Efficient training data extraction framework for intrusion detection systems

Abdelhamid Makiou, Ahmed Serhrouchni

Efficient Training Data Extraction Framework for Intrusion Detection Systems

Abdelhamid MAKIOU
Ahmed SERHROUCHNI

Telecom Paristech 46, Rue Barrault 75013 Paris France
INFRES Department
E-mails: makiou@telecom-paristech.fr, ahmed@telecom-paristech.fr

***Abstract*—One of the most important challenges in networks of future is how to provide security services in an efficient and quick manner without questioning the conceptual model of these networks. NIDS (Network Intrusion Detection system) is an effective approach for dealing with these security issues. Their detection engines use two main approaches to inspect malicious traffic; misuse detection and anomaly detection. Machine learning approaches have been widely investigated in anomaly detection systems. However, despite multiple academic researches, such systems are rarely employed in operational settings. They suffer from long time training, lack of good quality training data and a random selection of features (training attributes). This paper aims to resolve issues of limited quantity and quality of training data sets and investigates the impact of features selection process on the classification performance. It introduces a novel framework to extract efficient real *"ground-truth"* training data sets from the network traffic for supervised machine learning algorithms. We support this claim by performing the evaluation of results on a Naive Bayesian classifier.**

***Index Terms*—Attack Detection, machine learning, Security rules, Pattern-Matching**

I. Introduction

Each network carries data for numerous different kinds of applications, which should be inspected by security settings in order to detect malicious contents. Most of the current methods for intrusion detection are security rules based systems (misuse detection), which use pattern matching algorithms to inspect network traffic. In spite of the robustness of these systems, attackers can bypass them by substituting malicious pattern characters or hiding them. However, a basic solution is to write a specific rule for each type of evasion technique, but it requires a high mastery of both protocol semantics and the regular expressions programming. Furthermore, pattern matching algorithms, used by security rules in order to deeply inspect complex patterns, decrease the overall performance of detection systems. To overcome the drawbacks of these methods, machine learning algorithms are an alternative to provide a fast statistical and probabilistic detection of attacks. However, actual operational settings deploy rarely machine learning based modules for anomaly detection. The reasons are 1) the long time training process, 2) the lack of quantity and good quality of training data 3) and a random selection of features (training attributes). As regards the quantity of the training data sets, often in practical applications, the number of available ground-truth samples is not sufficient to achieve a reliable estimate of the classifier parameters in the learning phase of the algorithm. In particular, if the number of training samples is relatively small compared to the number of features (and thus of classifier parameters to be estimated) [1], the well-known problem of the curse of dimensionality (i.e., the Hughes phenomenon [2][3]) arises. This results in the risk of over-fitting the training data and may involve poor generalization capabilities in the classifier. The limited quantity and quality of the training samples involve the definition of ill-posed classification problems [1], [4]. This leads to a very high cost of errors (false negatives and false positives) and misinterpretation of operational results. In addition, a lot of academic researches suffer from the lack of "ground-truth" training data to perform the evaluation of their proposals. The most of them used an old data set originated from the 1998 DARPA Intrusion Detection Program, which is adopted in the Data-Mining and Knowledge Discovery (KDD) competition [5][6]. and investigate the impact of features selection process on the classification accuracy artifacts [8][7]
In this paper, we introduce a novel method to construct training data sets for intrusion detection systems based on Machine Learning Classifiers. We developed a collaborative framework between a misuse detection engine and a machine learning classifier.

II. Related Work

In [9], Kruegel and Vigna propose an anomaly-based intrusion detection system for web applications. It characterizes HTTP requests using a number of statistical characteristics derived from parameter's length, character distribution, structure, presence and order. This method focuses only on the incoming query parameters whereas it ignores the corresponding HTTP response. These results are either causing unnecessary false positives or missing certain attacks. Zhang and Zulkernine [10] employed the Random Forests technique in NIDSs (Network Intrusion Detection Systems) to improve the detection performance. To increase the detection rate of the minority intrusions, they built the balanced data set by over-sampling the minority classes and down-sampling the majority classes. Random Forests can build patterns more efficiently over the balanced data set, which is much smaller than the

original one. The experiments showed that the approach can reduce the time to build patterns and increase the detection rate of the minority intrusions. The results showed that the proposed approach provides better performance compared to the best results from the KDD99 contest [11]. An intrusion detection system based on neural networks has been presented in [12][13]. While it exhibits good detection with low false alarm rates when tested with the DARPA 1999 IDS Evaluation data, using multilayer perceptrons requires relatively more processing time for intrusion detection.

III. The Machine Learning Scheme

In this paper, we consider only supervised machine learning technique for our study. We also use a Naive Bayesian algorithm as a classifier for filtering malicious contents. Indeed, the Naive Bayes classifier provides a simple approach, with clear semantics, for representing, using and learning probabilistic knowledge[11]. The goal is to accurately predict the class of test instances where the training instances include the class information. This classifier can be seen as a specialized form of the Bayesian network, termed Naive because it relies on two important simplifying assumptions. It assumes that the predictive attributes are conditionally independent and no hidden or latent attributes influence the prediction process. According to the Bayes theorem and the total probabilities theorem [14], for a vector of attributes $\overrightarrow{x} = (x_1, \ldots, x_n)$, the probability to belong to the class c is defined as follows:

$$p(C=c|\overrightarrow{X}=\overrightarrow{x})=\frac{p(C=c)p(\overrightarrow{X}=\overrightarrow{x}/C=c)}{p(\overrightarrow{X}=\overrightarrow{x})} \quad (1)$$

Using the theorem of the total probabilities, we deduce:

$$p(C=c|\overrightarrow{X}=\overrightarrow{x})=\frac{p(C=c)p(\overrightarrow{X}=\overrightarrow{x}/C=c)}{\sum_{c\epsilon\{attack,Leg\}} p(C=c)\,p(\overrightarrow{X}=\overrightarrow{x}/C=c)} \quad (2)$$

A. Classification errors evaluation method

A false negative is mistakenly classifying an attack as a legitimate content, and a false positive is a legitimate content mistakenly classified as an attack. In our model, the cost of a false negative is much higher than the cost of a false positive. Indeed, wasting time in analyzing traffic requests is more acceptable than authorizing a malicious content. The two error types are defined as such follows:

- Classifying an attack as legitimate content: ($Attack \rightarrow Leg$)
- Classifying a legitimate content as an attacks: ($Leg \rightarrow Attack$)

In our classifier model, the first error is more serious than the second one. To illustrate it, we introduce the parameter λ, to give more importance to the first error by assuming that: $Attack \rightarrow Leg$ is λ times more costly than $Leg \rightarrow Attack$.

B. Classification criteria

According to the above two error types, the selection criteria is as follows:
The content of a vector (content) $\vec{x}$ is classified as legitimate if and only if:

$$p(C = Leg|\vec{X} = \vec{x}) > \lambda.p(C = Attack|\vec{X} = \vec{x}) \quad (3)$$

given that $p(C = Leg|\vec{X} = \vec{x}) + p(C = Attack|\vec{X} = \vec{x}) = 1$, the selection criteria becomes as follows:

$$p(C = Leg|\vec{X} = \vec{x}) > \alpha \quad (4)$$

Where :

$$\alpha = \frac{\lambda}{1+\lambda}, \lambda = \frac{\alpha}{1-\alpha} \quad (5)$$

C. Method and parameters evaluation

In this section, we define the parameters that allow us to evaluate our filter. To this end, two evaluation parameters are used: accuracy (Acc) and error ($Err = 1 - Acc$) [15]. They are defined as follows:

$$\begin{aligned} Acc &= \frac{n_{Attack\rightarrow Attack} + n_{leg\rightarrow leg}}{N_{Attack} + N_{leg}} \\ Err &= \frac{n_{Attack\rightarrow leg} + n_{Attack\rightarrow Attack}}{N_{Attack} + N_{leg}} \end{aligned} \quad (6)$$

where:

- $N_{Attack} = n_{Attack\rightarrow leg} + n_{Attack\rightarrow Attack}$
- $N_{leg} = n_{leg\rightarrow leg} + n_{leg\rightarrow Attack}$
- $n_{y\rightarrow z}$ denotes the number of patterns of class y that are mistakenly classified as patterns of class z.

Parameters defined above, do not support the notion of weight (on the two error types) introduced in the previous paragraph. This leads us to introduce the weighted accuracy ($Wacc$) and weighted error ($Werr = 1 - Wacc$). We assumed that $Attack \rightarrow Leg$ is λ times more devastating for our system than $Leg \rightarrow Attack$. To make accuracy and error rate sensitive to this cost, we should treat each attack as if it is λ inputs; when an attack is misclassified, then it will be counted as λ errors. While we will have λ successes, when it is classified correctly

$$\begin{aligned} Wacc &= \frac{\lambda n_{Attack\rightarrow Attack} + n_{leg\rightarrow leg}}{\lambda N_{Attack} + N_{leg}} \\ Werr &= \frac{\lambda n_{Attack\rightarrow leg} + n_{leg\rightarrow Attack}}{\lambda N_{Attack} + N_{leg}} \end{aligned} \quad (7)$$

To get a precise idea on the classification performance, we compare it with a non-filtered environment in which all network traffic is considered as legitimate.

For this purpose, we introduce the definition of the base-line weighted error and the base-line weighted accuracy (denoted by $Wacc^b$ and $Werr^b$ respectively) which are defined as follows:

$$Wacc^b = \frac{\lambda N_{leg}}{\lambda N_{Attack} + N_{leg}} \quad (8)$$
$$Werr^b = \frac{N_{Attack}}{\lambda N_{Attack} + N_{leg}}$$

The TCR (Total Cost Ratio) value measures the performance of a machine learning classifier to the same environment without a classifier. In the case where the TCR value is negligible, the best approach is to not use a classifier and to send all traffic to the rules based detection engine. An effective filter which could be used in real environments should have a TCR value higher than 1.

The TCR formula is defined as follows:

$$TCR = \frac{Werr^b}{Werr} = \frac{N_{Attack}}{\lambda n_{leg \to Attack} + n_{Attack \to leg}} \quad (9)$$

IV. The Training Data Extraction Framework

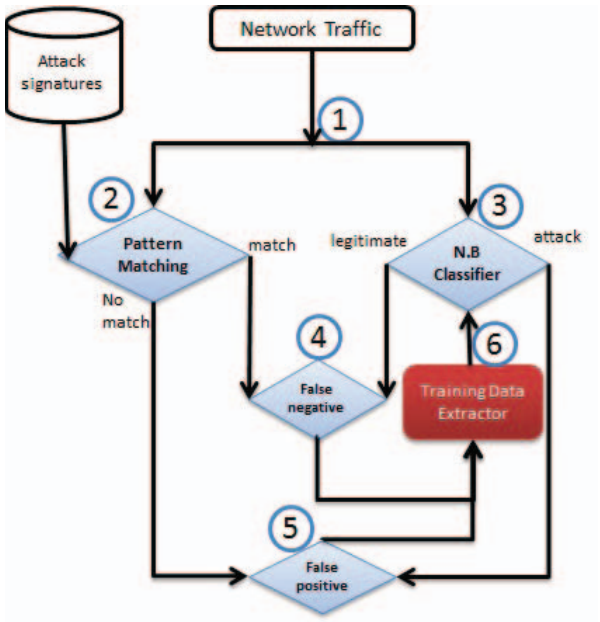

Fig. 1: The Training Framework

In [18] we developed a framework that collect training data from a log file of a web server. We used these training data sets as initial training data for our classifier.

Phase 1: The network packets are forwarded to both the pattern matching engine and the Bayesian classifier (a copy for each one).

Phase 2: The pattern matching engine analyzes packets and decides whether the contents match or not with an attack signature.

Phase 3: The classifier receives normalized data (features vectors) and applies the classification algorithm (with initial training data set). It decides if the vector belongs to a legitimate traffic or not.

Phase 4: The classifier has classified data as legitimate, in the same time the pattern matching engine has classified it as an attack. In this case, the decision of the classifier is considered as a false negative. Indeed, in this learning phase, the pattern matching engine result takes precedence over that of the classifier.

Phase 5: The classifier has classified data as an attack. On the other hand, the pattern matching engine has not filtered it (legitimate) . The result of the classifier is considered as a false positive. However, if the classification probability is much higher than the classification threshold (α previously-introduced). In this case the content is considered to be a true positive. Thus, we need to tune the α threshold in order to improve the quality of generated training data sets. In section VI, we performed a results analysis by varying the value of the threshold α from 50% to 99% and evaluate the impact of this parameter on the performance of our classifier.

Phase 6: In this phase, the training data extractor module receives classification errors (false positives and false negatives) from both the pattern matching detection engine and the Bayesian classifier. The module will generate a new training data vector for each error, thus, will improve the accuracy of the Bayesian classifier during the training process.

V. Training attributes Selection

Even within a single network, the network's most basic characteristics such as bandwidth, duration of connections, and payloads can exhibit immense variability, rendering them unpredictable over short time intervals (seconds to hours). Traffic diversity is not restricted to packet-level features, but extends to application-layer, information as well, both in terms of syntactic and semantic variability. Syntactically, protocol specifications often purposefully leave room for interpretation, and in heterogeneous traffic streams. Semantically, features derived from application protocols can be just as fluctuating as network-layer packets [7][16][17].

For these reasons, the choice of classification attributes (features vectors) plays an important role in the performance of a classifier. However, finding attributes that represent all potential malicious activities on all networks is simply mission impossible. Mutual Information (M.I.) an IF-TDF are two methods used in the process of features selection. We used in [18] M.I. to extract features from SQL injection attacks requests to build the training vectors, and we obtain good results. As conclusion, the best way to improve the accuracy of a intrusion detection system is to build for each class of attacks a specific classifier. In [19] the authors proposed a multi-classifier architecture that can be considered as a model for our proposal, but this is out of the scope of this paper.

VI. Results Analysis

We collected training data sets from the previous framework and used Weka program to simulate a stand-alone Naive Bayesian classifier for SQL injection attacks. Then results are

compared with our work [18] we have dealt with raw training data sets collected form logs of a web server.

Data (%) SQLi-Leg	λ Num	α (%)	False Pos.(%)	False Neg.(%)	TCR Value
80-20	1	50	4.6	0.6	15.38
66-34	1	50	2.0	0.3	28.57
50-50	1	50	3.0	0.5	14.29
80-20	2	66.67	6.0	0.5	8.16
66-34	2	66.67	2.0	0.3	15.38
50-50	2	66.67	3.0	0.5	7.69
80-20	5	83.33	5.6	0.4	3.39
66-34	5	83.33	2.9	0.3	6.45
50-50	5	83.33	4.0	0.6	3.23
80-20	9	90	5.8	0.6	1.90
66-34	9	90	4.0	0.3	3.54
50-50	9	90	5.0	0.5	1.82
80-20	99	99	6.6	0.4	0.18
66-34	99	99	4.0	0.3	0.34
50-50	99	99	5.0	0.4	0.17

TABLE I: Costs Table (Raw Training Data sets)

Data (%) SQLi-Leg	λ Num	α (%)	False Pos.(%)	False Neg.(%)	TCR Value
80-20	1	50	0.5	0.1	30.11
66-34	1	50	1.2	0.1	20.19
50-50	1	50	2.5	0.4	14.29
80-20	2	66.67	5.3	0.3	10.66
66-34	2	66.67	1.0	0.2	15.44
50-50	2	66.67	4.0	0.3	9.99
80-20	5	83.33	3.3	0.2	8.21
66-34	5	83.33	3.3	0.1	10.13
50-50	5	83.33	2.2	0.2	11.20
80-20	9	90	3.8	0.3	5.30
66-34	9	90	3.5	0.2	6.21
50-50	9	90	4.2	0.3	4.50
80-20	99	99	2.2	0.2	3.21
66-34	99	99	2.2	0.2	3.21
50-50	99	99	2.2	0.2	3.21

TABLE II: Costs Table (New Training Data sets)

In the experiment, we begin with evaluating the performance of the N.B. classifier trained with raw training data sets. We got best values within 0.3% of false negatives and 2% of false positives. The results of our new training data sets applied on the same classifier under the same conditions are better. We obtained less than 0.1% of false negatives and 0.5% of false positives.

VII. Conclusion an Future Works

The main purpose of our work is to present a novel method for collecting real ground-truth training data for intrusion detection system. The lack of academic research in this field and experimental results has led us to perform the evaluation of results comparing with our main previous work on SQL injection attacks and evasion techniques. Nevertheless, we have shown the effectiveness of our approach in one class of Web applications Attacks (SQL injections). The next step of this work would be to study the impact of *real-world* network traffic parameters (such as session duration, connection's origin, payload type,...) on the quality of training data for each context of detection.

References

[1] A. Baraldi , L. Bruzzone and P. Blonda "Quality assessment of classification and cluster maps without ground truth knowledge", IEEE Trans. Geosci. Remote Sens., vol. 43, no. 4, pp.857 -873 2005

[2] G. F. Hughes "On the mean accuracy of statistical pattern recognition", IEEE Trans. Inf. Theory, vol. IT-14, no. 1, pp.55 -63 1968

[3] Bruzzone, L.; Mingmin Chi; Marconcini, M., "A Novel Transductive SVM for Semisupervised Classification of Remote-Sensing Images," Geoscience and Remote Sensing, IEEE Transactions on , vol.44, no.11, pp.3363,3373, Nov. 2006

[4] M. Chi and L. Bruzzone "A semilabeled-sample-driven bagging technique for ill-posed classification problems", IEEE Geosci. Remote Sens. Lett., vol. 2, no. 1, pp.69 -73 2005

[5] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-line Intrusion Detection Evaluation," Computer Networks, vol. 34, no. 4, pp. 579-595, October 2000.

[6] McHugh, "Testing Intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratories," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262-294, November 2000.

[7] Sommer, R.; Paxson, V., "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," Security and Privacy (SP), 2010 IEEE Symposium on , vol., no., pp.305,316, 16-19 May 2010

[8] M. V. Mahoney and P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," in Proc. Recent Advances in Intrusion Detection, 2003.

[9] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. 10th ACM Conference on Computer and Communication Security (CCS 03), pages 251261. ACM Press, October 2003

[10] J. Zhang and M. Zulkernine. Network intrusion detection using random forests. Proceedings of the the Third Annual Conference on Privacy, Security and Trust, pages 53-61, October 2005.

[11] Gharibian, F.; Ghorbani, A.A., "Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection," Communication Networks and Services Research, 2007. CNSR '07. Fifth Annual Conference on , vol., no., pp.350,358, 14-17 May 2007

[12] W. W. Streilein, R. K. Cunningham, and S. E. Webster, "Improved detection of low-profile probe and denial-of-service attacks," in Proceedings of the 2001 Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, June 2001.

[13] Chi Cheng; Wee Peng Tay; Guang-Bin Huang, "Extreme learning machines for intrusion detection," Neural Networks (IJCNN), The 2012 International Joint Conference on , vol., no., pp.1,8, 10-15 June 2012

[14] C.P.Robert,Le choix Baysien. Principes et pratiques, Ed. Springer,2006

[15] Androutsopoulos I., J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos.2000a. An Evaluation of Naive Bayesian Anti-Spam Filtering. Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, Spain, pages 917.

[16] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, vol. 31, no. 23-24, pp. 2435-2463, 1999.

[17] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: A View From the Edge," in Proc. ACM SIGCOMM Internet Measurement Conference, 2008.

[18] Makiou, A.; Begriche, Y.; Serhrouchni, A., "Improving Web Application Firewalls to detect advanced SQL injection attacks," Information Assurance and Security (IAS), 2014 10th International Conference on , vol., no., pp.35,40, 28-30 Nov. 2014

[19] Te-Shun Chou; Fan, J.; Fan, S.; Kia Makki, "Ensemble of machine learning algorithms for intrusion detection," Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on , vol., no., pp.3976,3980, 11-14 Oct. 2009