Transactions on Machine Intelligence 2(1) (2019) 48–58

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

Intelligent Intrusion Detection of Computer Networks Using Random Forest Algorithm

S. Moshrefzadeh[1,*], O. Rahmani Seryasat[2], B. Ravaei[3]

[1] Department of Computer Engineering, Faculty of Technology and Engineering, Dana Institute of Higher Education, Yasouj, Iran.
[2] Department of Electrical Engineering, Faculty of Technology and Engineering, Shams Higher Education Institute, Gargan, Iran.
[3] Department of Computer Engineering, Faculty of Technology and Engineering, Yasouj University, Yasouj, Iran.

ARTICLE INFO

Article History:
Received 22 October 2018
Received in revised form
12 February 2019
Accepted 21 March 2019
Available online 22 March 2019

Keywords:
Intelligent Intrusion Detection,
Computer Networks, Random
Forest Algorithm

ABSTRACT

Intelligent Intrusion Detection Systems (IDS) are pivotal in safeguarding computer networks against unauthorized access and cyber threats. These systems are engineered to detect, identify, and classify potential attacks, while also recognizing security vulnerabilities, thereby enabling timely alerts for network administrators. This study delves into the application of the Random Forest algorithm as the core technique for intelligent intrusion detection. The efficacy of the proposed approach was evaluated using the NSL-KDD dataset, a widely recognized benchmark in intrusion detection research. This dataset comprises 125,973 samples with 41 distinct features representing various network traffic characteristics. The Random Forest algorithm, known for its ensemble-based nature, constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) of the individual trees. This method enhances predictive accuracy and controls overfitting. Experimental results indicate that the use of this algorithm significantly improves the accuracy of intrusion detection, achieving a remarkable detection rate of 99.89%. These findings underscore the potential of Random Forest in developing intelligent and reliable IDS, offering a robust solution for real-world network security applications. The study also discusses the algorithm's performance in terms of precision, recall, and F1-score, highlighting its effectiveness in various attack scenarios.

1. INTRODUCTION

In recent years, *Intrusion Detection Systems (IDS)* have emerged as a pivotal topic in the domain of cybersecurity, attracting widespread attention. With the rapid expansion and ubiquity of the Internet, many businesses and financial institutions have transitioned to online platforms, offering services directly to users. The convenience and efficiency of online service delivery have led to a proliferation of web-based

* Corresponding Author: sadeghmoshrefzadeh@yahoo.com
Department of Computer Engineering, Faculty of Technology and Engineering, Dana Institute of Higher Education, Yasouj, Iran.

http://dx.doi.org/10.47176/TMI.2019.48

services. However, this widespread adoption has simultaneously introduced significant vulnerabilities, particularly cyber intrusions and hacking attacks, which pose serious threats to digital enterprises. As data volumes grow exponentially, the importance of advanced data analysis methods becomes increasingly evident [1].

Intrusion detection systems now constitute a core component of network security, designed specifically to detect, identify, and track unauthorized activities within computer networks [2]. Traditional IDS approaches, while foundational, suffer from significant limitations. These systems typically monitor network traffic for suspicious behavior and may detect security breaches or anomalies. Nonetheless, they heavily depend on manual inspection and expert knowledge, making the process both time-intensive and complex.

To address these challenges, the integration of machine learning techniques into IDS has gained traction. Machine learning offers scalable and adaptive capabilities, enabling the classification of vast network data into normal and anomalous (attack) categories. This is especially vital given the increasing sophistication and frequency of cyber threats [3]. As the digital landscape expands, the prevalence of spammers, attackers, and fraudulent entities has surged. Consequently, IDS has evolved into an indispensable element of modern network infrastructure [4].

Historically, the concept of intrusion detection dates back to the mid-1980s. Between 1984 and 1986, Dorothy E. Denning, in collaboration with Peter G. Neumann, developed a real-time IDS based on expert systems a foundational milestone in the field [5]. Subsequently, in 1991, researchers at the University of California, Davis, designed a prototype Distributed Intrusion Detection System (DIDS), which also utilized expert system principles [6]. More recently, in 2015, Viegas and colleagues proposed an anomaly-based IDS architecture tailored for System-on-Chip (SoC) applications within the Internet of Things (IoT) ecosystem [7].

In this study, we employ the Random Forest algorithm to enhance the precision of intelligent intrusion detection systems. The implementation was conducted using the WEKA data mining platform, which facilitated the application of the algorithm. Our results demonstrate that the use of Random Forest significantly improved the accuracy of intrusion detection.

The structure of this paper is organized as follows: Section 2 reviews the existing literature; Section 3 outlines the research background; Section 4 presents the proposed methodology; Section 5 discusses the research findings; Section 6 offers an in-depth analysis and discussion; Section 7 concludes the study; and Section 8 provides suggestions for future research.

2. RESEARCH LITERATURE

In this section, familiarization with the intrusion detection system, types of intrusion detection system, increasing network security using an intrusion detection system, abilities and capabilities of the intrusion detection system, advantages of the intrusion detection system, the needs of IDS systems for the organization's network and common IDS tools are briefly explained.

2.1. What is an Intrusion Detection System?

An Intrusion Detection System (IDS) is a security mechanism designed to monitor and analyze network traffic in order to identify suspicious activities or abnormal patterns that may indicate security breaches. By continuously observing data packets traversing the network, IDSs aim to detect unauthorized or malicious behavior and generate real-time alerts to system administrators. Certain advanced IDS implementations are capable of responding automatically to detected threats, such as blocking traffic from suspicious IP addresses or isolating affected systems to prevent further intrusion.

The primary objective of an IDS is to detect potential intrusions and neutralize threats before they can compromise the integrity, confidentiality, or availability of the network. IDSs can be categorized broadly into two types based on their deployment location:

- **Network-Based Intrusion Detection Systems (NIDS):** These are positioned at strategic points within the network to monitor traffic between multiple devices and detect threats targeting network infrastructure.
- **Host-Based Intrusion Detection Systems (HIDS):** Installed on individual host machines or endpoints, these systems monitor system-level activities such as file access, system calls, and application logs.

Moreover, IDSs can be further classified according to the techniques they use to identify malicious behavior, including signature-based detection, which relies on known attack patterns, and anomaly-based detection, which flags deviations from established normal behavior.

2.2. Types of Intrusion Detection Systems

Intrusion Detection Systems (IDSs) are classified into various types based on their location of deployment and detection methodology:

- **Host-Based Intrusion Detection System (HIDS):** HIDS operates on individual devices within a network, monitoring internal activities and detecting anomalous or malicious behavior originating from within the host itself. It is particularly effective for identifying threats such as malware infections or unauthorized access attempts that arise internally.
- **Network-Based Intrusion Detection System (NIDS):** NIDS is deployed at strategic points within the network infrastructure, typically at gateways or switches, to monitor all inbound and outbound traffic. It analyzes data packets across the network to detect suspicious patterns and potential security breaches.
- **Signature-Based Intrusion Detection System (SIDS):** SIDS identifies known threats by comparing network traffic against a database of predefined attack signatures. This method is effective for detecting previously encountered threats but is limited in recognizing novel or zero-day attacks.
- **Anomaly-Based Intrusion Detection System (AIDS):** AIDS uses machine learning and statistical models to establish a baseline of normal network behavior. It detects deviations from this baseline as potential threats, making it suitable for identifying unknown or evolving attack patterns.

2.3. Enhancing Network Security through IDS

Intrusion Detection Systems can be further categorized into **passive** and **active** types. A **passive IDS** solely monitors and logs suspicious activity or generates alerts without intervening. In contrast, an **active IDS** (also known as an Intrusion Prevention System or IPS) not only detects threats but also initiates countermeasures, such as blocking traffic or isolating compromised nodes. This proactive approach significantly enhances the responsiveness of network defenses.

2.4. Functional Capabilities of IDS

IDSs provide a wide array of functionalities that support cybersecurity operations, including:

- Generating real-time alerts for suspected security breaches.
- Detecting unauthorized modifications in system files and notifying administrators.
- Assisting in interpreting audit trails and system logs, thus improving the transparency and manageability of security monitoring.

- Overseeing critical components such as servers, routers, firewalls, and control systems to detect, prevent, or mitigate cyber threats.

2.5. Benefits of Deploying Intrusion Detection Systems

The implementation of IDS offers numerous advantages for organizational security:

- Real-time monitoring and potential blocking of suspicious agents.
- Improved strategic planning for updating and strengthening security infrastructure.
- Elimination of inefficiencies associated with manual system monitoring.
- Comprehensive visibility into all network hosts and devices.
- Capability to analyze network packets and identify operating system types.
- Enhanced incident response efficiency through accurate data collection and analysis.

2.6. Organizational Need for IDS Implementation

For any organization, maintaining robust network security is paramount. Reliance solely on human expertise is insufficient to safeguard against today's sophisticated cyber threats. Intrusion Detection Systems offer automated surveillance of all incoming, outgoing, and internal network communications, enabling early detection of unauthorized access and abnormal activities. Despite strong security protocols, networks remain vulnerable due to the constantly evolving tactics of cyber attackers. IDSs act as essential tools for IT personnel, improving detection speed, reducing response times, and mitigating potential damages from cyber incidents.

3. BACKGROUND RESEARCH

Numerous studies have applied data mining and machine learning techniques to enhance IDS performance. Two significant approaches are discussed below:

3.1. Hybrid Data Mining and Machine Learning Approach for IDS

One proposed system employs data mining techniques combined with classification algorithms and feature reduction methods to improve detection accuracy. The integration of these methods led to a reported detection efficiency of **96.20%** [8].

3.2. Network-Based Anomaly Detection Using Neural Networks

This approach presents an anomaly-based IDS that leverages neural networks to detect abnormal patterns in network traffic. The system extracts features from network packets, preprocesses them, and applies neural classifiers such as Backpropagation (BP) and PBH (Perceptron-Based Hybrid). The PBH model demonstrated faster convergence and improved classification accuracy, resulting in enhanced detection performance and lower false alarm rates [9].

4. PROPOSED METHOD

This research proposes the use of the Random Forest algorithm for intelligent intrusion detection, focusing on improving detection accuracy and system performance.

4.1. Random Forest Algorithm

The Random Forest algorithm is an ensemble learning technique based on constructing multiple Classification and Regression Trees (CART). Each tree is trained on a different subset of data created through bootstrapping, a method that samples data with replacement. During training, at each split node, a random subset of features is selected, and the best feature among them is used to split the data. This dual-randomization of both data and features ensures diversity among the trees.

As a result, the model benefits from reduced variance and increased robustness. Once the forest is constructed, classification is performed by aggregating the outputs of individual trees through a majority voting mechanism. Approximately 63% of the data is used for training (in-bag), while the remaining 37% is retained for Out-of-Bag (OOB) evaluation, allowing for unbiased performance assessment [11, 12].

This technique's effectiveness lies in its ability to create diverse decision boundaries and reduce overfitting. In the context of IDS, the algorithm can efficiently classify vast amounts of traffic data into normal and attack classes, thereby enhancing the system's ability to respond to cybersecurity threats.

4.1.1. Bagging method

The bagging method was proposed by Leo Berryman in 1996.The bagging algorithm process is as follows; Consider a training set D of size m. Bagging produces a new training set D_i with initial size m by sampling uniformly and by replacing samples from D. Sampling with replacement allows some samples to be repeated in each D_i. This type of sampling is known as self-starting sampling. The output of a combination of n models is obtained by averaging for regression and voting for classification. Therefore, by using resampling and producing different data sets, the required variety will be obtained [12].

4.1.2. Characteristics of random forest algorithm

The random forest algorithm has unparalleled accuracy among the existing algorithms. This algorithm can be efficiently implemented on large databases and can deal with thousands of input variables without variable deletion.

4.1.3. Definition of random forest

Random forest is a classifier of ensembles consisting of decision tree classifiers. Each classifier for each input sample is of the form $h(x,\theta_k)$, where x is an input sample and θ_k is the training set for the k-tree. The θ are independent of each other but with the same distribution. For each sample x, each tree provides a prediction for the class of the sample x, and finally the classes with the highest number of votes of the trees on the input x are selected as the class of the sample, this process is called random forest. The random forest algorithm can increase the prediction accuracy compared to the individual classification tree. In an individual tree, with small changes in the training set, instability occurs, which disrupts the prediction accuracy in the test sample, but the group nature of the random forest algorithm makes it adapt to the changes and eliminates the instability [10].

4.1.4. Out-of-bag estimation

Suppose each classifier is built with a new training set by decision tree method. According to the training set θ and with self-starting method, training sets θ_k are formed. Then the tree classifiers $h(x,\theta_k)$ are made and voting is done from each tree to predict the class. The training samples in the original training dataset that are not in the training set of class k are called out-of-bag samples of class k. In each training set obtained by the self-boot method, out-of-bag samples are about one-third of the samples in the original training set, which are not included in the training set. Equation (1) shows the method of estimating the out-of-bag sample rank on the forest. To obtain the sample category, the prediction of trees whose training set does not contain samples must be collected first, and then the category with the highest average vote on forest tree predictions is considered as the sample category [10].

$$(y(x) = \arg max_c(\frac{1}{k}\sum_{k=1}^{k} I\,(h_k(x) = c, x\epsilon\, OOB_K)) \quad (1)$$

$$(I(h_k(x) = c, x\epsilon\ OOB_K) = \begin{cases} 1\ h_k(x) = c, x \in OOB_K \\ 0\ otherwise \end{cases} \quad (2)$$

Where K is the number of trees, c represents the category h_k (x), the prediction of the kth tree on sample x, and OOB is the set of OOB_K samples of the k tree. Equation (2) shows that the value of the index function will be one if x is in the set of k-tree samples (it is not a member of the k-tree training set) and also the k-tree classifies the sample x to class c. Otherwise, the value of the index function becomes zero.

To obtain the estimation of OOB samples on the forest error (OOB) in relation (3), we use the forest classification error on the OOB samples of tree k [10].

$$I(y_i,\ x_i)\epsilon\ OOB_K = \begin{cases} 1(x_i,\ y_i)\epsilon\ OOB_K \\ 0(x_i,\ y_i) \notin\ OOB_K \end{cases} \quad (3)$$

$$error_k(OOB) = \frac{\sum_{i=1}^{N}(y(x_i)=y_i;(x_i,y_i)\epsilon\ OOB_K}{\sum_{i=1}^{N} I(x_i,y_i)\epsilon\ OOB_K} \quad (4)$$

N is the number of all samples of the main training set, x_i is the i sample on the main training set, y_i is the actual rank of x_i, $y(x_i)$ is the predicted rank for x_i according to equation 1. In relation (4), the value of function I will be one, if the sample (x_i,y_i) belongs to the OOB set of tree k, and otherwise, it is zero [7].

4.1.5. Power and dependence

In random forest, an upper bound is considered for the error. Two parameters are measured for this upper limit. The first parameter is the strength of individual classifications and the second parameter is the dependence between forest trees [10].

$$PE^* \leq \rho(1 - s^2)/s^2 \quad (5)$$

In relation (5), s shows the strength of individual classifications in the forest and $\bar{p}$ shows the dependence between forest trees. Based on this Relationship PE^*, the higher the value of s and the smaller the value of $\bar{p}$, the smaller the error will be. The $\bar{p}$/s^2 ratio for a random forest is a useful guide in understanding its performance. This ratio is the division of dependence by the square of power, and the smaller it is, the better the performance of the forest and the lower the error.

4.1.6. Category prediction by random forest algorithm

To obtain the rank of the test sample for the random forest, the prediction of all the trees in the forest is used. To determine the grade of the test sample, we use the relation (7 and 6) [10].

$$(x) = \arg max_c(\frac{1}{k}\sum_{k=1}^{k} I\ (h_k(x) = c \quad (6)$$

$$(h_k(x) = c) = \begin{cases} 1\ h_k(x) = c \\ 0\ otherwise \end{cases} \quad (7)$$

5. RESEARCH FINDINGS

In this section, which includes the implementation method and evaluation criteria, it will be briefly explained.

5.1. Implementation

For implementation, we use several steps according to Figure (1).

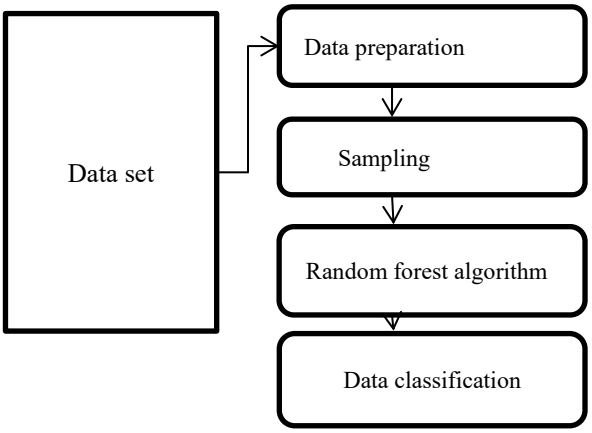

Fig. 1. Flowchart of implementation steps of random forest algorithm.

Dataset: This stage contains 125973 samples, which include 67343 normal data and 58630 abnormal data, loaded from the github site, in which each sample has 41 features (shown in Table 1).

Data preparation: In this step, we add the data set to the Weka tool.

Data sampling: In this step, the number of training and test data sets is selected, which includes seed=1 and fold=10.

Random Forest Algorithm: This step, whose explanation was given in the previous section. To implement the control, we add this algorithm to the tool.

Classification of data: according to the percentage of test samples or test data sets that are classified, the classification accuracy is estimated. The implementation results are shown in Table (2).

Table 1. Important features of intelligent intrusion detection [13].

Attributes	Concept	variable type
Duration	Connection time	Numerical continuum
protocol type	Protocol type	discrete
service	Network service at the destination	discrete
flag	Normal status or connection error	discrete
Src_bytes	Number of data bytes from source to destination	Numerical continuum
dst_bytes	Number of data bytes from destination to source	Numerical continuum
land	A value of 1 if the connection is established and a value of 0 if the connection is disconnected	Discrete two-quantity
wrong_fragment	The number of defective parts	Numerical
urgent	Number of necessary packets	
hot	Number of important indicators	Numerical
num_failed_logins	The number of login attempts	Numerical
logged_in	Value 1 in case of successful login and 0 in case of failed login	Discrete two-quantity

num_compromised	Number of compromised conditions	Numerical
root_shell	The value is 1 if the root shell is obtained and zero if it is changed	discrete
su_attempted	If the root command is done	discrete
num_root	Number of root accesses	Numerical
num_file_creations	Number of file creation operations	Numerical
num_shells	The number of queries in the shell	Numerical
num_access_files	Number of operations on access control files	Numerical
num_outbound_cmds	Number of outgoing commands in ftp session	Numerical
is_host_login	Value 1 if login is required	discrete
is_guest_login	If the user is logged in as a guest	discrete
count	Number of connections to the same host as the current connection in the last two seconds	Numerical
serror_rate	Percentage of connections that have SYN errors	Numerical
rerror_rate	Percentage of connections that have REg errors	Numerical
same_srv_rate	Percentage of connections to the same service	Numerical
diff_srv_rate	Percentage of connections to other services	Numerical
srv_count	Number of connections to the same service as the current connection in the last two seconds	Numerical
srv_serror_rate	Percentage of connections that do not have SYN errors	Numerical
srv_rerror_rate	Percentage of connections that do not have REg errors	Numerical
srv_diff_host_rate	Percentage of connections to other hosts	Numerical
Dst_host_count	Number of connections to the host at the destination	Numerical
Dst_host_srv_count	Percentage of connections to the host at the destination	Numerical
Dst_host_same_srv_rate	Rate of connections to the same host as the current destination	Numerical
Dst_host_diff_srv_rate	The rate of connections to other hosts at the destination	Numerical
Dst_host_same_src_port_rate	The rate of port connections to the same host on the destination	Numerical
Dst_host_srv_diff_host_rate	The rate of connections to other hosts at the destination	Numerical
Dst_host_serror_rate	Destination host error rate	Numerical
Dst_host_srv_serror_rate	The percentage of host error rates at the destination	Numerical
Dst_host_rerror_rate	The rate of repeated host errors at the destination	Numerical
Dst_host_srv_rerror_rate	Percentage of host retry errors rate at the destination	Numerical

Table 2. The results of implementing the random forest algorithm.

Number	**Evaluation criteria**	**Random forest algorithm**
1	(Correctly) The percentage of accuracy and the correct number of samples	99.89%
		125835
2	(Incorrectly) The degree of inaccuracy and the number of incorrect samples	0.1095%
		138
3	(Kappa) access to the quality of performance assurance	0.9978
4	(Mean absolute error) The average amount of real error	0.0028
5	Root mean squared error	0.0313
6	(Relative absolute error)	0.5645%
7	Relative true error rate	6.2742%
8	(Root relative squared error) The amount of root relative square error	125973

5.2. Evaluation criteria

According to table (2), the number of samples that are correctly classified is equal to 125,835 samples. This means that the accuracy of the random forest algorithm is 99.89%. Equation (8) is used to calculate the accuracy:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

According to table (2), the number of samples that are incorrectly classified is equal to 138 samples. This criterion means that the classification error is equal to 0.1095%. The amount of error is calculated based on equation (9):

$$Error = 100 - \frac{TP+TN}{TP+TN+FP+FN} \tag{9}$$

Table 3. Classification details of random forest algorithm

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.999	0.002	1	0.998	0.999	1	normal
0.998	0.001	0.996	0.999	0.999	1	anomaly
0.999	0.001	0.999	0.999	0.999	1	

Table (3) shows the classification details by random forest algorithm.

The first column (TP Rate): shows the correctness of the classification by the random forest algorithm for each type of class.

The second column (FP Rate): the degree of incorrect data classification.

The third column (Precision): indicates the accuracy of the classification of each of the classes in the data set. It is calculated based on equation (10).

$$Precision = \frac{TP}{TP+FP} \tag{10}$$

The fourth column (Recall): The ratio of the number of correct items classified by the algorithm from one class to the number of items in the said class, which is calculated based on equation (11).

$$ReCall = \frac{TP}{TP+FN} \tag{11}$$

The fifth column (F-Measure): According to the calculations made for the Precision and Recall criteria, at this stage the value of the F-Measure weighted quantity can be calculated. F-Measure is a suitable parameter for evaluating the classification quality and also describes the weighted average between two quantities. Precision and Recall. For a classification algorithm in ideal conditions, the value of this quantity is equal to one and in the worst case it is equal to zero. This parameter is calculated using equation (12).

$$\mathrm{F-Measure} = 2 * \frac{Precision*ReCall}{Precision+ReCall} \tag{12}$$

The sixth column (ROC Area): expresses the correctness and incorrectness of the classification according to ROC.

6. DISCUSSION

In the discussion, the proposed method is compared with the previous methods. As shown in Table 4, the proposed method is compared with the previous methods. The previous methods have a low detection accuracy compared to the proposed method. The explanations related to the previous methods are explained in the methods section. The accuracy obtained by the proposed method has a very high accuracy. I hope that in the future it will be possible to increase the accuracy of intelligent intrusion detection by combining data mining algorithms.

Table 4. Comparison of the proposed method with previous methods.

Random forest algorithm	Dimension reduction + decision tree	Bagging algorithm	Logistic regression algorithm
99.89%	97.19%	99.84%	97.10

7. CONCLUSION

Intrusion detection system has become an essential part of computer network security. Which is used to identify and track intruders in the network. We used the random forest algorithm for the accuracy of intelligent intrusion detection. This algorithm can be a very efficient algorithm among other data mining algorithms due to the fact that it is a hybrid algorithm and the connection of several decision trees is created and formed as a forest. For implementation, we used a data set that includes 125,973 samples in which each sample has 41 characteristics, and finally, we used the random forest algorithm and were able to increase the accuracy of intelligent intrusion detection to 99.89%.

8. SUGGESTIONS

By combining different data mining algorithms such as rotating forest algorithm and J48 algorithm, the accuracy of intelligent intrusion detection increased.

CONFLICTS OF INTEREST

The authors declare that there are no financial or non-financial conflicts of interest related to this research.

REFERENCE

[1] Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques* (3rd ed.). Elsevier.

[2] Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16–24. https://doi.org/10.1016/j.jnca.2012.09.004

[3] Salo, F., Nassif, A. B., & Essex, A. (2019). Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Computer Networks*, *148*, 164–175. https://doi.org/10.1016/j.comnet.2018.11.010

[4] Senthil Murugan, N., & Usha Devi, G. (2018). Detecting streaming of Twitter spam using hybrid method. *Wireless Personal Communications*, *103*(2), 1353–1374. https://doi.org/10.1007/s11277-018-5513-z

[5] Denning, D. E. (1986, April). *An intrusion detection model* [Paper presentation]. Proceedings of the Seventh IEEE Symposium on Security and Privacy, Oakland, CA, USA. https://doi.org/10.1109/SP.1986.10010

[6] Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., Ho, C. L., Levitt, K. N., Mukherjee, B., Smaha, S. E., Grance, T., Teal, D. M., & Mansur, D. (1991, October). *DIDS (Distributed Intrusion Detection System) – Motivation, architecture, and an early prototype* [Paper presentation]. Proceedings of the 14th National Computer Security Conference, Washington, DC, USA.

[7] Viegas, E., Santin, A. O., França, A., Jasinski, R., Pedroni, V. A., & Oliveira, L. S. (2017). Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems. *IEEE Transactions on Computers*, *66*(1), 163–177. https://doi.org/10.1109/TC.2016.2560839

[8] Boroumandzadeh, M. (2014). *Presenting a combined data mining and machine learning method for detecting intrusions in computer networks* [Paper presentation]. National Conference on Engineering Sciences, New Ideas, Tonekabon, Iran. https://civilica.com/doc/308424

[9] Alishzadeh, Y., Sadeghian, B., & Safabakhsh, R. (2003). *Network-based intrusion detection using anomaly detection with neural networks* [Paper presentation]. 9th Annual Conference of the Iranian Computer Association, Tehran, Iran. https://civilica.com/doc/45714

[10] Amini Khoei, Z., & Puri, A. (2017). Network traffic classification using improved random forest algorithm. *Computer Science*, *2*(2), 24–38.

[11] Venkatesan, N., & Priya, G. (2015). A study of random forest algorithm with implementation using WEKA. *International Journal of Innovative Research in Computer Science and Engineering*, *1*(6), 156–162.

[12] Cutler, D. R., Cutler, A., & Stevens, J. R. (2012). Random forests. In L. I. Kuncheva (Ed.), *Ensemble machine learning* (pp. 157–175). Springer. https://doi.org/10.1007/978-1-4419-9326-7_5

[13] Wong, J. (2016). *NSL-KDD Dataset (KDDTrain+.arff)* [Data set]. GitHub. https://github.com/jmnwong/NSL-KDD-Dataset/blob/master/KDDTrain%2B.arff