Evaluating Evasion Attack Methods on Binary Network Traffic Classifiers

Rishi Shah
rshah36@mit.edu

Jeff Gaston
jg856674@wcupa.edu

Matthew Harvey
msharve2@asu.edu

Michael McNamara
mcnamaramj20@mail.vmi.edu

Osvaldo Ramos
10502244@uvu.edu

Yeonsang You
yyou@hawaii.edu

Elie Alhajjar
elie.alhajjar@westpoint.edu

Army Cyber Institute
United States Military Academy
West Point, NY 10996, USA

Abstract

Adversarial machine learning focuses on attempting to deceive machine learning classifiers by carefully manipulating input data. A lot of work has been done in the context of image classification, where image classifiers can be fooled by a "small" change in the input data that is not recognizable to the human eye. In this paper, we investigate the vulnerability phenomenon in a different setting, namely the network traffic domain. We conduct six experiments with various combinations of machine learning algorithms, network traffic data sets, and perturbation methods, and we measure the effects of such perturbations on binary network traffic classifiers. The algorithms used include neural networks (NN), support vector machines (SVM), decision trees (DT), and random forests (RF). The perturbations methods implemented include random perturbation and generative adversarial networks (GAN). Within the data sets, malicious network traffic data are perturbed while benign data are left unchanged. Across all experiments, we observe a drastic reduction in the accuracy of the classification models. This supports the hypothesis that evasions attacks can indeed fool not only image classifiers but also binary network classifiers.

Keywords: adversarial machine learning, evasion attacks, network traffic classifiers, data perturbation.

1. INTRODUCTION

Recent studies on image classification models have elucidated a class of vulnerabilities which cause misclassification through malicious input. Such input data are called "adversarial examples" (Goodfellow et al., 2015), they can be created using specific methods to exploit gaps in a model's performance or the latent features upon which the classifiers are based (Ilyas, et al., 2019). This study aims to elucidate how this vulnerability manifests in binary network traffic classifiers.

Adversarial attacks can be classified into two categories: targeted attacks and untargeted attacks. In the first instance, an attacker aims to find a perturbed input that is classified in the same target as the original input. In the second instance, the objective is to search for a perturbed input that is different target than the original one. In terms of the learning model. We distinguish between two types of attacks: white-box attacks and black-box attacks. The first type assumes the adversary knows everything related to the trained model, while in the second type it is assumed that the adversary has no access to the trained model but has access only to the output of the model. In the latter case, the effective transferability of evasion attacks across different classification models was demonstrated in several places (see for example Demontis et al., 2018).

Image classification is considered an unconstrained domain, in the sense that the input data can vary freely since it simply consists of a collection of pixels. On the other hand, network traffic is a constrained domain due to the complex and mixed nature of the input data, a fact that renders its manipulation very sensitive and complicated. There are several attack algorithms in the literature, we mention a few of them: the fast gradient sign method (FGSM), the Jacobian-based saliency map attack (JSMA), Deepfool, CW attack among others. We refer the reader to the book by Joseph et al. (2019). In a recent paper (Alhajjar et al., 2019), the authors studied two new techniques for generating adversarial examples in the intrusion-detection setting, namely particle swarm optimization and genetic algorithms. Their results clearly show that machine learning algorithms are vulnerable to adversarial perturbation in unconstrained domains.

Given the ample success demonstrated in various recent studies about adversarial examples in multiple domains, this paper aims to provide more evidence about this phenomenon in machine learning, in particular to highlight the weaknesses in binary network traffic classifiers.

2. METHODOLOGY

In this study, six independent experiments were conducted, each testing a different permutation of training datasets, model architectures and data perturbation methods. Table 1 lists each experiment, and the following sections provide an overview of the technical details of each implementation.

Exp #	Dataset	Model	Perturbation
1	A	NN	Random
2	A	SVM	Random
3	B	RF	AG
4	B	NN	Random
5	C	DT	Random
6	C	NN	GAN

Table 1. Overview of the six experiments conducted.

Datasets

Datasets A and B are 2 of the 13 network captures of botnet traffic of the CTU-13 dataset (García et al., 2014). In each of the datasets, real malware is executed on the computer system and a traffic capture is recorded. The traffic capture includes the malicious botnet traffic with real and background traffic. For each of the 13 datasets, the actions the malware took are different. Specifically, dataset A showcases botnet traffic using *http* to conduct spam and port scanning over the course of 11.63 hours with close to 4.5 million packets. In dataset B, Internet Relay Chat, spam, and click fraud are captured, the duration was 4.21 hours with close to 71.9 million packets. The dataset has 11 raw features: duration, protocol, source IP/port, destination IP/port, flags, Tos, packet count, byte count, and flows. For both datasets, the IP addresses and ports are further subdivided by byte for a total of 8 feature values from these raw features. This process results in an engineered feature set of 15 values. All numerically continuous features are normalized to the interval [0,1]. For features with unique values such as strings, one-hot encoding is used to represent them.

Dataset C is obtained from the Canadian Institute for Cybersecurity's 2017 traffic collection, meant for the testing of Intrusion Detection Systems (IDS). The dataset spans five days of network traffic and includes the execution of more than eight different cyberattacks, including DoS, Botnet, DDoS, and Brute Force SSH. The dataset contains over 80 raw features, which are narrowed down to 16, using cross-correlation analysis with a binarized label (benign vs malicious traffic).

Classification Models
Six independent classification models are trained, of varying type and architecture. Each is developed in Python using the Sklearn and Tensorflow libraries. Neural networks (NN) are the most common classifier used in this series of experiments. Experiments 1 and 4 are implemented using shallow NNs with an Adam optimizer. Learning rates are chosen experimentally based upon accuracy and F1-score. Input size is dependent upon respective datasets, while output is always binary (malicious vs benign).

This is similar to experiment 6, where a deep NN is constructed to conduct binary classification of network traffic. The classification network consists of two dense hidden layers of 500 nodes each and uses a stochastic gradient descent (with Nesterov momentum) for optimization. The architecture is experimentally defined based upon accuracy and F1-score on dataset C.

Besides NNs, experiment 2 utilizes a generic support vector machine (SVM) to classify the network traffic. Experiment 5 uses a decision tree (DT) model that allowes for more accurate characterization of dataset C, since each leaf node represents a class label, i.e. the decision taken after computing all attributes (García et al., 2014). Furthermore, a DT allows for a visual representation of weaknesses and possible target points in the algorithm's function.

Lastly, experiment 3 utilizes a random forest framework to construct a network traffic classifier. Since all decision tree-based models can be overfitted easily based upon maximum height, and because dataset B is particularly unbalanced (with 100:1 benign to malicious data points), the random forest classifier is chosen to accurately model the function of an IDS. In addition, 8 of the 15 features are derived from the IP address and port number raw features.

Perturbation Methods
For each data set, two separate perturbation methods are applied. Each data set is then tested with at least one random perturbation, and (for datasets B and C) variants of a generative method of data perturbation are also implemented.

Randomization
Experiments 1, 2, 5 all use a standard randomization method to perturb their respective datasets. Randomization is bounded by the possible ranges of each feature value. In addition, certain features are bounded such that randomization could only modify input data points in one direction. This is specifically used for features which would be difficult for an adversary to artificially decrease (e.g. duration, packets, bytes) without affecting other feature values.

Given the structural similarities between datasets A and B, experiment 4 utilizes a slightly improved iteration of the random perturbation method. In this modified random method, samples are randomly drawn from a normal distribution of the feature value, as represented in Figure 1. The distribution is created based off the original dataset, while the additional samples drawn from the distribution are used as the perturbed samples. Drawing from the normal distribution helps elucidate more specifically how small modifications in feature values could possibly affect a classifier's output.

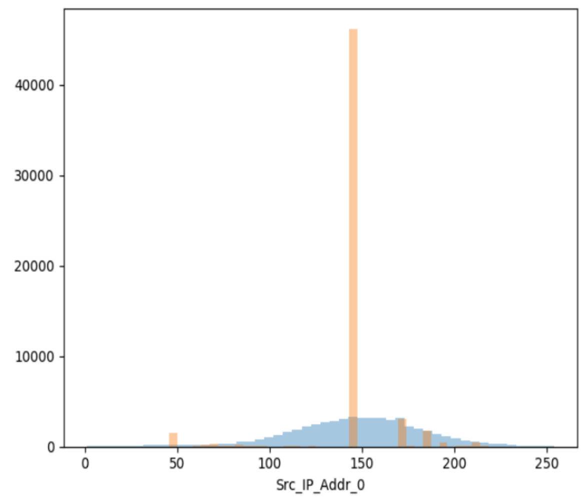

Figure 1. Feature values before (orange) and after (blue) random perturbation.

Augmented Generative
If a machine learning classifier for network traffic flow is considered to be a "black-box", the attackers can only use the input network traffic flow data and output results (benign or malicious) to conduct the adversarial perturbation. The augmented generative method is created in this study for experiment 3, in order to emulate the process of an adversary learning about a network traffic classifier.

In order to elucidate details about the classifier, the random network traffic flow samples are first generated from a uniform distribution of the past data and fed into the machine learning classifier. Several features are kept fixed due to their unique characteristics such as Destination IP address and Destination Port. After the random samples are classified, we obtain the distribution of samples classified as malicious or benign. By using these distributions, the decision boundaries of the classifier are visualized (as shown in Figure 2). In order to further isolate a particular decision boundary, new samples are generated based on the distributions of samples already classified and fed into the machine learning classifier. This process can be conducted multiple times based on the characteristics of the decision boundaries.

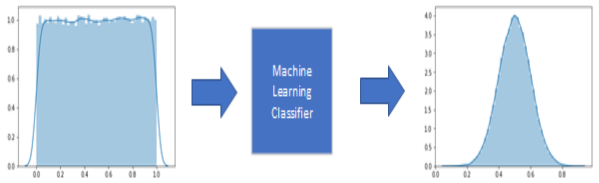

Figure 2. The process to find the probability distribution of decision boundaries.

In figure 3, the decision boundaries are shown as statistical models. This process defines the characteristics of the decision boundaries with probability distributions. If the attackers know the distributions of the decision boundaries, they can fool the machine learning classifier with a masking process. In figure 4, the attackers can turn malicious data to be classified as benign data with a modification of network traffic data.

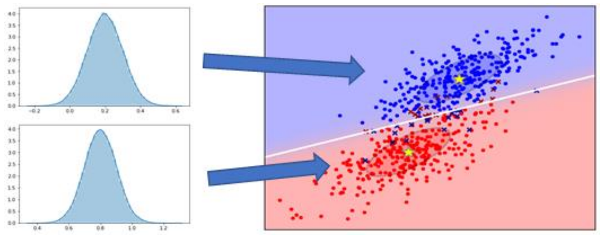

Figure 3. Visualizing decision boundary based on the distribution of classified random sample.

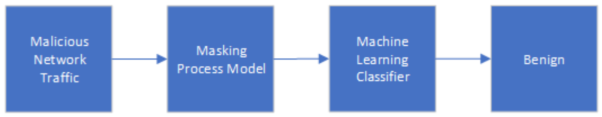

Figure 4. Perturbation workflow for experiment 3.

Generative Adversarial Network

Experiment 6 implements a deep neural network against data points perturbed with a Generative Adversarial Network (GAN). This technique involves the creation of a generative neural network, which takes inputs of a 100-dimensional array of noise (latent variables), and returns outputs of the dimensions of network traffic data points. This network consists of three dense hidden layers, of sizes 256, 512, and 1024 units. Using a "Leaky ReLu" activation function on each layer and an Adam optimization function, the generative network is trained by plugging its outputs into the classification model and conducting weight updates for data points that the classifier was able to tell were fake. In this way, the generator and the classifier are trained simultaneously, both improving their accuracies and F1-score across multiple alternating training rounds.

3. RESULTS & DISCUSSIONS

In each model presented, the dependency on certain features can be adjusted to reduce vulnerability. However, these models and corresponding weaknesses represent models optimized for accuracy and F1-score, rather than security. In this way, these results reveal valuable insights into the feature engineering of secure network traffic classifiers in the future.

Random Perturbation on NNs

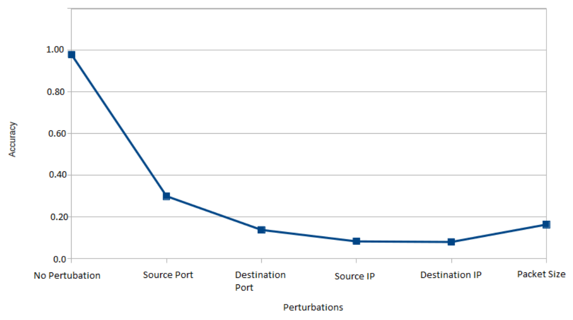

Figure 5. Accuracy levels after successive, compounding perturbations.

As shown in figure 5, the baseline accuracy for experiment 1 is represented, and followed by compounding perturbations. Experiment 1 achieves a baseline accuracy score of 97.91%. The test set after perturbation consists only of malicious data points to measure if the perturbations can cause a misclassification of malicious to benign, the most dangerous case in real-life applications. The most significant decrease in accuracy after perturbing one feature is achieved after perturbing the source port, resulting in a 68.54% decrease in accuracy. In addition, the final data point displays an upward trend in accuracy from 7.966% to 15.26%. This may be caused by over manipulation of the data, resulting in the specific data points becoming obscure and too far from an original non-perturbed data point (and therefore being classified as malicious).

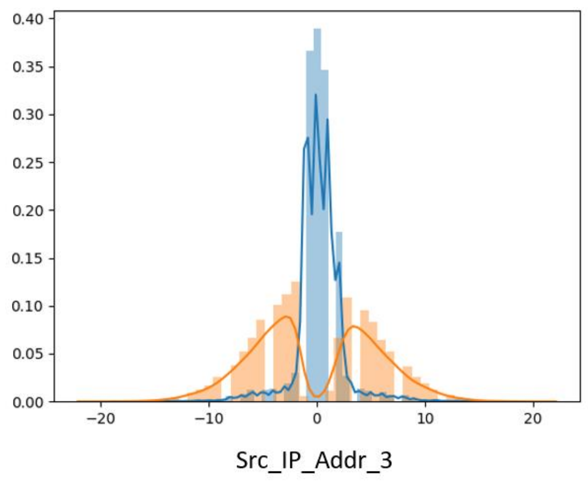

Figure 6. Correct classification (blue) vs misclassification (orange) based on source IP.

Experiment 4 also applies random perturbation to a NN. The original accuracy of its neural network module is 90.49%. After complete randomization of all features, the new accuracy is 12.41%. Each feature is then randomized individually. The most notable changes in accuracy are from the second, third, and fourth bytes of the source IP address with 34.67%, 57.70%, and 27.30% respectively. The only other feature with an accuracy below 90% is source port with an accuracy of 89.88%. Figure 6 shows the result from perturbing the fourth byte of the source IP address. It is important to note that the change needed for the source IP address to cause misclassification of the neural network is relatively small (3-6) compared to the feature size (255). This can be easily done by an attacker using a VPN, spoofing the IP address, or even using a different computer on the same subnet. Source port, while still having a low change in accuracy, is the only other feature to have an accuracy below 90%. A larger change would be needed to cause misclassification; however, these changes could be done by opening another port to send packets on. Another important observation is the abundance of features that do not cause a large change in accuracy; the bytes and packet features cause no change in accuracy. A possible explanation as to why the source IP address and port number are responsible for a huge shift in accuracy is because the dataset is a capture of a malware infected computer, the source IP address of the malicious packets would be mostly constant throughout the dataset. It is worthwhile to note that the destination IP address and port are not as vulnerable as their source counterparts. A probable reason to explain this discrepancy is that the neural network might have found a correlation between source IP address and malicious data before any pattern was found in the destination IP address and port.

Random Perturbation on an SVM

Experiment 2 utilizes a support vector machine on the data set A. Feature values are normalized (e.g. port numbers were all divided by 65,535 which is the largest possible unsigned port number not registered under ICANN). Within this framework, the data points labeled "malicious" in the data set are perturbed in the following way. The upper and lower bounds are fixed using averages of upper and lower quartiles, which made it possible to find a threshold where the classifier misclassifies malicious connections as benign.

Training and testing the model on unperturbed data led to an accuracy of 93.12% and F1-score of 92.6% as shown in Figure 7. When the classifier is tested from an adversarial perspective with perfect knowledge of the system, the classifier performs poorly. The highest accuracy it achieves when tested against the perturbed subset was only 39%.

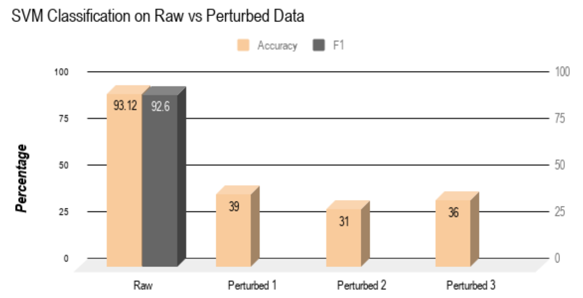

Figure 7. Accuracy (orange) and F1 score (grey) after successive perturbations.

Random Perturbation on a Decision Tree

In experiment 5, a DT is used which shows 98% baseline accuracy and allows for a visual representation of the decision boundaries. Figure 8 shows a sample of one of those branches enlarged to give an idea of how big this decision tree grows based on the classifier with the sample data. The accuracy of each leaf node is recorded as well for the sake of completeness. Therefore, taking that data and perturbing it with a randomized algorithm would allow it to pick between the sixteen classifiers used in our data sample.

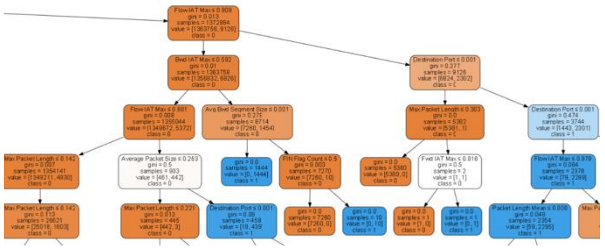
Figure 8. Sample branch of DT. Dark orange represents confidence in a benign label, while dark blue represents confidence in a malicious label.

It is found that the "Max Inter-packet Interval" feature is the most susceptible to targeting by an adversary who tries to cause misclassification via perturbation, as can be seen in the first perturbation of Figure 9. Specifically, perturbing this feature leads to a decrease in accuracy down to 76%.

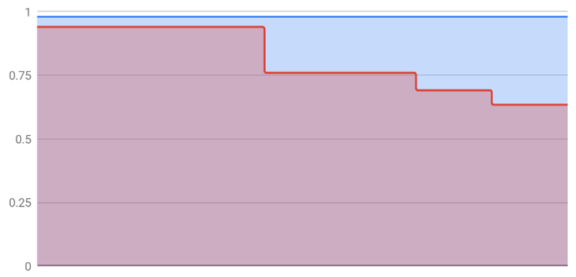

Figure 9. Decreases in accuracy (red) compared to the baseline (blue) across perturbations.

Augmented Generative Method on a Random Forest

The augmented generative perturbation method is applied to one million samples labeled as malicious in experiment 3. The samples are iteratively generalized and reselected into benign data with the masking process described previously. Testing this data on the target classifier, 99.9% of malicious network traffic is classified as benign. With this specialized and realistic perturbation method, an adversary is able to isolate the decision boundaries of a classifier and mask the malicious data as benign data with a great accuracy. The most significant feature of the boundary that was attacked is source IP address. Based on target boundaries characteristics, the most important feature will be perturbed. However, across models and datasets, source IP address seems to not only be heavily dependent upon for classification, but also can be used to cause misclassification.

Generative Adversarial Network on an NN

Constructing and training a GAN resulted, in part, in a generative network able to consistently produce misclassified data samples. When saved off, the unsupervised generative model could be used to consistently produce fake data samples, given only noise from a normal distribution.

As shown in Figure 10, training iterations result in the discriminator and the generator accuracy fluctuating around 50% in the first half of the training batches. Although at first sight this seems poor, the fluctuation shows repeating cycles of generator and discriminator improvements, the sign of successful GAN training. However, most training iterations are not quite successful. Since this type of model is unstable, many training iterations result in complete convergence to an over-trained classifier (a generative network that created poor data samples) too early in the training process. Since successful iterations are rare and seemingly random, the creation and training of the model shows the need for further research into optimization of GANs on network traffic data sets. With this successful model, however, fake data points are created from the latent noise inputs, and subsequently are passed into a separate classifier (with the same architecture as the version in the GAN but trained separately). This separate classifier achieves a 97.54% accuracy with a 93.38% F1-score after being trained on a binary version of the data set, using the feature engineering described in the methods section. However, when tested against fake generated data points, 87% of them are classified as malicious. This result implies that the generator, during GAN training, converges to malicious feature values more than benign and/or the additional classification network learns to classify data points in a discrete range as benign, and all others as malicious.

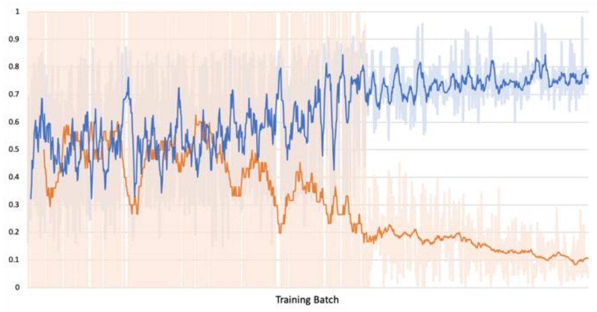

Figure 10. Accuracy of the discriminator network (blue) and generator network (orange) during training iterations.

Overall, GANs prove to be insightful in automating the process of data perturbation and could provide an avenue to discover new perturbation methods. However, this study also showcases the need for significantly more research into the application, given its instability during training and ambiguity regarding the reason behind misclassification of the generated points.

4. CONCLUSIONS

This study represents an additional step forward in the field's understanding of the effect of perturbed data on various network traffic classification models. Random perturbation elucidates possible attack vectors in Neural Networks, an SVM, and a decision tree. Specifically, vulnerabilities in these models are found by analyzing the distribution of feature values for those data samples that are misclassified. These effective perturbation ranges are developed for features such as IP addresses and ports.

In addition, a novel perturbation method is tested in the augmented generative experiment. Not only does it showcase a new potential attack vector against linear classification models, but also should encourage further investigation of other perturbation methods where an adversary evades detection via black-box analysis. Similarly, experimentation with the GAN also showcases a black-box adversarial process, given a classifier can be built which is similar to the target.

Although much work is left to be done in the investigation of adversarial machine learning in the cyber domain, this study supports the general hypothesis that machine learning classifiers are not robust in unconstrained domains the same way they are not robust in constrained domains. Moving forward, more permutations of data sets, classification models, and perturbation methods should be tested. In addition, in order to better model the actions of potential adversaries, methods of constructing genuinely malicious network packets within the bounds specified here should be investigated.

5. REFERENCES

Alhajjar, E., Bastian, N., Maxwell, P. (2019). Generating adversarial examples using PSO and GS. *Preprint*.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. *arXiv:1905.02175*.

Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., Roli, F. (2018). Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. *ArXiv:1809.02861*.

Sharafaldin, I., Lashkari, A. and Ghorbani, A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, pages 108-116.

García, S., Grill, M., Stiborek, J. and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples, *stat 1050*.

Joseph, A. D., Nelson, B., Rubinstein, B. I. P. and Tygar, J. D. (2019). Adversarial Machine Learning. *Cambridge University Press*.