Azerbaijan Journal of High Performance Computing, Vol. 2, Issue 2, 2019, pp. 97-112
https://doi.org/10.32010/26166127.2019.2.2.97.112

Introducing a New Intrusion Detection Method in The SDN Network to Increase Security Using Decision Tree and Neural Network

Ebrahim Zaheri Abdevand[1], Shamsollah Ghanbari[1], Zhanat Umarova[2], Zhalgasbek Iztayev[2],

[1]Islamic Azad University, Ashtian, Iran, server_kia@yahoo.com, myrshg@gmail.com
[2]South Kazakhstan State University, Shymkent, Kazakhstan, zhanat-u@mail.ru, zhalgasbek71@mail.ru

*Correspondence:
Shamsollah Ghanbari,
Islamic Azad University,
Ashtian, Iran,
myrshg@gmail.com

Abstract

Computer networks are difficult to use due to the large number of devices such as router, switch, hop, and many sophisticated security management protocols, but in networks defined with integrated management and configuration software, software-based networks are nowadays important. They are high-end and will become one of the most used and important communication tools in the IT world in the future. In these networks, like all other networks, data security and protection is crucial because a network that is not secure will not work, in this paper, we present a new method of intrusion detection in this network, which consists of two parts: training and testing. Looking to determine if the network is normal or not? By checking the output of these two categories, the current status of the network is determined. The proposed method uses decision tree and neural network. In each first class of tree, the classification of abnormal data is classified and in the second class, the norm data is in decision tree. The output of the decision tree is neural network input Shows that the proposed method performs well.

Keyword: SDN network, security, intrusion detection

1. Introduction

The weaknesses demonstrate the penetration of different political, financial, military and even motivational motives in the security systems of computer networks. In order to counter intruders with computer systems and networks, several methods have been developed, known as intrusion detection methods. The purpose of intrusion detection is to identify unauthorized use, misuse, and damage to computer systems and networks by both internal users and external attackers. Intrusion Detection Systems Effective classification models and patterns for detecting normal behaviors are abnormal behaviors. In the area of intrusion detection, two different methods can be considered: malware detection and malware detection. The main idea in detecting abuse is to display patterns in patterns so that even changes in these attacks can be identified. Based on these patterns and templates, the detection of malware can be detected by the known changes in the network traffic that cause the attack. The main disadvantage of this method is its inability to detect attacks whose traffic behavior is not predicted.

The main purpose of the anomaly detection method is to construct a statistical model to define normal traffic. In this case, any deviation from this model can be considered as anomalous and can be diagnosed as an attack. Using this method is theoretically feasible to diagnose unfamiliar attacks, although in some cases, this may result in a high rate of error reporting. The ability to detect unfamiliar attacks has been a source of growing interest in developing new methods for detecting malformations in recent years (Ali, Sivaraman, Radford & Jha, 2015), Scott-Hayward, Natarajan & Sezer, 2015, Kim & Feamster, 2013).

The emergence of software-defined network architecture - and the separation of control - from the data-level - is accompanied by advantages and disadvantages that have posed challenges in designing anomaly detection systems. Attacks can have a major impact on SDN architecture (Nunes, Mendonca, Nguyen, Obraczka & Turletti, 2014). The most dangerous situation that can occur is when the SDN controller is attacked, which can occur by exploiting the weaknesses of the processes and services implemented in the controller. As a result, the entire SDN network will be affected, and attackers will be able to take control of the entire network (Van Trung, et al., 2015, October, Narayanan, 2013, October).

2. Software-based networks (SDN)

SDN has been recognized as the most promising solution to the challenges of controlling complicated and complex networks. SDN made its commitment by separating network control functions from network switch components. By moving control plans from network components to stand-alone servers, switching components can be simple, cost-effective for general purposes, and at the same time control plans can rely on the design principle of distributed systems rather than distributed routing protocols in their implementations (Dang-Van & Truong-Thu, 2017).

SDN is a scheme where the central software program calls the controller to dictate the behavior of the overall network. In SDN network equipment, packet forwarding equipment is simple (data plan), while “brain” or control logic is implemented in the controller (control plan). This scheme offers many advantages over traditional methods. First, it is much easier to introduce a new idea in the network by software, modify and manipulate the fixed commands on the proprietary network equipment. Second, SDN introduces the benefits of centralized methods in network configuration. In contrast to distributed management, operators do not have to configure the entire network equipment to change the network method, instead making massive traffic forwarding decisions at a rational location in the controller with global information creates network status (Narantuya, 2015).

SDN has changed the way design and management of networks based on the logic of a centralized SDN controller, which is directly controlled by the packet management functions of network switches. It uses the API as standard as OpenFlow. The OpenFlow switch has a table of closed management rules, where each rule has a pattern (corresponding to bits in the closed header), a list of actions (e.g., drop, flood, forward out header, changing header field, or sending a packet to the controller), the set of counters (tracking the number of bytes and packets), and the priority (ambiguity between rules with patterns that overlap). By the received packet, the OpenFlow switch identifies the highest priority that matches the rule and authorizes the action and increases the counter. According to the rule set by the controller program, the OpenFlow switch can behave like a router, switch, network address interpreter or something (Bozakov & Papadimitriou, 2014, May).

The concept of the standard was first put forward by researchers at Stanford, Nick McEwen, Martin Casado, and others. The first specification was released in December 2007. The researchers aimed to create an environment for communication with the campus industrial production network protocols. The study was conducted in an environment free of cost or the construction of experimental networks or the impact of generated traffic on the network. The first out-of-campus focus on OpenFlow as a bandwidth scaling tool came from large data centers. The ONF Foundation began in 2008 intending to promote a new form of SDF networks compatible with the OpenFlow protocol. To this end, the Foundation is responsible for standardizing the OpenFlow protocol. Unlike most IT standardization groups or consortiums, the ONF Foundation was not founded by infrastructure technology providers but by companies that were eager to use the technology, such as Google, Facebook, Microsoft, Yahoo, and four others (Campbell, & Ying, 2011).

3. Previous methods

Researchers have worked on the detection of anomalies and have used neural network combinations with fuzzy clustering to solve both the problems of low accuracy of diagnosis and the stability of diagnosis based on neural network based intrusion system. The authors of the article (Scott-Hayward, S., 2015, April) used a two-way neural network to detect malformations and malformations. Radial basis function networks have been used as a real-time clustering model, and elemental networks are used to reconstruct memory from past events.

Dangovas, V., & Kuliesius, F. (2014, January) present an important study of the use of neural networks for the detection and classification of intrusions. The purpose of the study is to determine which neural networks categorize the attacks well, leading to a higher detection rate for the attack. This study focuses on categorizing two types of records: a single class (normal or attack) and several classes. Where neural networks also classify attacks. In this work, five types of neural networks have been tested.

Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., & Maglaris, V. (2014) used a reduced neural network whose size was reduced based on the grouping of inputs. Reducing network size speeds it up. Incoming features are divided into four categories Content features Basic features Time-based traffic features and Host-based traffic features. In this network, the connection between the input layer and the hidden layer is based on the category of attributes. The advantage of the above method is to reduce the impact of the attack because the network is quickly cleared of the attack, but its drawback is the accumulation and congestion in some switches.

Dotcenko, S., Vladyko, A., & Letenko, I. (2014, February) conducted a face-to-face research in which two different methods were examined in terms of accuracy and time. Part of the KDD dataset was selected as the training data and then applied to the dataset for better result of the feature selection algorithm PCA used in the next step, SVM + DT + Boosting and SVM + DT + Bagging. The comparison is applied and evaluated. The above method is very accurate, but the processing time is high.

The paper [1] presented a research need on different machine learning algorithms. In the method presented by the author, first, by using the genetic algorithm, the effective features of the KDD dataset, which is 2 in number, are reduced, and several different classifications based algorithms are used to investigate and compare the results in this study. It is suggested that the use of the feature selection module improves the speed and precision of the performance of the segment and ultimately compares several machine learning methods as classifiers, the full results of which are presented by Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). The disadvantage of the method described above is that it should be fully information about the network and not applicable to short circuits where not much information is available.

Porras et al. (2015, February) used an algorithm to detect infiltration in software-centric networks, and in this algorithm, each particle is distributed in the network environment and case of abnormal states, the pheromone value is increased and finally the particles that More than average they have intrusive states, one of the major disadvantages of this method is the dependence of the proposed method on the network topology, which makes it not available in all networks.

In the paper by Wang et al. (2015, June), a combination of three genetic algorithms, decision tree, and neural network is used to detect the intrusion. In this algorithm, the neural network input is optimized with genetics and decision tree and five-layer neural network is used. Separating and combining data with one module increases processing time.

Kim et al. (2015, October) used data mining techniques to analyze network data. This research, with a comprehensive review of all the articles in this topic, has provided a review article to summarize the methods presented in evaluating intrusion detection systems. Finally, a comprehensive comparison between the existing methods is presented.

In a study by Sayeed et al. (2015, September), multiple credit rating methods including logistic regression, linear discriminant analysis, k-nearest neighbor, neural network, decision tree, backup vector machine on eight real datasets have been compared and concluded that the methods are neural network and support vector machine have been better than other methods in classification. The main disadvantage is the high consumption of resources involved.

4. The proposed method

We propose a method for detecting intrusion into the sdn network by combining the following classification methods.

- Selecting 80% of data as training and 20% as testing
- Classification of test data by three methods neural network, svm vector machine, and decision tree
- The result of the status of the test data is equal to the majority vote of the three classes proposed

4.1. Classification of Test Data by Neural Network

RBF networks have the most use after pre-release networks. As its name implies, this network uses radial functions. The RBF network consists of a hidden layer of basic functions or neurons. In each input neuron, the input distance and the center of the neuron are calculated. Then they give the distance value as the input to the base function, and its value is the output of the neuron. The final output value of the RBF network is calculated based on the weighted sum of the outputs of the base functions with the unit bias value. Based on the research done, RBF neural networks have the following two advantages over other neural networks:

They can model nonlinear functions using only one hidden layer, making it easy for the user to guess how many hidden layers there are.

By changing the linear composition of the output layer neuron, learning of the RBF network can be accelerated.

In this study, as the goal was to detect attacks from normal access, so we assigned a class of normal number and otherwise assigned the number 0 so that the neural network could perform smoothly.

The details of the neural network are as follows:

This neural network has eight inputs, five hidden layers and outputs. The inputs are the same as the vector properties of the IOT network; in each hidden layer, there is a four-value property and the output is the test data class. For each output, there is a relation as follows:

$$\text{Result}=w_1F_1+w_2F_2+w_3F_3+w_4F_4+w_5F_5$$

F_1, F_2, and ... are the same k attributes selected and w is the weight of each attribute. The values of w are obtained based on the values of the training data.

The neural network uses the perceptron algorithm to learn the coefficients as follows.

1. We assign random values to weights
2. Apply perceptron to individual training examples. If the example is found to be incorrect, we correct the values of the perceptron weights.
3. Are all training examples evaluated correctly?

• Yes (end of algorithm)

• No; Return to Step 2

After the training data is obtained using the coefficients, the test data is entered into test data, which is the vector whose normal attack or normal status is not specified to be computed. And then, the calculation of the neural network coefficients is updated to make it more accurate for subsequent records. Each test record becomes a training record after the result.

• The percentage of similarity of each test data is calculated with the training data in the abnormal state

• The percentage of similarity is equal to the number of common features that the vector has with the training data vectors; the latter is expressed as a percentage, and the highest value is chosen as a percentage of similarity (Karimazad, R., & Faraahi, A., 2011, September).

4.2. Data classification by SVM

SVM support vector machine is a learning process based on statistical learning theory, which is one of the best machine learning methods used in data mining. It has been very successful in issues of data classification and pattern recognition, such as text categorization, face recognition in images, handwriting recognition and bioinformatics.

SVM is a binary classifier that separates two classes using a linear boundary. This method uses all bands and an optimization algorithm to obtain the samples that form the boundaries of the classes. These examples are called support vectors. Some training points that are the closest to the decision-making boundary can be considered as a subset for defining decision-making boundaries and as a support vector. Suppose the data is made up of two classes, and the classes in total have x_i = i, i = 1, …, L. The training point is that xi is a vector. These two classes are labeled with y = 1. The optimal boundary method is used to calculate the decision boundary of two completely separate classes. In this method, the linear boundary between two classes is calculated as follows:

1- All class +1 samples are located on one side of the border, and all class-1 samples are located on the other side of the border.

The decision-making boundary shall be such that the distance between the closest teaching samples of the two classes to each other is perpendicular to the decision-making boundary as far as possible.

A linear decision boundary, in general, can be written as follows:

w. x +b =0

X is a point on the decision boundary, and w is a n-dimensional vector perpendicular to the decision boundary. b / ‖ w ‖. Distance between source and decision boundary and w. X represents the interior multiplication of two vectors w and x.

Since multiplying a constant on both sides of the equation will still be equal.

The first step in calculating the optimal decision boundary is to find the closest two-class training examples. In the next step, the distance between those points is calculated perpendicular to the boundaries that completely separate the two classes. The optimal decision boundary is the border that has the maximum margin. The optimal decision boundary is calculated by solving the following optimization problem.

$$\min_{w,b} \min_{i=1,\dots,l} \left[y_i \frac{(w.x_i + b)}{|w|} \right]$$

Given the above equation and performing a series of mathematical operations, the above relation becomes the following relation.

$$\min_{w,b} \frac{1}{2} |w|^2, y_i(w.x + b) - 1 \geq 0 \quad i = 1, \dots, L$$

It is difficult to solve the relation as mentioned above optimization problem. To simplify it by using indefinite Lagrange coefficients, this optimization problem can be transformed into the following form: λ_i are Lagrange coefficients.

$$\max_{\lambda 1,\dots\lambda L}\left[-\frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L}\lambda_i y_i (x_i . x_j) y_j \lambda_j + \sum_{i=1}^{L}\lambda_i\right]$$
$$\lambda_i \geq 0 \qquad i = 1, \dots L$$

$$w = \sum_{i=1}^{L}\lambda_i y_i = 0$$

After solving the above problem and finding the Lagrange coefficients, w is calculated using the following equation.

$$w = \sum_{i=1}^{L}\lambda_i y_i x_i$$

The λ_i of the support vectors will be greater than zero, and λi of the other points will be zero. Therefore, given the high and zero relation of λi to xi that are not support vectors, only a limited number of training points that are support vectors are needed to reach the decision boundary.

4.3. Decision Tree

In this sense, the target is represented in the form of a tree in which the tree is constructed using the principle of recursive partitioning. There are features in this type that are useful as a partitioning feature or as a node based on information criteria, and then the process continues, repeated over and over again for each child node until all features are considered.

Moreover, build a decision tree. Some pruning techniques may be taken to reduce tree size and thus to avoid over-sized. In this section, we use the C4.5 decision tree algorithm in machine learning tools for our experiments.

Algorithm C4.5 is a process for generating an initial decision tree from a training dataset in which the training data is produced as a decision tree. The decision tree is structured with two types of nodes, the leaf node representing a set of data, and the decision node that identifies some tests that must be performed on the value of a single attribute with a branch and sub tree for each test result. The decision tree can be used to classify a new instance. By starting from the root of the tree and moving across it to reach a leaf. In each node without leaf decision, we calculate the result of the test properties in the node, moving on to the root side under the selected tree.

Once the vector attributes are specified, the C5 decision tree for the attributes is constructed as follows:

- This tree is four regular, meaning each node has four children.
- Nothing at the root.
- At the first level (root children) having four nodes, the first field information is stored which has values 1 to 4, from left to right, respectively.
- In the second level, where each node has four nodes in the first level, the second field information is stored, with values from 1 to 4, from left to right, respectively.

At the third level, where each node has a second level of 4 nodes, the third field information is stored, with values from 1 to 4 in the left to right order, respectively.

• In the fourth level, where each node has a level 4 node of 4 nodes, the Num_ Unique_Operators field information is stored, which has values 1 to 4, from left to right, respectively.

• At the fifth level, where each node of the fourth level has four nodes, the fourth field information is stored, which has values 1 to 4, from left to right, respectively.

• At the sixth level of each node, the fifth level has two nodes and the tree leaves are stored, which is the desired vector class.

• Based on the above, the C5 tree is built (Alghuried, 2017).

4.4. Determining the Status of Test Data

So far, the test data has been classified by three classes of neural network, decision tree and svm. According to the output of these three methods, the test data status is as follows:

• If the three classes of clauses + state the status of the test vector + is the test data +

• If two classes of clauses + declare the status of the test vector +, the test data is +

• If three classes of clauses state the status of the test vector - declare the test data

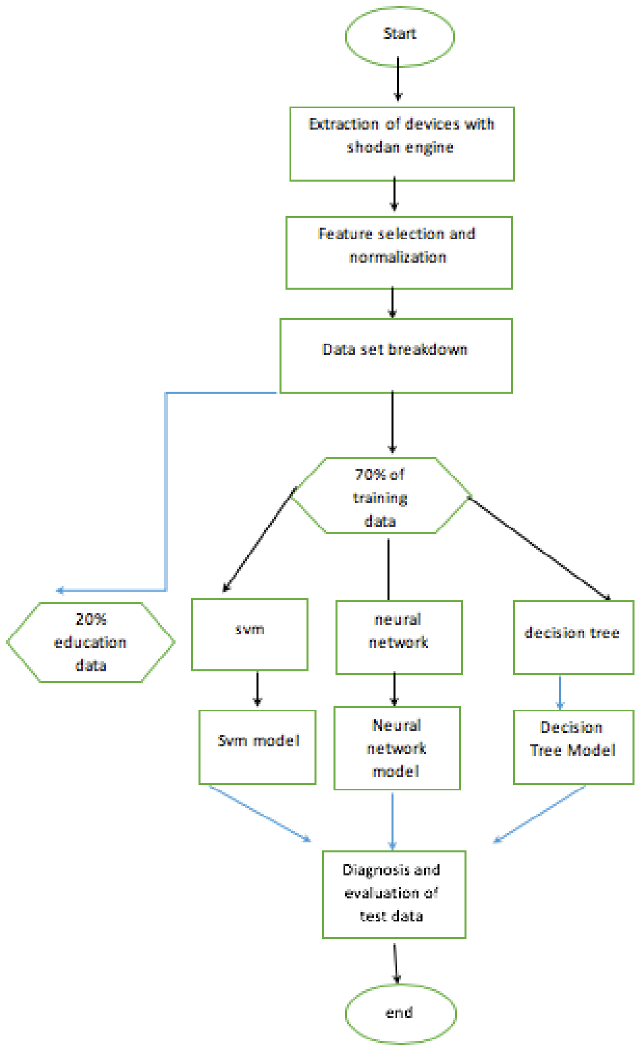
• If two classes of clauses state the status of the test vector - declare the test data

5. Evaluation

To simulate the proposed method, two training and testing datasets were used. In the test phase, first, the data whose status is known are classified into positive and negative groups. The negative group is the data that should be alerted and the positive group is the normal data group and no alert is required. The test dataset is for data that needs to be checked for status that needs to be alerted; according to what was said in the previous chapter, there are four modes for which several criteria then evaluate them. Matlab software was used for the simulation. The following two methods have been selected to compare and evaluate the proposed method:

In the reference written by Boero et al. (2017, September), a mixed negative selection algorithm for the detection of anomalies is presented. The purpose of this algorithm was to improve the negative selection algorithm when faced with high dimensional problems and data. In this study, binary and real identifiers have been used simultaneously. Each input sample is first evaluated with binary identifiers. If binary identifiers identify this sample, this sample is considered to be anomalies. If binary identifiers do not recognize the input sample, it will be evaluated using true identifiers. If binary identifiers do not recognize this sample, this sample will be considered as a normal sample and if identified using these true identifiers, it is considered an anomaly in the system.

Ajaeiya et al. (2017, July) believe that network intrusion detection based on anomaly detection methods plays an important role in protecting networks. Numerous meta-heuristic techniques have been used to produce anomaly detectors. Their paper proposes a hybrid approach to detect anomalies in large-scale databases, using



Flowchart is the proposed method as follows:

multi-start metaheuristic-based discoveries and genetic algorithms. This proposed approach considers the production of negative selection-based discoveries. This method was evaluated using the NSL-KDD dataset, which is a modified version of the KDD CUP 99 dataset. The results (96.1% accuracy) compared to machine learning algorithms indicate the success of this method.

5.1. Data Model

The IST team from the MIT Lincoln Laboratory under the supervision of DARPA and AFRL.SNHS collected the first standardized data to evaluate and assess intrusion detection systems. This information has been used for several weeks in a simulation to test the DARPA intrusion detection system. This dataset is categorized by year of data collection (0-1). The data set of year 6, which was carefully collected and supervised by Bai during his Ph.D. project, was used in the 3rd International Knowledge and Data Mining Competition, D KDDCUP, and the 5th conference in this field. This database contains standard connection records that contain a set of simulated attacks and intrusions into a military network. The attacks in this dataset fall into four main categories: DOS, R2L, U2R and Probe. The KDD99 dataset is organized into multiple files. In this set of files, there are two files called percent_10 and corrected, which in many studies serve as a basis for training and testing of designed systems, respectively (Cup, 1999). In this study, simulations were used to evaluate the features of the Probe section, with a set of 1000 for training and a set of 5000 for testing and evaluation.

5.2. Precision criteria

One of the most important evaluation criteria is the accuracy of the output result, which is compared in Figure 2 with the accuracy of the training data criterion. This criterion shows how accurate the intrusion detection output is.

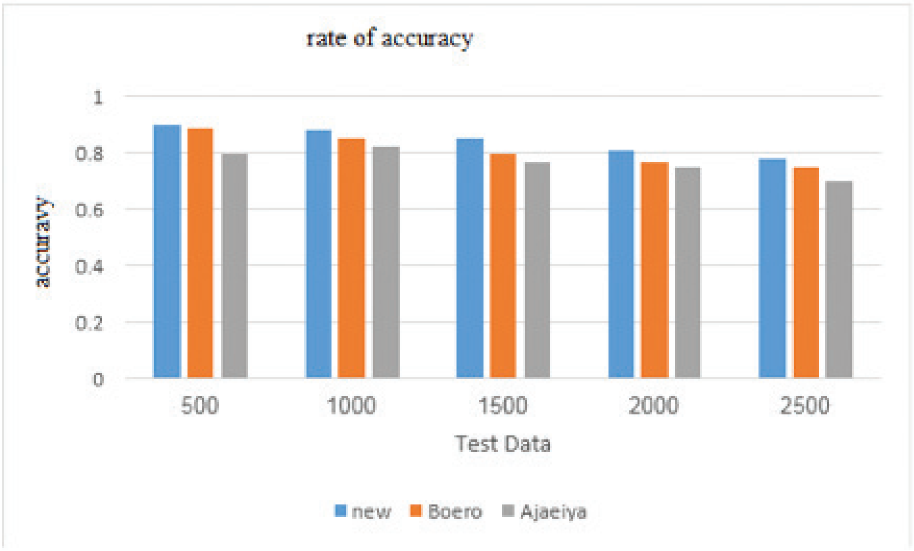


Fig. 2: Comparison of accuracy based on test data

As Figure 2 shows, the proposed method is more accurate and increases as the number of test data increases, as each test data is added to the training data after execution, which increases the accuracy of the test data.

5.3. Runtime criterion

In this section the time of implementation of the proposed algorithm is compared, which is visible in Figure 3.

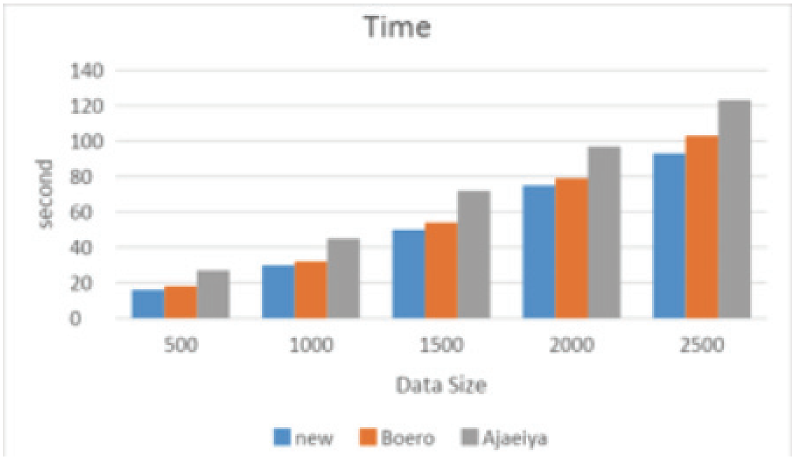


Fig. 3: Comparison of execution time

As can be viewed in Figure 3, the proposed method is faster than the previous methods.

5.4. False Positive Rate Criterion (FPR)

This measure indicates what percentage of alerts are false. To simulate this scenario, the number of attacks on the data is increased at each step. The simulation result is shown in Fig. 4.

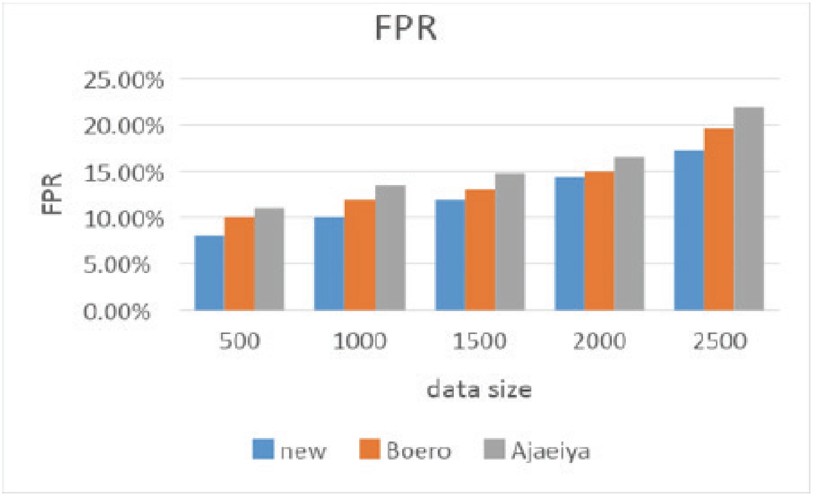


Fig. 4: FPR criterion

According to the results of the simulation, the proposed method performs better. The proposed method is more accurate and the accuracy of the test data increases as the number of test data is added to the training data after execution and this increases the accuracy.

5.5. Incorrect FRR Rejection Criterion

This error occurs when a licensed user system does not erroneously. FRR means the rate of error rejection, also called error number one, and the percentage of times that error occurred.

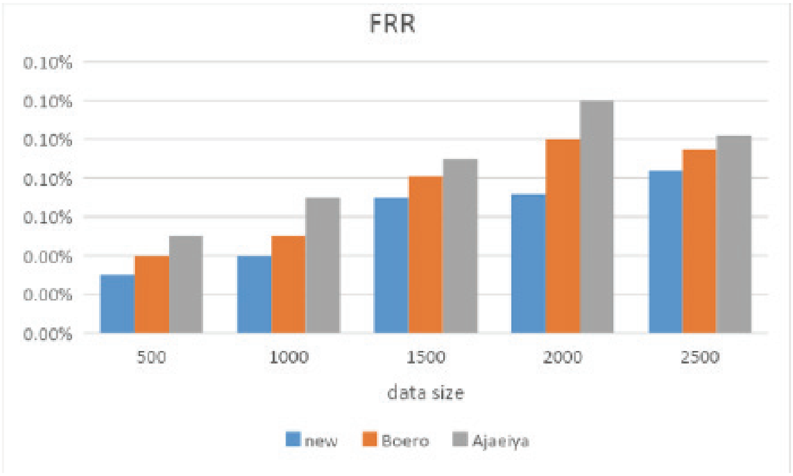


Fig. 5: FRR criterion

As the graph above shows, the proposed method performs better because the proposed method performs better and is less error-prone because it uses two categories for normal and abnormal data, in which both groups of data Normal and abnormal are processed with high accuracy.

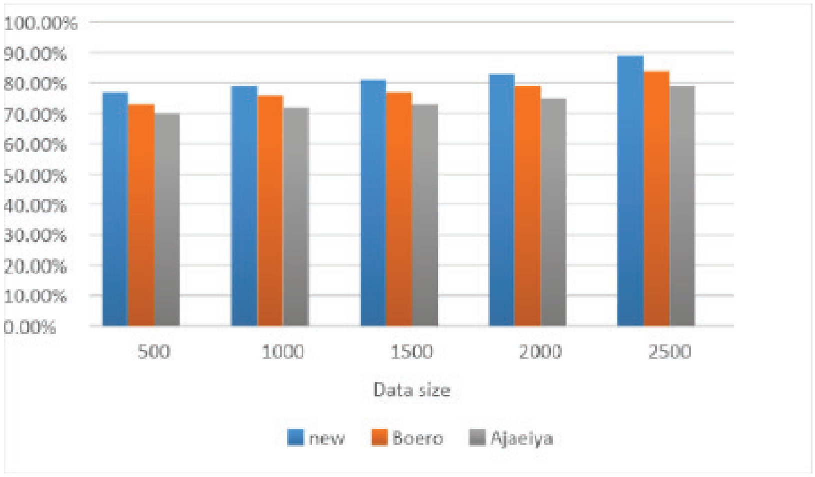


Fig. 6: FRR criterion

5.6 Detection Rate Criterion

To evaluate this scenario the scenario is as the number of training data is increased at each stage and the DR values at each stage are shown in Figure 6.

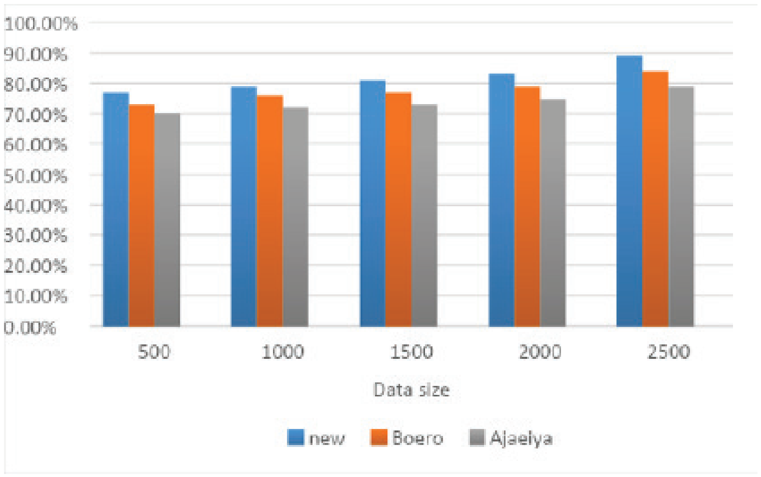


Fig. 7: DR criterion

The simulation results show that the proposed method performs better than the previous method. The reason for the high accuracy of the proposed method is the use of the distance vector mechanism in the ambiguous data segment. When the ambiguous data is detected, the probability of error is increased, but in the proposed method, we apply a statistical method that reduces the error percentage and improves the accuracy.

5.7. EER Equal Error Rate Criterion

Reducing the rate of wrongful acceptance will lead to an unintentional increase

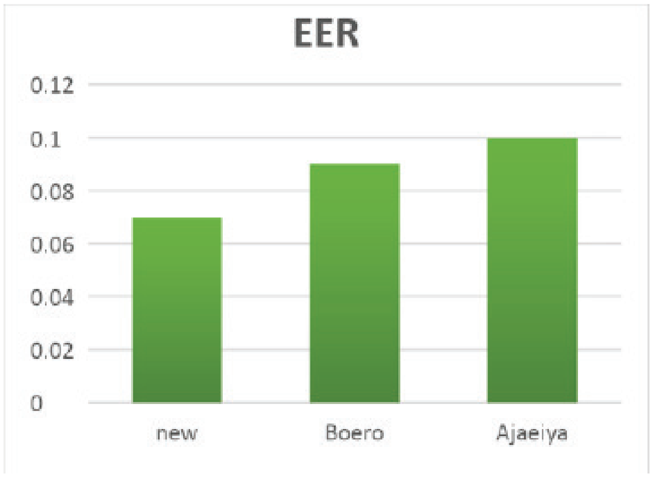


Fig. 8: EER criterion

in the rate of wrongful acceptance. The point at which the rate of error of rejection is equal to the rate of error of acceptance is the point of error rate. The lower this parameter indicates that the system has a better sensitivity and good balance. The result of comparing this criterion can be seen in Figure 7.

As the graph above shows, the proposed method performs better because the proposed method performs better and is less error-prone because it uses two categories for normal and abnormal data, in which both groups of data Normal and abnormal are processed with high accuracy.

6. Conclusion

Users who use the service on the network must be highly secure. One important security solution is the intrusion detection system, which can prevent attacks. Intrusion detection allows us to prevent network attacks to increase data security and users. We used the KDD dataset in the proposed method. This dataset consists of two parts of training and testing. In the training section, we used the classes to classify the statuses by comparing the current situation with the training data. In this paper, we simulate the proposed method, and, as the simulation results show, the proposed method performs better than the pre-performance method and can deliver better results with less accuracy and time.

References

Ajaeiya, G. A., Adalian, N., Elhajj, I. H., Kayssi, A., & Chehab, A. (2017, July). Flow-based intrusion detection system for sdn. In *2017 IEEE Symposium on Computers and Communications (ISCC)* (pp. 787-793). IEEE.

Alghuried, A. (2017). A model for anomalies detection in internet of things (IoT) using inverse weight clustering and decision tree.

Ali, S. T., Sivaraman, V., Radford, A., & Jha, S. (2015). A survey of securing networks using software defined networking. *IEEE transactions on reliability, 64* (3), 1086-1097.

Boero, L., Marchese, M., & Zappatore, S. (2017, September). Support vector machine meets software defined networking in ids domain. In *2017 29th International Teletraffic Congress (ITC 29)* (Vol. 3, pp. 25-30). IEEE.

Bozakov, Z., & Papadimitriou, P. (2014, May). Towards a scalable software-defined network virtualization platform. In *2014 IEEE Network Operations and Management Symposium (NOMS)* (pp. 1-8). IEEE.

Campbell, C., & Ying, Y. (2011). Learning with support vector machines. *Synthesis lectures on artificial intelligence and machine learning, 5 (1),* 1-95.

Cup, K. D. D. (1999). Dataset. *available at the following website http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html, 72.*

Dangovas, V., & Kuliesius, F. (2014, January). SDN-driven authentication and access control system. In *The International Conference on Digital Information, Networking, and Wireless Communications (DINWC)* (p. 20). Society of Digital Information and Wireless Communication.

Dang-Van, T., & Truong-Thu, H. (2017). A multi-criteria based software defined networking system Architecture for DDoS-attack mitigation. *REV Journal on Electronics and Communications, 6* (3-4).

Dotcenko, S., Vladyko, A., & Letenko, I. (2014, February). A fuzzy logic-based information security management for software-defined networks. In *16th International Conference on Advanced Communication Technology* (pp. 167-171). IEEE.

Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., & Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks, 62,* 122-136.

Hong, S., Xu, L., Wang, H., & Gu, G. (2015, February). Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. In ***NDSS*** (Vol. 15, pp. 8-11).

Karimazad, R., & Faraahi, A. (2011, September). An anomaly-based method for DDoS attacks detection using RBF neural networks. In *Proceedings of the International Conference on Network and Electronics Engineering* (Vol. 11, pp. 44-48).

Kim, H., & Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine, 51 (2),* 114-119.

Kim, J., Firoozjaei, M. D., Jeong, J. P., Kim, H., & Park, J. S. (2015, October). SDN-based security services using interface to network security functions. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 526-529). IEEE.

Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *arXiv preprint arXiv:1406.0440.*

Narantuya, J. (2015). *Distributed Sampling Algorithm for Intrusion Detection in SDN Environment'* (Doctoral dissertation, Master Thesis. Gwangju Institute of Science and Technology).

Narayanan, R., Lin, G., Syed, A. A., Shafiq, S., & Gilani, F. (2013, October). A framework to rapidly test SDN use-cases and accelerate middlebox applications. In *38th Annual IEEE Conference on Local Computer Networks* (pp. 763-770). IEEE.

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials, 16* **(3),** 1617-1634.

Porras, P. A., Cheung, S., Fong, M. W., Skinner, K., & Yegneswaran, V. (2015, February). Securing the Software Defined Network Control Layer. In ***NDSS*.**

Sayeed, M. A., Sayeed, M. A., & Saxena, S. (2015, September). Intrusion detection system based on Software Defined Network firewall. In ***2015*** *1st International Conference on Next Generation Computing Technologies (NGCT)* (pp. 379-382). IEEE.

Scott-Hayward, S. (2015, April). Design and deployment of secure, robust, and resilient SDN Controllers. In *Proceedings of the 2015 1st IEEE conference on network Softwarization (NetSoft)* (pp. 1-5). IEEE.

Scott-Hayward, S., Natarajan, S., & Sezer, S. (2015). A survey of security in software defined networks. *IEEE Communications Surveys & Tutorials, 18* **(1),** 623-654.

Van Trung, P., Huong, T. T., Van Tuyen, D., Duc, D. M., Thanh, N. H., & Marshall, A. (2015, October). A multi-criteria-based DDoS-attack prevention solution using software defined networking. In *2015 International Conference on Advanced Technologies for Communications (ATC)* (pp. 308-313). IEEE.

Wang, H., Xu, L., & Gu, G. (2015, June). Floodguard: A dos attack prevention extension in software-defined networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (pp. 239-250). IEEE.

Submitted 10.07.2019
Accepted 21.09.2019