SECURING MACHINE LEARNING CLASSIFIERS WITH INPUT HASHING RE-WEIGHT STRATEGY

IGOR FRANC
Belgrade Metropolitan University, Faculty of Information Technologies and SECIT Security Consulting, igor.franc@metropolitan.ac.rs

NEMANJA MAČEK
School of Electrical and Computer Engineering of Applied Studies, Belgrade and eSigurnost Association, Belgrade, macek.nemanja@gmail.com

MILAN GNJATOVIĆ
University of Novi Sad, Faculty of Technical Sciences, milangnjatovic@uns.ac.rs

BRANIMIR TRENKIĆ
School of Electrical and Computer Engineering of Applied Studies, btrenkic@viser.edu.rs

MITKO BOGDANOSKI
Military Academy General Mihailo Apostolski, Skoplje, Macedonia, mitko.bogdanoski@ugd.edu.mk

DRAGAN ĐOKIĆ
Belgrade Metropolitan University, Faculty of Information Technologies, dragan.djokic@metropolitan.ac.rs

Abstract: *Adversarial machine learning resides at the intersection of machine learning and computer security. Originally, machine learning techniques were designed for environments that do not assume the presence of an adversary. However, in the presence of intelligent adversaries, this working hypothesis is likely to be violated to at least to some degree, depending on the skillset of an adversary. A skilful adversary can carefully manipulate the input data exploiting specific vulnerabilities of learning algorithms. This results in misclassification of malicious instances, which may compromise the whole system security. For example, by carefully modifying values of features with largest weight without changing the outcome of malicious packet, an adversary may trick an intrusion detection system to allow malicious packet into the network. Solutions presented in research studies by other authors consider the classifier protection using re-weight strategies; typically, this results in compromise between accuracy and robustness. Unlike those, the research presented in this paper deals with a re-weight strategy based on hashing all the numeric features without classification accuracy degradation. System becomes robust as feature weights are even and avalanche effect makes virtually impossible for an attacker to modify the input data and trick the learner into misclassification. Research hypotheses are experimentally validated on custom intrusion detection dataset consisting of numeric features.*

Keywords: *Machine Learning, Adversarial Learning, Hashing*

1. INTRODUCTION

Machine learning algorithms independently collect knowledge from the machine readable information, i.e. they learn from data. Such algorithms build a model from example inputs and use it to make predictions or decisions [1]. Tom Mitchell provided a widely quoted, formal definition of machine learning: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*" [2].

There are two types of machine learning algorithms: unsupervised (no "teachers") and supervised (with "teachers"). Unsupervised algorithms learn from unlabeled examples; the objective of unsupervised learning may be to cluster examples together on the basis of their similarity [3]. Unsupervised learning is suitable for finding patterns in the data. Supervised learning algorithms build a model from a training set (given in the form of feature vectors) with class label assigned to each instance. Once trained, supervised algorithms assign class labels to previously unseen examples of the same task, on the basis of the formed model [4].

In the case of machine learning, most classification algorithms were developed to learn and operate in secure, controlled environment. Transition from controlled environment to a potentially hostile environment may result in significant security failures [5].

In adversarial classification tasks like spam filtering and intrusion detection, a skilful malicious adversary can carefully manipulate the input data exploiting specific

vulnerabilities of learning algorithms. Thus, aside from achieving good classification performances, machine learning algorithms have to be robust against adversarial data manipulation [6].

2. ADVERSARIAL LEARNIG: ATTACKS ON CLASSIFIERS AND COUNTERMEASURES

Potential vulnerabilities of the learning system in adversarial environment can be categorized according to the influence on the classifier, security policy violation and specificity of the attack [7].

According to the influence on the classifier, attacks can be either causative, if they are aimed at compromising the training phase of the classifier, or exploratory, if they are carried out at the classification phase, with the aim to gather knowledge about the classifier (the knowledge can, for example, be gathered from feedback on class labels assigned to malicious traffic instances.) According to the security policy violation, attacks can be either integrity violation, if the adversary's goal is to have malicious instances classified as legitimate activity by the system, causing false negative rate to increase, or availability violation, if the adversary's goal is to perform DoS attacks and render the classifier useless. Specificity of the attack refers to the scope of malicious samples that will be misclassified as legitimate activities. According to the specificity, attacks can be either targeted, if the adversary's goal is to have a limited range of malicious instances misclassified, or indiscriminate, if the adversary's goal is to have all malicious samples misclassified as legitimate.

When designing a machine learning based security mechanism, it is necessary to identify system vulnerabilities, the possibilities to execute attacks on a system that will exploit these vulnerabilities, and the consequences of successfully executed attacks. In other words, system architect should take the role of adversary and try to anticipate all possible attacks, such as compromising the training set or detection evasion.

Several defence techniques against attacks on the classifier are proposed in [7]: regularization (defence from causative attacks), randomization (defence from targeted attacks) and information hiding or disinformation (defence from exploratory attacks). In some circumstances, the learning system may alter the information seen by the adversary, thus providing the adversary with a misleading picture of the classifier. More sophisticated systems can lead the adversary to believe that a certain type of attack is not included in the training set. Allegedly “allowed attack” will lead the adversary to reveal himself.

There are several ways to detect attacks on the classifier. For example, exploratory attacks can be identified by running a separate clustering algorithm against the classified data: the sudden appearance of a large cluster near the decision boundary could indicate probing attacks on machine learning based intrusion detection system. Detecting attacks on the classifier is very important because it provides information about adversary's capabilities; this information can be used to re-adapt defence strategies.

3. RE-WEIGHT STRATEGIES

One of the re-weight strategies that improves the robustness of linear classifiers in spam filters is proposed in [8]. The technique is based on the normalization of feature weights, thus avoiding over-emphasizing or under-emphasizing feature weights. If there are features with over-emphasized weights, the adversary will adapt the values of most significant features (features with the highest impact on classifier decision) and trick the learner to classify malicious as legitimate instance. These attacks are typically executed on spam filters (for example, by increasing the number of innocent words), but are also feasible to evade detection by intrusion detection classifier.

If the distribution of feature weights is uniform, an adversary will have to change more feature values to trick the classifier. If adaptation of each feature requires the same effort, then this re-weight strategy increases the robustness of the classifier, i.e. its resistance to attacks based on exploiting knowledge about the decision function. However, the proposed technique is a compromise between accuracy and robustness of the classification system.

Re-weight strategy presented in this paper is based on hashing all the numeric features, both in the training and operational phase. By doing so, system becomes robust resulting feature weights become even and avalanche effect resulting from hash function is a huge troublemaker even for a skilled adversary. Aside, if generated hash is bitwise longer than the original input value, input data is virtually casted into higher-dimensional space (similar to Support Vector Machines’ kernel trick.) According to experimental evaluation given in this paper, we can conclude that this re-weight strategy is applicable to several machine learning based security mechanisms, including, but not limited to intrusion detection systems. The only downside of the proposed solution is that it operates only with numeric features, thus not being applicable to systems that operate with categorical features without additional feature vector transformation.

4. APPLICATION TO CUSTOM INTRUSION DETECTION DATASET

The intrusion detection dataset used in this research is built from traffic captured on the simulated virtualized networking environment. Synthetic dataset consists of normal, healthy traffic and a number of successful exploitations of unpatched Windows XP operating system, executed with variety of open source and commercial software products. Both healthy and malicious traffic have been recorded separately and cleansed from virtualization protocol and service leftovers (noise removal). This reassembles a scenario for two-class supervised learning problem. Numerical features (representing statistical data) were extracted from PCAP files and data instances were created, labelled and shuffled into a separate training and test sets, both consisting of 10.000 instances.

Following machine learning algorithms were used to train models and classify test sets: decision trees [9, 10], Naive Bayes [11, 12], Random Forest [13] and AdaBoost [14] using C4.5 decision tree as the base learner. See references

[9-14] for more details on aforementioned algorithms. Hashed values were calculated using MD5 and Secure Hash Algorithm (SHA-2).

Let TP, TN, FP and FN denote number of true positives, true negatives, false positives and false negatives [15]. Accuracy of the classifier is calculated with the following equation:

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

A simple algorithm used to calculate feature weights operates follows [15]: let a denote the accuracy of classifier trained with all features, and let a_i denote the accuracy of a classifier trained with all features except feature i. Accuracy change for that classifier is given with the expression:

$$\Delta a_i = a - a_i. \quad (2)$$

The smallest and the largest accuracy changes are given with expressions (3) and (4):

$$\Delta a_{min} = min(\Delta a_i), i = 1, \ldots n \quad (3)$$

$$\Delta a_{max} = max(\Delta a_i), i = 1, \ldots n, \quad (4)$$

where n denotes the number of features in the dataset. Feature weight w_i of the feature i is given with the equation:

$$w_i = \frac{\Delta a_i - \Delta a_{min}}{\Delta a_{max} - \Delta a_{min}}. \quad (5)$$

All feature weights are scaled to a range [0, 1]. Classifier used to calculate weights is C4.5 decision tree.

Experiments with original input data

Models have been trained and tested using Python with scikit-learn. The first set of models have been trained with the original datasets (feature values have not been hashed). Test set classification accuracy is given in the table 1.

Table 1: Classification accuracy (original input data)

Algorithm	Accuracy (%)
C4.5	81,43%
Naive Bayes	85,68%
Random Forest	95,19%
AdaBoost	93,87%

Feature weights of 27 features are presented on graph on Image 1. As one may see, the distribution of feature weights is not uniform, as there are features with over-emphasizing or under-emphasizing weights. In this scenario, the adversary familiar with the classifier and statistical features of data instances will adapt the values of most significant features and trick the learner to classify malicious as legitimate instance.

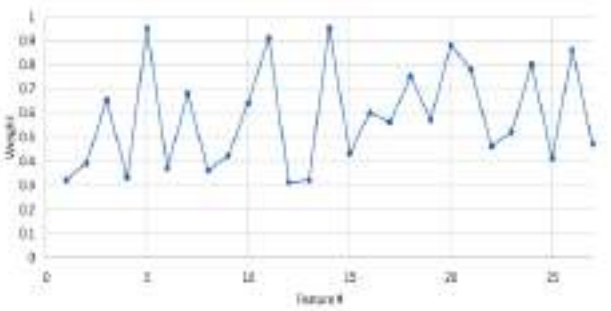

Image 1: Feature weights scaled to range [0, 1], original input data

Experiments with hashed input data

The second set of models have been trained with two datasets containing values from original dataset processed with MD5 and SHA-2 hash functions. Feature values in the test sets have been also processed with the aforementioned hash functions. Test set classification accuracy is given in the table 2. According to the result, one may notice that the degradation of accuracy is almost negligible.

Table 2: Classification accuracy (hashed input data)

	Accuracy (%)	
Algorithm	MD5	SHA-2
C4.5	79,95%	79,89%
Naive Bayes	84,52%	85,01%
Random Forest	94,07%	93,87%
AdaBoost	91,77%	90,82%

Feature weights of 27 features are presented on graph on Image 2. As one may see, the distribution of feature weights is now uniform, as there are no features with over-emphasizing or under-emphasizing weights. Due to uniform distribution is uniform, an adversary will have to change more feature values to trick the classifier. As the classifier operates with hash values it may be concluded that (1) adaptation of each feature requires the same effort and that (2) due to the avalanche effect, minor adaptations in original data before hashing in the IDS will result in major change in the resulting data, making it virtually impossible for a skilled attacker to trick the classifier.

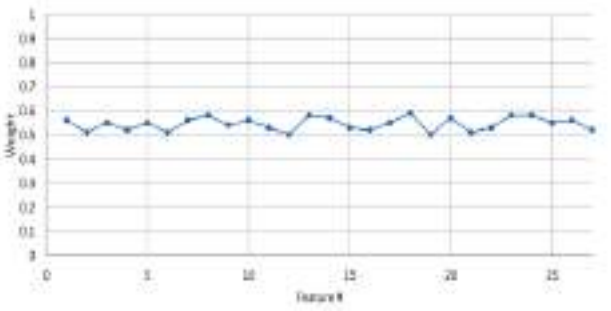

Image 2: Feature weights scaled to range [0, 1], hashed input data

5. CONCLUSION

In this paper we have presented an approach to feature re-weight strategy based on hashing all the numeric feature values. Classifier is trained and tested with all feature values of each instance being previously processed by a hash function. Hashing provides a robust classifier as resulting feature weight distribution is uniform and avalanche effect resulting from hash function is an obstacle even for a skilled adversary. Re-weight strategy presented in this paper is applicable to several machine learning based security mechanisms. However, the downside of the proposed solution is that it operates only with numeric features, which means that additional feature vector transformation is required if categorical features exist.

REFERENCES

[1] M. A. Hall and L. A. Smith, "Practical feature subset selection for machine learning," in C. McDonald (Ed.), Computer Science '98 Proceedings of the 21st Australasian Computer Science Conference ACSC'98, Perth, 4-6 February, 1998, pp. 181-191. Berlin: Springer.

[2] T, Mitchell, "Machine Learning", McGraw-Hill Science/Engineering/Math, page 2, 1997.

[3] Z. Ghahramani, "Unsupervised Learning," in O. Bousquet et al. (Eds.): "Machine Learning", LNAI 3176, Springer-Verlag Berlin Heidelberg, 2003.

[4] I. Hendrickx, "Local Classification and Global Estimation: Explorations of the k-nearest neighbor algorithm", PhD Thesis, Tilburg University, 2005.

[5] T. Woods, M. Evans, D. Rust, and B. Podoll, "Security in Machine Learning: Measuring the relative sensitivity of classifiers to adversary-selected training data," CSCI 5271 Project Final Draft, University of Minnesota, Minneapolis, USA, 2008.

[6] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in Asian Conference on Machine Learning, pp. 97-112, November, 2011,

[7] M. Barreno, B. Nelson. R. Sears, A. D. Joseph and J. D. Tygar, "Can machine learning be secure?", in Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pp. 16-25), 2006.

[8] A. Kołcz, and C. H. Teo, "Feature weighting for improved classifier robustness", in CEAS'09: sixth conference on email and anti-spam, 2009, no pagination.

[9] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, "Classification and Regresssion Trees", Wadsworth, Belmont, 1984.

[10] R. Quinlan, "C4.5: Programs for machine learning", Morgan Kaufmann Publishers, Inc., 1993.

[11] V. Cherkassky and F. M. Mulier, "Learning from Data: Concepts, Theory and Methods. 2nd ed.", John Wiley - IEEE Press, 2007.

[12] I. H. Witten, E. Frank and M. A. Hall, "Data Mining: Practical machine Learning Tools and Techniques, 3rdEd", Elsevier Inc., 2011.

[13] L. Breiman, "Random Forests", Machine learning, 45(1), pp. 5-32, 2001.

[14] [21] B. Kégl, "The return of AdaBoost.MH: multi-class Hamming trees", arXiv: 1312.6086, Dec. 20. Last time visited: Aug 15, 2016..

[15] N. Maček, B. Đorđević, V. Timčenko, M. Bojović, M. Milosavljević, "Improving Intrusion Detection with Adaptive Support Vector Machines", Elektronika ir elektrotechnika, Vol. 20, No. 7, pp. 57-60, 2014.