Detection and Classification of Anomalies in IP Communications Networks

George Radu Floristean, Andreea Udrea

Faculty of Automatic Control and Computers
University Politehnica of Bucharest, Bucharest, Romania
(email: george.floristean@stud.acs.upb.ro)

Abstract: At this moment, the implementation and development of digital technologies in all fields has involved an overall increase in communications and the number of interconnected devices. In this case, communication systems and networks are constantly threatened, so the detection of adverse security events is an important step in maintaining vital security services, such as their confidentiality, integrity and availability. Intrusion detection systems are the most widely used line of defense in information and communication technology for monitoring and detecting security events. This article aims to identify and classify the anomalies encountered in network traffic by using different machine learning methods and comparing their performances. Results show that random forests are able to classify attack types with high accuracy of about 99%, while support vector machine have a marginal poorer performance at this task.

Keywords: Intrusion Detection Systems, Random Forests, Support Vector Machine, Anomalies detection and classification

1. INTRODUCTION

Communication systems and networks are constantly threatened by attackers, therefore monitoring and detecting security-related events represent the main way to maintain vital security services. For example, two recent major security threats that have been extensively analyzed are worth mentioning: Dyn cyberattack (Chaabouni et al., 2019) and VPNFilter malware (Sapalo et al., 2019). These interfered with thousands of Internet of Things (IoT) devices which have been compromised, causing a high economic impact and, on the other hand, very high personal costs.

Regarding this subject, Intrusion Detection Systems (IDS) are the most appreciated solutions for defense in information and communication technology (ICT) when it comes to monitoring and detecting security events, which are powerful tools to combat various types of attacks. While analyzing network traffic in order to detect normal and abnormal behavior (Dobrescu et al., 2009) and creating fault detection systems (Rughinis and Gheorghe, 2013) have been of interest for the past two decades, continuous improvements have been made recently in this direction (Jabez and Muthukumar, 2015) while their adoption also reached developing countries (Lwoga and Sangeda, 2018).

These systems are classified (Magán-Carrión et al., 2020) into Network-IDS (NIDS) and Host-IDS (HIDS). The purpose of the first one is to analyze network traffic flows, which usually come from firewalls, routers, switches or other network devices, while the second analyzes information from a host (e.g., syslog files). IDSs can also be classified as signature-based IDS (SIDS) or anomaly-based IDS (AIDS). SIDS usually causes malicious behavior compared to predefined attack signatures. AIDS detects system anomalies that are different from the normal behavior of network traffic.

The evaluation of this type of system is generally performed using predefined data sets from simulated scenarios very similar to the final network or application system.

Recently, it can be observed that solutions based on Machine Learning (ML) algorithms (Kim and Kyung-Joon, 2021) and also Deep Learning algorithms (Nicholas et al., 2018; Abu Al_Haija and Zein-Sabatto, 2020) for implementing IDS became extremely popular.

In this context, we can differentiate the proposed solutions based on the training strategy: supervised, semi-supervised and unsupervised approaches. For supervised learning, tagged data sets are needed to classify known attacks and this fact lead to a high interest in creating public datasets with labels (Khraisat and Alazab, 2021).

In the case of unsupervised learning algorithms that do not require labeled data, marked data is recommended anyway to validate the performance of the methods. Finally, both approaches are found in semi-supervised learning. At the unattended stage, abnormal behavior can be detected later during the supervised stage.

Machine learning techniques like Random Forests (RF) and Support Vector Machine (SVM) require uniform input data for their operations, i.e. the characteristics taken into consideration for the training process must be available before the phase in which the models will be implemented and tested.

This article aims to identify and classify the anomalies captured in network traffic using different machine learning strategies, but selecting an algorithm is not a trivial task and depends heavily on the required features and the final application. Finally, in this paper we compare the results of two classical approaches to supervised learning: RF and SVM on a large public dataset CICIDS2017. For these algorithms implementation and testing, Python programming language, along with some important modules, such as: sklearn (for

creating the models), pandas (for data presentation), labelencoder (for labels manipulation), minmaxscaler (for data standardization) have been used. The algorithms obtained results are also compared to other published solutions in terms of performances in detecting abnormal behaviour.

2. DATASET DESCRIPTION AND PREPROCESSING

2.1 Dataset description and analysis

Since the dataset CICIDS2017 (Sharafaldin et al., 2018) have been made public, it has attracted the attention of researchers in terms of analysis and development of new models and algorithms for abnormal traffic behavior detection. According to the publishers, the dataset spans eight different files containing five days of normal traffic and attacks.

Table 1. CICIDS2017 dataset by days.

File Name	Day Activity	Attacks Found
Monday-WorkingHours.pcap ISCX.csv	Monday	Benign (Normal human activities)
Tuesday-WorkingHours.pcap_ ISCX.csv	Tuesday	Benign, FTPPatator,SSH-Patator
Wednesday-workingHours.pcap_ISCX.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoSSlowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.pca p_ ISCX.csv	Thursday	Benign, Web Attack – BruteForce, Web Attack – SqlInjection, Web Attack – XSS
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHours-Afternoon-PortScan.pcap ISCX.csv	Friday	Benign, PortScan
Friday-WorkingHours-Afternoon-DDos.pcap ISCX.csv	Friday	Benign,DDoS

The data set was created using the B-Profile system, which outlines the behavior of human interactions on the Internet and generates a normal benign background traffic. It was created from the behavior of 25 internet users, which are based on HTTP, HTTPS, FTP, SSH and e-mail protocols.

In Table 1 presents an overview of the data acquired each day. Regarding the information about attacks within five days it can be observed that the data gathered on Thursday and Friday (during working hours, afternoon) are suitable for binary classification, and the data collected on Tuesday, Wednesday and Thursday (morning) are best for designing the multiclass detection model.

Nonetheless, it should be kept under consideration that the best model obtained should be capable of detecting attacks of any kind. Thus, to design such an IDS, the traffic data presenting during the 5 days should be combined to form an unique data set, so that the detection model can use it.

In Table 2 are presented the 15 labels for the captured traffic classes, classes exhaustively described in section 2.3: the benign class that represents the normal traffic from the 5 days of traffic, as well as other 14 classes that represent the simulated and captured attacks.

Table 2. CICIDS2017 classes and their instances.

Class labels	Number of instances
BENIGN	2271320
MALIGN	556556
Bot – 1	1956
DdoS – 2	128025
DoS GoldenEye – 3	10293
DoS Hulk – 4	230124
DoS Slowhttptest – 5	5499
DoS Slowloris – 6	5796
FTP-Patator – 7	7935
Heartbleed – 8	11
Infiltration – 9	36
PortScan – 10	158804
SSH-Patator – 11	5897
Web Attack – BruteForce – 12	1507
Web Attack – SqlInjection – 13	21
Web Attack – XSS – 14	652

In addition, the malignant class was created in which all attack classes were concatenated, for a subsequent binary classification.

2.2 Dataset features description

The first packet determines the forward (source to destination) and backward (destination to source) directions, so the 83 features, such as: duration, number of packets, number of bytes, packet length, are calculated separately in both directions.

The data file contains labeled columns for each stream, as well as FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort and Protocol, plus about 80 other network traffic features.

It should be noted that TCP streams are usually stopped when the connection is removed (through the FIN packet), while UDP streams are terminated after a certain time. This flow interval can be assigned arbitrarily, e.g. 600 seconds for TCP and UDP.

2.3 Description of attacks scenarios

In this dataset, six attack profiles are created based on the latest updated list of common attack families as specified by the dataset publisher:

BruteForce Attack: Can be used to discover passwords, but also to discover content and pages hidden in a web application. Brute Force Attack is basically a trial and error attack.

Heartbleed: The Heartbleed bug is exploited by sending a malformed “heartbeat request” with a small payload and a long

data field to the vulnerable destination (usually a server) to evoke and extort the victim's response.

Botnet: is a group of devices, connected to the Internet, used to perform various malicious activities, to steal data, to transmit malware or spam, or to launch attacks allowing the attacker access to the device and its connection.

DoS Attack: The "Denial-of-Service" attack is meant to shut down a device or network, which will make it inaccessible to its users by flooding it with traffic. This cyber attack is usually done by flooding the server or device with many unnecessary requests., interrupting the services of a host connected to the Internet.

DDoS Attack: The "Distributed-Denial-of-Service" attack or distributed server denial of service typically occurs when multiple computers with Internet connections want to flood a victim's bandwidth or resources.

Web Attack: In this dataset, an SQL injection is used in which an attacker can create a string of SQL commands, which can be used to force the database to respond to information; Cross-Site Scripting (XSS) that happens when developers don't test the code properly to find the ability to inject the script and BruteForce over HTTP tries a list of possible or probable passwords to find the administrator's password.

Infiltration: After a successful infiltration, a "backdoor" attack will be executed on the victim's computer and can perform various attacks on the victim's network, such as full port scanning, obtaining IP (ICMP sweep) and enumerating services using NMAP security scanner.

Based on the attack scenarios explained above, to execute each attack, the publisher have used the available public tools developed in Python.

2.4 Data preprocessing

Examining the instances of the combined files, it was discovered that the dataset contains 291469 instances that are missing the class tag or have missing information for the labels of certain instances. By eliminating such instances with missing information a dataset of 2827876 instances with 82 features each is obtained.

Thus, preprocessing is done in 4 simple steps, such as: encoding applied to columns that are not represented numerically, removing instances that lack the class tag, that contain null or invalid values, balancing the data set by matching the classes as the number of records (the number of Benign traffic records to be approximately equal to the number of Malign traffic records) and standardizing the data set to keep the shape of the original distribution (by scaling each feature to a given range).

Balancing the data set as an equal number of instances in the malignant class and the benign class has been achieved by randomly eliminating benign instances, so the number of benign class reaches about half.

2.5 Data set organization in training, validation and testing sets

A training set for machine learning algorithms is a data set of instances used for learning, i.e. for adjusting model parameters. Most approaches that search through training data for empirical relationships tend to “overfit” data (overlap), which means they learn the training set but no longer behave well on new data, called test set (Ripley, 1996).

A validation set is a data set of examples used to identify the optimal hyperparameters of a classifier. The validation set works like a hybrid: it represents training data used for testing, not being part of the training, but not part of the final testing.

A test set is a data set that is independent of the training set but follows the same probability distribution. If a model that matches the training data also matches the test set, a minimal "overfit" has occurred.

All three sets were created by randomly selecting instances from the original dataset, in order not to create dependencies between them. Moreover, the test set was constructed from the initial data set before standardizing the training set and the validation set.

For this work, the data set is divided into 70% instances for the training set, 15% for validation and 15% for testing.

3. A BRIEF DESCRIPTION OF THE USED ALGORITHMS AND USED HYPERPARAMETERS

Selecting the right algorithm is a key part of any machine learning project and, as there are dozens of methods to choose from, it is essential to understand the strengths and weaknesses of different applications.

Thus, as mentioned in the introduction of the paper, it was desired to obtain good results with high predictions using supervised learning algorithms, such as: random forests(RF) and support vector machines(SVM).

3.1 Support Vector Machine(SVM)

Support vector machines, or SVMs, are a supervised learning algorithm that can be used for both classification and regression problems (Chauhan et al., 2019).

The classification is performed by finding the hyperplane that differentiates the two classes very well. The use of SVMs in cybersecurity has a certain limitation, because being a supervised machine learning method requires labeled information for effective learning and can only handle binary classification very well, while intrusion detection requires the classification of several classes.

Machines with support vectors treat each data characteristic equally. In real intrusion detection datasets, many features are redundant or less important. SVM training takes a long time for the IDS domain and requires a large storage space for the data set.

Each instance in the training set contains a target value (eg. class tags) and several attributes (eg. observed features or variables). The objective of the SVM is to produce a model (based on training data) that can predict the target values or labels of the test data, to which only the attributes of the test data have been provided.

In this paper it is used a SVM with radial basis function kernel (or RBF). The implementation of the SVM algorithm was

carried out in the Python programming language, its hyperparameters being obtained by an exhaustive search (using the GridSearch method) with a search space between 0.01 and 1000 for the gamma parameter and the range [0.1; 1000] for C Penalty Parameter C both for the binary classification and for the multiclass problem. The penalty parameter C is set at 1000, the gamma parameter at 1 following the previously mentioned search.

3.2 Random Forests(RF)

Random forests are a set of classification or decision trees. The random forest generates many classification trees. Each tree is constructed by a different bootstrap sample from the original data, using a tree classification algorithm.

Decision trees use graphs to model decision making, where each node in the graph represents a question about data, and the branches between nodes represent possible answers to this question (Parmar et al., 2019).

Because each tree is built using the bootstrap method, about a third of cases are left out of the bootstrap instances and are not used in training. These cases are called 'ouf-of-bag' cases and are used to obtain an unbiased estimate of the classification error during the run-in period (execution), while the trees are added to the forest.

After the formation of the forest, a new object to be classified is placed on each tree in the forest for classification. Each tree gives a vote indicating the decision of the tree regarding the class of the object. The forest combines the votes from different decision trees to decide the final class of the object in the test set.

Following an exhaustive search, the parameters for obtaining a robust model are: total number of trees in the "forest" equal with 5 after searching for the optimal number of trees for each type of the data set, a parameter "oob"(out-of-bag) which is a method to reduce the number of instances of the majority classes, the parameter for maximum depth where the algorithm chooses its own depth depending on the number of features, being smaller or equal to their number and the Gini criterion, instead of entropy, to minimize the likelihood of misclassification.

4. RESULTS

In order to achieve the more robust, accurate and very suitable results for a further analysis and comparison of the algorithms, I first used a binary classification using the SVM versus RF, then on all attacks a multiclass classification using the SVM versus RF.

4.1 Binary classification results (abnormal behavior detection)

While analyzing network traffic, the first task is to distinguish between normal scenarios and malicious cases. The attacks should be identified with high precision.

4.1.1 Support Vector Machines(SVM)

The results obtained by SVM on the test set are:

- Benign SVM - test set: 94.23%
- Malign SVM - test set: 98.34%

The prediction percentage of the malignant class (attacks) obtained by SVM in the test set is higher than the prediction percentage of the benign class (normal traffic), as it is preferable for certain instances of benign trafficking to be predicted as attacks (malignant traffic) and not the other way around.

In table 3 is presented the confusion matrix that can be used to measure the performance of a machine learning algorithm, usually a supervised learning one. Each row of the confusion matrix represents the instances of an actual class and each column represents the instances of a predicted class.

Also, in table 4 is presented the classification report on key metrics in a classification problem: precision, recall, f1-score and support for each class we're trying to find or predict.

Recall indicates the proportion of the class that is predicted taking into account the number of elements of the class. The precision shows the number of correctly classified elements in that class.

The **f1-score** is the harmonic mean between precision and recall. The support is the number of occurrences of the given class in the dataset. The thing is, precision and recall is highly used for an imbalanced dataset because in a highly imbalanced dataset, a 99% accuracy can be meaningless.

The parameters "Macro-Avg" and "Weighted-Avg" stand for:

- macro-average says the function is computing the f1-score for each label, and returns the average without considering the proportion for each label in the dataset.

- weighted-average says the function is computing the f1-score for each label, and returns the average considering the proportion for each label in the dataset.

Table 3. Confusion Matrix for Test Set – SVM.

	Predicted values		
Actual values	Confusion Matrix (403982 packets of traffic that includes attacks and normal IP traffic)	0 - Benign	1 - Malign
	0 – Benign Traffic	305444	18866
	1 – Malign Traffic	1301	78371

Table 4. Binary Classification Report for Test Set – SVM.

	Precision	Recall	F1-score	Support
0 - Benign Traffic	0.995	0.942	0.968	324547
1- Malign Traffic	0.807	0.983	0.886	79435
Accuracy	0.95	0.95	0.95	0.95
Macro-Avg	0.901	0.963	0. 927	403982
Weighted- Avg	0.958	0.95	0.952	403982

4.1.2 Random Forests(RF)

The results obtained by RF on the test set are:

- Benign RF - test set: 99.989%
- Malignant RF - test set: 99.983%

The overall accuracy increased slightly from 0.95 to 0.99 because the number of instances given to the algorithm was sufficient for the model to recognize the patterns for attacks and increase the total recall.

Table 5. Confusion Matrix for Test Set – RF.

	Predicted values		
Actual values	Confusion Matrix (403982 packets of traffic that includes attacks and normal IP traffic)	0 - Benign	1 – Malign
	0 - Benign Traffic	324274	36
	1 - Malign Traffic	6	79666

Based on the results obtained with the help of SVM and RF for the binary classification phase, a slight increase of the precision and accuracy was observed for RF compared to SVM. As a consequence of the problem not being linearly separable, RF algorithm is more efficient on large datasets than SVM.

Table 6. Binary Classification Report for Test Set – RF.

	Precision	Recall	F1-score	Support
0 – Benign Traffic	0.99996	0. 9998	0.99992	324547
1 – Malign Traffic	0.99947	0.99984	0.99965	79435
Accuracy	0.9998	0.9998	0.9998	0.9998
Macro-Avg	0.99971	0.99985	0.99979	403982
Weighted- Avg	0.9998	0.9998	0.9998	403982

4.2 Multiclass classification results

While binary classification helps in detecting treats, we are also interested in identifying a treat by its class. In what follows the performances at this task for the SVM and RF algorithms are presented.

4.2.1 SVM – Multiclass classification results

The results obtained by SVM on the test set for all attacks, respectively for each one are:

- Total Malignant LSVC: 99.12%
- Bot (LSVC): 99.46%
- DDoS (LSVC): 99.95%
- DoSGoldenEye (LSVC): 97.17%
- DoSSlowhttptest (LSVC): 96.85%
- DoSHulk (LSVC): 99.12%
- DoSSlowloris (LSVC): 80.19%
- FTP_Patator (LSVC): 99.74%
- SSH_Patator (LSVC): 98.87%
- Heartbleed (LSVC): 100%
- Infiltration (LSVC): 50%
- PortScan (LSVC): 99.90%
- Web Attack - BruteForce (LSVC): 83.38%
- Web Attack - SqlInjection (LSVC): 0%
- Web Attack - XSS (LSVC): 0.74%

The total prediction percentage of attacks is higher than the average predictions for each attack, because for individual predictions the classes were unequal, some being almost zero, so they have prediction results less than or equal to 50%. This happens due to the fact that many classes such as Heartbleed, Infiltration and SqlInjection are underrepresented. In table 7 details on the SVM performances are presented.

Table 7. Multiclass Classification Report for Test Set – SVM.

	Precision	**Recall**	**F1-Score**	**Support**
1 - Bot	0.995	0.997	0.996	395
2 - DDoS	0.986	0.991	0.992	25567
3 – DoS GoldenEye	0.979	0.960	0.969	2061
4 - DoSHulk	0.997	0.991	0.994	46147
5 – DoS Slowhttptest	0.963	0.967	0.965	1117
6 – DoS Slowloris	0.976	0.808	0.884	1181
7 - FTP	0.897	0.995	0.943	1567
8 – Heartbleed	0.5	1	0.66	2
9 - Infiltration	1	0.63	0.77	8
10 - PortScan	0.9994	0.998	0.9991	31685
11 - SSH	0.922	0.991	0.955	1195
12 - BruteForce	0.625	0.831	0.713	261
13 - SqlInjection	0	0	0	4
14 - XSS	0	0	0	122
Accuracy	0.991	0.991	0.991	0.991
Macro-Avg	0.774	0.797	0.775	111312
Weighted-Avg	0.9903	0.991	0.9904	111312

4.2.2 RF – Multiclass classification results

The results obtained by RF on the test set for all attacks, respectively for each one are:

- Total Malign RF- test set: 99.90%

- Bot (RF): 100%
- DDoS (RF): 100%
- DoSGoldenEye (RF): 99.95%
- DoSSlowhttptest (RF): 99.81%
- DoSHulk (RF): 99.98%
- DoSSlowloris (RF): 99.73%
- FTP_Patator (RF): 100%
- SSH_Patator (RF): 100%
- Heartbleed (RF): 100%
- Infiltration (RF): 83.33%
- PortScan (RF): 99.97%
- Web Attack - Brute Force (RF): 86.26%
- Web Attack - SqlInjection(RF): 100%
- Web Attack - XSS (RF): 71.64%

The total prediction percentage of attacks is higher than the average prediction for each attack. Also, due to the fact that the 14 attack classes are not balanced, a slight decrease in precision and F1-score compared to accuracy can be observed (Table 8).

Table 8. Multiclass Classification Report for Test Set – RF.

	Precision	**Recall**	**F1-Score**	**Support**
1 – Bot	1.000000	1	1	395
2 – DdoS	1	0. 99	0.99	25567
3 – DoS GoldenEye	0.998	0.998	0.998	2061
4 – DoSHulk	0.99	0.99	0.99	46147
5 – DoS Slowhttptest	0.998	0.996	0.997	1117
6 – DoS Slowloris	0.995	0.997	0.996	1181
7 – FTP	1	1	1	1567
8 – Heartbleed	1	1	1	2
9 – Infiltration	1	0.875	0.933	8
10 – PortScan	0.99	0.99	0.99	31685
11 – SSH	1	1	1	1195
12 - BruteForce	0.837	0.869	0.853	261
13 - SqlInjection	1	0.5	0.66	4
14 - XSS	0.698	0.663	0.68	122
Accuracy	0.99	0.99	0.99	0.99
Macro-Avg	0.966	0.921	0.938	111312
Weighted-Avg	0.999	0.999	0.999	111312

4.3 Comparison of performance measures for the most prominent attacks

In this section it's a comparison of the performance measures obtained by the 2 algorithms on the most common 3 types of attacks of the 14 known, namely DDoS, DoSHulk and PortScan.

Thus, for the 3 classes of attacks detected with SVM we have the following performances:

- Accuracy - DDoS: 99.91%
- Accuracy - DoSHulk: 99.1%
- Accuracy - PortScan: 99.89%
- Recall - DDoS: 100%
- Recall - DoSHulk: 100%
- Recall - PortScan: 100%
- Precision - DDoS: 99.91%
- Precision - DoSHulk: 99.11%
- Precision - PortScan: 99.89%

And for the attacks detected with RF, the following measures were obtained:

- Accuracy - DDoS: 99.99%
- Accuracy - DoSHulk: 99.99%
- Accuracy - PortScan: 99.96%
- Recall - DDoS: 100%
- Recall - DoSHulk: 100%
- Recall - PortScan: 100%
- Precision - DDoS: 99.99%
- Precision - DoSHulk: 99.99%
- Precision - PortScan: 99.96%

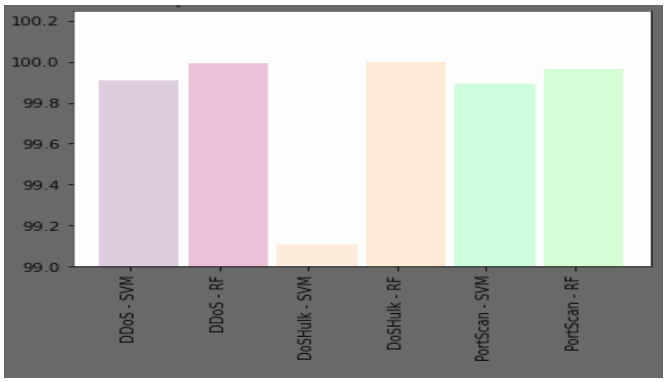

Fig. 1. Accuracy comparison between the 2 algorithms for the most prominent 3 attacks.

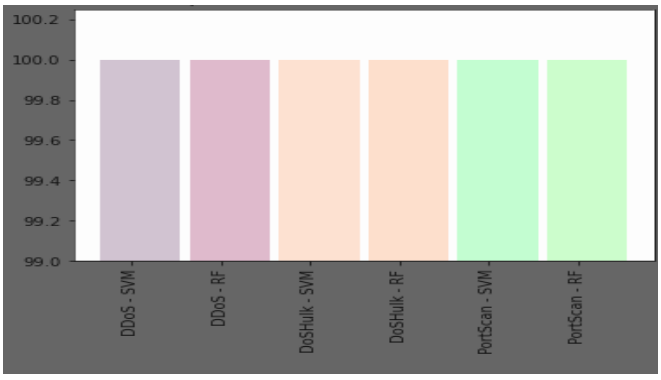

Fig. 2. Recall comparison between the 2 algorithms for the most prominent 3 attacks.

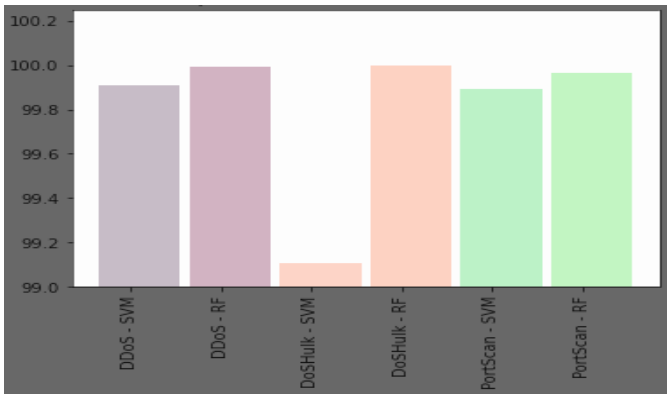

Fig. 3. Precision comparison between the 2 algorithms for the most prominent 3 attacks.

As one can see in Figures 1-3, the accuracy and recall obtained by the 2 algorithms are almost equal to the accuracy (100%) due to the fact that these classes had a very large number of instances compared to other types of attacks, which resulted in more robust and accurate learning by the models.

5. DISCUSSION AND CONCLUSIONS

In conclusion, it was noticed that RF method is able to classify attack types with a high accuracy of about 99%, unlike the classifier obtained with SVM which has slightly poorer performance. Also, in order for the SVM algorithm to obtain better results and close to those made with RF, a larger space for hyperparameters search is recommended, as well as the elimination of less important features from the data set. In this paper we did not want to eliminate the features that characterize network traffic just to simulate a real-time IDS, online, on a continuous flow of traffic.

There are other classification algorithms, like K-nearest neighbour algorithm (Kachavimath et al., 2020) or decision trees (Abdulrahman and Ibrahem, 2019), that, in some circumstances achieve comparable performances. However, all other IDS systems based on machine learning techniques were tested on the KDDcup99 or NSL-KDD dataset (Tavallaee et al., 2009) which are 20 years old and, respectively, 10 years old. Since then, the network packet attributes have changed and are continuously being updated. Hence, KDD-cup99 has only 41 attributes. This project uses the CICIDS 2017 data set which has 85 attributes. The size and input of the data set are 20 times larger than KDDcup99 or NSL-KDD.

The Random Forest algorithm offers very good accuracy compared to other models. In the application, this model could be implemented to detect any attack. Also, one of the most important concerns in achieving a robust IDS is to make sure that the data is "clean", i.e. does not contain feature values or even missing labels. Further developments or improvements:

- Combine the investigated dataset with other public sets
- Extension of the initial work on intrusion detection using the TCP protocol or other communication protocols.
- IDS uses network packets as an input source. The packet structure covers all layers of the network, including the application layer. Thus, the realization of the application for the detection of specific intrusions is an additional extension to this research. This application can detect and possibly stop attacks in real time (being an online IDS)
- Using other techniques to detect intrusions, such as the use of neural networks, fuzzy logic or probabilistic logic.
- Intrusion detection in encrypted traffic.

REFERENCES

Abdulrahman, A.; Ibrahem, M.K. (2019): Evaluation of ddos attacks detection in a new intrusion dataset based on classification algorithms. *Iraqi J. Inf. Commun. Technol.* **1**, 49–55.

Abu Al-Haija, Qasem & Zein-Sabatto, Saleh. (2020). An Efficient Deep-Learning-Based Detection and Classification System for Cyber-Attacks in IoT *Communication Networks. Electronics*. 9. 21-52. 10.3390/electronics9122152.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., Faruki, P. (2019) Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Commun. Surv. Tutor. 2019, 21*, 2671–2701.

Chauhan, V.K., Dahiya, K. & Sharma, A. (2019) Problem formulations and solvers in linear SVM: a review. *Artif Intell Rev 52***,** 803–855.

Dobrescu, R. & Hossu, D. & Ulrich, R. (2009). Self-similarity Tests for Internet Traffic. *Control Engineering and Applied Informatics*. 11.

Jabez, J. & Muthukumar, B.. (2015). Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach. *Procedia Computer Science*. 48. 338-346. 10.1016/j.procs.2015.04.191.

Kachavimath, A.V.; Nazare, S.V.; Akki, S.S., (2020): Distributed denial of service attack detection using naïve bayes and k-nearest neighbor for network forensics, In 2020 *2nd International conference on innovative mechanisms for industry applications* (ICIMIA), pp. 711–717.

Khraisat, A. & Alazab, A. (2021) A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges.

Kim, S., & Kyung-Joon P. (2021) A Survey on Machine-Learning Based Security Design for Cyber-Physical Systems. *Applied Sciences, 11*(12), 5458.

Lwoga, E. T., & Sangeda, R. Z. (2018). ICTs and development in developing countries: A systematic review of reviews. *The Electronic Journal of Information Systems in Developing Countries*, e12060. doi:10.1002/isd2.12060.

M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani(2009): "A Detailed Analysis of the KDD CUP 99 Data Set," *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA).*

Magán-Carrión, R., Urda, D., Díaz-Cano, I., Dorronsoro, B. (2020) Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches. *Appl. Sci., 10*, 1775.

Nicholas, L., Ooi, S.Y., Pang, Y.H., Hwang, S.O. , Tan, S. (2018) Study of long short-term memory in flow-based network intrusion detection system, *Journal of Intelligent*

& Fuzzy Systems, Pre-press, 1-11.

Parmar A., Katariya R., Patel V. (2019) A Review on Random Forest: An Ensemble Classifier. In: Hemanth J., Fernando X., Lafata P., Baig Z. (eds) International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018. ICICI 2018. *Lecture Notes on Data Engineering and Communications Technologies*, vol 26. Springer, Cham.

Ripley, B. (1996). Pattern Recognition and Neural Networks. *Cambridge University Press*. p. 354. ISBN 978-0521717700.

Rughiniș, R., & Gheorghe, L. (2013) TinyAFD: Attack and Fault Detection Framework for Wireless Sensor Networks, *Journal of Control Engineering and Applied Informatics - CEAI, 15(1)*, pp. 33–44, 2013. ISSN:1454-8658.

Sapalo Sicato, J.C., Sharma, P.K., Loia, V., Park, J.H. (2019) VPNFilter Malware Analysis on Cyber Threat in Smart Home. Network. *Appl. Sci., 9*, 2763.

Sharafaldin, I., Arash, H. L., & Ghorbani, A. (2018) Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *4th International Conference on Information Systems Security and Privacy (ICISSP)*, Portugal.