Advances in Military Technology
Vol. 18, No. 1, 2023, pp. 101-117
ISSN 1802-2308, eISSN 2533-4123
DOI 10.3849/aimt.01794

Detection of Malicious Network Activity by Artificial Neural Network

M. Turčaník* and J. Baráth

Department of Informatics, Armed Forces Academy of gen M.R. Štefánik, Liptovský Mikuláš, Slovak Republic

The manuscript was received on 15 December 2022 and was accepted after revision for publication as technical information on 27 April 2023.

Abstract:

This paper presents a deep learning approach to detect malicious communication in a computer network. The intercepted communication is transformed into behavioral feature vectors that are reduced (using principal component analysis and stepwise selection methods) and normalized to create training and test sets. A feed-forward artificial neural network is then used as a classifier to determine the type of malicious communication. Three training algorithms were used to train the neural network: the Levenberg-Marquardt algorithm, Bayesian regularization, and the scaled conjugate gradient backpropagation algorithm. The proposed artificial neural network topology after reducing the size of the training and test sets achieves a correct classification probability of 81.5 % for each type of malicious communication and of 99.6 % (and better) for normal communication.

Keywords:

artificial neural networks, malicious communication system, principal component analysis, stepwise selection method

1 Introduction

Detection, elimination and prevention of security incidents in cyberspace is an important prerequisite for safe and effective use of information technology in globally connected economies [1, 2]. This work uses a trained neural network to search and identify attacks in cyberspace, using pre-trained and pre-intercepted data communications for its operation.

* *Corresponding author: Department of Informatics, Armed Forces Academy of gen M.R. Štefánik, Demänová 393, Liptovský Mikuláš, Slovak Republic. Phone: +421 960 42 30 11, E-mail michal.turcanik@aos.sk. ORCID 0000-0002-6773-213X.*

1.1 Problem Definition

The paper focuses on finding malicious activity in intercepted data communications using a trained artificial neural network. In doing so, it involves the recognition of learned types of malicious activity in forward captured communication obtained from the CSE-CIC-IDS2018 dataset [3]. The idea behind the approach is to transform the coherent data exchanges between nodes in the network into behavioral feature vectors, which are then preprocessed and used to classify the type of attack. Due to the necessary extraction of complete data exchanges between communicating network nodes and the necessary preprocessing, the method is intended for offline analysis, in contrast to approaches described in [4, 5], for example.

The course of the experiments is as follows. From the CSE-CIC-IDS2018 dataset, a training set consisting of 1 800 records (200 records of normal traffic and 200 records each of 8 attack variants) and a test set consisting of 7 200 records (800 records of normal traffic and up to 800 records each of 8 attack variants) were extracted. Each record contained 77 input parameters. For training and testing the feed-forward neural network, three variant sets were used (original – 77 input parameters, transformed using PCA – 18 input parameters, and transformed using stepwise selection method - 23 input parameters).

The reason for optimizing the training and testing sets is to reduce the number of input parameters describing the data stream and to remove noise, while we expect to increase the classification accuracy.

A trained feed-forward neural network is used in the role of the classifier. For each variant of the normalized training and test set, the neural network is trained using three different algorithms (the Levenberg-Marquardt algorithm, Bayesian regularization, and the scaled conjugate gradient backpropagation algorithm) in order to maximize the probability of correctly classifying malicious code and minimize misclassifications.

The paper presents the results of the classification of the malicious activity and the optimization of the input parameters of the training set. The malicious activity classification results are obtained from nine experiments (3 training and test sets times 3 training algorithms) by using an independent test set.

2 Types of Malicious Communication

As examples of malicious communication, we used different variants of Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks, Brute Force attacks, Botnet and Infiltration. These were a total of eight attack variants, and the training and test sets we created included legitimate data communications that we considered to be secure. Each of the attacks is characterized by the type of protocols used, the intensity, as well as the number of attackers or victims. We validated the ability of the neural network to classify the different types of attacks and distinguish attacks from legitimate communications on selected communication samples.

For a better introduction to the issue of malicious communication observation and behavior of selected attacks (DDoS LOIC-HTTP and Infiltration), we show the numbers of attackers and victims, including a sample of the intercepted malicious communication, which is then used for the creation of training and test sets, and recognition using a neural network.

2.1 DoS Attack

The essence of a DoS attack is that (one) attacker, by making a large number of legitimate requests to a selected service, in our case a web service, tries to ideally crash or extremely slow down the service of the selected victim. Several freely available tools can be used for this purpose, in our case the Slowloris Perl-based tool was used to crash the web server.

2.2 DDoS LOIC-HTTP Attack

The category of DDoS attacks is made up of three attack variants (DDoS-LOIC-HTTP, DDOS-LOIC-UDP and DDoS-HOIC), with the individual attacks differing in the target of the attack and the tool that was used for the attack. We will describe the principle of the attack using the first variant of the DDoS attack. The DDoS LOIC-HTTP attack is implemented using the open-source Low Orbit Ion Cannon tool originally developed by Praetox Technologies as a network load testing tool and currently freely available on several open-source platforms [4, 5]. In this case, the attack behavior is well-defined and clearly distinguishable from other types of attacks. Tab. 1 shows the victim's address (Address B) and the attackers' addresses (Address A) on lines 1-10 for the attack, and provides summary information about the amount of data transmitted between the attackers and their victim. Lines 11 to 14 represent an example of normal communication, and the difference in the characteristics of the communication is evident when comparing the data in the Packets, Bytes, and Packets A-B and Bytes A-B columns as well.

Tab. 1 Addresses of the attackers and the victim and the volume of communication during the duration of the attack

Line number	Address A	Address B	Packets	Bytes	Packets A to B	Bytes A to B	Duration (seconds)
1	18.216.24.42	172.31.69.25	144254	8712920	144254	8712920	10830.85
2	18.216.200.189	172.31.69.25	145713	8803158	145713	8803158	10864.12
3	18.218.11.51	172.31.69.25	143828	8687480	143828	8687480	10817.06
4	18.218.55.126	172.31.69.25	145207	8770490	145207	8770490	10820.70
5	18.218.115.60	172.31.69.25	138803	8384642	138803	8384642	10858.60
6	18.218.229.235	172.31.69.25	146691	8862466	146691	8862466	10860.99
7	18.219.5.43	172.31.69.25	144458	8725964	144458	8725964	10834.02
8	18.219.9.1	172.31.69.25	147264	8897152	147264	8897152	10862.58
9	18.219.32.43	172.31.69.25	143178	8647964	143178	8647964	10794.51
10	52.14.136.135	172.31.69.25	145894	8812004	145894	8812004	10831.44
11	60.191.38.77	172.31.69.25	7	735	7	735	5.99
12	71.6.202.198	172.31.69.25	8	649	8	649	139.62
13	79.137.70.1	172.31.69.25	2	108	2	108	0.12
14	103.45.108.96	172.31.69.25	8	693	8	693	473.13

Thus, in our case, the simulated attack is conducted from 10 devices simultaneously against a single victim. In a real deployment, in order to achieve the goal of disabling the service, it is necessary to increase the intensity of the attack, i.e., to deploy an order of magnitude more attackers and increase the duration of the attack as well. Development versions of a tool to perform such an attack and its use are described in reference [4].

Tab. 1 describes the time, intensity and number of attackers involved in the attack, which are input for later analysis and determination of the attack. For a closer look at one complete packet exchange between attacker and victim, see Tab. 2. This is a complete behavioral pattern of the selected attack consisting of 9 packets, which is transformed into the input vector of the test or training set using CICFlowMeter [5, 6].

Tab. 2 Example of communication between attacker and victim

Protocol	Source IP	Source port	Destination IP	Destination port	Attributes	Size (bytes)
TCP	18.219.9.1	64546	173.31.69.25	80	SYN	66
TCP	173.31.69.25	80	18.219.9.1	64546	SYN,ACK	66
TCP	18.219.9.1	64546	173.31.69.25	80	ACK	54
TCP	18.219.9.1	64546	173.31.69.25	80	PSH,ACK	74
TCP	173.31.69.25	80	18.219.9.1	64546	ACK	54
TCP	173.31.69.25	80	18.219.9.1	64546	PSH,ACK	1018
TCP	173.31.69.25	80	18.219.9.1	64546	FIN,ACK	54
TCP	18.219.9.1	64546	173.31.69.25	80	ACK	54
TCP	18.219.9.1	64546	173.31.69.25	80	RST,ACK	54

2.3 Brute Force Attacks

The essence of brute force attacks (in our case there were two variants of the attack – Brute Force-Web, BruteForce-FTP) is an attempt to gain access to a service by guessing login credentials during the authentication phase. The attacker uses a large dictionary of login credentials (or generates them randomly) and repeatedly enters them when prompted to access the service of interest on the victim's computer. A large number of tools and extensive dictionaries of stolen and frequently used credentials are available to perform such an attack; in our test suite, Patator, a multithreaded tool written in Python, was used for the attack to gain access with web and ftp services.

2.4 Botnet

The essence of this category of attacks is to create a network of controlled (malware-infected) computers that perform malicious activities according to the commands of the controlling computer. Such compromised computers can carry out targeted massive attacks on other computers and services on the network.

In the scenario used in the creation of the dataset, the compromised computers do not attack other devices; the botnet is designed to collect and periodically send screenshots of the compromised computers (which are part of the botnet) to the controlling computer, in order for the attacker to obtain information about the data processed on them. The Zeus and Ares tools were used to implement the botnet.

2.5 Infiltration Attack

Unlike the previous attack, this is a group (of two) of activities performed to gain access to the (single) compromised device and then perform malicious activities that vary from case to case and reflect the specific interest of the attacker. The attack starts with the attacker attacking the vulnerable device, gaining control of it, and then another attack is conducted from the already attacked machine. The commentary on the dataset used states that an attacker from within the organization has exploited a vulnerability in a selected application on the victim's computer. He then used the Metasploit framework to create a backdoor so that he could exploit the victim's computer as a source of further attacks on the internal network.

This attack is characterized by a primary attacker and a primary victim who then attacks secondary victims, with the intensity of communication between the different pairs (primary attacker vs. primary victim and primary victim vs. secondary victims) being different. For a closer look at the packet exchange part between the attacker and the initial victim, see Tab. 3.

Tab. 3 Example of communication between the attacker and the initial victim

Protocol	Source IP	Source port	Destination IP	Destination port	Attributes	Size (bytes)
TCP	172.31.69.13	53445	13.58.225.34	31337	SYN	66
TCP	13.58.225.34	31337	172.31.69.13	53445	SYN,ACK	66
TCP	172.31.69.13	53445	13.58.225.34	31337	ACK	54
TCP	13.58.225.34	31337	172.31.69.13	53445	PSH,ACK	58
TCP	13.58.225.34	31337	172.31.69.13	53445	ACK	1514
TCP	172.31.69.13	53445	13.58.225.34	31337	ACK	54
TCP	13.58.225.34	31337	172.31.69.13	53445	ACK	1514
TCP	13.58.225.34	31337	172.31.69.13	53445	ACK	1514
TCP	172.31.69.13	53445	13.58.225.34	31337	ACK	54
TCP	13.58.225.34	31337	172.31.69.13	53445	ACK	1514

The LOIC-HTTP and Infiltration DDoS examples demonstrate the variety of approaches to executing attacks, while highlighting the difficulty of a one-size-fits-all approach for detecting them.

3 The Use of Artificial Neural Network

3.1 Training and Test Set Design

In the training and test set design, we ensured that the training and test sets had a balanced representation of legitimate and malicious communication samples from the CSE-CIC-IDS2018 dataset [3], with the number of input parameters for each approach varying depending on the input parameter optimization method used. The training set

contained 200 samples and the test set contained 800 samples for each category from normal traffic and recognized attacks, and the sets did not overlap with each other. The data were normalized before learning and recognition of the samples by the neural network.

3.2 Artificial Neural Network Topology

The most used ANN topology for the task of classification is the feed-forward neural network. In the next figure is presented ANN for the classification of legitimate and malicious communication. The ANN consists of four layers: an input layer, the first and the second hidden layers, and an output layer.

Input data are represented by seventy-seven parameters (details in CSE-CIC-IDS2018 dataset) and each parameter has been assigned one input of the ANN. The biggest obstacle to using the ANN is setting an appropriate number of neurons in hidden layers. In our case, ANNs with a number of hidden neurons from ten to fifty were analyzed. The result of this optimization process is the ANN with two hidden layers, where the first and the second layers have 20 neurons. The definition of the output layer depends on the specific problem to be solved by the ANN. In our scenario, the ANN will classify the specific threat in the communication stream. The number of output neurons corresponds to the number of classes of analyzed malicious communication that can be found in the data communication.

The output values represent the following types of legitimate and malicious communication: “clean and safe communications”, “Bot”, “DoS attacks-SlowHTTPTest”, “Brute Force-Web”, “DDoS attacks-LOIC-HTTP”, “Infilteration”, “FTP-BruteForce”, “DDOS attack-HOIC” and “DDOS attack-LOIC-UDP”.

The structure of the designed neural network was created in the Matlab development environment. Very good support for work with ANN can be found in Deep Learning Toolbox, which was also used. A pre-trained network is a possible solution that can be used for most deep-learning applications after adaptation to a specific problem. In this paper, we created and trained feed-forward neural networks from scratch using the trainNetwork function. The complete architecture of the neural network was determined by layer parameters and training options were defined for the realization of the training process.

3.3 Artificial Neural Network Training

To correctly implement the defined task by the chosen topology of the artificial neural network it is necessary to select the optimal training algorithm. Since the beginning of the use of the ANN, many training algorithms were invented and applied [7]. In order to avoid the the risk that we will not achieve optimal results by choosing an inappropriate training algorithm, three algorithms were used for training. The same training and testing sets were used by all algorithms to make the results of the trained neural networks comparable.

We will use three of the existing algorithms: Levenberg-Marquardt algorithm, Bayesian regularization, and scaled conjugate gradient backpropagation (SCGB) algorithm.

The Levenberg-Marquardt algorithm is based on iterations that find the minimum of a multivariate criterion function [8]. The Bayesian regularization algorithm is an artificial neural network training algorithm that updates the weight and bias values according to Levenberg-Marquardt optimization [9]. The scaled conjugate gradient

algorithm was invented by Moller and is based on conjugate directions [10]. Presented algorithms will be used to train proposed topologies of the artificial neural networks for detection of selected malicious communication and the results will be compared to select the optimal architecture.

Although the data set used contained a sufficient number of samples, cross-validation was used to achieve the best possible results. In cross-validation, the original data is divided into two subsets, training and testing, and the testing metrics are recorded. Similarly, multiple rounds of cross-validation are performed on different data splits into test and training sets. The test results from all rounds are finally averaged to produce a more accurate estimate of the model's predictive performance. Cross-validation helps reduce variability and thereby limits problems such as the overfitting of the neural network to the training data.

4 Results

The purpose of the proposed feed-forward neural network will be to detect and distinguish the type of malicious communication in network traffic. The size of the input layer of the neural network will be gradually changed according to the number of parameters under investigation, with 77 neurons for the full number of input parameters, 18 neurons for the reduced number of parameters by using PCA method and 23 neurons for the stepwise selection method. The next section will describe the methods for reducing the size of the datasets and the results obtained for training and pattern recognition with a neural network trained with three different training algorithms.

4.1 Defining a Default State with a Complete Set of Input Parameters

To define the initial state and compare the impact of dataset reduction on the accuracy of malicious data stream determination, we first used all 77 input data stream parameters as input. The neural network structure for this task is shown in Fig. 1. The input layer is composed of 77 input neurons, followed by two hidden layers of 20 neurons each, and an output layer composed of 9 neurons (one neuron for each category). Three algorithms were used for training: the Levenberg-Marquardt algorithm, Bayesian regularization and the Backpropagation scaled conjugate gradient algorithm. The experiment resulted in three trained neural networks, and their success rate of correctly recognizing the type of malicious data stream is shown in Fig. 2.

The selected training algorithms achieved approximately the same correct classification rate of 89.2 % to 91.2 % in learning the type of malicious code. The learning was followed by the testing stage, where we used an independent test set of data communication samples. Neural networks trained with the scaled conjugate gradient backpropagation algorithm and the Levenberg-Marquardt algorithm achieved correct classification probabilities of 81.4 % and 81.9 %, respectively. The neural network trained by the Bayesian regularization algorithm achieved unsatisfactory results at 31.1 %, although this algorithm was the best at recognizing the training set. The best results in the classification of the test set were achieved using the Levenberg-Marquardt algorithm 81.9 %. But this number does not reflect the results in the classification of individual types of malicious communication.

A very good way of visualization is the confusion matrix (Fig. 3). In the first row of the confusion matrix, we can see the ANN results for the clean and safe communications classification, where 100 percent success was achieved.

The second row represents "Bot" communication. All 800 samples of this type of attack were classified correctly but 5 were classified incorrectly. Four of them belonged to the class "Brute Force-Web" and one to the class "DDOS attack-LOIC-UDP" malicious communications.

Malicious communications of the type "DoS attacks-SlowHTTPTest" are shown in the third row. Only 251 samples from the test set were classified as an attack of this type. The similarity of this type of attack with the "FTP-BruteForce" type is amplified by the classification error, which is shown in the 7th row of the matrix. An attack classified as "FTP-BruteForce" was actually "DoS attacks-SlowHTTPTest" in 549 cases. In three cases, the samples were classified as "DoS attacks-SlowHTTPTest" although it was 1 sample "Brute Force-Web" and 2 samples "FTP-BruteForce".

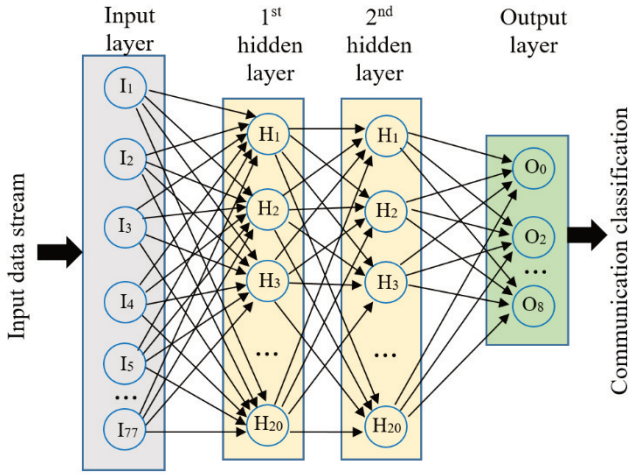

Fig. 1 Topology of feed-forward ANN

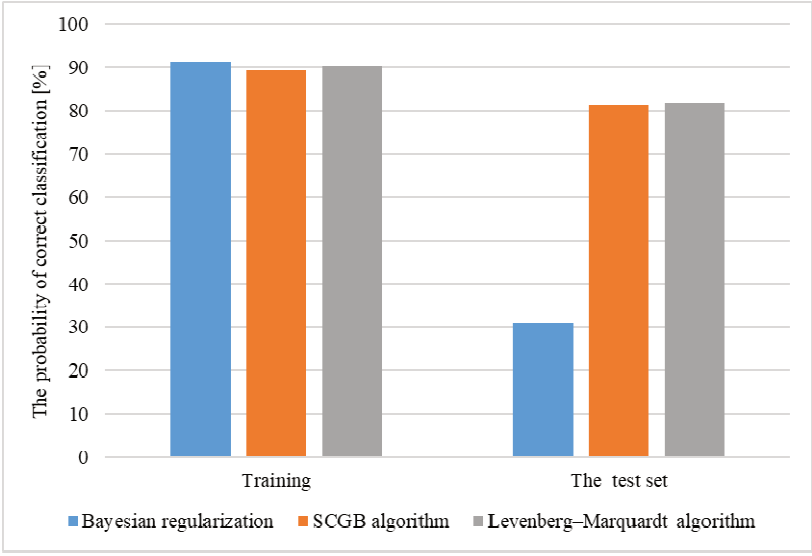

Fig. 2 Feed-forward neural network results for the full set of input parameters

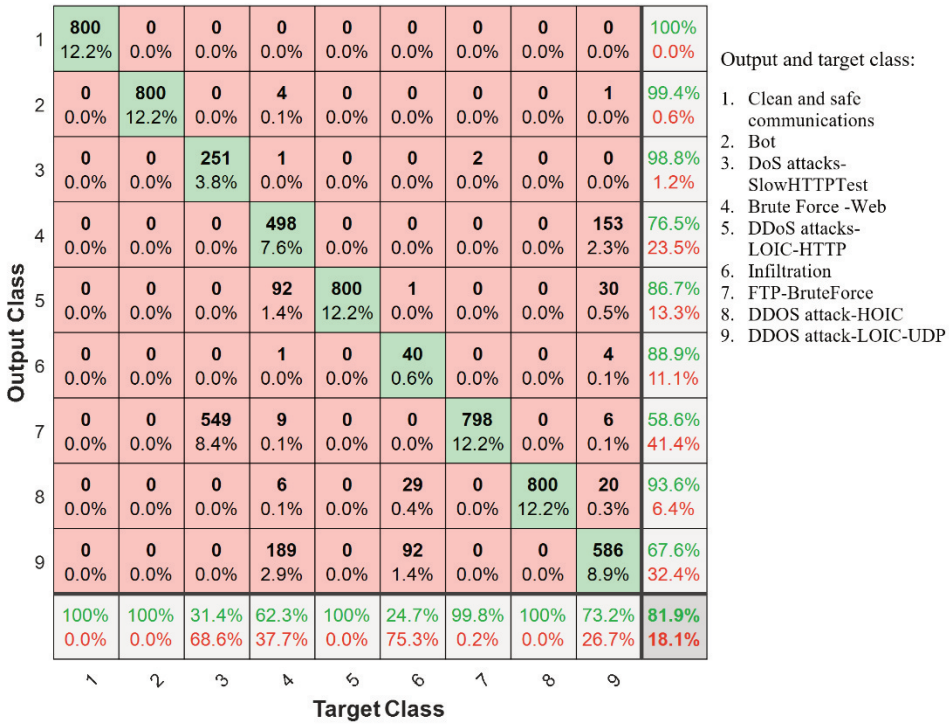

Fig. 3 Confusion matrix for a testing set for the full set of input parameters

"BruteForce-Web" malicious communications are shown in the 4th row. 498 samples from the test set were classified as an attack of this type, while 153 samples were actually malicious communications type "DDOS attack-LOIC-UDP". The similarity of this type of attack with the "DDOS attack-LOIC-UDP" type is amplified by the classification error, which is shown in the 9th row of the matrix. An attack classified as "DDOS attack-LOIC-UDP" was actually "BruteForce-Web" in 189 cases. In 92 cases the samples were classified as "DDOS attack-LOIC-HTTP" although they were samples of "Brute Force-Web".

All 800 samples of "DDOS attack-LOIC-HTTP" were classified correctly (5th row of the table). In addition, 92 ("BruteForce-Web") and 30 ("DDOS attack-LOIC-UDP") samples were incorrectly identified as this type of malicious communications. Forty samples of "Infiltration" were identified correctly. The rest of the samples were classified as "DDOS attack-HOIC" (29 samples) and "DDOS attack-LOIC-UDP" (92 samples). In the same way, we can analyze the results for all types of malicious communications.

4.2 Reduction of Input Parameters Using PCA

Real applications can involve hundreds or even many more measurable factors which must be considered during the learning process. The number of parameters affects the time of the training process and can influence the probability of finding a reasonable solution. Nevertheless, it is frequently feasible to decrease the number of features with the goal of reducing computational complexity. The loss of information and possibly

lower system performance is the main disadvantage of dimensionality reduction, even though the learning process is realized faster. If we realize the training of the proposed artificial neural network with the full set of parameters before the dimensionality reduction of the training data, we can analyze the influence of the decreased number of parameters. The reduction of parameters very often presumes to remove some noise and redundant data. Based on our experiences, we expect a better efficiency of the proposed system for network attack identification. Another advantage of decreasing dimensionality in addition to learning acceleration is a better data visualization. The number of dimensions from two to five is the best from the human point of view. Two or three dimensions are the best to draw or plot on a graph and discover some important observations thanks to visual perception.

There are several methods to solve the dimensionality reduction problem. Examples of feasible methods are principal component analysis [11], factor analysis [12], and others. For our task of malware identification, we will use principal component analysis.

Principal component analysis (PCA) is an ordinal method that allows reducing the number of dimensions in the Euclidean space (defined by correlated variables) so that no information is lost. The number of original mutually correlated (observed) variables is replaced by the new number of mutually uncorrelated (orthogonal) unmeasurable "synthetic" variables so that the first new coordinate axis (the first principal component) is directed in the direction of maximum variability between objects. The second axis (the second principal component) is perpendicular to the first axis and is guided in the direction of the second largest variability between objects, etc.

The relative position of objects in the original space and in the new space (given by principal components) is the same. The original coordinate system is rotated in the direction of max. variability between objects, while the Euclidean distances between objects are preserved.

Fig. 4 presents the value of variability for a given principal component. The first main component has the highest value – approximately 36 % of the total data variability and therefore this component can be taken as the most significant. The value for the second principal component is approximately 21 % and for the third principal component 12 %. The fourth principal component has a variability close to 11 % and the fifth principal component has close to 7 % of the total variability. The sixth and seventh principal components have a value of variability above 2 %. From the eighth component, the value of variability is going close to 1 % or less.

Fig. 5 shows the value of cumulative variability for a given principal component. To achieve a cumulative variability over 90 %, we need to use 7 principal components.

If we use 11 principal components, we achieve a cumulative variability of 95 % percent. The value of cumulative variability changes to 99 % for 18 principal components. We can continue to increase the number of principal components, but the increase in the value of cumulative variability will be imperceptible.

To determine the optimal number of principal components, we have several methods. The most frequently used method, the Kaiser-Gutmann criterion, is based on all principal components whose variability is greater than 1 %. In our application, we set a border to the total variability in the data to 99 %. It means that we need to use the first 18 principal components.

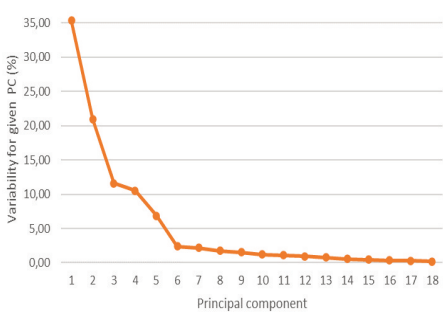

Fig. 4 The variability for the given principal component in % from the total data variability

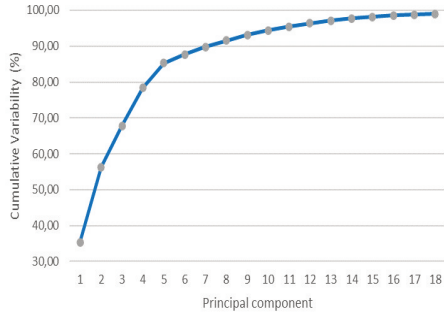

Fig. 5 The cumulative variability for the given principal component

The use of the PCA method ensured the reduction of the initial 77 input parameters to the final 18 new independent variables, which are classified by a trained feed-forward neural network (Fig. 1). The structure of the neural network is as follows: 18 input neurons, 20 neurons each in the two hidden layers and 9 output layer neurons. The training of the neural network defined in this way was carried out by 3 algorithms: the Levenberg-Marquardt algorithm, Bayesian regularization and the scaled conjugate gradient backpropagation algorithm. The training of the neural network was stopped at the moment of the optimal learning point for the corresponding algorithm so as to avoid overfitting the training dataset and having poor performance on the test set. During the training phase, each algorithm achieved a correct classification rate ranging from 88.2 % to 91.7 % (Fig. 6). This was followed by testing the trained neural networks with the prepared test set. The trained neural networks were tested with the prepared test set in the next step. The neural networks trained with Bayesian regularization algorithm and Levenberg-Marquardt algorithm obtained the probability of correct classification at 64 % and 72.4 %, respectively. The best results were obtained with the neural network trained by the scaled backpropagation conjugate gradient algorithm with 81.4 %.

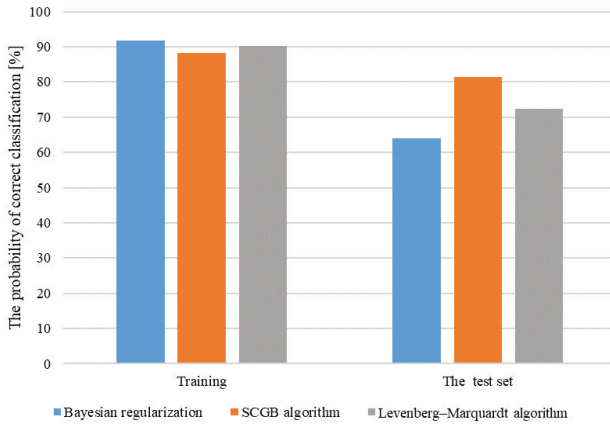

Fig. 6 Feed-forward neural network results for PCA variable selection

More detailed results for testing the classification using ANN with reduction of the input parameters by PCA are displayed in the confusion matrix (Fig. 7). The best results were achieved for "Clean and safe comms", "Bot" and "DDOS attack-HOIC". Other types of malicious communications have similar results (with minor differences for specific cases) as in the previous variant with a complete set of input parameters (Fig. 3).

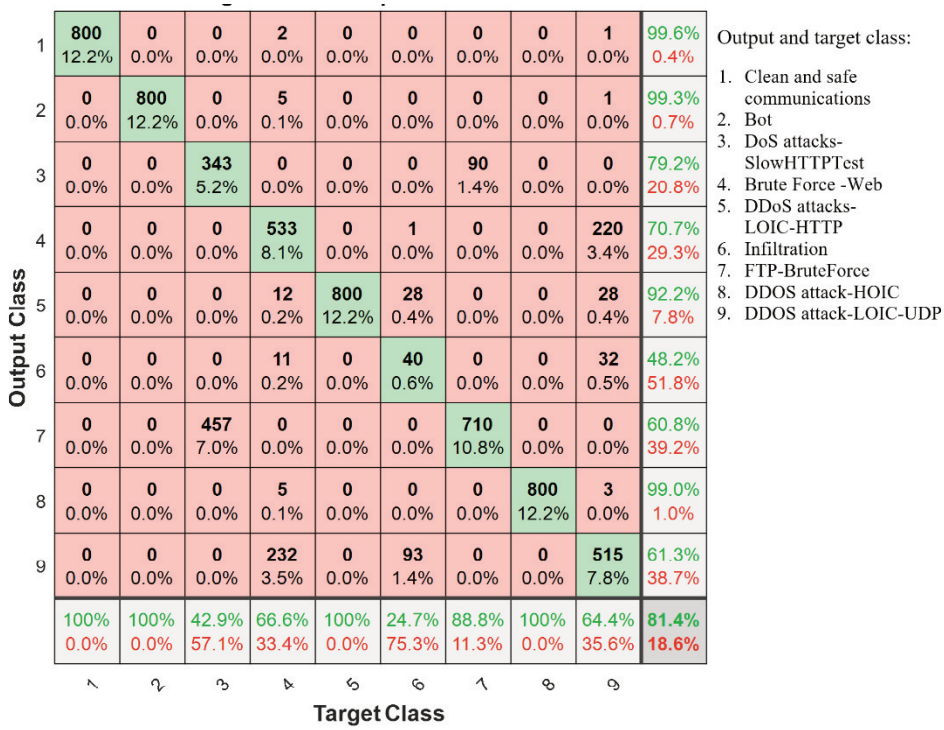

Fig. 7 Confusion matrix for a testing set for the PCA reduction of input parameters

4.3 Reduction of Input Parameters Using the Stepwise Selection Method

The stepwise selection method is used to construct a linear model of the training and test set by using stepwise regression to add or remove predictors to or from the constant models, while requiring knowledge of the system response to the variable. While it is true that larger models have less error in determining the output variable, on the other hand, their prediction ability is reduced [13-15].

Experiments aimed at identifying malicious communication were conducted using our independent training and testing datasets extracted from CSE-CIC-IDS2018 [16], while the characteristics and possible use of the original set are described in [17-19]. From the original set of captured communication, we extracted attack and normal communication patterns using Security Onion 2.3 [20-22] which were then transformed using CICFlowMeter [5, 6], then normalized and ingested into the training and test sets. The sets included normal communication and Botnet attacks, DoS attacks-SlowHTTPTest, Brute Force-Web, DDoS attacks-LOIC-http, Infiltration, FTP-BruteForce, DDOS attack-HOIC, and DDOS attack-LOIC-UDP.

The model created is represented by a function:

$$\text{Label} \sim 1 + \text{FwdIAT}_{\text{Max}} + \text{BwdIAT}_{\text{Max}} + \text{PktLen}_{\text{Mean}} + \text{FwdSegSize}_{\text{Min}} + \cdots \quad (1)$$

where Label is the estimated attack type, followed by a list of 23 predictors. The original output from the CICFlowMeter contained data flows of size 83 variables and a label indicating the flow's affiliation with the attack or normal operation – Label. Irrelevant variables (Timestamp, FlowID, SrcIP, DstIP ...) were removed from the output and the remaining part (consisting of 77 input parameters) of the data stream was used as input to build a linear model using a stepwise selection method.

The obtained predictors allowed reducing the sizes of the training and test sets and, after normalization, were used to train the neural network to recognize the attacks. Fig. 8 shows the estimated effects of changes in the selected predictors on the output value (Label variable) in the linear regression model, with the horizontal line through the effect value indicating the 95 % confidence interval for the effect value.

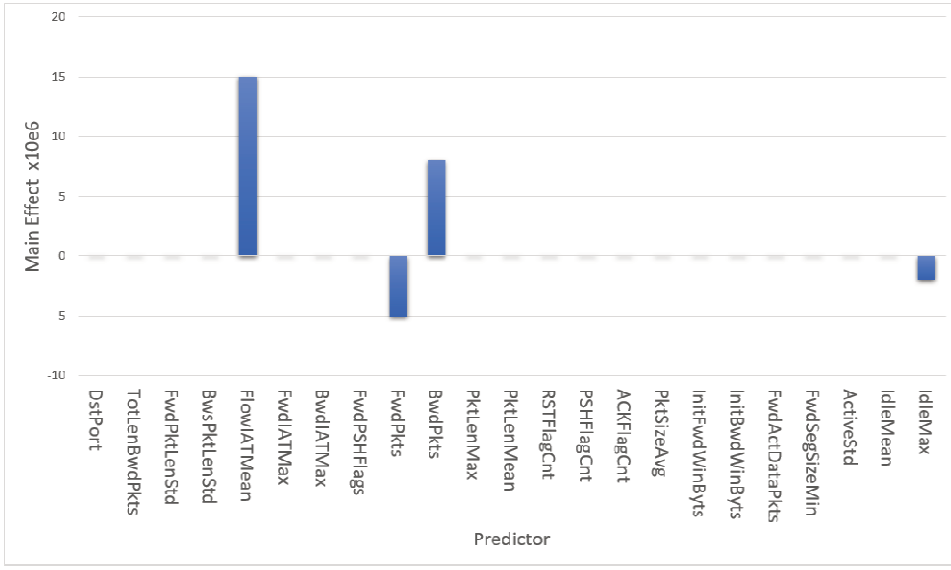

Fig. 8 Effects of selected predictors in the stepwise selection method

This time the neural network (Fig. 1) had 23 input neurons, the number of neurons in the hidden layers and the output layer remained unchanged. For training using the reduced training set, the same training algorithms were used as in the previous cases. The malicious communication recognition results for the training and test sets are shown in Fig. 9. The Bayesian regularization and Levenberg-Marquardt algorithms achieved 91 % and 90 % correct malicious communication type identification success rates in training, and the SCGB algorithm achieved 87.6 %.

The results of malicious communication recognition from the test set show that the best results were achieved by Levenberg-Marquardt algorithm – 81.5 % followed by Bayesian regularization – 80.4 % and SCGB algorithm – 78.8 %. All three algorithms achieved very reasonable results during neural network training (probability of correct classification: 91 %, 87 % and 90 %). The trained neural networks were tested with the test set in the next step, and all three algorithms achieved balanced results of correct classification of each type of attack at a level of just around 80 %.

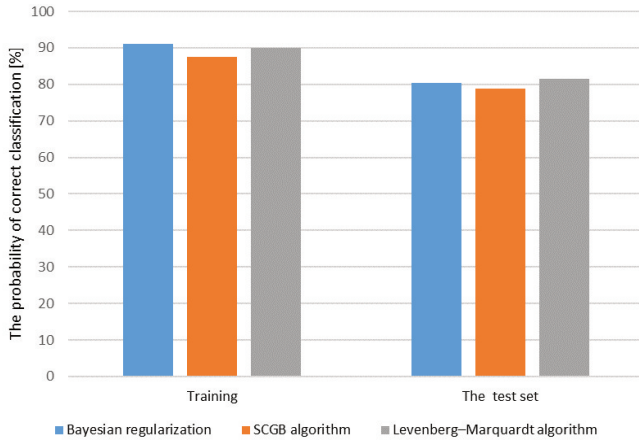

Fig. 9 Feed-forward neural network results for the stepwise selection method

More detailed results for testing the classification using ANN with reduction of the input parameters by the stepwise selection method are displayed in the confusion matrix (Fig. 10). The best results were again achieved for classification of "Clean and safe communications", "Bot" and "DDOS attack-HOIC". Other types of malicious communications have similar results (with minor differences for specific cases) as in the previous variant with a complete set of input parameters (Figs 3 and 7). The similarity according to the legend of the Fig. 10 can be identified in pairs of attacks type 3 and 7, 4 and 9, 4 and 5, 5 and 9, 6 and 9, where errors occurred in the classification of malicious communications.

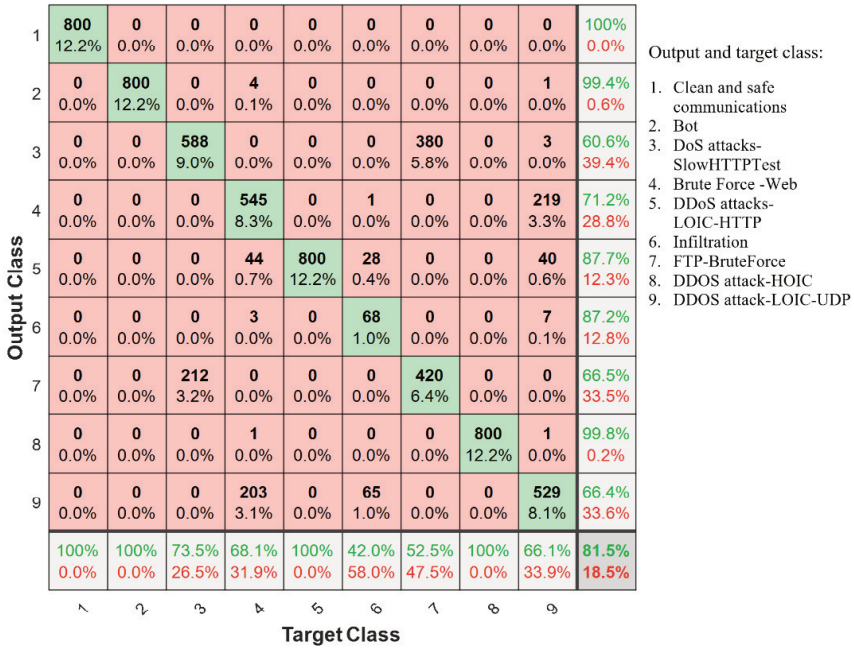

Fig. 10 Confusion matrix for a testing set for the stepwise selection method

5 Conclusions

In the present paper, we have analyzed the possibility of detecting selected types of malicious communication using behavioral feature vectors and artificial neural networks. The size of behavioral feature vectors was reduced by PCA and stepwise selection method. The reduced behavioral feature vectors were used to form a separate training and testing set and three different training algorithms were used to train the artificial neural network: the Levenberg-Marquardt algorithm, Bayesian regularization and the scaled conjugate gradient backpropagation algorithm. If we decrease the number of input parameters from 77 to a smaller number (18 for PCA and 23 for stepwise selection method), we do not significantly change the probability of correct classification of malicious communications. But the decreased number of input parameters allows us to create hardware implementation of the neural network with simplified architecture, which can be retrained faster. In the summarizing Tab. 4, results of all analyzed scenarios in the training and testing phase are presented.

The achieved results have clearly confirmed the similarity between the samples of the given dataset of individual types of malicious communications, which did not allow to achieve better results than stated in 4th part. What is more important, the proposed artificial neural network topology achieves a correct classification probability 100 % in two scenarios and 99.6 % in one scenario for normal communication (confusion matrices Figs 3, 7 and 9). Based on these results, we can distinguish normal and malicious communications almost without error. The proposed artificial neural network can be used not only in offline analysis, but also in almost real-time after preprocessing of input data streams. Smaller ANN can be also retrained faster in this way of use.

Tab. 4 Comparison of results of analyzed methods

Scenario	Training algorithm	Training set	Test set
Full set of input parameters	Bayesian regularization	91.2	31.1
	SCGB algorithm	89.2	81.4
	Levenberg–Marquardt algorithm	90.1	81.9
PCA	Bayesian regularization	91.7	64.0
	SCGB algorithm	88.2	81.4
	Levenberg–Marquardt algorithm	90.2	72.4
Stepwise selection method	Bayesian regularization	91.0	80.4
	SCGB algorithm	87.6	78.8
	Levenberg–Marquardt algorithm	90.0	81.5

Acknowledgement

The article was written as part of the scientific project “Artificial intelligence and its influence on the development of defence capabilities” contract number SEMOD-EL76/9-195/2022-OdPaPV.

References

[1] DROPPA, M. and M. HARAKAL Analysis of Cybersecurity in the Real Environment. In: *Proceedings of the Communication and Information Technologies Conference KIT 2021*. Vysoke Tatry: IEEE, 2021, pp. 92-98. DOI 10.1109/KIT52904.2021.9583748.

[2] DULIK, M. Deploying Fake Network Devices to Obtain Sensitive User Data. In: *Proceedings of the Communication and Information Technologies Conference KIT 2021*. Vysoké Tatry: IEEE, 2021, pp. 87-91. DOI 10.1109/ KIT52904.2021.9583751.

[3] *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)* [online]. [viewed 2022-03-02]. Available from: https://registry.opendata.aws/cse-cic-ids2018/

[4] SAUTER, M. "LOIC Will Tear Us Apart" The Impact of Tool Design and Media Portrayals in the Success of Activist DDOS Attacks. *American Behavioral Scientist,* 2013, **57**(7), pp. 983-1007. DOI 10.1177/0002764213479370.

[5] SHARAFALDIN, I., L.A. HABIBI and A.A. GHORBANI. A Detailed Analysis of the CICIDS2017 Data Set. In: *Information Systems Security and Privacy, 4th International Conference*. Funchal: ICISSP, 2019, pp. 172-188. DOI 10.1007/ 978-3-030-25109-3_9.

[6] SHARAFALDIN, I., A.H. LASHKARI, S. HAKAK and A.A. GHORBANI. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST).* Chennai: IEEE, 2019. DOI 10.1109/CCST.2019.8888419.

[7] BAPTISTA, F.D., S. RODRIGUES and F. MORGADO-DIAS. Performance Comparison of ANN Training Algorithms for Classification. In: *2013 IEEE 8th International Symposium on Intelligent Signal Processing*. Funchal: IEEE, 2013, pp. 115-120. DOI 10.1109/WISP.2013.6657493.

[8] KUMARASWAMY, B. 6 - Neural networks for data classification. In D. BINU and B.R. RAJAKUMAR, eds. *Artificial Intelligence in Data Mining*. Cambridge: Academic Press, 2021, pp. 109-131. ISBN 0-12-820601-2.

[9] BURDEN, F. and D. WINKLER. *Bayesian Regularization of Neural Networks*. Totowa: Humana Press, 2009. ISBN 1-58829-718-7.

[10] BABANI, L., S. JADHAV and B. CHAUDHARI. Scaled Conjugate Gradient Based Adaptive ANN Control for SVM-DTC Induction Motor Drive. In: *Artificial Intelligence Applications and Innovations.* Springer: Cham, 2016, pp. 384-395. DOI 10.1007/978-3-319-44944-9_33.

[11] KARAKAYA, D., O. ULUCAN and M. TURKAN. Pas-Mef: Multi-Exposure Image Fusion Based on Principal Component Analysis, Adaptive Well-Exposedness and Saliency Map. In: *Proceedings of the ICASSP 2022*. Singapore: IEEE, 2022, pp. 2345-2349. DOI 10.1109/ICASSP43922.2022.9746779.

[12] YUE, Y., X. MA and C. ZHANG. Comprehensive Performance Evaluation of the Listed Companies in Coal Mining Industry Based on Factor Analysis and Cluster Analysis. In: *2010 Asia-Pacific Conference on Wearable Computing Systems*, Shenzhen: IEEE, 2010. DOI 10.1109/APWCS.2010.75.

[13] WANG, C., J. SUN, Y. LI, J. ZHAO and B. TIAN. A Comparison of Stepwise Cluster Analysis and Multiple Linear Regression for Hydrological Simulation. *Journal of Physics: Conference Series,* 2022, **2224**, 012026. DOI 10.1088/1742-6596/2224/1/012026.

[14] JAIN, K. and A. SINGH. Data-Prediction Model Based on Stepwise Data Regression Method in Wireless Sensor Network. *Wireless Personal Communications*, 2023, **128**, pp. 2085-2111. DOI 10.1007/s11277-022-10034-3.

[15] HAMID, N.B., M.E. SANIK, H.M. NOOR, J. PRASETIJO, M. MOKHTAR, M.A.M. AZMI, M.I. YAHAYA and M.Z. RAMLI. Prediction Model of Mass Rapid Transit Noise Level Using the Stepwise Regression Analysis. In: *Proceedings of the 7th International Conference on the Applications of Science and Mathematics 2021*. Singapore: Springer, 2022, pp. 379-389. DOI 10.1007/978-981-16-8903-1.

[16] SHARAFALDIN, I., A.H. LASHKARI and A.A. GHORBANI. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*. Funchal: SciTePress, 2018, pp. 108-116. DOI 10.5220/0006639801080116.

[17] DRAPER-GIL, G., A.H. LASHKARI, M.S.I. MAMUN and A.A. GHORBANI. Characterization of Encrypted and VPN Traffic Using Time-Related Features. In: *Proceedings of the 2nd International Conference on Information Systems Security and Privacy – ICISSP*. Roma: SciTePress, 2016, pp. 407-417. DOI n10.5220/0005740704070414.

[18] LASHKARI, A.H., G.D. GIL, M.S.I. MAMUN and A.A. GHORBANI. Characterization of Tor Traffic Using Time based Features. In: *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*. Porto: SciTePress, 2017, pp. 253-262. DOI 10.5220/0006105602530262.

[19] ANDRÉ, C.D.S., S.C. NARULA, S.N. ELIAN and R.A. TAVARES. An Overview of the Variables Selection Methods for the Minimum Sum of Absolute Errors Regression. *Statistics in Medicine*, 2003, **22**(13), pp. 2101-2111. DOI 10.1002/sim.1437.

[20] BEJTLICH, R. *The Practice of Network Security Monitoring: Understanding Incident Detection and Response*. San Francisco: No Starch Press, 2013. ISBN 1-59327-509-9.

[21] *Security Onion* [online]. [viewed 2022-03-02]. Available from: https://securityonionsolutions.com/

[22] *Security Onion Documentation* [online]. [viewed 2022-03-15]. Available from: https://docs.securityonion.net/en/2.3/index.html