2021 IEEE 7th International Conference on Network Softwarization (NetSoft) | 978-1-6654-0522-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/NetSoft51509.2021.9492526

Mitigating Evasion Attacks on Machine Learning based NIDS Systems in SDN

Aparna Ganesan
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080
Aparna.Ganesan@utdallas.edu

Kamil Sarac
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080
ksarac@utdallas.edu

Abstract—**Today, network-based intrusions are among the most prevalent security threats our networked systems face. In the case of software-defined networks (SDN), not only the connected devices and services but also the SDN controllers may be subjected to intrusion attempts. The advent of efficient and robust machine learning (ML) algorithms along with the availability of a large number of network datasets enabled the development of ML-based network intrusion detection systems (NIDS). Recent work has demonstrated that ML-based NIDS systems are vulnerable to evasion attacks where the adversary targets the ML classifier in the NIDS system to evade detection by performing various packet perturbations. In this work, we propose an approach to build robust ML based NIDS systems that use multiple ML classifiers trained with reduced feature sets. Our approach depends on a careful feature selection procedure based on Permutation Feature Importance, a wrapper based feature engineering method. Our evaluations on well-known datasets show that the proposed hybrid multi-classifier system is robust and performs well against the packet perturbation attacks considered in this work.**

Keywords—**-Intrusion detection systems, software defined networks, adversarial evasion attacks, machine learning, feature engineering, multi-classifier systems.**

I. Introduction

Due to ever expanding size and fully connected nature of the Internet, assuring the security of our networks and their connected services as well as devices has become a significant challenge. Today, network-based intrusions are among the most prevalent security threats our networked systems face [1]. In the case of software-defined networks (SDN), not only the connected devices and services but also the SDN controllers may be subjected to intrusion attempts.

Defense against intrusion attempts involves utilizing intrusion detection systems (IDS). IDS that mainly focus on detecting network-based intrusion attempts are called network-based IDS or NIDS. The advent of efficient and robust machine learning (ML) algorithms along with the availability of a large number of network datasets enabled the development of ML-based NIDS. In case of SDN, the centralized nature of the network control provides strong support to implement ML-based NIDS that leverage the ability to have an easy access to network wide monitoring data in SDN [2].

While ML-based NIDS are effective in detecting intrusion attempts, it has been recently shown that they can be misled by capable adversaries who aim to evade ML classifiers in NIDS [3]. There are different types of adversarial attacks that can be launched against ML-based NIDS including evasion, poisoning, and over-stimulation attacks [1]. Here, we aim to work against evasion-based adversarial attacks on ML-based NIDS in SDN. In evasion attacks, the adversary launches an attack by perturbing the features of the packet with the goal of evading the ML-based NIDS. Using packet crafting techniques, the adversaries can make an ML classifier inaccurately classify attack packets as benign, thereby bringing the accuracy of the ML-based NIDS from 99+% all the way down to 0% [3]. When we use ML classifiers in the conventional way, with all useful features in the dataset to train the model, the adversary with domain knowledge and access to packet crafting techniques can manipulate packet headers and launch attacks on ML-based NIDS.

Our goal is to improve the effectiveness of ML-based NIDS so as to detect such packet perturbations by making these systems immune and robust against evasion attacks. To combat these attacks, we propose to identify multiple sets of essential features and utilize several ML classifiers on these RFS. We require that the ML classifiers utilizing the RFS perform as well as the ones that utilize the entire feature set in the absence of adversarial attacks. We need the individual ML classifiers to be robust to some type of evasion attacks such that the collection of these ML classifiers are robust to all known evasions and using them together in some fashion helps us better fight against evasion attacks.

Using KDD'99 [4], CICIDS [5] and DARPA [6] datasets, (1) we demonstrate that working with a subset of important features can be as effective as using the full feature set in ML-based NIDS, (2) we propose a methodology to identify important feature sets in a given dataset, and (3) we use ML classifiers that utilize these RFS and show that they perform better than the conventional classifiers when subjected to adversarial evasion attacks. Our experimental results show that the accuracy of the resulting ensemble classifier has improved across all considered types of evasion attacks.

II. Related Work

ML algorithms perform well in applications that involve pattern recognition and anomaly detection. With the availability of large number of network datasets, several ML-based solutions are implemented in IDS. The centralized nature of network control in SDN provides strong support to implement ML-based IDS in these networks. In [2], authors study existing ML-based solutions for intrusion detection in SDN. They provide an extensive analysis of various types of learning methods for IDS in SDN. In [7], authors implement a scalable ML-based anomaly detection framework, called *Athena*, in SDN. *Athena* uses several supervised and unsupervised ML algorithms and achieves good detection accuracy. In addition to the ML techniques, deep learning methods are applied to build IDS with good performance results [8]. For example, in [9], the authors propose Reconstruction from Partial Observation (RePO) in NIDS by using denoising autoencoders in unsupervised manner to deal with adversarial evasion.

Adversarial attacks make up an important attack vector for IDS. Corona et al. [1] provide a taxonomy of adversarial attacks on IDS and outline approaches to address them. A more recent work by Aiken and Scott-Hayward [3] demonstrates the impact of feature manipulation by adversaries. The adversary performs feature engineering to learn the features that could be used by the ML classifiers in the IDS to generate attack packets. By perturbing those features, they successfully bypass the IDS using SYN flood DDoS attacks bringing down the accuracy of the IDS drastically. Here, we propose a solution to the problem demonstrated in this work. We also make use of the adversarial tool developed in this work, called *Hydra*, to perform evasion based attacks in our evaluations.

In [10], authors propose a wrapper based feature engineering technique as an adversary aware method using both forward selection and backward elimination. Although the feature set is proved to perform better than the alternatives, the resulting feature set is fixed. This is important because the adversary can now figure out that feature set and can come up with new packet manipulation approaches to evade detection.

In [11], the authors propose a defense mechanism against the adversarial attacks on NIDS based on Neural Networks. They generate their evasion dataset using a part of the training dataset by performing mathematical transformations. In our work, we use multiple ML classifiers to build an ensemble based solution and test our solution by generating evasion datasets using an adversarial tool in an emulated SDN. In [12], authors perform a feature engineering to find the best RFS to train considered ML models for DDoS attack defense in SDN. In our work, we develop multiple RFS to be used by several ML algorithms in an ensemble ML classifier for defense against evasion attacks.

III. Proposed Solution

We observe that evasion attacks are effective because of successful feature manipulation. Here, we perform extensive feature engineering to address the problem. First, we conduct an experiment to demonstrate that working with a carefully selected RFS can provide good results in intrusion detection under no evasion case. Next, we use a feature ranking method along with four ML classifiers, namely Support Vector Machine (SVM), Logistic Regression (LR), Neural Networks (NN), and Random Forests (RF), to identify feature sets to be utilized in our ML-based NIDS. Last, we conduct extensive experiments to evaluate the performance of the resulting ML classifiers in the absence as well as presence of evasion attacks.

A. Feature Engineering

While designing a ML model, the number of features and the contribution of those features to the model play an important role in forming the resulting decision boundary, the geometrical surface that separates the feature space into regions belonging to each class of the target variable (benign/malicious). We see that evasion attacks are successful because the adversary is able to manipulate certain packet features to evade the ML based IDS. That is, when we perform feature engineering to identify the importance of each feature, form several subsets of the overall feature set, and use only these subsets of the features during training of the ML classifiers, we obtain different decision boundaries that may be robust against different evasion attacks. If the selected feature is not impacting the decision of the ML classifier in use, we may expect the perturbation attacks targeting the feature may not be effective in misleading the classifier.

To make the ML-based IDS robust, we propose using multiple classifier models trained with different feature sets. Consider a dataset with 12 usable features and assume that the minimum number of features with which the models can be generated is 7. Even with this limit, there are $\sum_{r=1}^{5} 12P_r$ possible feature sets, where $12P_r = \frac{12!}{(12-r)!}$, which is quite large. Therefore, it becomes essential to introduce some structure while choosing the feature sets for training. Here, we remove features based on feature importance. There are several methods that can be used to rank the features based on their importance, namely, wrapper based methods, filter based methods, and embedded methods [13]. Here, we use a wrapper based method for ranking features for their importance.

B. Permutation Feature Importance

We use permutation feature importance (PFI), a wrapper based method, to rank features based on their importance. PFI is model agnostic, i.e., it can be used with different ML classifiers. We use PFI with four ML classifiers (SVM, LR, NN, and RF) to obtain feature rankings. PFI ranks features through a greedy approach and the resulting feature ranking can be considered as an optimal feature ranking for the selected ML model and the training dataset.

PFI works by changing the values of the features in the dataset and evaluates the impact of those changes in the model score. This is an iterative method, where during each iteration, one feature is picked and the values of that feature column is randomly shuffled multiple times to generate multiple corrupted versions of the dataset. Based on the change in the model score with the corrupt data, importance value of that particular feature is evaluated. The module gives out the importance value of each feature and based on which the feature ranking is obtained. This operation is performed for every feature of the dataset iteratively.

C. Hypothesis

We propose to find multiple subsets of essential features and develop multiple ML models using those feature sets.

Features	No evasion	pairflow	pay_pair	payload	rate_pair	rate_pay	rate	stealth
Support Vector Machine	0.97522	0.28642	0.99012	0.98678	0.27386	0.98645	0.43747	0.99095
Logistic Regression	0.97759	0.53037	0.98653	0.98129	0.52153	0.98083	0.43183	0.98720
Neural Networks	0.95188	0.26640	0.28740	0.39921	0.25584	0.40913	0.40977	0.27318
Random Forests	1.0	0.54083	0.99685	0.99944	0.53351	1.0	0.45097	0.99701

TABLE I: Accuracy values of original classifiers including all features with Evasion

We make sure that those models do not perform worse than the original classifier. That is, we need the accuracy of those classifiers to be acceptable when compared to the original classifiers that we would normally use in the absence of adversarial attacks. Then, we could use these classifiers together in multiple different ways to design a robust IDS. In this work we propose one way to use the combination of multiple classifiers with RFS by programming the resulting IDS to pick an ML classifier randomly so as to predict the class of the incoming packets in a periodic manner. The classifiers can also be used in a time shared manner, that is, a certain model with a certain feature set can be used for a period of time before it is replaced with another one. The reason behind this approach is that, when the IDS uses the entire feature set, there is no uncertainty for the adversary, that is, with the right packet manipulation, evasion is straightforward. With the proposed scheme, a successful evasion will require that the adversary choose to focus on a certain feature that is important to the classifier currently in use. Since the scheduling of the classifier will be kept confidential, it will be difficult for the adversary to successfully launch these type of evasion attacks.

IV. Evaluations and Results

In this work, the metric used for evaluation of the ML models is accuracy as the problem in hand is a classification problem. The accuracy of a model is defined as the ratio of the total number of data points classified correctly to the total number of data points the model encounters during testing and it can be calculated as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

where *TP = True Positives*, *FP = False Positives*, *TN = True Negatives*, *FN = False Negatives*.

The evaluations of the proposed method are done in two phases. In phase 1, we use the well-known KDD'99 [4] dataset to evaluate the performance of ML models with RFS. This phase does not consider evasion attacks and mainly focuses on the accuracy of the models with smaller number of features as compared to the case with full feature set. In phase 2, we train the models using CICIDS [5] and DARPA [6] datasets to evaluate the performance of the classifiers during evasion attacks. The dataset for evasion is generated by using Hydra [3] which emulates an SDN in Mininet with Faucet controller to launch attack traffic. The final model is tested against the evasion dataset generated by this emulation environment.

A. Phase 1: Hypothesis Validation

Here, we used KDD'99 dataset to analyse the efficiency of classifiers built with lesser number of features after removing the features not ranked important by the PFI. We want to ensure that the accuracy of the system with RFS is as well as that of the system with the full feature set. While trying to make an ML-based IDS robust against evasion attacks, we cannot accept a system that will have poor accuracy during the normal operation. The results showed us that with proper feature engineering, even as the number of features goes as low as 4 or 5 from 41, the accuracy values are comparable. In case of LR trained with the features obtained from SVM, the accuracy value improved even with just 3 features when compared to 41. We also found that there are large number of models that perform extremely well with RFS. These results were encouraging and showed that building an ensemble classifier with multiple classifiers with RFS will be efficient during normal operations (i.e., no evasion attacks). The evaluations are not presented due to space constraints.

B. Datasets with RFS for Evasion

In this part, we use the dataset that is used in [3]. In [3], the authors use a dataset consisting of CICIDS and DARPA datasets for training. They have designed and implemented an adversarial tool, called *Hydra*, to generate evasion based attacks on an emulated SDN environment for testing. We use the same setup to run our experiments.

The training dataset comprises of benign data points from CICIDS dataset and malicious data points from DARPA SYN flood dataset. The original CICIDS dataset after preprocessing includes about 40,000 benign data points and the original DARPA data set has 32,000 malicious data points. We randomly select 80% of the data points from the CICIDS datasets and all of the data points from the DARPA dataset to include in our training dataset. The 20% of data points from the CICIDS dataset are used as part of the evasion dataset, that is used for testing. The datasets used in this part have a total of 12 usable features.

Evasion dataset is generated using the Hydra adversarial tool on an emulated SDN environment. The evasion dataset includes seven types of packet manipulations performed on some prominent features of the flows. The features are manipulated separately as well as in combinations of two features. Every generated dataset is mixed with the isolated 20% of the data points from CICIDS dataset to make sure the attack dataset is not skewed and has both benign and evasion traffic. This makes the experiments look closer to real time network

Rank	Support Vector Machine	Logistic Regression	Neural Networks	Random Forests	Overall Feature Ranking (OFR)
1	bytes_per_second	bytes	dst_bytes	bytes_per_second	bytes_per_second (8)
2	dst_bytes	dst_bytes	bytes_per_second	pair_flow	bytes(12)
3	src_pkts	src_bytes	bytes	bytes_per_packet	dst_bytes(13)
4	bytes	bytes_per_second	pkts	bytes	bytes_per_packet(18)
5	bytes_per_packet	bytes_per_packet	bytes_per_packet	src_pkts	src_pkts(25)
6	pkts	pair_flow	dst_pkts	packet_pair_ratio	pair_flow(27)
7	dst_pkts	ip_proto	src_pkts	pkts_per_sec	src_bytes(30)
8	dst_bytes	packet_pair_ratio	src_bytes	pkts_per_sec	pkts(33)
9	pair_flow	pkts_per_sec	pkts_per_sec	src_bytes	pkts_per_sec(33)
10	src_bytes	src_pkts	pair_flow	dst_pkts	dst_pkts(34)
11	packet_pair_ratio	dst_pkts	packet_pair_ratio	pkts	packet_pair_ratio(36)
12	ip_proto	pkts	ip_proto	ip_proto	ip_proto(43)

TABLE II: Feature engineering on the evasion dataset

Models	Classifiers	# Features	No Evasion	pairflow	pay_pair	payload	rate_pair	rate_pay	rate	stealth
Support Vector Machine	C_1	11	0.95273	0.99458	0.99415	0.99188	0.99479	0.99167	0.99166	0.99444
	C_2	10	0.90889	0.28433	0.60673	0.42625	0.27339	0.43604	0.43689	0.53092
	C_3	9	0.94890	0.26883	0.29002	0.40285	0.25818	0.41286	0.41356	0.27571
	C_4	8	0.79775	0.27682	0.98012	0.97239	0.26464	0.97171	0.42270	0.98110
	C_5	7	0.97653	0.27660	0.97988	0.97206	0.26442	0.97136	0.42235	0.98087
Logistic Regression	C_6	11	0.96455	0.94145	0.93530	0.91472	0.94495	0.91306	0.91281	0.93803
	C_7	**10**	**0.97229**	**0.96682**	**0.96420**	**0.95027**	**0.96813**	**0.94904**	**0.94896**	**0.96597**
	C_8	9	0.96422	0.96375	0.95989	0.94815	0.96677	0.94686	0.94683	0.96183
	C_9	8	0.95338	0.97078	0.96698	0.95867	0.97348	0.95862	0.95844	0.96815
	C_{10}	7	0.95387	0.26711	0.96815	0.96030	0.25685	0.96029	0.41114	0.96926
Neural Networks	C_{11}	11	0.95163	0.26625	0.28720	0.39899	0.25581	0.40902	0.40959	0.27303
	C_{12}	10	0.95196	0.26558	0.28648	0.39798	0.25505	0.40787	0.40850	0.27234
	C_{13}	9	0.95008	0.26606	0.58618	0.39871	0.25552	0.40861	0.40925	0.51122
	C_{14}	8	0.95750	0.51453	0.96787	0.95991	0.50785	0.96017	0.41109	0.96899
	C_{15}	7	0.95391	0.51382	0.96710	0.95884	0.50717	0.95908	0.41	0.96838
Random Forests	C_{16}	11	0.99991	0.29203	0.99504	0.99764	0.28042	0.99804	0.44896	0.99482
	C_{17}	10	0.99991	0.54281	0.99995	0.99994	0.53347	0.99994	0.45097	0.99996
	C_{18}	9	0.99991	0.54281	0.99995	0.99994	0.53347	0.99994	0.45097	0.99996
	C_{19}	**8**	**0.99983**	**0.99996**	**0.99995**	**0.99994**	**0.99996**	**0.99994**	**0.99994**	**0.99996**
	C_{20}	**7**	**0.99975**	**0.99996**	**0.99995**	**0.99994**	**0.99996**	**0.99994**	**0.99994**	**0.99996**

TABLE III: Combined results with the RFS

operation. The evasion cases are pairflow, payload, payload and pairflow, rate, rate and pairflow, rate and payload, stealth.

Using the training dataset we trained four ML classifiers including SVM, LR, NN and RF and performed the experiments with and without evasions. Table I shows the effect of evasion on the ML classifiers trained with the full feature set of 12 features. We see that out of the seven evasions analysed, *pairflow, rate-pairflow* and *rate* evasions are successful across all four models trained with 12 features. Rest of the evasions, namely, *payload-pairflow*, *payload*, *rate-payload* and *stealth* evasion attacks are successful in at least one of the classifiers. We see that not a single classifier is robust across the entire range of evasion attacks. Hence, using one model in an IDS makes it vulnerable to at least three evasion attacks. Thus we need to design an IDS that is robust and efficient in catching all known evasion attacks. We also observe in Table I that, in the absence of evasion, all four classifiers perform extremely well, which is validated by [3].

C. Phase 2: Evaluations with Evasion Attacks

Our experimental study in Section IV-A demonstrated that the use of RFS could be effective in defending against non-evasion attacks. In this section, we use a similar methodology to develop a number of ML classifiers trained with RFS and test their performances in the presence of evasion attacks.

In the first step, we rank the features of our evasion dataset based on their importance. We use PFI with four ML classifiers including SVM, LR, NN and RF to calculate importance values of each feature and rank them from most to least important as shown in Table II. From these rankings, we calculate an *overall feature ranking (OFR)* for each feature in the dataset. OFR value of a feature i is calculated as the sum of the ranks for feature i as $OFR_i = \sum_j r_{ij}$, where r_{ij} refers to the rank of feature i in ranking $j \in \{SVM, LR, NN, RF\}$. The last column in Table II shows the ranking of the features based on their OFR values. We use the OFR ranking of the features in building ML classifers with RFS. We expect that the nature of OFR calculation will help normalize the impact of feature removal across all the ML classifiers with RFS.

From the training dataset, using the OFR ranking, datasets with RFS are obtained. The original dataset has 12 features and in this process we obtain 5 datasets with number of features going from 11 to 7 by removing the top five highest ranked features, consecutively, from the main feature set. Using these five datasets, we train four ML algorithms and get 20 ML classifiers named as C_i for $i = [1..20]$ in Table III. Each RFS when used for training a model will result in a different decision boundary and hence different performance results. The performance of the models with RFS show that some of these models perform on par with models trained with full feature set in the absence of evasion and perform better in the presence of various type of evasion attacks.

We performed another analysis keeping the number of features constant as 11 and swapping out one of the top five ranked features from the feature set where ranking of the features are obtained from PFI with the corresponding ML algorithm, i.e., we do not use OFR but use the ranking corresponding to the ML algorithm given in Table II. This analysis is performed to increase the available model choices when building an ensemble classifier. The evaluations are not included due to space constraints. With the above experimental cases, we have a total of 40 classifiers to be considered. The classifiers are named $C_1, C_2, \ldots C_{40}$ for reference. We see that, out of the 40 cases observed, some classifiers are not as good as the original classifiers in case of no evasion scenario (e.g., C_{11}). Therefore, from these classifiers, we choose the ones that are robust both with and without evasion for our ensemble classifiers. We build two different the ensemble classifiers: one with just two models, C_{19} and C_{20} and the other one with 6 models, namely, $C_7, C_{19}, C_{20}, C_{23}, C_{24}, C_{28}$ (C_{23}, C_{24}, C_{28} - not included due to space limitations).

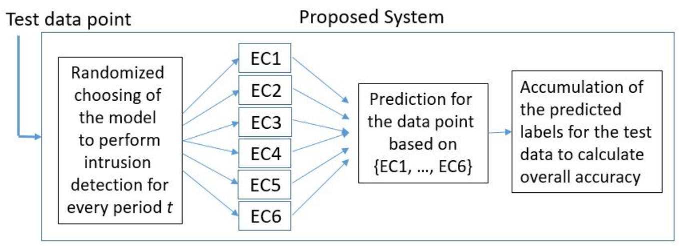
Test data point
Proposed System
Randomized choosing of the model to perform intrusion detection for every period t
EC1
EC2
EC3
EC4
EC5
EC6
Prediction for the data point based on {EC1, ..., EC6}
Accumulation of the predicted labels for the test data to calculate overall accuracy

Fig. 1: Ensemble IDS (IDS with multiple ML classifiers)

Evasion	Accuracy of ensemble classifier with 2 models	Accuracy of ensemble classifier with 6 models
No evasion	1.0	0.98940
pairflow	0.99996	0.98277
pay_pair	0.99995	0.98049
payload	0.99994	0.97435
rate_pair	0.99996	0.98399
rate_pay	0.99994	0.97463
rate	0.99994	0.97373
stealth	0.99996	0.98064

TABLE IV: Accuracy of the ensemble classifier

The working of the proposed method is shown in Figure 1. The individual models used in the ensemble classifier are trained with the training data as discussed above. During testing (or operational) time, for each data point, we randomly select one of the models in the ensemble classifier to label the data point as benign or malicious. Once we go through the entire evasion dataset by going through the data points individually, we accumulate the individual predictions and calculate the accuracy of the overall ensemble classifier for the evasion dataset and report it. The accuracy of the two ensemble classifiers is tabulated in Table IV. For this training dataset and evasion cases considered, we find that the first ensemble classifier performs extremely well. In addition to this, we are interested in the second ensemble classifier with six ML models to establish the idea of increasing the uncertainty for the adversary so that it is not possible to launch an attack targeted at a particular model or feature. The accuracy values in column 2 of Table IV show that the ideal case of robustness is achievable by using models with RFS whereas the column 3 shows that in case we do not find models like C_{19} and C_{20}, using multiple models as an ensemble together definitely improves the performance when compared to the original case as shown in Table I. We see that the proposed ensemble classifiers are robust against the tested types of evasion attacks.

V. Conclusion

We have focused on the evasion based adversarial attacks on ML-based IDS in SDN. Observing that the evasion attacks mainly target at manipulating certain features to confuse the ML classifiers via packet perturbations, we have proposed to use multiple feature sets of reduced size to identify multiple ML classifiers that would be more robust to evasion attempts when used in an ensemble. Our experimental evaluations using well-known datasets has shown that the proposed ensemble classifier is effective in detecting several evasion strategies that were gone undetected by the traditional use of a single ML classifier with the full training dataset. This work is partially supported by NSF award DGE-1820640.

References

[1] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *IS*, 2013.

[2] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking: Research issues and challenges," *IEEE CST*, 2019.

[3] S. S. J. Aiken, "Investigating adversarial attacks against network intrusion detection systems in sdns," in *IEEE Conference NFV-SDN*, 2019.

[4] "Kdd cup 1999 dataset." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[5] I. Sharafaldin, A. H. Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018.

[6] M. Gharaibeh, "Darpa 2009 intrusion detection dataset," (2009). [Online]. Available: http://www.darpa2009.netsec.colostate.edu

[7] S. Lee, J. Kim, S. Shin, P. Porras, and V. Yegneswaran, "Athena: A framework for scalable anomaly detection in software-defined networks," in *47th Annual International Conference on DSN*, 2017.

[8] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking," *EAI Endorsed Transactions on Security and Safety*, 2017.

[9] M. J. Hashemi and E. Keller, "Enhancing robustness against adversarial examples in network intrusion detection systems," 2020.

[10] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Transactions on Cybernetics*, 2016.

[11] M. Pawlicki, M. Choraś, and R. Kozik, "Defending network intrusion detection systems against adversarial evasion attacks," *FGCS*, 2020.

[12] H. Polat, O. Polat, and A. Cetin, "Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, 2020.

[13] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, Mar. 2003.