Neural Comput & Applic
DOI 10.1007/s00521-016-2418-1

ORIGINAL ARTICLE

An effective combining classifier approach using tree algorithms for network intrusion detection

Jasmin Kevric[1] · Samed Jukic[1] · Abdulhamit Subasi[2]

Received: 15 May 2015 / Accepted: 8 June 2016
© The Natural Computing Applications Forum 2016

Abstract In this paper, we developed a combining classifier model based on tree-based algorithms for network intrusion detection. The NSL-KDD dataset, a much improved version of the original KDDCUP'99 dataset, was used to evaluate the performance of our detection algorithm. The task of our detection algorithm was to classify whether the incoming network traffics are normal or an attack, based on 41 features describing every pattern of network traffic. The detection accuracy of 89.24 % was achieved using the combination of random tree and NBTree algorithms based on the sum rule scheme, outperforming the individual random tree algorithm. This result represents the highest result achieved so far using the complete NSL-KDD dataset. Therefore, combining classifier approach based on the sum rule scheme can yield better results than individual classifiers, giving us hope of better anomaly based intrusion detection systems in the future.

Keywords Intrusion detection · Tree-based classifiers · NSL-KDD · Combining classifiers approach

✉ Abdulhamit Subasi
absubasi@effatuniversity.edu.sa

Jasmin Kevric
jasmin.kevric@ibu.edu.ba

Samed Jukic
sjukic@ibu.edu.ba

[1] Faculty of Engineering and Information Technologies, International Burch University, Francuske Revolucije bb. Ilidza, 71000 Sarajevo, Bosnia and Herzegovina

[2] Computer Science Department, College of Engineering, Effat University, Jeddah 21478, Saudi Arabia

1 Introduction

Computer system security is defined as the process of protecting three factors: confidentiality, integrity, and availability (CIA) [1]. Network security is becoming progressively more important due to the huge growth of computer networks usage and network applications. The 2005 annual computer crime and security survey, jointly conducted by the Computer Security Institute and the FBI, indicated that financial losses incurred by the respondent companies due to network attacks/intrusions were US $130 million [2]. Therefore, intrusion detection (ID) is a major research problem in network security, although the concept of ID was proposed by Anderson way back in 1980 [3].

Intrusion detection system (IDS) can be defined as a security service that monitors and investigates system events so that the attempts to access system resources in an unauthorized manner can be identified [1]. An intrusion can also be defined as "any set of actions that attempts to compromise the integrity, confidentiality, or availability of a resource" [4].

ID is based on the assumption that the behavior of intruders is different from a legal user [5]. The goal of intrusion detection systems (IDS) is to identify attacks from intruders with high accuracy in order to secure internal networks [6]. Network-based IDS is a valuable tool which looks for known or potential malicious activities in network traffic and raises an alarm whenever a suspicious activity is detected [7].

Misuse and anomaly detection are two basic approaches for detecting network intrusions. Misuse detection (also known as the knowledge-based or signature-based detection) is based on storing all features of known attacks in a database. In other words, the signatures of attacks are known and described a priori and IDS attempts to recognize and notify the activity as either normal or intrusion by comparing the network connection records to the known intrusion patterns obtained from the human experts. Misuse IDS has high detection rates for well-known attacks, but is usually unable to deal with new or unknown attacks. Thus, the database needs continuous updating in order to cope with novel intrusion types [8].

Alternatively, anomaly or behavior-based IDS is based on detecting behavioral change of users in the network [8], assuming that an intrusion can be detected by observing deviations from a normal or expected behavior of a monitored entity. Thus, it is extremely important to form an accurate opinion of what the subject's normal behavior is. For that purpose, various data records observed through processes on the same network must be examined. These data records and features that can be extracted from them represent one of the most important factors. Whenever input features have more information about the benign and intruder users, the intrusion detection system can better separate different type of users, improving the security of the protected system/network. The anomaly detection techniques have the advantage of detecting unknown attacks over the misuse detection technique [9].

Misuse-based detection is generally favored in commercial products due to its predictability and high accuracy. But, cyber hackers, knowing that intrusion detection systems are installed in our network, will always try to develop and launch "new" attacks. Due to its theoretical potential for addressing novel attacks, anomaly detection is typically conceived as a more powerful method in academic research that must be well studied [10, 11]. Generally, various machine learning algorithms are employed for anomaly detection in IDS. In this work, a combining classifier model using different decision tree algorithms is proposed. The suggested detection algorithm is able to classify the incoming network traffics as being normal or an attack.

An effective intrusion detection system needs to reduce false positives/alarms rate. At the same time, it needs to be effective at catching attacks, maintaining high detection rate. One of the main problems with IDS is the overhead, which can become prohibitively high. The detection response time is another major problem in the IDS. Computer networks have a dynamic nature in a sense that information and data within them are continuously changing. Therefore, detecting an intrusion accurately and promptly is crucial especially in real time IDS [12].

This paper is organized as follows. Section 2 summarizes the related work done in developing anomaly based IDS. Section 3 describes the algorithms and techniques we used to develop our own anomaly based IDS. All results and analysis in form of tables can be found in Sect. 4. A thorough discussion about the results is provided in Sect. 4. Lastly, we finalize the paper by drawing conclusions in Sect. 5.

2 Literature review

Conducting a thorough analysis of the recent research trend in anomaly detection, several machine learning methods were reported to have a very high detection rate while keeping the false alarm rate very low [13]. However, there is no evidence of using anomaly detection approaches in the state-of-the-art IDS solutions and commercial tools. In addition, practitioners still think that anomaly detection is an immature technology. The cause of this contrast was partially explained in [10]. They studied the details of the research done in anomaly detection and considered various aspects such as learning and detection approaches, training and testing data sets, and evaluation methods. They showed that there are some inherent problems in the KDDCUP'99 data set [14], which is widely used as one of the few publicly available data sets for network-based anomaly detection systems.

The first important deficiency in the KDD data set was the huge number of redundant records. It was found that about 78 and 75 % of the records are duplicated in the KDD train and test set, respectively [10]. Therefore, learning algorithms are biased toward the more common records, thus ignoring rare records which are usually more harmful. Alternatively, the existence of these repeated records in the KDDTest set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records [10].

The difficulty level of the records in KDD data set was also analyzed in [10]. They employed 21 classifiers (7 classifiers, each trained 3 times with different train sets) to label the records of the entire KDDTrain and KDDTest sets. Shockingly, about 98 % of the records in the KDDTrain set and 86 % of the records in the KDDTest set were correctly classified with all the 21 learners. These numbers are important as random parts of the KDDTrain set are used as test sets in many papers achieving about 98 % classification rate applying very simple machine learning methods. Even applying the KDDTest set will result in having a minimum classification rate of 86 % [10].

The authors were able to conduct such analysis since every record in the original KDD train and test set was associated with a #successfulPrediction integer value initiated to zero before any record was labeled [10]. Once a record was correctly classified by a classifier, this value was increased by 1 for the given record. In other words, #successfulPrediction value showed how many classifiers were able to correctly classify a particular record, so its maximum value was 21 [10].

The #successfulPrediction value was used as a basis for generating new NSL-KDD datasets, providing a solution to solve the two aforementioned issues of the original KDD dataset [10]. In short, new train and test sets (KDDTrain+ and KDDTest+), consisting of selected records of the complete KDD dataset, were proposed. The provided datasets do not suffer from any of the mentioned problems. Even the 20 % of the KDDTrain+ set was provided along with a KDDTest-21 set, which did not include any of the records that had been correctly classified by all 21 classifiers. In other words, KDDTest-21 set did not contain records whose #successfulPrediction value was equal to 21. As a result, the number of records in the new sets is reasonable, so the experiments could be run on the complete sets without the need to randomly select a small portion. By this method, evaluation results of different research work will be consistent and comparable [10]. The new version of KDD dataset, NSL-KDD, is publicly available for researchers [15].

Despite all the improvement over KDDCUP'99 dataset, NSL-KDD is not perfect. This dataset still suffers from some of the problems discussed by McHugh [16]. For example, normal and attack data have unrealistic data rates; training datasets for anomaly detection are not adequate for its intended purpose; no effort have been made to validate that false alarm behavior of IDSs under test shows no significantly difference on real and synthetic data. Malhony and Chan confirmed McHugh's findings by their experiments, which discovered that many attributes had small and fixed ranges in simulation, but large and growing ranges in real traffic [13, 17]. NSL-KDD may also not be a perfect representative of existing real networks due to the lack of public datasets for network-based IDSs. However, it still can be applied as an effective benchmark data set to help researchers compares different intrusion detection methods [10]. In [10], they actually employed 20 % of KDDTrain+ as the train set and three different test sets: KDDTest (original KDD test set), KDDTest+, and KDDTest-21, achieving the highest accuracies of 93.82, 82.02, and 66.16 % respectively. Overall best performing algorithms were NB Tree [18], C4.5 [19], and random tree [20].

A number of researchers have used NSL-KDD dataset to test their intrusion detection algorithms since 2009, when NSL-KDD set became available. In particular, machine learning techniques were developed as classifiers which are used to classify the incoming network traffics as normal or an attack [12, 21–25]. However, NSL-KDD dataset (as well as the original KDD dataset) contains attack-type labels for every attack sample. These attack labels belong to four attack groups: DoS, Neptune or Probe, U2R, and R2L. Therefore, some researchers have taken an approach of developing machine learning techniques to classify the incoming network traffics as being either normal or belonging to one of four attack groups [7, 8, 26–31]. However, some researchers were not specific about whether they classified network traffic into two or five groups [9, 32–35].

In [10], the number of records in the KDDTrain+ and KDDTest+ sets was made reasonable so that the experiments could be run on the complete sets without the need to randomly select a small portion. As a result, evaluation results of different research works would be consistent and comparable. However, random parts of the NSL-KDD dataset are used as train, and test sets in many papers [7, 26–29, 32, 35], without explicitly stating which NSL-KDD datasets (KDDTrain+ and/or KDDTest+) were used to randomly choose the samples. Their approaches make their results pretty incomparable as their proposed methods have been trained and tested on different datasets. Detection accuracies ranges from 89 to 94.7 % have been reported.

On the other hand, substantial number of published studies reported the results of their proposed solution where the machine learning algorithm was trained and tested on the same dataset [12, 22–24, 33]. In [12, 22, 23], tenfold cross-validation was performed on entire 20 % of KDDTrain+ set to evaluate the proposed methods. Detection accuracies above 96.5 % were reported. In [33], tenfold cross-validation was performed on roughly half of the records in KDDTrain+ set, reporting detection accuracies as high as 99.7 %. Sethuramalingam and Naganathan [24] divided 20 % of KDDTrain+ set into TCP, UDP, and ICMP samples, reporting low detection accuracies for UDP samples only.

Authors were able to find only two more papers following the same approach of training and testing the proposed methods as suggested in [10]. In [9], class-dependent feature transformation (CDFTR) technique for reducing the dimension of features was presented. Detection accuracies for five different feature transformation techniques and three different classifiers were compared and reported. Strangely, all 15 results were reported as identical for both KDDTest+ and KDDTest-21 sets. The highest detection accuracy was 80.14 % using decision tree classifier, which makes it comparable to [10]. In [21], the overall detection accuracy of the fuzzy classifier found by genetic programming on KDDTest+ set was reported to be 82.74 %, slightly outperforming all methods in [10].

Every instance of network traffic in NSL-KDD dataset is defined by 41 features. Therefore, many feature selection (feature transformation, feature reduction, or data filtering) techniques have been compared and applied to decrease the number of features of NSL-KDD dataset such as: class-dependent feature transformation [9], rough set algorithm [32], principal component analysis [7, 9, 22, 23, 28, 30], decision trees [23], random projection and nominal to binary [22], information gain [24], information gain and genetic algorithm [24], correlation-based and consistency-based feature selection [25], deep belief network, gain ratio and Chi-square [7], PSO-OPF, HS-OPF and GSA-OPF [35], correlation-based, information gain and gain ratio [31].

Many research studies have employed hybrid intelligent techniques using combination of classifiers with the intention of improving the overall performance of the resultant model. The most popular hybrid approaches were Meta classifiers such as: END and Grading [23], and AdaBoost ensemble [12]. One of the easiest ways to implement the combining classifiers approach is to use data mining tool Weka [8, 10, 25, 29, 31, 33, 34, 36].

The aim of this paper is to follow the same approach of evaluating the performance of our detection algorithm as in [10]. We will use decision tree-based machine learning tools to classify whether the incoming network traffics are normal or an attack, using 41 different features. No feature selection was performed. The performance of suggested detection algorithm is evaluated on the NSL-KDD dataset which represents an improved version of the original KDDCUP'99 dataset. We will employ hybrid intelligent techniques by combination of classifiers with the intention of improving the overall performance of the resultant model using Weka data mining software.

3 Materials and methods

3.1 NSL-KDD dataset

The NSL-KDD dataset and the way how it was derived from KDDCUP'99 dataset has already been described in Sect. 2. The generated data sets, KDDTrain+ and KDDTest+, included 125,973 and 22,544 records, respectively. KDDTest-21 dataset, which did not include any of the records that had been correctly classified by all 21 learners in [10], incorporated 11,850 records. For experimental purposes, 20 % of the records in KDDTrain+ were also generated. All four of these dataset sets are publicly available at [15], with binary class labels (normal or attack), or including attack-type labels. Every pattern of network traffic is defined by 41 features [10].

3.2 NBTree and random tree classifiers

Naive-Bayes [37, 38] uses Bayes rule to compute the probability of each class given the instance, assuming that the attributes are conditionally independent given the label. The data are prediscretized using an entropy-based algorithm [39, 40]. The probabilities are estimated directly from data based directly on counts. In cases when the conditional independence notion is violated, Naive-Bayes algorithms were shown to be unexpectedly accurate on numerous classification tasks. But most of these studies took advantage of small databases. In some larger databases, the accuracy of Naive-Bayes does not scale up as well as decision trees [18].

On the other hand, decision trees [19, 41] are normally built by recursive partitioning. A univariate (single attribute) split is selected for the root of the tree according to some criterion, and the process repeats recursively. The process known as pruning, which decreases the tree size, is performed after a full tree has been built [18]. The most popular representative of decision trees is C4.5.

An NBTree algorithm induces a hybrid of decision tree classifiers and Naive-Bayes classifiers. This hybrid approach attempts to utilize the advantages of both decision trees and Naive-Bayes. A decision tree is built with univariate splits at each node, but with Naive-Bayes classifiers at the leaves. The final classifier resembles Utgoff's Perceptron trees [42], but the induction process is very different and geared toward larger datasets. Therefore, the decision tree segments the data, a task that is considered as an essential part of the data mining process in large databases [43]. Each segment of the data, represented by a leaf, is described through a Naive-Bayes classifier. The induction algorithm segments the data so that the conditional independence assumptions required for Naive-Bayes are likely to be true. Therefore, this method keeps the interpretability of both Naive-Bayes and decision trees. At the same time, their combination often performs better than both individual classifiers, particularly in the larger databases [18].

Random tree classifier [20] represents an ensemble of single decision trees; each being built by taking into account K arbitrarily selected features at each node expansion. No strictly defined condition (like information gain or gini index) for feature selection nor pruning is used. Random tree is a scalable and efficient ensemble tool outperforming the base classifier and is comparable to other popular ensemble algorithms. Interestingly, random tree configuration is determined prior to training data inspection. After the data have been studied, random tree is adjusted and ready for application resulting in a technique which scales very well for large datasets such as IDS [44].

In order to build tree structures, a categorical (discrete) feature from the training data like gender can be considered on one occasion only during a decision path. On the other hand, a continuous feature may be selected on more occasions along a particular decision path, but never with the same threshold (splitting value) which is randomly selected. All trees in the ensemble generate a class probability distribution which is recorded in the leaf node level of the tree. The class distribution probabilities from numerous decision trees in the ensemble are then averaged to produce the definitive class distribution guess [45].

If the information about the features combinations which are most predictive of the class label is required, Random decision tree classifiers should be avoided. Similarly to artificial neural networks, Random decision trees are very good black-box predictors only [44].

A random tree algorithm is very efficient during the training phase as it generates a tree after only one pass through the training data. In addition, Random decision tree optimally estimates the true probability of being a member of a given class for each record. In other words, they represent capable application of Bayes optimal classifiers [44].

3.3 Combining classifiers

Accomplishing the highest possible classification performance represents the key objective of designing pattern recognition systems. For this purpose, various classification schemes for a particular pattern recognition task can be developed. Even though one of the systems would produce the best overall performance, the sets of patterns that are correctly classified by the different classifiers would not necessarily overlap. In other words, different classification schemes possibly provide complementary information about the patterns. We can use this complementary information to enhance the performance of the individual classifiers. Here lies the motivation behind the recent interest in combining classifiers approach, where the idea is to consider more than one classification scheme. A theoretical underpinning of many existing classifier combination schemes for fusing the decisions of multiple experts each employing a distinct pattern representation was provided in [46]. Combining classifier schemes such as the sum rule, product rule, max rule, min rule, median rule, and majority voting have been developed. Although the experiments demonstrated that some of these combination schemes outperform a best individual classifier, we have poor understanding why some of them perform better than others. A surprising outcome of the comparative study is that the combination rule developed under the most restrictive assumptions—the sum rule—outperforms other classifier combinations schemes. This empirical finding was explained by the sensitivity analysis which showed that the sum rule is most resilient to estimation errors [46].

4 Results and discussion

For experimental purposes, we employed 20 % of the records in KDDTrain+ as the train set, having trained the detection algorithms. We applied the learned models on two test sets, namely KDDTest+ and KDDTest-21. Besides the overall accuracy, the performances of the classification are also expressed in terms of sensitivity and specificity. Sensitivity measure shows how good our proposed algorithms are when detecting network attacks. Specificity, on the other hand, represents the measure of false detections, or the performance of a classifier in detecting normal attacks. Three detection algorithms achieving highest accuracies were selected: random tree, C4.5, and NBTree. Other algorithms used in the comparison were: Naive-Bayes, random forest, multilayer perceptron, and support vector machines. Random "seed" parameter was set to 2. For C4.5, the minimum number of instances per leaf was 6, and the fivefold was used for reduced error pruning. NBTree was used with the default parameters.

4.1 Individual classifiers

The results of detection performance on two test sets for individual classifiers are shown in Tables 1 and 2.

Now, we will show the detection performance of combining classifier models based on different combinations of these three algorithms. The sum rule scheme, which is most resilient to estimation errors, was used [46].

4.2 Combining classifiers

The results of detection performance on two test sets for combining classifiers are shown in Tables 3 and 4.

The detection performance of learning algorithms on KDDTest+ is higher than KDDTest-21 in all cases. This is quite reasonable as KDDTest-21 is more challenging dataset than KDDTest+. In case of individual classifiers, random tree performed the best although C4.5 had least number of false alarms. NBTree performed the worse based on any evaluation criteria we used in this paper.

In case of combining classifiers approach, it was interesting to see that the combination of the best and worst individual classifier (random tree and NBTree) delivered the best detection performance on both test datasets, slightly outperforming the random tree + C4.5 combination. The explanation for this could be hidden behind the motivation for combining classifiers approach. Actually, the sets of patterns misclassified by the different classifiers do not necessarily overlap. In other words, different classifier designs can potentially offer more complementary information about the patterns to be classified. Because of this, random tree + NBTree combination provides better detection performance than the individual random tree classifier or any other combination of classifiers. The only other combination of classifiers outperforming the individual random tree classifier was random tree and C4.5 combination.

Table 1 Detection performance of individual classifiers on KDDTest+

Individual classifiers	Evaluation criteria		
	Sensitivity (%)	Specificity (%)	Accuracy (%)
Random tree	82.6	96.2	88.46
C4.5	78.0	97.2	86.29
NBTree	75.0	91.3	82.02

Table 2 Detection performance of individual classifiers on KDDTest-21

Individual classifiers	Evaluation criteria		
	Sensitivity (%)	Specificity (%)	Accuracy (%)
Random tree	77.0	85.5	78.51
C4.5	70.9	87.5	73.92
NBTree	66.9	62.9	66.17

Table 3 Detection performance of combining classifiers models on KDDTest+

Combining classifiers	Evaluation criteria		
	Sensitivity (%)	Specificity (%)	Accuracy (%)
Random tree + NBTree	83.9	96.2	89.24
Random tree + C4.5	83.0	96.5	88.81
NBTree + C4.5	76.7	95.6	84.98
All three	79.8	96.3	86.95

Table 4 Detection performance of combining classifiers models on KDDTest-21

Combining classifiers	Evaluation criteria		
	Sensitivity (%)	Specificity (%)	Accuracy (%)
Random tree + NBTree	78.8	85.6	80.0
Random tree + C4.5	77.5	86.7	79.15
NBTree + C4.5	69.2	81.8	71.46
All three	73.3	85.5	75.54

Table 5 Comparison table

Comparison table	Feature selection	Overall accuracy (%)	
		KDDTest+	KDDTest-21
NBTree [10]	No	82.02	66.16
Decision tree [9]	CDFTR	80.141	80.141
Fuzzy classifier [21]	No	82.74	–
Random tree + NBTree (our approach)	No	89.24	80.0

Authors were able to find only two more papers following the same approach of training and testing the proposed methods as suggested in [10]. We compare our result with the results presented in these papers in Table 5.

We can see that our approach significantly outperforms all others in terms of overall accuracy on KDDTest+, being only slightly defeated by CDFTR followed by decision tree classifier in [9]. However, they reported the same overall accuracy for both test sets, which is unlikely to happen.

We need to note that the NBTree classifier used in [10] was implemented in Weka package using default values of the parameters. On the other hand, decision tree classifier from [9] was developed in MATLAB, but the authors did not mention whether default or tuned values of parameters were applied. In [21], a genetic programming (GP) was employed to evolve fuzzy classifier, and GP parameters were provided in the study. However, authors did not mention the environment in which the algorithm was developed or whether the provided parameters were default

Table 6 Comparison table for tenfold cross-validation

Comparison table	Feature selection	Overall accuracy (%) 20 % KDDTrain+
Adaboost + GA [12]	No	99.57
DMNB [22]	N2B	96.5
Random tree + NBTree (our approach)	No	99.53

ones. Our approach was also implemented in Weka, but tuned values of parameters were used since they resulted in increased overall accuracy when compared to default values. Considering the scarcity in the number of studies following the proper approach of training and testing IDS, the fact that certain studies do not report if values of parameters had been altered [9, 21], we think the comparison provided in Table 5 is compatible and worth mentioning.

However, substantial number of published studies reported the results of their proposed solution where the machine learning algorithm was trained and tested on the same dataset (usually 20 % of KDDTrain+ set), using tenfold cross-validation. We compare our result with the results presented in these papers in Table 6.

Our approach significantly outperforms [22], whereas it is comparable to the result obtained in [12]. It is interesting that no approach from Table 6 evaluated the performance of detection algorithm on either version of the test set.

In cases when our approach did not perform the best, it was outperformed by only 0.17 % (Tables 5 and 6). On the other hand, our proposed solution achieved the best detection performance when following the most proper way of evaluation the detection algorithm as proposed in [10].

It is also worth mentioning that all approaches in Tables 5 and 6 were developed as classifiers which try to label the incoming network traffics as normal or an attack, except in [9], where researchers were not specific about whether they classified network traffic into two or five groups.

5 Conclusion

In this paper, we developed a combining classifier model based on tree-based algorithms for network intrusion detection. The NSL-KDD dataset, a much improved version of the original KDDCUP'99 dataset, was used to evaluate the performance of our detection algorithm. The task of our detection algorithm was to classify whether the incoming network traffics are normal or an attack, based on 41 features describing every pattern of network traffic. We conclude that combining classifier approach based on the sum rule scheme can yield better results than individual classifiers, although the opposite is also possible. This is very evident from Tables 1, 2, 3, and 4. We also conclude that selecting the two best individual classifiers may not lead to the best overall performing combination.

The effect of feature reduction both on the training time and detection accuracy can be tested in the future. We may also develop a similar detection algorithm to classify the incoming network traffics as being either normal or belonging to one of four attack groups.

References

1. Stallings W, Brown L (2008) Computer security principals and practice. Pearson Education, Upper Saddle River
2. C. S. Institute and F. Investigation (2005) In: Proceedings of the 10th annual computer crime and security survey
3. Anderson JP (1980) Computer security threat monitoring and surveillance. James P. Anderson Co., Fort Washington
4. Debar H, Dacier M, Wespi A (2000) A revised taxonomy for intrusion detection systems. Ann Telecommun 55(7):361–378
5. Stallings W (2006) Cryptography and network security principles and practices. Prentice Hall, Englewood Cliffs
6. Tsai C, Hsu Y, Lin C, Lin W (2009) Intrusion detection by machine learning: a review. Expert Syst Appl 36:11994–12000
7. Salama MA, Eid HF, Ramadan RA, Darwish A, Hassanien AE (2011) Hybrid intelligent intrusion detection scheme. In: Gaspar-Cunha A, Takahashi R, Schaefer G,Costa L (eds) Soft computing in industrial applications, Springer, Berlin, Heidelberg, pp 293–303
8. Natesan P, Rajesh P (2012) Cascaded classifier approach based on Adaboost to increase detection rate of rare network attack categories. Paper presented at the international conference on recent trends In information technology (ICRTIT), pp 417–422.
9. Mohammadi M, Raahemi B, Akbari A, Nassersharif B (2011) Class dependent feature transformation for intrusion detection systems. In: 19th Iranian conference on electrical engineering (ICEE)
10. Tavallaee M, Bagheri E, Lu W, Ghorbani A-A (2009) A detailed analysis of the KDD CUP 99 data set. In: Second IEEE symposium on computational intelligence for security and defence applications
11. Wang T, Mabu S, Lu N, Hirasawa K (2011) A novel intrusion detection system based on the 2-dimensional space distribution of average matching degree. In: SICE annual conference, Tokyo
12. Harb HM, Desuky AS (2011) Adaboost ensemble with genetic algorithm postoptimization for intrusion detection. Int J Comput Sci Issues 8(5):28–33
13. Wu S, Banzhaf W (2010) The use of computational intelligence in intrusion detection: a review. Appl Soft Comput 10:1–35
14. KDD Cup (1999) [Online]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
15. The NSL-KDD Data Set [Online]. http://iscx.ca/NSL-KDD/

16. McHugh J (2000) 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 3(4):262–294
17. Mahoney MV, Chan PK (2003) An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. In recent advances in intrusion detection (RAID2003), Lecture Notes in Computer Science, vol. 2820. Springer-Verlag, pp 220–237
18. Kohavi R (1996) Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the second international conference on knowledge discovery and data mining
19. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, Inc., Los Altos
20. Aldous D (1991) The continuum random tree. I. Ann Probab 19:1–28
21. Kromer P, Platos J, Snasel V, Abraham A (2011) Fuzzy classification by evolutionary algorithms. In: IEEE international conference on systems, man, and cybernetics
22. Panda M, Abraham A, Patra MR (2010) Discriminative multinomial Naive Bayes for network intrusion detection. In: Sixth international conference on information assurance and security
23. Panda M, Abraham A, Patra MR (2012) A hybrid intelligent approach for network intrusion detection. Procedia Eng 30:1–9
24. Sethuramalingam S, Naganathan ER (2011) Hybrid feature selection for network intrusion. Int J Comput Sci Eng 3(5):1773–1780
25. Koshal J, Bag M (2012) Cascading of C4.5 decision tree and support vector machine for rule based intrusion detection system. Int J Comput Netw Inf Secur 4(8):8–20
26. Naoum RS, Abid NA, Al-Sultani ZN (2012) An enhanced resilient backpropagation artificial neural network for intrusion detection system. Int J Comput Sci Netw Secur 12(3):11–16
27. Naoum RS, Al-Sultani ZN (2012) Learning vector quantization (LVQ) and k-nearest neighbor for intrusion classification. World Comput Sci Inf Technol J 2(3):105–109
28. Eid HF, Darwish A, Hassanien AE, Abraham A (2010) Principle components analysis and support vector machine based intrusion detection system. In: 10th International conference on intelligent systems design and applications (ISDA)
29. Zhang Yichi, Wang Lingfeng, Sun Weiqing, Green Robert C, Mansoor A (2011) Distributed intrusion detection system in a multi-layer network architecture of smart grids. IEEE Trans Smart Grid 2(4):796–808
30. Lakhina S, Joseph S, Bhupendra V (2010) Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD. Int J Eng Sci Technol 2(6):1790–1799
31. Saurabh M, Neelam S (2012) Intrusion detection using Naive Bayes classifier with feature reduction. Procedia Technol 4:119–128
32. Kumar S, Nandi S, Biswas S (2011) Research and application of one-class small hypersphere support vector machine for network anomaly detection. In: Third international conference on communication systems and networks (COMSNETS)
33. Gandhi GM, Appavoo K, Srivatsa SK (2010) Effective network intrusion detection using classifiers decision trees and decision rules. Int J Adv Netw Appl 2(3):686–692
34. Kim G, Lee S, Sehun K (2014) A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Syst Appl 41:1690–1700
35. Pereira CR, Nakamura RYM, Costa KA, Papa JP (2012) An optimum-path forest frame work for intrusion detection in computer networks. Eng Appl Artif Intell 25:1226–1234
36. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18
37. Good IJ (1965) The estimation of probabilities: an essay on modern Bayesian methods. MIT Press, Cambridge
38. Langley P, Iba W, Thompson K (1992) An analysis of Bayesian classifiers. In: Proceedings of the tenth national conference on artificial intelligence
39. Fayyad UM, Irani KB (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th international joint conference on artificial intelligence
40. Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. In: Prieditis A, Russell S (eds) Machine learning: proceedings of the twelfth international conference. Morgan Kaufmann, Los Altos, pp 194–202
41. Breiman L, Friedman J, Olshen R, Stone P (1984) Classification and regression trees. Wadsworth, Belmont
42. Utgoff PE (1988) Perceptron trees: a case study in hybrid concept representation. In: Proceedings of the seventh national conference on artificial intelligence
43. Brachman RJ, Anand T (1996) The process of knowledge discovery in databases. In: Advances in knowledge discovery and data mining. AAAI Press, CA, USA, pp 37–57
44. Jagannathan G, Pillaipakkamnatt K, Wright RN (2012) A practical differentially private random decision tree classifier. Trans Data Priv 5:273–295
45. Zhang K, Fan W (2007) Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. Knowl Inf Syst 14(3):299–326
46. Kittler J, Hatef M, Duin RPW, Matas J (1998) On combining classifiers. IEEE Trans Pattern Anal Mach Learn 20(3):226–239