Investigating adversarial attacks against Random Forest-based network attack detection systems

Philippe Owezarski
LAAS-CNRS, Université de Toulouse, CNRS
Toulouse, France

***Abstract*—A significant research effort in cybersecurity currently deals with Machine Learning-based attack detection. It is aimed at providing autonomous attack detection systems that require less human expert resources, and are then less expensive in time and money. Indeed, such systems are able to autonomously learn about benign and malicious traffic, and to classify further traffic samples accordingly. In such context, hackers start designing adversarial learning approaches in order to design new attacks able to evade from the Machine Learning-based detection systems. The work presented in this paper aims at exhibiting how easy it is to modify existing attacks to make them evade from the Machine Learning-based attack detectors. The Random Forest algorithm has been selected for this work as it is globally evaluated as one of the best Machine Learning algorithm for cybersecurity, and it provides informations on how a decision is made. Indeed, the analysis of the related Random Forest trees helps explaining the limits of this Machine Learning algorithm, and gives some information that could be helpful for making attack detection somewhat explainable. Several other Machine Learning algorithms as SVM, kNN ans LSTM have been selected for evaluating their ability to detect the adversarial attack presented in this paper.**

***Index Terms*—Adversarial learning, Machine Learning, Random Forest, Network Attacks Detection**

I. Introduction

Attack/intrusion detection is one of the hot topics in cybersecurity, and nowadays, most of the efforts on network attack detection deal with machine learning (ML) and deep learning (DL) based approaches, for their capabilities of working and making decision autonomously. It is then a step toward correcting the main limits of signature based attack detection (time to proceed, amount of human resources involved, and cost). Despite ML based attack detection is not widely deployed, it is raising most of the research efforts nowadays, and the related literature exhibits very good detection results, often evaluated over 99% of detection accuracy in labs [1] [2].

Despite these high accuracy results, Elmrabi et al. [1] and Ahmad et al. [2] pointed out limits of ML based detection approaches, and especially that none of the existing papers exhibit a full detection of all attacks contained in any dataset. Many papers dealing with adversarial learning also exhibit weaknesses of ML based anomaly detection approaches (cf. II) when facing attacks whose statistical profile has been modified not to relate to the trained model [3] [4]. This new paper then deals with investigating how some attacks that have never been considered in previous adversarial learning based works can easily evade from ML based detection systems, and this, targeting as simple as possible changes on attack profiles. Throughout this paper, and for deeper analysis, the Random Forest (RF) supervised ML algorithm has been selected: the Random Forest (RF) algorithms is mostly used, as it has been widely studied and used for attack detection, and it appears as one of the best algorithms for attack detection, with a detection accuracy often evaluated over 99% [5]. It also provides a lot of informations on the different trees of the forest, especially on what parameters and related values impacted the decision finally taken when facing a bin of traffic. RF then allows deep analysis for explaining its Artificial Intelligence (AI) process for decision making. Note also that several other supervised ML algorithms for attack detection have been evaluated on the new modified adversarial attacks.

The rest of this paper is as follows: section II presents some significant related contributions in the domain of adversarial learning. Section III provides some information on the dataset that has been selected for running the experiments performed for the works described in this paper. Section IV shows how the RF-based detection tool has been built, and its detection performance on the selected dataset. Section V then investigates on few examples how it is possible to tune some generally well detected legacy attacks to make them evade the RF detection tool. Section VI also shows that the tuned attacks also easily evade SVM, kNN and LSTM [6] machine Learning algorithms. Finally section VII concludes this paper.

II. Related Works

The use of ML in cybersecurity has been raising a lot of efforts since at least a decade. The literature is rich in papers reporting ML-based attack detection experiments that exhibit very high performance and accuracy. However, only few of these works integrate some adversarial learning aspects. This is a significant issue as these few papers on adversarial learning in cybersecurity exhibit strong limits of ML-based attack detection, showing examples on how to evade from ML-based detectors, and pointing out the lack of robustness of these ML-based detection approaches [7]. They also show, based on adversarial learning results, how to improve the ML algorithms for more efficient and robust attack detection (as [8] in the case of RF). Some works on adversarial learning [9] more specifically demonstrate that very limited adversarial perturbations can have a strong negative impact on the classification accuracy of ML algorithms. It is also the case for DL algorithms [10]. In papers as [3] [9], authors propose

adversarial attacks (often they are only modified well-known attacks) that evade some ML algorithms that are nevertheless considered as very efficient for detecting such attacks. It has then be proved that famous ML algorithms as K-mean, SVM, kNN or Logistic Regression, to quote a few, can often miss simple adversarial attacks. Adversarial attacks have also been designed for evading famous algorithms as Gradient Boosted Decision Trees [11], or Neural Networks [12].

Note however that two approaches can be considered when designing adversarial attacks: most of the times, researchers adopt a white-box point of view for ML-based anomaly detectors, i.e. they can enter the ML engine and data to analyze the ML algorithm behavior and its decision. But some researchers adopt the black-box position of hackers, i.e. they cannot have access to internal ML algorithm informations, and they can just make remote tests for getting information on what kind of attacks ML-based detectors can detect or miss, and explain why. Such black-box approach has been adopted by authors of papers like [13]. In our paper we adopt an intermediate approach, i.e. the adversarial attacks are designed with a black-box approach for the RF algorithm. However, for analyzing the results, we use internal RF information on built RF trees to understand the reasons for a success or a miss of the detection of an attack.

III. The CIC-IDS-2017 Dataset

For illustration purpose, the CIC-IDS-2017 dataset has been selected [14]. CIC-IDS-2017 is nowadays a new reference dataset for security tool assessment. This dataset has been created by the Canadian Institute for Cybersecurity with the purpose of giving researchers an open source Dataset for designing and evaluating network intrusion detection systems. The CIC-IDS-2017 dataset consists of eight .csv files. It represents a total of five days of traffic. Each gathered day contains one attack or more that were specifically generated. Table I indicates the list of attacks contained in the different files of the CIC-IDS-2017 dataset.

Each of the .csv files is the result of the network traffic analysis performed with CICFlowMeter, an open source tool that re-constructs bidirectional flows from the gathered packet pcap files, and extracts the specific features of these flows. In bidirectional flows, the first packet determines the forward (source to destination) and backward (destination to source) directions. Hence the statistical time-related features can be calculated separately in the forward and backward directions. CIC-IDS-2017 comes with 78 features.

For using this dataset, we had to merge all the files into one single .csv file. We then cleaned the dataset by taking off the rows with NaN, infinite or missing values. We also noticed that there were 8 features that had a variance of 0. We then took these features off the dataset. Finally we added a binary label column. The dataset is given with types of attack as labels, but we chose to add the option of using binary class with "1" to indicate an attack, and "0" for benign traffic.

After this preprocessing stage, the final dataset consists of 2 827 876 samples with 70 features and two label columns: a MultiClass label (which is the attack type) and a binary label. Table II details the distribution of the different attack types in the dataset.

TABLE I
CIC-IDS-2017 files description

File name	Day	Attacks found
Monday-WorkingHours.csv	Monday	Benign (Normal Activity)
Tuesday-WorkingHours.csv	Tuesday	Benign, FTP-Patator, SSH-Patator
Wednesday-workingHours.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.csv	Thursday	Benign, Web Attack - Brute Force, Web Attack - SQL injection, Web Attack - XSS
Thursday-WorkingHours-Afternoon-Infilteration.csv	Thursday	Benign, Infiltration
Friday-WorkingHours-Morning.csv	Friday	Benign, Bot
Friday-WorkingHours-Afternoon-PortScan.csv	Friday	Benign, PortScan
Friday-WorkingHours-Afternoon-DoS.csv	Friday	Benign, DDoS

TABLE II
Distribution of Samples in CIC-IDS-2017

Attack Type	Number of samples in the dataset
BENIGN	2271320
DoS Hulk	230124
PortScan	158804
DDoS	128025
DoS GoldenEye	10293
FTP-Patator	7935
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1956
Web Attack - Brute Force	1507
Web Attack - XSS	652
Infiltration	36
Web Attack - Sql Injection	21
Heartbleed	11

IV. RF-tool design and performance

As it is generally the case in the related literature, we selected the RandomForestClassifier [15] function from Python Scikit-Learn as the RF-based attack detection tool. It appears to be a very good implementation of the RF algorithm. It always exhibited very high detection performance on any dataset [5].

For running the experiments, it is first required to optimize the three RF hyper-parameters, i.e. number of trees in the forest, maximum depth of the trees, and maximum number of features in the trees. Indeed, RF ranks all features of the dataset from the most to the less important ones for the classification. By working only on the most important features, a significant

TABLE III
FN RATE PER ATTACK TYPE FOR RF DETECTION

Attack Type	Number of samples in the dataset	Percentage of undetected samples
DoS Hulk	230124	0.09%
PortScan	158804	0.02%
DDoS	128025	0.08%
DoS Golden-Eye	10293	1.58%
FTP-Patator	7935	0.06%
SSH-Patator	5897	0.16%
DoS slowloris	5796	1.23%
DoS Slowhttptest	5499	1.66%
Bot	1956	63.01%
Web Attack - Brute Force	1507	5.29%
Web Attack - XSS	652	4.20%
Infiltration	36	80%
Web Attack - Sql Injection	21	57.14%
Heartbleed	11	0%

reduction of the computing times is gained without reducing much the detection/classification accuracy. The methodology for finding out the 3 optimal RF hyper-parameters follows the classically used 5-fold Cross-Validation approach. For each parameter, several runs with a wide range of values are performed, the other hyper-parameters being fixed to the best values previously determined. For the scoring of each run, we use the function score() of Scikit-learn which gives the accuracy of the detection. The accuracy is defined as the number of correctly identified samples over the total number of samples.

The optimization stage gives for each hyper-parameter the following values:

- Number of estimators (i.e. number of trees): 80
- Max depth of trees: 15
- max number of features : 15

The optimal detection performance for each of the 14 attacks contained in the CIC-IDS-2017 with RF is depicted in Table III. Table III displays the percentage of False Negatives per attack kind, together with the number of samples of each attack present in the dataset.

As in most papers of the related literature, in this paper, performance of attack detectors are evaluated thanks to values and evaluation metrics as the number of True Negatives (TN), False Positives (FP), False Negatives (FN), True Positives (TP), Precision, Recall, and F1-Score. These three last metrics are often completed with an accuracy score defined by:

$$\text{Accuracy.Score} = \frac{\text{\# of samples correctly classified}}{\text{total number of samples}} \quad (1)$$

The optimal detection results with our RF-based detection tool on CIC-IDS-2017 full dataset are :

- Precision = 0.999075
- Recall = 0.999260
- F1-score = 0.999167
- Accuracy score = 0.998663

These statistical results appear as very good, and based on such figures, it is generally accepted that RF performs very efficiently for attack detection. These results have however to be balanced with the results depicted in table III that clearly shows significant lacks of the RF-based detector. It clearly appears in Table III that despite a very low global FN rate, some attacks remain mostly undetected by RF-based attack detection: some attacks as SQL web injection, Bot or infiltration attacks are very often missed by the RF-based detector. This limit of the detection performance does not appear on the classical performance metrics because attacks are statistically merged with a very big number of benign sample of traffic that are well classified by the RF-based detector. It also appears that the attacks that are not well detected are the ones with the fewest samples in the dataset. But this is not an absolute rule, as the Heartbleed attack - that is the one with the less samples in the dataset - is perfectly detected. The underrepresentation of some attacks in the dataset is then not the (only?) cause of miss-detection.

Anyway, the classical statistical evaluation metrics for attack detection are not representative of the real performance of the detectors. In the remainder of the paper, the evaluation, both in quantity and quality, is based on the number of false negatives. This is the most representative indicator when dealing with adversarial attacks, the aim being to investigate how attacks can evade the RF-based attack detector.

V. ADVERSARIAL ATTACKS

Adversarial attacks have already been the topic of many works [3] [4]. Some of them notably relate to RF-based attack detection as [3]. They especially investigate the optimal attack that always (or almost always) evade from the detection algorithms. For that, the design of the adversarial attack often takes advantage of some information got from the internal analysis of the ML-based detection system, as the main features used by the ML algorithms for making the decision. In this paper, we target the design of a generic method for building any kind of adversarial attacks as simple as possible, with a high efficiency whatever the kind of ML algorithm used for attack detection. For that purpose, ML algorithms are considered as black-boxes, and we then design the adversarial attacks without benefiting from any knowledge on how the considered ML algorithm behaves when facing attacks. We are considering the ML algorithm a a white-box only after its experimental performance evaluation, and this for analyzing the reasons of the detection changes with regard to the different kinds of adversarial attacks.

This section particularly aims at showing how easy it is to design attacks that evade RF-based attack detection. It just leverages that RF makes its attack classification/detection when observing statistical deviations between current traffic samples and the trained traffic. The idea there is then just

to modify the volume and temporal profile of attack flows. It consists in replaying existing attacks whose traces are available in public datasets (here the CIC-IDS-2017 dataset) after changing:

- The size of the packets
- The inter-arrival times of packets

Just changing these two features impacts several others, as packet rate per second, throughput per second, packets bursts sizes and rates, etc. that are often features that the ML algorithms are taking advantage of for making their decision.

The changes will be applied to the 14 kinds of attacks contained in the CIC-IDS-2017 dataset then constituting a new dataset with modified attacks profiles. The methodology then deals with training the RF-detector with the original CIC-IDS-2017 dataset, and then to measure the FN rates on each attack kind when the RF detector is applied on the modified dataset.

Practically, the changes on the packet sizes and inter-arrival times are applied in a random manner, but with a maximum amplitude parameter C expressed in percent. For example, the packet size cannot be modified by more than C% of its initial value.

The packets sizes of the adversarial attack flows are modified as follows:

$$\forall n \in [0; Max\_Packets], S'(n) = (1 + C.Rand(-1; +1))S(n) \quad with : S'(n) \in [40; 1500] \tag{2}$$

For the Inter-Arrival Times, by definition:

$$\begin{cases} t'(0) = t(0) \\ \forall n \in [0; Max\_Packets - 1], IAT(n) = t(n+1) - t(n) \end{cases} \tag{3}$$

The Inter-Arrival Times are modified as follows:

$$\forall n \in [0; Max\_Packets - 1], \quad IAT'(n) = (1 + C.Rand(-1; +1)).IAT(n) \tag{4}$$

Finally, for the modified dataset, the arrival time of the packets of the adversarial attack are :

$$\forall n \in [0; Max\_Packets-1], t'(n+1) = t'(n)+IAT'(n) \tag{5}$$

where:

- $S(n)$ is the size of the n^{th} packet of the flow in the original dataset.
- $S'(n)$ is the size of the n^{th} packet of the flow in the modified dataset.
- $IAT(n)$ is the Inter-Arrival Time between the n^{th} and $(n+1)^{th}$ packets of the flow in the original dataset.
- $IAT'(n)$ is the Inter-Arrival Time between the n^{th} and $(n+1)^{th}$ packets of the flow in the modified dataset.
- $t(n)$ is the arrival time of the n^{th} packet of the flow in the original dataset.
- $t'(n)$ is the arrival time of the n^{th} packet of the flow in the modified dataset.

TABLE IV
FN RATE PER ATTACK TYPE FOR RF DETECTION

Attack Type	**FN rate with the original dataset**	**FN rate with 10%modified attacks profiles**	**FN rate with 25%modified attacks profiles**	**FN rate with 50%modified attacks profiles**
DoS Hulk	0.09%	14.52%	38.87%	72.23%
PortScan	0.02%	1.94%	5.16%	12.06%
DDoS	0.08%	21.49%	54.31%	82.92%
DoS Golden-Eye	1.58%	41.88%	81.52%	97.77%
FTP-Patator	0.06%	1.12%	3.25%	6.09%
SSH-Patator	0.16%	1.84%	4.36%	8.67%
DoS slowloris	1.23%	35.29%	69.41%	86.89%
DoS Slowhttptest	1.66%	39.55%	74.42%	91.63%
Bot	63.01%	88.31%	100.00%	100.00%
Web Attack - Brute Force	5.29%	57.26%	79.03%	94.49%
Web Attack - XSS	4.20%	63.62%	98.80%	100.00%
Infiltration	80.00%	100.00%	100.00%	100.00%
Web Attack - Sql Injection	57.14%	98.73%	100.00%	100.00%
Heartbleed	0%	0%	0%	0%

- $Rand(-1; +1)$ is the random function that returns a random floating number in the [-1; +1] interval.
- $Max\_Packets$ is the number of packets in the flow.
- C is the maximum modification value in percent for the size of packets or Inter-Arrival times.

In the following, three different modified datasets serve for exhibiting how RF detector performs depending on the level of modifications applied on the attack profiles, respectively with C = 10, 25 and 50%. FN ratio for each attack kinds are given in Table IV.

It clearly appears in Table IV that changing the size and the IAT of packets has a negative impact on the detection accuracy, except for Heartbleed and Patators attacks that have a clear signature related to specific protocols, and of course PortScan attacks whose detection mostly leverages source and destination addresses pairs that are not impacted by the changes performed on the dataset. To analyze these results we have been using the Scikit-learn library [16] that provides all required information on the trees inside the RF (here 80 trees). For each tree, much information on the nodes and leaves are provided, as the features and thresholds considered. And in depth analysis of the RF trees obtained with the original vs. the modified CIC-IDS-2017 datasets exhibits two main differences:

- For 37% of the trees of the RF, the considered features in the leaves close from the root (the most significant features for the decision making) have changed. The RF algorithm then classifies the samples differently for the two datasets, and they then converge to different classification results.

- For 54% of the trees of the RF, it appears that the changes make the figures contained in the tree nodes and leaves lower than the detection thresholds. The related samples then are not detected as being part of an attack class, and remain classified as benign traffic.

For the remaining trees, they do not change significantly for the two datasets and correspond to the traffic classes whose detection is not significantly impacted by the dataset changes (Heartbleed, PortScan, and Patator attacks).

VI. Comparison of adversarial attack detection with several other supervised learning algorithms

This section deals with evaluating how several other supervised Machine Learning algorithms perform when facing the new adversarial attacks. SVM, kNN and LSTM have been selected as they cover the main different approaches of supervised machine learning from vectors to neural networks. The experimental methodology is the same as for RF: the Machine Learning based detection algorithms are trained on the original CIC-IDS-2017 dataset, and their detection performance is then evaluated on the modified datasets. The SVM, kNN and LSTM algorithms have been configured with the parameters that provided the lowest FN rate when running on the original CIC-IDS-2017 dataset. These parameters have been determined using an empirical approach, i.e. fixing arbitrarily values to the parameters at the beginning, and then applying a dichotomy on the parameters for the following experiments based on the obtained detection results for the previous experiment.

The results are presented (as for RF) in tables: table V presents the comparative results between RF, SVM, kNN and LSTM when the testing dataset has not been modified compared to the training dataset. Tables VI, VII, and VIII presents the quality of detection, based on the FN rate, when the testing dataset has been modified with C value equals 10, 25 and 50 respectively.

These results clearly exhibit the effectiveness of the new adversarial technique as almost all the modified attacks easily evade from all the detection algorithms. It also appears that SVM and kNN have the worst detection results, especially compared to RF. The result was expected as RF combines several trees that helps considering many parameters at the same time. RF can be considered as working on the same principle as ensemble learning approaches that combine several detection tools to make the decision, except that for RF, all models are similar (i.e. binary trees) but focusing on different features. This however gives RF more detection capabilities than ML models as SVM or kNN. The low detection performance of LSTM is more surprising. The analysis of the logs of the neurons seems to exhibit that LSTM is not able to easily adapt to the brutal changes in the profile of the attacks: the detection threshold are not violated with the new packet sizes and IAT of the attacks. Note that this is particularly difficult to get into the neurons behaviors, and understand how they make their final decision. The same question arises when considering how the neural network behaves during the training phase.

TABLE V
FN rate per attack type for RF, SVM, kNN and LSTM detection with the original dataset

Attack Type	RF	SVM	kNN	LSTM
DoS Hulk	0.09%	3.12%	2.97%	0.17%
PortScan	0.02%	6.86%	1.73%	0.03%
DDoS	0.08%	0.09%	0.12%	0.11%
DoS Golden-Eye	1.58%	5.77%	4.34%	6.28%
FTP-Patator	0.06%	2.84%	0.18%	0.09%
SSH-Patator	0.16%	0.68%	1.22%	0.38%
DoS slowloris	1.23%	3.16%	6.32%	42.08%
DoS Slowhttptest	1.66%	1.74%	3.42%	2.19%
Bot	63.01%	97.04%	100.00%	100.00%
Web Attack - Brute Force	5.29%	28.73%	6.06%	6.76%
Web Attack - XSS	4.20%	12.83%	7.67%	5.19%
Infiltration	80.00%	100.00%	100.00%	100.00%
Web Attack - Sql Injection	57.14%	100.00%	100.00%	87.44%
Heartbleed	0%	0%	0%	0%

TABLE VI
FN rate per attack type for RF, SVM, kNN and LSTM detection with 10% modified attacks profiles

Attack Type	RF	SVM	kNN	LSTM
DoS Hulk	14.52%	68.31%	47.44%	28.17%
PortScan	1.94%	36.14%	.33,89%	7.35%
DDoS	21.49%	77.11%	89.37%	36.18%
DoS Golden-Eye	41.88%	44.08%	45.92%	45.03%
FTP-Patator	1.12%	3.15%	4.74%	2.03%
SSH-Patator	1.84%	5.32%	4.28%	1.93%
DoS slowloris	35.29%	100.00%	100.00%	52.06%
DoS Slowhttptest	39.55%	100.00%	100.00%	41.73%
Bot	88.31%	100.00%	100.00%	100.00%
Web Attack - Brute Force	57.26%	100.00%	100.00%	89.91%
Web Attack - XSS	63.62%	100.00%	100.00%	97.16%
Infiltration	100.00%	100.00%	100.00%	100.00%
Web Attack - Sql Injection	98.73%	100.00%	100.00%	100.00%
Heartbleed	0%	0%	0%	0%

Anyway, the bad result of this experiment is that the 4 selected supervised machine learning algorithms are not able to detect attacks when the packet sizes and IAT have been changed compared to the original training dataset, IAT appearing as the key feature generally used by ML algorithms.

VII. Concluding remarks

This paper deals with investigating adversarial attacks for evading supervised Machine Learning-based attack detection systems. It proposes two main contributions. The first one aims at showing how easy it is to create attacks that have significant chances to evade the supervised Machine Learning-

TABLE VII
FN RATE PER ATTACK TYPE FOR RF, SVM, KNN AND LSTM DETECTION WITH 25% MODIFIED ATTACKS PROFILES

Attack Type	RF	SVM	kNN	LSTM
DoS Hulk	38.87%	100.00%	100.00%	84.21%
PortScan	5.16%	78.33%	69.42%	47.17%
DDoS	54.31%	100.00%	100.00%	100.00%
DoS Golden-Eye	81.52%	100.00%	100.00%	100.00%
FTP-Patator	3.25%	11.76%	16.46%	6.17%
SSH-Patator	4.36%	19.39%	23.19%	7.87%
DoS slowloris	69.41%	100.00%	100.00%	100.00%
DoS Slowhttptest	74.42%	100.00%	100.00%	100.00%
Bot	100.00%	100.00%	100.00%	100.00%
Web Attack - Brute Force	79.03%	100.00%	100.00%	100.00%
Web Attack - XSS	98.80%	100.00%	100.00%	100.00%
Infiltration	100.00%	100.00%	100.00%	100.00%
Web Attack - Sql Injection	100.00%	100.00%	100.00%	100.00%
Heartbleed	0%	0%	0%	0%

TABLE VIII
FN RATE PER ATTACK TYPE FOR RF, SVM, KNN AND LSTM DETECTION WITH 50% MODIFIED ATTACKS PROFILES

Attack Type	RF	SVM	kNN	LSTM
DoS Hulk	72.23%	100.00%	100.00%	100.00%
PortScan	12.06%	100.00%	100,00%	94.67%
DDoS	82.92%	100.00%	100.00%	100.00%
DoS Golden-Eye	97.77%	100.00%	100.00%	100.00%
FTP-Patator	6.09%	27.72%	31.19%	12.84%
SSH-Patator	8.67%	47.32%	51.34%	28.87%
DoS slowloris	86.89%	100.00%	100.00%	100.00%
DoS Slowhttptest	91.63%	100.00%	100.00%	100.00%
Bot	100.00%	100.00%	100.00%	100.00%
Web Attack - Brute Force	94.49%	100.00%	100.00%	100.00%
Web Attack - XSS	100.00%	100.00%	100.00%	100.00%
Infiltration	100.00%	100.00%	100.00%	100.00%
Web Attack - Sql Injection	100.00%	100.00%	100.00%	100.00%
Heartbleed	0%	0%	0%	0%

based detection. It is shown that it is enough to modify the size of packets and the temporal profile of packets arrivals, not to fall in cases learned by the ML algorithm during its training phase. This result is a very bad news as is is then very easy to apply such a simple methodology to all existing attacks for them to remain undetected by ML-based detection algorithms. The second contribution deals with explaining, in the case of RF, the reasons of the bad detection performance when facing our new adversarial attacks. The goal of this second contribution deals with opening the path to explainable AI, and then providing to security managers some explanations why a given flow is classified as benign or malicious.

The strongest contribution of the work presented in this paper certainly deals with studying how the RF algorithm makes its classification decision by going deeply in the analysis of the RF trees. It appears that very small changes on some parameters of the traffic samples can make the RF trees change completely, notably with changes in the order of the most significant features for making the decision, or the decision thresholds. The RF algorithm then appears as very sensitive to the data it takes advantage of for the training and the classification phases. It can even appear as chaotic. This specificity is a lack that makes the design of adversarial attacks very easy.

REFERENCES

[1] N. Elmrabit, F. Zhou, F. Li, H. Zhou, “Evaluation of Machine Learning Algorithms for Anomaly Detection”, IEEE International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2020

[2] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, “Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches”, Transactions on Emerging Telecommunications Technologies, 32(1), 2021

[3] J. Aiken, S. Scott-Hayward, ”Investigating Adversarial Attacks against Network Intrusion Detection Systems in SDNs”, IEEE Conference on Network Functions Virtualization and Software Defined Networks, Dallas, TX, USA, 12-14 November 2019

[4] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, M. Colajanni, ”Deep Reinforcement Adversarial Learning against Botnet Evasion Attacks”, IEEE Transactions on Network and services Management, Vol. 17, No. 4, December 2020

[5] P.A. Alves Resende, A. Costa Drummond, “A Survey of Random Forest Based Methods for Intrusion Detection Systems”, ACM Computing Survey, vol. 51, No. 3, May 2018

[6] A. H. Mirza, S. Cosan, ”Computer network intrusion detection using sequential LSTM neural networks autoencoders”, In IEEE 26th Signal Processing and Communications Applications Conference (SIU), IEEE, 2018

[7] C. Yin, Y. Zhu, S. Liu, J. Fei, H. Zhang, ”Enhancing network intrusion detection classifiers using supervised adversarial training”, Springer Journal of Supercomputing, 2019

[8] G. Apruzzese, M. Andreolini, M. Colajanni, M. Marchetti, ”Hardening random forest cyber detectors against adversarial attacks”, IEEE Transactions on Emerging Topics in computational Intelligence, Vol. 4, No. 4, 2020

[9] G. Apruzzese, M. Colajanni, L. Ferretti, M. Marchetti, ”Addressing Adversarial Attacks Against Security Systems Based on Machine Learning”, IEEE International Conference on Cyber Conflicts, May 2019

[10] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, ”The Limitations of Deep Learning in Adversarial Settings”, IEEE European Symposium on Security and Privacy (Euro S&P’2016), March 2016

[11] S. Calzavara, C. Lucchese, G. Tolomei, ”Adversarial Training of Gradient-Boosted Decision Trees”, ACM International Conference on Information Knowledge management, 2019

[12] D.J. Miller, Z. Xiang, G. Kesidis, ”Adversarial Learning targeting Deep neural Network Classification: A Comprehensive Review of Defense Against Attacks”, proceedings of the IEEE, Vol. 108, 2020

[13] Y. Senzaki, S. Ohata, K. Matsuura, ”Simple Black-Box Adversarial Examples Generation with Very Few Queries”, IEICE Transactions on Information and Systems, Vol. 103, No. 2, 2020

[14] I. Sharafaldin, A. Habibi Lashkari, A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, pages 108-116, 2018

[15] RandomForestClassifier Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[16] Understanding the decision tree structure Scikit Learn : https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html