2 could further improves the efficiency after feature selection.

Most existing anomaly detection algorithms focus on improving prediction ability. Nevertheless, in the face of the challenge of a large amount of data to predict, computing time and space consumption are important factor to be optimized. Consequently, in this paper, we apply a cascading model to improve the modeling efficiency without sacrificing the accuracy. Many previous algorithms are employed and get a great promotion in our model. And this model could be employed to improve the performance of existing system such as distributed system or parallel system. They have the same goal to accelerate processing without any conflict.

3 Features and Preprocessing

3.1 Data Model

Feature extraction and feature selection can eliminate strong correlated, redundant and irrelevant features to reduce data redundancy and storage consumption. It also provides a superb way for better and easier understanding of dataset. Many investigators have explored on feature selection. For better performance, we take advantage of previous studies to select the feature of our dataset of web log. Many analyses about web logs regard URL or URI as an important property. Owning to its universality and intelligibility in log records, the most of features we used are based on the URL or URI. A method of token is employed to deal with URI without any query in this paper. We classify all features into 4 categories: Number-related Features, Length-related Features, Character -related Features, Structure-based Features, as Table 1 shown. The main features will be introduced later.

Table 1. All features

ID	Name	Type	Introduction
Number-related features			
1	Num_digit	Integer	Number of digit in URI string
2	Num_letter	Integer	Number of letter in URI string
3	Num_punctuation	Integer	number of punctuation in URI string
4	Num_token	Integer	Number of token in URI string
5	Num_parameter	Integer	Number of parameter in URI string
Length-related features			
6	Length_URI	Integer	Length of URL
7	Length_max_token	Integer	The max length of token
8	Length_min_token	Integer	The min length of token
9	Length_min_parameter	Integer	The min length of parameter
10	Length_max_parameter	Integer	The max length of parameter
11	Length_cookie	Integer	Length of cookie
12	Length_post	Integer	Length of post
13	Length_referer	Integer	Length of referer
Character-related features			
14	Character_cookie	Integer	Anomaly probability in cookie character
15	Character_parameter	Integer	Anomaly probability in parameter character
16	Character_URI	Integer	Anomaly probability in URI character
17	Character_post	Integer	Anomaly probability in post character
Structure-based Features			
18	Relative entropy	Float	Relative entropy of URI string
19	Depth	Integer	The URI path depth
20	Token	Integer	Anomaly probability of token

Each record in the Web Log are processed and converted into a associated vector $\mathbf{V} = [\mathbf{v_1}, \mathbf{v_2}, ...; \mathbf{v_k}]$, where v_i represents the value of i^{th} feature, and k is the total number of features we used. The classifiers of machine learning are applied to categorize traffic. Records are sperated into two categories: normal data are labeled as 0, while abnormal data are labeled as 1. Relative values of several features are depicted in Fig. 1. For simplicity of exposition, all values are standardized between $0-1$. It's clear to see that, the Depth, Digit, Letter, Token of attack are higher than access, while Relative Entropy is contrary. The details of processing for some important feature are described as follow.

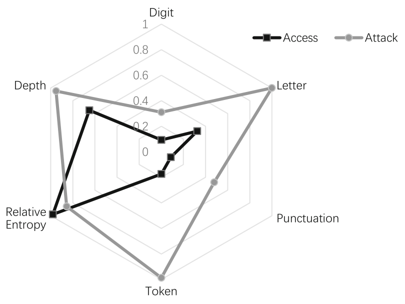


Fig. 1. Rader map of relative value of features

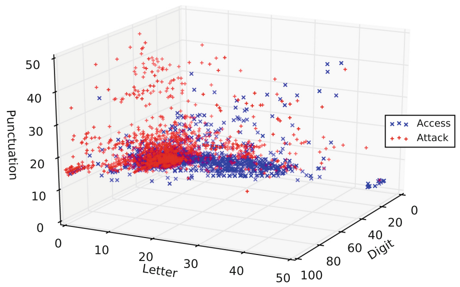


Fig. 2. The frequency of letter, digit, punctuation in URI

3.2 URI Token

URI is regarded as an important feature in many previous studies on HTTP web logs, and they usually use the parameters of URI query. However, there are some URIs without query in our dataset. We benefit from the method that Naive Bayes handles the problem of spam emails and it considers each word in path is equal. In this paper, every URI path is divided into some tokens. Naive Bayes are employed to calculate the abnormal probability of a URI path. Finally simplify the probability value to 0 or 1. During the actual operation, the URI path sring is segmented by delimiters. For example, URI "www.example/show_shiji/id/46485.php" is convert to token set [*'www'*, *'example'*, *'show'*, *'shiji'*, *'id'*, *'php'*]. For easier analysis, pure digital and single letter are ignored.

Statistical analysis of tokens frequency contributes to anomaly detection. For the frequency of these tokens appears differently in normal activities and abnormal. We can observe in Fig. 1, the anomaly value of *Token* is higher than normal. For example, in SQL injection, the "select" and "from" is easy to observe. We can simply regard an URI string as a Token set. In many existing researches, features in language are expressed by N-gram, usually "1-Gramm" and "2-Gramm", using Markov model to describe the structure information with higher prediction ability. However, the structure information displayed in the URI is not really obvious. We use Naive Bayes method to determine the abnormal or normal probability of a URI string, which is faster than Markov model in computation, and shows better prediction performance also.

Every token is regarded as independent in Naive Bayes. The particular arithmetics are described as follow.

1. Split URI string to get the Token list by recognizing delimiter, such as []=?@|\${}.
2. Filter out the pure individual letters and numbers of token. We can get *Num_token* in this step also.
3. Union these token lists to a token set.
4. According to the labeled data, calculate the prior probability $P(Y)$ and the conditional probability $P(X_i|Y)$ of every token in normal and abnormal respectively.
5. Calculate the normal and abnormal posteriori probability of every URI path string as:

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(X)}. \tag{1}$$

6. Simplify the normal and abnormal probability of every path string as the values of feature *Token*. $Ft = 1$ means that the URI is more possibly abnormal than normal:

$$Ft = \begin{cases} 0, & \text{if } P(Y=0|X) > P(Y=1|X) \\ 1, & \text{if } P(Y=0|X) \leq P(Y=1|X) \end{cases}. \tag{2}$$

3.3 Relative Entropy

Entropy is quoted in statistics and information theory discipline to represent the uncertainty of an event. Threepak and Watcharapupong [4] inspect web attacking scripts usually have more sophisticated request patterns than legitimate ones. Web Log file is one of the important sources to obtain evidence of the attack. And in URI, web attack requests may contain more repeated contents than normal requests.

We imitate the same splitting method as URI token for simplification, and calculate the entropy of URI path string referring to existing studies [4,18]:

$$E_t = \frac{1}{\lambda}\sum_{i=1}^{N} p_i\left[\log(\lambda) - \log(p_i)\right] \tag{3}$$

$$E_{\max} = \frac{1}{\lambda}\sum_{i=1}^{\lambda} 1\times\left[\log(\lambda) - \log(1)\right] = \log(\lambda) \tag{4}$$

$$E_{rel} = \frac{E_t}{E_{\max}}. \tag{5}$$

where an URI, contains λ tokens with n distinct ones ($n < \lambda$), p_i ($i = 1$ to n) denotes the frequency that the i^{th} word appears. The relative entropy (E_{rel}) is formulated for simplification and normalization in formula (5). The maximum entropy ($E_{\max}$), is the entropy of URL with all tokens occur only once, formulated in formula (4).

In general, abnormal attacks are more sophisticated than the normal requests. The farther to 1 the relative entropy value is, the more sophisticated the URL request is. This method eliminates some complicated unusual URL, but it does not work on these anomalies with high entropy.

3.4 Character

According to the observation, the character distribution varies in different websites. Nevertheless, characters appear in normal activities are generally steady, mostly human-readable and only printable. A normal character set could be built by statistics of characters in normal URI strings. There are also some attack character groups like '..', '.fg./' only in abnormal URI request instead of normal. The value of *Character_URI* is to present the probability of anomaly with character. The Character of normal URI request is 0 and abnormal is 1 respectively.

If there are characters of an URI path outside the normal character set, the value is 1, otherwise is 0. The method can also be applied to detect the anomaly of other string as well as URI request. Details of method are described as followed.

1. Learning phase: we aim to get the normal character set C in this stage. C represents initialized an empty set. For every URI string in normal URI, $C = C \cup S_i$, where S_i denotes all characters of i^{th} URI string, $i = 1$ to M and M denotes the number of normal activities;
2. Predicting phase: for every URI string, $\forall c \in S_i$, if $c \in C$ the probability value of URI character is 1; else, is 0. If there are characters like ' ./ ', ' /. ', ' .. ', ' ** ', ' */ ', ' select% ', '<script>', the value of URI character is 1;

It is interesting to notice that the approach for character also can be adopted to detect the anomaly of other string, such as *Character_cookie*, *Character_parameter*, *Character_post* in Table 3, not only URI path string.

3.5 Digit, Letter and Punctuation

There are some rules and syntaxes in natural language. The frequency of the characters is different in English such as "E" is the most common. So we have a hypothesis that this phenomenon is also possible in the distribution of URI string. We analyze the statistical result of the normal and abnormal character frequency and find that the attack and normal access frequency distribution are different in characters. For clear understanding and easy computation, all characters are classified into three categories: digit, letter and punctuation. As Fig. 1 showing, the relative frequencies of digit, letter and punctuation are distinct in normal and abnormal. Because of the complexity of anomaly, some attack contents could be contained in the URI quest. What's more, there may be more specific punctuations in abnormal attack. Such as one attack URI quest:

/article/youbianjc/you?seq=../../../../../../../../etc/passw

The frequencies of punctuation '/' and '.' are higher than normal quest. As Fig. 2 illustrates, there are more punctuations and digits in anomaly than normal access, so we add *Num_digit*, *Num_letter*, *Num_punctuation* to the feature set, whose contributions are verified positive in the experiments.

3.6 Depth

The URI path depth of a normal site is usually 3 or so commonly. Because the deeper link is, the more difficult it is for user to obtain the needed information quickly, and for search engine to crawl the web. We simply identify '/' to determine the depth, and analyze its distribution. The depth of normal access are concentrated in smaller value than anomaly, which is painted in Fig. 1. For example, there are many of '/' in an anomaly URI request:

/fgs/index.php?s=/article/show/id/{*$*{*@phpinfo()*}}

4 Cascading Model Based on Machine Learning

This paper aims to maintain high accuracy as well as efficiency. The key is to build a two-stage model, which could complete anomaly detection work in two phases efficiently and effectively.

4.1 Motivation

Under the great pressure of massive data, it makes no sense to pursue only on high accuracy and ignore the time consumption, especially in the circumstance of real-time detection. Hence to minimize the time in a acceptable range of accuracy is one of the important studies in anomaly detection. While now many researches focus on the improvement in accuracy of algorithm, ignoring the time consumption. We conduct the optimization in the detection model instead algorithm.

Different features have different contributions to the classifier. Feature selection and feature Extraction can reduce the number of features, which are an important for machine learning to improve efficiency and avoid curse of dimensionality. Removing irrelevant or redundant features or components brings obvious benefits in terms of computational resources, such as reducing resource consumption for processing, storing and transmitting data, improving the prediction performance of the classifiers, and providing a better understanding of data. After Feature selection, for the same objective, we further assume that using several features with high contribution could detect most of the anomalies in a shorter time than using all of features. And for feature Extraction, such like PCA, WebAD^2 can also used to speed up the performance. The ranking of eigenvalues is of significance to select the best components for quick recognition in the first step, which could enhance the time performance of PCA. A statistics experiment is conducted on our dataset of web log. Figure 3 reveals the variation of accuracy and time on average in our web log dataset. It is observed that with the increase of feature counts, the growth of accuracy levels off gradually, but the increase of time is still clearly. When feature numbers go over 4, the accuracy starts to be in slow growth.

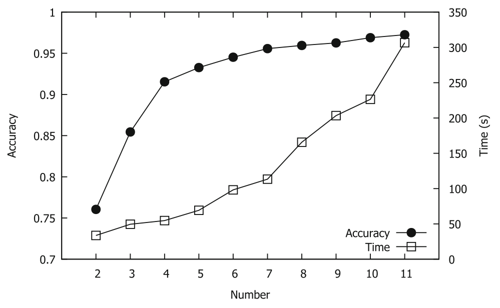


Fig. 3. Variation of accuracy and time with feature number increasing

4.2 Cascadinng Model

We draw on cascading model to hierarchically settle the conflict of efficiency and accuracy. The first stage of WebAD^2 is designed to reduce the cost of time and space. Several high cost-efficient features are picked out according to the practical situation. The second stage aims at increasing the accuracy, trained with all features to ensure the performance of detection. Figure 4 illustrates the structure of two-stage model. As can be seen from the left half, during the process of training, all labeled samples in learning set are preprocessed into the feature vectors. The first model adopts the feature vectors with several cost-efficient features, while the second employs all features to refine the result. The right half displays predicting process. Samples in the testing set are preprocessed into an associated vector only involving partial cost-efficient features. Then classifiers in the first model determine the category in the light of the confidence score. These samples exceeding the threshold θ of confidence score are identified by the first stage. The others whose confidence score are dissatisfactory stream down to second stage, and determined the category with all features. Normal data are labeled as 0, while abnormal data are labeled as 1.

The first level model constructs with the several high cost-efficient features, which is ideal for fast data collection in both machine learning and predicting stages. In weblog dataset, we select four most cost-efficient features. The features in first stage are self-tuned according to the actual requirement, and also can be captured with the method in Sect. 4.3. In predicting set, unlabeled samples are preprocessed into feature vectors, only consisting of these cost-efficient features. Then classifiers in the first model calculate the confidence scores, which present the anomaly and normal probability of these vectors, and every classifier can be assigned with different threshold. This threshold value is to determine the destination of one sample. There are two outputs of every classifier: (1) If its confidence score is below the threshold θ, this sample must stream to the second stage. (2) If its probability exceeds θ, the classification process will suspend, and this sample is identified as the corresponding label. As for the value of θ, in practice, first we can set a small threshold, and increase it gradually until reaching the acceptable accuracy.

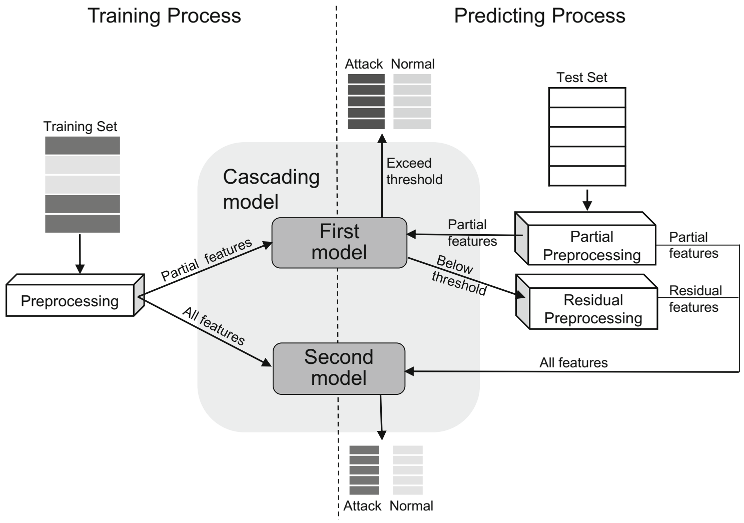


Fig. 4. The structure of two-stage model WebAD^2

Any classifier can be employed to learn and predict samples, such as Decision Trees, Random Forest, Logistic Regression, Adaboost, and Support Vector Machines. Every classifier can be assigned different threshold. The confidence score is a indicator to determine whether the sample should be passed to the second stage for further inspection or not. Sample is labeled as the anomaly if confidence score exceeds the threshold. The threshold makes a noticeable difference on the performance of second stage by impacting the number streaming down from the first stage. The higher threshold is, the more samples the second stage will process. More samples will increase the time and storage consumption, resulting in a lower efficiency. Nevertheless, a lower threshold may retain more samples in the first stage, which make the classification result suspect. When very large volumes of data need to be processed, the setting of threshold is crucial to balance the accuracy and efficiency.

The second level model collects more information from all the features of these samples difficult identified in first stage. Any classifier can be employed to learn and predict samples, the same as the first stage. When training, for better understanding of dataset, all training samples are exploited. The second stage only predicts these data streaming down from the first stage, whose vectors are added with the residual features for higher accuracy and recognition rate. Although the residual features may be generally time-consuming to calculate and possibly make less contribution than the highest cost-efficient features.

For most samples are retained in the first stage, the reduction in the number of samples results in the decrease of prediction time. What's more, the threshold set in the first stage makes a crucial difference on the samples number in second stage, which then impacts the time of second step. We set a small value of θ, and increase it to satisfy an acceptable accuracy. Especially, suppose that the threshold θ is set as 1.0, which means all the sample will stream down to the second stage. Whereas, it is futile, even counterproductive, as it has cost plenty of time in the first stage.

4.3 Combination Selection of Features

This paper addresses a method to select partial features for the first stage model, which is self-adaptive to select the highest cost-efficient features for distinct datasets involving totally different features.

We understand that several features in the first stage make a great difference in the promotion of performance. Another problem to be solve is how to pick out the highest cost-efficient features. It is an apparent fact that different combinations of features lead to different performance results. We create a situation to study how they behave in the same environment, where 2 million of identical web log data with different combinations of features need to be detected by Decision Tree. It takes 4.71 s for one combination consisting of *Num_digit*, *Num_letter*, *Num_punctuation* and *Depth* to get the accuracy of 91.17% in the second stage. Another combination is composed of *Depth*, *Token* and *Relative Entropy*, whose accuracy reaches up to 94.24% but in 21.55 s. This distinction is reasonable that higher accuracy are probably at the cost of longer time.

So we should as far as possible reduce time consumption at a prerequisite of satisfying accuracy. There are a lot of research on feature selection for anomaly detection, ranking the traffic features according to their contribution. Research [26] proposes a multi-stage feature selection method using filters and stepwise regression wrappers. There are two ways to select features for the first stage. If your original data have employed the feature selection or feature extraction, the features combination in first stage can be selected on the basis of the ranking information. Another method, as formula (6), (7), is to calculate the combination score (CS) to balance accuracy and time consumption. These formulas is derived from F-Measure, as formula (8).

$$T' = 1 - \frac{T}{T_{\max}} \tag{6}$$

$$CS = \frac{1}{\frac{1}{\lambda F} + \frac{1}{(1-\lambda)T'}} = \frac{\lambda(1-\lambda)FT'}{\lambda F + (1-\lambda)T'} \tag{7}$$

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P+R} \tag{8}$$

Where F denotes the F-score, T denotes the time of detection. T_{max} is the maximum time and T' is the normalization of T. λ $(0 < \lambda < 1)$ is adopted to adjust the weights of F and T, often set as 0.5. The F-score is a measurement to evaluate accuracy in statistical analysis of binary classification. We also take advantage of F-score as the evaluation standard of the performance of our model in experiments. Combination selection method will rank the balance scores of different combination of features and select the highest value after the setting of features number in the first stage. This method does not take the internal information of dataset into consider like correlation and redundancy of features. It only believes the results as important, which is straightforward but effective and more universal in different IDSs.

5 Experiments and Analysis

In this section, validation experiments are performed to verify the effectiveness of classification in WebAD^2 compared with a basic model. Basic model refers to one-stage model proceeding all features in one time. We apply the metrics of precision, recall, FPR and F-score to evaluate the performance of models. All experiments are run in a machine with Intel Core i5-4570, 2 GB memory and 3.20 GHz CPU under Windows 10.

5.1 Dataset

Due to the variety of networks, traffic profiles and attack types, the representativeness of any dataset for intrusion detection is circumscribed. The network research community still lacks of a representative accessible network traffic dataset. Many of the published researches in anomaly detection still apply the darpa'98 and kdd'99 cup. However, because of the lack of public datasets for network-based IDSs, KDD dataset still suffers from some of the problems discussed by McHugh [10] and may not be a perfect representative of existing real networks.

In light of this, we verify the performance of WebAD^2 on two datasets. First, we use a dataset of web log data from The National Computer Network

Table 2. Attack type in web log dataset

Type	Introduction
SQLI	SQL injection attack
XSS	Cross-site scripting
CODE	Arbitrary code execution vulnerability
COLLECTOR	A malicious content acquisition
SCANNER	Malicious scanning
FILEI	Access to sensitive directory/file
RLFI	Remote/local file contains
OS_COMMAND	Arbitrary command execution
WEBSHELL	Webshell access, contains PHP DDOS
SPECIAL	Particularity of the attack
OTHERS	Others

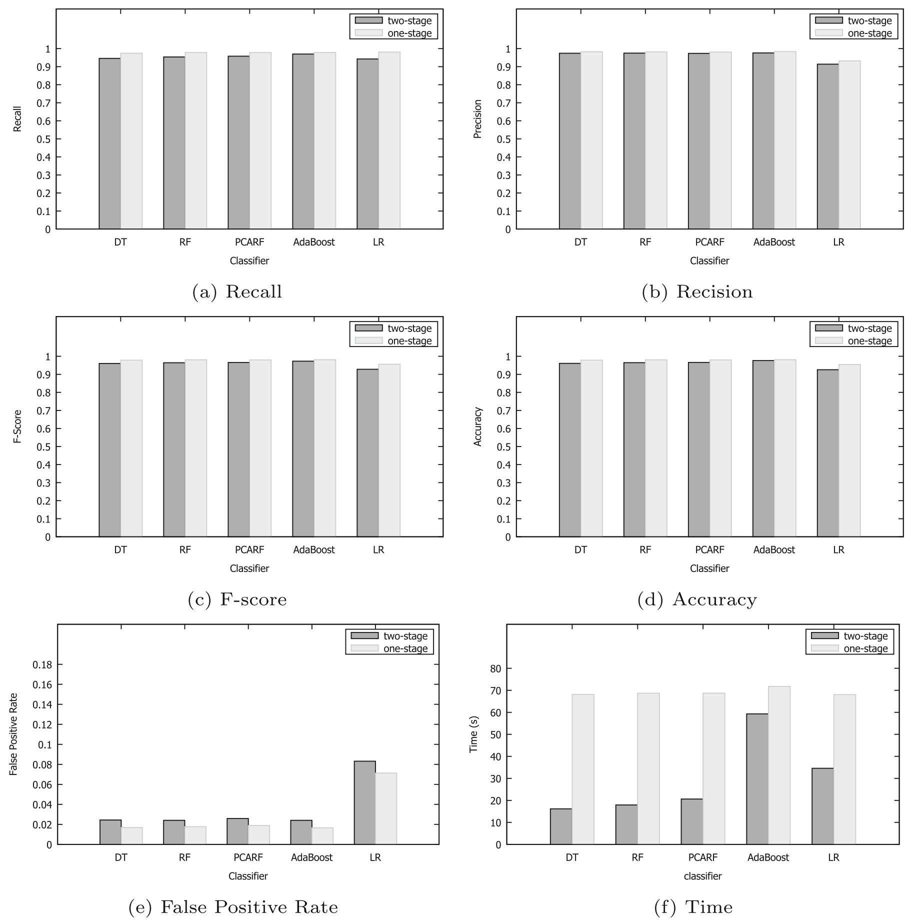
Emergency Response Technical Team/Coordination Center of China (known as CNCERT or CNCERT/CC). The data information is actual and reliable from large-scale web sites. As shown in the Table 2, this dataset contains more than 11 attack types such like SQLI, XSS, DDos. And there are more anomalous samples in the web log dataset compared with KDD CUP99, that half are anomalous samples. This 440 GB dataset consists of 2,000,000 records selected randomly. Both normal and abnormal records are 1,000,000. What's more important, the data are labeled, making great contribution to the learning of classification. Normal records are labeled as 0, while abnormal records are 1.

5.2 Experiment on Web Log Dataset

First in CNCERT web log dataset, we conduct experiments about the performance of two-stage WebAD^2 and one-stage basic model to compare the accuracy and time consumption. The threshold θ of confidence score is set as 0.8. In Fig. 5, we depict the distinction of two different models using the same dataset and setting. The time consumption including preprocessing and detecting significantly decrease in WebAD^2. For example, in the classification Decision Tree, when 400,000 data to be predicted, WebAD^2 spends 16.18 s while basic model needs 68.12 s, reducing more than 76%. The total accuracy of two stage reaches to 0.9627 and basic model is 0.9717, basically maintaining a high accuracy in a short period of time. So when facing challenge of the large data processing and real-time prediction, WebAD^2 has such an enormous advantage for rapid identifying. Random Forest (RF) and PCA Random Forest (PCARF) are based on decision tree, so they almost achieve the same performance. Random Forest constitutes of many decision tree, which could improve accuracy and reduce false alarm rate without over fitting. PCA Random Forest using the top components form principal component analysis (PCA) in the first stage. The one with best effectiveness is AdaBoost classification, but it has a lower efficiency, because of almost half samples stream down to second stage with the limit of confidence score. Logistic Regression (LR) also can improve the performance. Both classifications in two-level cascading model have remarkable advantage in recognition speed, meanwhile keep a good accuracy.

We take five-fold cross-validation for classification evaluation. In each validation experiment, the number of training dataset and test dataset are in a proportion of 4:1. Cross-validation process is repeated 5 times in total, and the average of all results from the folds is consider as a single estimation. Train all classifiers using train set and employ them to detect the anomalies in the test set. In the first stage, we only select *Num_digit*, *Num_letter*, *Num_punctuation* and *Depth* for classifier learning. Results are shown in Table 3.

We specify the procedure now. First of all, the combination selection of features in the first stage proceeds to select the highest cost-efficient in dataset. The combination of *Num_digit*, *Num_letter*, *Num_punctuation* and *Depth* reaches the best balance of accuracy and time consumption ultimately. We employ multiple classifier in the experiments to obtain the most suitable one. Each classifier can set different threshold. The threshold plays a noticeable role on the samples



(a) Recall (b) Recision

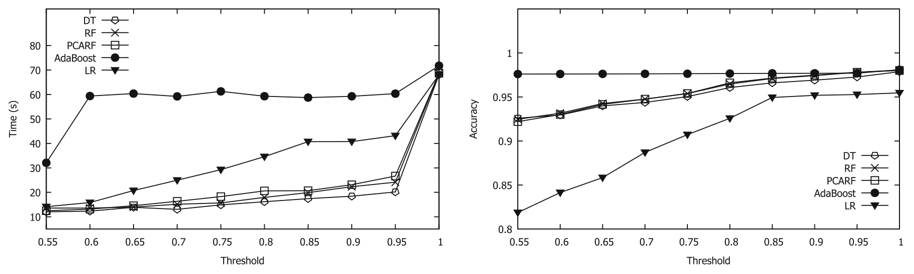
(c) F-score (d) Accuracy

(e) False Positive Rate (f) Time

Fig. 5. Performance in dataset of web log

number of second stage to impact the time and accuracy. Figure 6 illustrates the accuracy and time cost with the change of threshold. The higher threshold is, the more samples the second stage will process. Especially, when the threshold is set as 1.0, all the sample will stream down to the second stage. You can set a lower threshold, and constantly augment the threshold, until the correct rate in an acceptable range.

The second part of table shows the effectiveness of second stage in WebAD^2. More informations remain to be gathered for refined detection. The values in second stage are computed only in the samples streaming down from the first stage. Multiple classifiers are evaluated with five-fold cross-validation and AdaBoost also perform best. It is apparent and reasonable that the samples in second stage are hard to recognized. But the results remain higher accuracy, precision and recall, because all the features are employed in this stage. And the last part of Table 3 sums up the performance of two stage.



(a) Time (b) Accuracy

Fig. 6. Performance in dataset of web log

Table 3. The classifier result in web log dataset

Classifier		Accuracy	Precision	Recall	FPR	F-score	Number
First stage	Decision Tree	0.96269	0.97934	0.94609	0.02036	0.96243	337104
	Random Forest	0.96890	0.98180	0.95584	**0.01790**	0.96864	332136
	PCA Random Forest	0.97039	0.98020	0.96021	0.01942	0.97010	322831
	AdaBoost	**0.97877**	**0.98627**	**0.98419**	0.03513	**0.98523**	160782
	Logistic Regression	0.88511	0.96603	0.88529	0.11555	0.92390	144767
Second stage	Decision Tree	0.94977	0.94835	0.94533	0.04623	0.94684	62896
	Random Forest	0.94454	0.94288	0.94334	0.05432	0.94311	67864
	PCA Random Forest	0.94775	0.94680	0.94857	0.05306	0.94768	77169
	AdaBoost	**0.97514**	**0.96919**	0.96002	**0.01662**	**0.96458**	239218
	Logistic Regression	0.94877	0.88428	**0.97553**	0.06482	0.92767	255233
WebAD^2	Decision Tree	0.96066	0.97447	0.94597	0.02443	0.96001	400000
	Random Forest	0.96477	0.97520	0.95372	0.02408	0.96434	400000
	PCA Random Forest	0.96602	0.97375	0.95796	0.02591	0.96579	400000
	AdaBoost	**0.97660**	**0.97605**	**0.96973**	**0.02406**	**0.97288**	400000
	Logistic Regression	0.92573	0.91387	0.94287	0.08318	0.92814	400000

From the Fig. 5 we can see that, all classifiers improve the modeling efficiency without sacrificing the accuracy. AdaBoost has the best effectiveness performance but costs more time than others. Because the majority of samples stream down to the second stage, which increases the time consumption, as the last column in Table 3 shown. Synthetically, Decision Tree, Random Forest and PCA Random Forest are the best choice for high accuracy and less time.

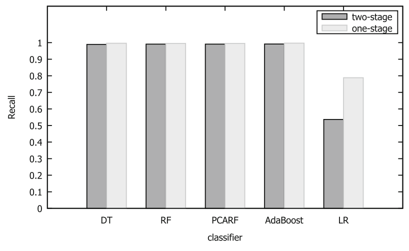
5.3 Experiments on NSL-KDD

Another set of experiments is requisite to indicate the universality of two-stage cascading model in different dataset. NSL-KDD is employed as an effective benchmark dataset to evaluate the intrusion detection methods. First, for there are some features in character form and classifiers only can predict the sample in numeric form, we preprocess the data turning them into numeric values. The number of the values of this features are constant. For example, one feature protocol type contains 4 different value: (1) ICMP; (2) TCP; (3) UDP; (4) others, so that can be marked as the sequence number correspondingly.

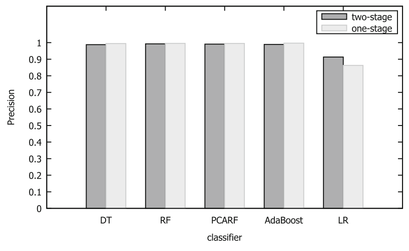
Table 4. The classifier result in NSL-KDD

Classifier		Accuracy	Precision	Recall	FPR	F-score	Number
First stage	Decision Tree	0.99006	0.98915	0.99080	0.01066	0.98997	29357
	Random Forest	**0.99462**	**0.99521**	**0.99395**	**0.00472**	**0.99458**	28878
	PCA Random Forest	0.99387	0.99388	0.99382	0.00608	0.99385	28473
	AdaBoost	0.99126	0.98898	0.99225	0.00960	0.99061	26018
	Logistic Regression	0.88173	1.00000	0.00063	0.00000	0.00127	11591
Second stage	Decision Tree	0.87719	0.89000	0.86829	0.86829	0.87901	346
	Random Forest	0.91158	0.91607	0.88631	0.06743	0.90094	825
	PCA Random Forest	0.94001	0.93069	0.92916	0.05185	0.92993	1230
	AdaBoost	**0.98724**	**0.99299**	**0.98903**	**0.01716**	**0.99101**	3685
	Logistic Regression	0.80385	0.85784	0.87950	0.40779	0.40779	18112
WebAD^2	Decision Tree	0.98875	0.98799	0.98937	0.02066	0.98868	29703
	Random Forest	0.98932	**0.99301**	0.99096	0.00646	**0.99198**	29703
	PCA Random Forest	**0.99164**	0.99126	0.99114	**0.00797**	0.99120	29703
	AdaBoost	0.99076	0.98948	**0.99185**	0.01054	0.99066	29703
	Logistic Regression	0.83424	0.91332	0.53654	0.24866	0.67597	29703

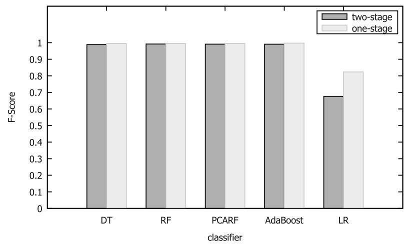
The results of experiments are showed in Table 4. In the classifier PCA Random Forest, the accuracy of in WebAD^2 is up to 0.99164, and the basic model is 0.99526. The performance in NSL-KDD is more obvious than web log dataset. WebAD^2 spends 0.25494 s to classify almost all anomalies and normal accesses, only 27.9% of 0.9144 s the basic model costs. These trials robustly demonstrate that two-level cascading model can greatly improves the modeling efficiency without sacrificing the accuracy as the Fig. 7 depicts. The accuracy of all the classifiers are close to 1, with time falling by about one third. The performance in NSL-KDD is even better than in the web log dataset, which indicates the universality of two-level cascading model.



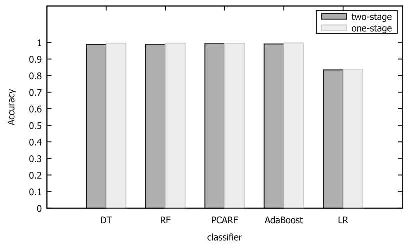
(a) Recall



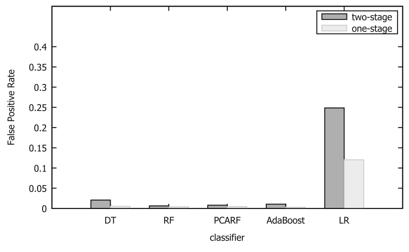
(b) Recision



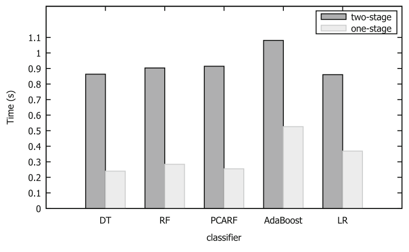
(c) F-score



(d) Accuracy



(e) False Positive Rate



(f) Time

Fig. 7. Performance in dataset of NSL-KDD

To summarize the above experiments, we can get that:

1. The first stage can identify most of anomalies in a short time. It manages the balance of time and accuracy, and ensures reliability through the setting of suited threshold. The first stage makes a significant contribution to the reduction of time in cascading model;

2. The second stage can reach a higher accuracy exploiting all features but need more time. But due to it only dispose a small quantity of samples streaming from the first stage, its total time is less than the basic model. The excellent contribution of second stage is to improve the accuracy;
3. The second stage adopts more features than first stage, so that shows better power to accurately distinguish normal traffics and abnormal traffics. But note that this model relays on the first stage model to classify most of the samples no matter what classifier is used.

6 Conclusions

In this paper, we propose a two-stage anomaly detection model, named WebAD^2, to identify web attacks and anomalies. Our studies reveal that about 85% attacks or anomalies could be identified by exploiting only a small set of features. Therefore, in the first stage, we select partial but key features to differentiate anomalies from normal web logs and gain significant performance boost. In the second stage, all features are exploited to finer-tune the detection results and achieve a satisfactory detection accuracy. WebAD^2 also could be employed to improve the performance of existing system such as distributed system or parallel system to further accelerate the processing. This paper also puts forward a feature selection method to choose cost-efficient features in the first stage, which ensures that an appropriate balance between accuracy and detection efficiency could be maintained.

The experimental results show that WebAD^2 can greatly reduce the time consumption without sacrificing anomaly detection accuracy. Besides, WebAD^2 could deal with massive web logs, and still meets the demand of real-time detection. In future work, we will apply clustering algorithms to further classify the anomalies and obtain fine-grained detection results.

References

1. Prokhorenko, V., Choo, K.K.R., Ashman, H.: Context-oriented web application protection model. Elsevier Science Inc. (2016)
2. Prokhorenko, V., Choo, K.K.R., Ashman, H.: Intent-based extensible real-time php supervision framework. IEEE Trans. Inf. Forensics Secur. **11**(10), 2215–2226 (2016)
3. Kruegel, C., Vigna, G., Robertson, W.: A multi-model approach to the detection of web-based attacks. Comput. Netw. **48**(5), 717–738 (2005)
4. Threepak, T., Watcharapupong, A.: Web attack detection using entropy-based analysis. In: The International Conference on Information Networking 2014 (ICOIN 2014), pp. 244–247. IEEE (2014)
5. Peng, J., Choo, K.K.R., Ashman, H.: User profiling in intrusion detection: a review. J. Netw. Comput. Appl. **72**, 14–27 (2016)
6. Osanaiye, O., Cai, H., Choo, K.K.R., Dehghantanha, A., Xu, Z., Dlodlo, M.: Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. Eurasip J. Wirel. Commun. Netw. **2016**(1), 130 (2016)

7. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutor. **16**(1), 303–336 (2014)
8. Nadiammai, G., Hemalatha, M.: Effective approach toward intrusion detection system using data mining techniques. Egypt. Inform. J. **15**(1), 37–50 (2014)
9. Osanaiye, O., Choo, K.K.R., Dlodlo, M.: Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework. J. Netw. Comput. Appl. **67**(C), 147–165 (2016)
10. McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans. Inf. Syst. Secur. (TISSEC) **3**(4), 262–294 (2000)
11. Zhang, S., Li, B., Li, J., Zhang, M., Chen, Y.: A novel anomaly detection approach for mitigating web-based attacks against clouds. In: 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 289–294. IEEE (2015)
12. Akamai: Q1 2017 state of the internet/security report. Technical report, Akamai Technologies, Inc (2017). https://www.akamai.com/us/en/about/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp
13. Hu, W., Hu, W., Maybank, S.: Adaboost-based algorithm for network intrusion detection. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **38**(2), 577–583 (2008)
14. Hu, W., Gao, J., Wang, Y., Wu, O., Maybank, S.: Online adaboost-based parameterized methods for dynamic distributed network intrusion detection. IEEE Trans. Cybern. **44**(1), 66–82 (2014)
15. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 251–261. ACM (2003)
16. Robertson, W.K., Vigna, G., Krgel, C., Kemmerer, R.A.: Using generalization and characterization techniques in the anomaly-based detection of web attacks. In: Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, USA (2006)
17. Mabu, S., Chen, C., Lu, N., Shimada, K., Hirasawa, K.: An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **41**(1), 130–139 (2011)
18. Yao, D., Yin, M., Luo, J., Zhang, S.: Network anomaly detection using random forests and entropy of traffic features. In: 2012 Fourth International Conference on Multimedia Information Networking and Security, pp. 926–929. IEEE (2012)
19. Zhang, J., Chen, X., Xiang, Y., Zhou, W., Wu, J.: Robust network traffic classification. IEEE/ACM Trans. Netw. **23**(4), 1257–1270 (2015)
20. Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: detecting the unknown without knowledge. Comput. Commun. **35**(7), 772–783 (2012)
21. Owezarski, P.: A near real-time algorithm for autonomous identification and characterization of honeypot attacks. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, pp. 531–542. ACM (2015)
22. Hasan, M.A.M., Nasser, M., Pal, B., Ahmad, S.: Support vector machine and random forest modeling for intrusion detection system (IDS). J. Intell. Learn. Syst. Appl. **6**(1), 45 (2014)
23. Bhavsar, Y.B., Waghmare, K.C.: Intrusion detection system using data mining technique: support vector machine. Int. J. Emerg. Technol. Adv. Eng. **3**(3), 581–586 (2013)

24. Fan, W.K.G.: An adaptive anomaly detection of web-based attacks. In: 2012 7th International Conference on Computer Science and Education (ICCSE), pp. 690–694. IEEE (2012)
25. Casas, P., Vaton, S., Fillatre, L., Nikiforov, I.: Optimal volume anomaly detection and isolation in large-scale IP networks using coarse-grained measurements. Comput. Netw. **54**(11), 1750–1766 (2010)
26. Iglesias, F., Zseby, T.: Analysis of network traffic features for anomaly detection. Mach. Learn. **101**(1–3), 59–84 (2015)
27. Hota, H.S., Shrivas, A.K.: Decision tree techniques applied on NSL-KDD data and its comparison with various feature selection techniques. In: Kumar Kundu, M., Mohapatra, D.P., Konar, A., Chakraborty, A. (eds.) Advanced Computing, Networking and Informatics- Volume 1. SIST, vol. 27, pp. 205–211. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07353-8_24