Poster: Adversarial Examples for Classifiers in High-Dimensional Network Data

Muhammad Ejaz Ahmed
Sungkyunkwan University, Suwon
Republic of Korea
ejaz629@skku.edu

Hyoungshick Kim
Sungkyunkwan University, Suwon
Republic of Korea
hyoung@skku.edu

ABSTRACT

Many machine learning methods make assumptions about the data, such as, data stationarity and data independence, for an efficient learning process that requires less data. However, these assumptions may give rise to vulnerabilities if violated by smart adversaries. In this paper, we propose a novel algorithm to craft the input samples by modifying a certain fraction of input features as small as in order to bypass the decision boundary of widely used binary classifiers using Support Vector Machine (SVM). We show that our algorithm can reliably produce adversarial samples which are misclassified with 98% success rate while modifying 22% of the input features on average. Our goal is to evaluate the robustness of classification algorithms for high demensional network data by intentionally performing `evasion` attacks with carefully designed adversarial examples. The proposed algorithm is evaluated using real network traffic datasets (CAIDA 2007 and CAIDA 2016).

KEYWORDS

Network attacks; Adversarial Machine Learning; Network Intrusion; Evasion Attacks; High Dimensional Data.

1 INTRODUCTION

Machine learning is now entrenching in modern technology, allowing an overwhelming number of tasks to be performed automatically at lower costs. Giant tech companies like Google, Microsoft, and Amazon have already taken initiatives to provide their customers with APIs, allowing them to easily apply artificial intelligence techniques into their applications. For instance, machine learning (ML) applications includes image classification to scrutinize online contents, automatic caption generation for an image, spam email detection, and many more [4].

However, there have been some limitations of ML algorithms due to which an adversary able to craft inputs would profit from evading detection. An adversarial sample is an input crafted such that it causes an ML algorithm to misclassify it [5]. Note that adversarial samples are created at test time, i.e., the ML algorithm is already trained and deployed by the defender, and do not require any alteration of the training process. Many ML methods require the data `stationarity` assumption where training and testing data are drawn from the same distribution (even when it is unknown). However, and attackers can often break this assumption by manipulating the input to the detector to get evasion because real-world data sources are not stationary. Moreover, another common assumption is that each datapoint is independent and identically distributed (i.i.d.); clearly this assumption can also be compromised by crafting the input with a little effort.

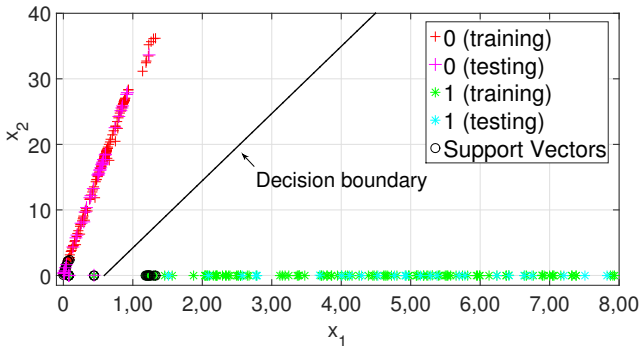

Figure 1: Network traffic features with classification (decision) boundary for a ML algorithm (SVM).

Since the real-world data sources do not follow the stationarity and i.i.d. assumptions, the ML models trained on such data sources could be vulnerable to be exploited by the attackers [1, 4, 6]. Fig. 1 shows real-world network traffic data (for two features x_1 and x_2) with the decision boundary of a binary classifier (SVM). Note that both features (x_1 and x_2) are linearly correlated for the class + (malicious traffic), where the correlation in the second feature (x_2) for class $*$ (legitimate traffic) is extremely low (almost zero).

Under the uneven separation of both classes from the decision boundary of the classifier, an attacker can evade the detection system by slightly modifying a malicious datapoint (even a single feature) at the lower end of the decision boundary. For example, if a malicious datapoint is modified at the lower axis of x_2 by the attacker (by updating $x_1 \geq 100$), it can evade the decision boundary and will be labeled as `legitimate` by the classifier.

In this paper, we propose a novel algorithm for adversarial sample creation against a popular ML algorithm, i.e., SVM. Furthermore, our approach alters a small fraction of input features leading to significantly reduced distortion of the source inputs. Our algorithm relies on heuristic searches to find minimum distortions leading to the misclassification for target inputs. We show that the input samples can be perturbed such that it is misclassified as a legitimate class with 98% success while perturbing 22% of the input features

CCS '17, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10.
https://doi.org/10.1145/3133956.3138853

on average. We performed extensive simulations on real-world network traffic with widely used 17 traffic features. In Section 2, we briefly describe the proposed approach for modifying the input features to bypass the classifier. Section 3 discusses the data used in our approach and the experiment results.

2 PROPOSED APPROACH

In this section we describe the proposed approach for identifying the features to be modified so that the classifier yields the adversarial output. We validate the proposed algorithm by using real traffic datasets.

2.1 Threat model

In our approach we assume that adversary can obtain query responses from the classifier. In case of intrusion detection system, the packets from malicious users are dropped by the intrusion detection system (IDS). Therefore, an adversary is aware of the dropped packets, thus features of these packets are recorded as training data. We further assume that adversary can observe legitimate packets by sniffing the network and therefore record the feature values of legitimate traffic [4]. In this way, dataset comprise of legitimate and malicious packets can be obtained by the adversary. Given a training set of m measurements consisting each of n features, our feature space becomes $\mathbf{X} \in \mathbb{R}^{m \times n}$. Note that our goal is to introduce minimum perturbation to a malicious datapoint so as to evade detection system.

2.2 Feature selection and modification

It has been shown that by using reduced set of features may require attackers to manipulate less features to evade the detection system [3]. The underlying idea of our approach is to select the features having most discriminative power for classification. Different methods in literature are proposed to obtain most distinguishing features such as information gains (IG), Chi Squared statistics, entropy, principal component analysis (PCA), mean and standard deviation of each dimension, skewness/kurtosis, cross correlation between dimensions, and ARIMA models, etc. In this paper, we restrict ourself to use IG to select the top K features having highest IG. After selecting-K top features, we take combination of the top K features to calculate two important metrics: distance and cross correlation.

In literature the distance between legitimate and malicious datapoints are used to quantify the differential between them [4]. However, in the proposed approach, in addition to calculate the distance (Euclidean) between each datapoint from legitimate and malicious traffic sources, we also compute the cross correlation $\gamma(x_1, x_2)$ between them. The Euclidean distance is the distance between two datapoints (one from each class, i.e., malicious and legitimate). Fig. 2 (left) show the Euclidean distance between each datapoint from legitimate and malicious datapoints for traffic features (total bytes received and total number of packets). Fig. 2 (right) show the the SVM classifier's decision boundary. Note that due to the non i.i.d. nature of real-world traffic sources, the adversary can craft (modify) datapoints (features) in the vulnerable area (shown in red-dashed circle) to misclassify the malicious datapoint as a legitimate. Also

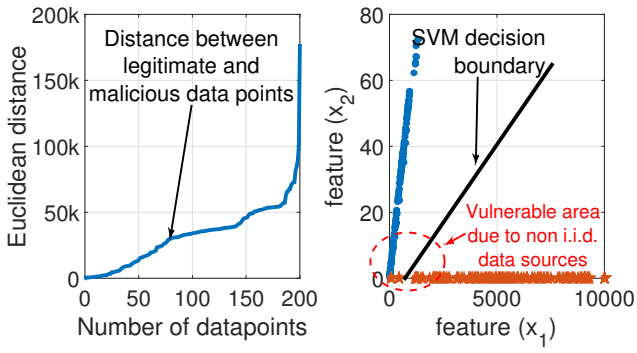

Figure 2: Euclidean distance between legitimate and malicious datapoints of Fig. 1, and classifier's uneven separation yields a vulnerable area which is exploited by the adversary.

note that if only one feature out of the two features is modified, the adversary can bypass the SVM's decision boundary.

Algorithm 1 Crafting adversarial sample.
$\{\mathbf{x}^i\}_{i=1}^m$ is the training data, each input feature vector has corresponding $\{y^i\}_{i=1}^m$ labels, y^* is the target network output (legitimate) class.

Input: $\mathcal{D} = \{\mathbf{x}^i, y^i\}_{i=1}^m, \mathbf{Y}^*$.
Result: $\mathbf{x}^*$ an adversarial datapoint.

1: K ← select top-K feature based on the IG($\{\mathbf{x}^i, y^i\}_{i=1}^m$)
2: $\mathbf{X}_{\text{mal}}$=$\{\mathbf{x}\}_{y=0}$ /* all malicious datapoints from the training data.*/
3: $\mathbf{X}_{\text{leg}}$=$\{\mathbf{x}\}_{y=1}$ /* all legitimate datapoints from the training data.*/
4: **for all** $c \in$ Comb(K) **do**
5: $(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})$ is a feature-pair relative to cth combination.
6: $E_d(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)}) \leftarrow$ Min(EuclideanDistance$\{\mathbf{x}^i_{\text{mal}}, \mathbf{x}^i_{\text{leg}}\}_{i=1}^{n/2}$)
7: $R_{\text{mal}}(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})$= Avg$\Big(xcorr\left(\mathbf{X}_{\text{mal}}(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})\right)\Big)$.
8: $R_{\text{leg}}(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})$= Avg$\Big(xcorr\left(\mathbf{X}_{\text{leg}}(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})\right)\Big)$.
9: ($\mathbf{x}_1$,$\mathbf{x}_2$) ← select the combination having maximum $R_{\text{mal}}(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})$ and $R_{\text{leg}}(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})$, and minimum $E_d(\mathbf{x}_{(c,1)}, \mathbf{x}_{(c,2)})$.
10: **while** not successfully misclassified **do**
11: $\mathbf{x} \leftarrow$ take a malicious datapoint having minimum $E_d(\mathbf{x}_1, \mathbf{x}_2)$ from $\mathbf{X}_{\text{mal}}$.
12: $\mathbf{x}^* \leftarrow$ replace the value of either feature ($\mathbf{x}_1$ or $\mathbf{x}_2$) with the $Avg(\mathbf{X}_{\text{leg}})$.

Algorithm 1 enumerates the steps involved in modifying a malicious datapoint to bypass the detection system. In the first step, information gain (IG) for every feature is calculated and top-K features are selected. Malicious and legitimate traffic are represented by $\mathbf{X}_{\text{mal}}$ and $\mathbf{X}_{\text{leg}}$, respectively, (line 2-3). The combinations of the selected top-K features are computed, and for each feature-pair combination, the Euclidean distance between datapoints from malicious and legitimate traffic is computed, and a minimum of them

Table 1: Input features for a ML detector.

Feature	Description
#pkts	number of packets
#bytes	number of bytes
pkts (src -> dst)	packets from source to destination
bytes (src -> dst)	bytes from sources to destination
pkts (dst -> src)	packets from destination to source
bytes (dst -> src)	bytes from destination to source
Rel. Start	Relative start time of a connection
dur.	duration of a connection
bits (src -> dst)	bits from source to destination
bits (dst -> src)	bits from destination to source
IP-H-len	IP header length
pkt-len	packet length
ttl-avg	average time to live (TTL)
ttl-std	TTL standard deviation
inter-arr	interarrival time
proto-type	protocol type

is taken (line 6). We do so because we want to figure-out which features could minimally be perturbed to evade the detector. Then, cross-correlation is calculated between the selected features of malicious and legitimate traffic (line 7-8). The reason for this is to find features which could be exploited based on the i.i.d. assumption made by most ML classifiers. For example feature-pairs having higher cross-correlation will bias the decision boundary of SVM as shown in Fig. 2 (right), consequently an adversary can exploit lower regions of the feature-pairs to bypass the classifier's decision boundary. In the next step, from all the feature-pair combinations with respective minimum Euclidean distance and cross-correlations, we select feature-pairs having minimum Euclidean distance and maximum cross-correlations (line 9). It is to be noted that we used heuristics to obtain the maximum (cross-correlation) and minimum (Euclidean distance) since the problem under consideration is non-convex multi-variable optimization problem. The selected features (from line 9) are modified by our proposed approach so as to misclassify them. For that, a malicious datapoint having minimum E_d is selected for modification, and either feature (as obtained from line 9) is modified and replaced with the average of the same feature of legitimate data traffic. The modified feature is then tested if it is evades the detector.

3 DATA DESCRIPTION AND RESULTS

We evaluate the performance of the proposed approach by using real network traffic measurements from CAIDA2007 and CAIDA2016 traces. We extracted 16 features from those network traces. The features used are given in Table 1. CAIDA2007 dataset is five minutes, i.e., 300 s of anonymized traffic obtained during a DDoS attack in August 2007. These traffic traces recorded only attack traffic to the victim and response from the victim, whereas the legitimate traffic has been removed as much as possible. The CAIDA2016 dataset [2] consists of anonymized passive traffic measurements from passive monitors in 2016. It contains traffic traces from the 'equinix-chicago' high-speed monitor.

Table 2: Results of the proposed approach.

Dataset	Successfully misclassified	Features modified
Training	98.5%	20%
Validation	98.1%	25%
Test	97.3%	22%

In our experiments, we randomly extracted malicious and legitimate traffic from each dataset. The labels are provided for both the classes (legitimate and malicious). We trained the SVM classifier with providing 200 datapoints from each class. For the top-K, we set $K = 5$.

After the classifier (SVM) is trained, we carryout the evasion attack. Note that we do not consider the whole training data whereas we take a subset (50 datapoints from each class) of training and testing data from the whole dataset and craft them according to Algorithm 1. Then each modified datapoint is input to the trained SVM classifier to check if the datapoint is misclassified. The results are shown in Table 2. The success rate is defined as the percentage of adversarial datapoints that were successfully misclassified as `legitimate` where in actual they were `malicious`. The feature modification percentage is defined as the number of features that were modified out of the total number of input features. The proposed approach is aware of what features are needed to be modified at which location to evade the detection system.

4 CONCLUSIONS

We introduced an algorithm to modify a malicious input so that it cannot be undetected by a classifier using SVM. To achieve this goal, we exploited the non i.i.d. assumption of SVM classification which can lead to a biased decision boundary. By exploiting a certain region around the decision boundary and crafting the adversarial sample accordingly, we show that the adversary can successfully bypass the detector with the success rate of 98% by modifying 22% of input features on average. For future work, we will extend the proposed idea to study other machine learning algorithms employed in identifying malicious traffic.

ACKNOWLEDGMENTS

The work was supported in parts by National Research Foundation of Korea (No. 2017R1D1A1B03032966), IITP (No. B0717-16-0116), and ITRC (IITP-2017-2012-0-00646).

REFERENCES

[1] F. Giorgio B. Biggio and R. Fabio. 2013. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering* 26, 4 (2013), 984–996.
[2] CAIDA dataset. 2016. https://www.caida.org/data/. (2016).
[3] B. Biggio D. S. Yeung F. Zhang, P. P. Chan and F. Roli. 2016. Adversarial feature selection against evasion attacks. *IEEE Transactions on Cybernetics* 46, 3 (2016), 766–777.
[4] H. Zheng G. Wang, T. Wang and B. Y. Zhao. 2014. Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers. *USENIX Security Symposium* (2014), 239–254.
[5] S. Jha M. Fredrikson Z. B. Celik N. Papernot, P. McDaniel and A. Swami. 2016. The limitations of deep learning in adversarial settings. *IEEE European Symposium on Security and Privacy (Euro S&P)* (2016), 372–387.
[6] Y. Qi W. Xu and D. Evans. 2016. Automatically evading classifiers. *In Proceedings of the Network and Distributed Systems Symposium (NDSS)* (2016), 1–15.