Anomalous Web Payload Detection: Evaluating the Resilience of 1-grams Based Classifiers.

Sergio Pastrana, Carmen Torrano-Gimenez, Hai Than Nguyen, and Agustín Orfila

Abstract Anomaly payload detection looks for payloads that deviate from a predefined model of normality. Defining normality requires an intelligent approach. Machine learning algorithms have been widely applied to build classifiers that distinguish normal from anomalous activity. These algorithms construct vectors of features extracted from raw payloads of a given dataset and train the classifier with them. The success of the detection highly depends on the potential of the training dataset to properly represent network traffic. In this paper we show that an adversary knowing the distribution of the dataset and the specific feature construction method may generate attack vectors evading the classifier. Particularly, in the case the classifier uses a simple feature construction method based on 1-grams, getting real-world payloads to evade the classifier is feasible. We present experimental results regarding four well-known classification algorithms, namely, C4.5, CART, Support Vector Machines (SVM) and MultiLayer Perceptron (MLP).

1 Introduction

Intrusion Detection Systems (IDSs) look for malicious activities in network and system data. Concretely, payload-based detection looks for intrusive patterns encapsulated in application data, such as web traffic, e-mails, instant messaging, etc. Due to the high complexity and variety of application data, a common approach is the use of Machine Learning (ML) algorithms [9, 7] to learn a model which is used in detection time to distinguish intrusive from normal payloads. ML algorithms require data

Sergio Pastrana and Agustin Orfila
Carlos III University of Madrid, Spain e-mail: `[spastran,adiaz]@inf.uc3m.es`

Carmen Torrano-Gimenez
Institute of Physical and Information Technologies, CSIC, Spain e-mail: `carmen.torrano@iec.csic.es`

Hai Than Nguyen
Telenor Research, Norway e-mail: `hai.nguyen_thanh@inria.fr`

represented as feature vectors, which are obtained by performing a feature construction process on raw traffic. A common approach to extract features from payloads is to use text processing methods, like *n*-grams [10, 8], which consider all the words of size *n* from the text.

ML algorithms assume that both the training and testing data have similar distributions. However, due to the presence of adversaries (which is the common scenario for IDS), the research community has lately focused on designing robust ML algorithms [5, 3]. In this scenario, it must be assumed that the data used for training and testing is different because of the possible changes performed by an adversary in the payloads. Biggio et al. [3] have recently presented an approach to assess the security of pattern classifiers against these attacks. Still, there is a substantial lack of experimental work exploring the problems derived from an attacker who can modify instances at will to subvert the detection function.

In this work we analyze the resilience of IDSs that use ML as classification algorithms and 1-grams as feature construction method. Although these classifiers are effective for a particular distribution of the training dataset, they may learn patterns that are specific for this distribution. We show that an adversary can reverse engineer the detection surface and discover these specific patterns. Then, she can evade the system by properly modifying some features from the attack vector. Nevertheless, it is still required to build raw payloads from these modified vectors to obtain real-world evasions, which is suitable if the feature construction process can be inverted. For example, if the feature construction uses 1-grams, the adversary only needs to include or remove single bytes wherever she chooses into the payload. Next, we summarize the main contributions of our work:

1. We conduct experiments with modern HTTP traffic containing real world attacks.
2. We study four classification algorithms that have been widely used in the research literature for IDSs.
3. We discuss the robustness of IDSs using these algorithms and 1-grams in the presence of smart adversaries. Concretely, we present a reverse engineering and evasion attack that allows to carefully modify malicious payloads and evade the IDSs.

The remainder of this document is structured as follows. Section 2 explains the experimental setup and Section 3 describes the attacks. Then, Section 4 presents the results and finally, Section 5 provides the conclusions.

2 Experimental Setup

We use the **CSIC 2010 HTTP dataset** [1], which has been successfully used for malicious payload detection in previous works [8, 6]. The traffic of the dataset contains normal and anomalous requests targeted to an e-commerce web application, with different values for those web pages that includes parameters. In total 36,000 normal requests and 25,000 anomalous requests are included. As the traffic is generated, all the requests are labeled (either as normal or anomalous). The dataset includes modern web attacks such as SQL injection or Cross-Site Scripting (XSS).

We use the following automatic **feature construction method** based on the 1-grams method for intrusion detection: for every HTTP request p, a feature vector $x(p) = (x_1, x_2, \ldots, x_{256})$ contains the number of appearances x_i of the 1-gram i in the method, path and arguments of p. After extracting the 1-grams from the dataset, we observe that from the 256 possible features (i.e., total number of ASCII characters), only 89 (34.77%) appear one or more times in the HTTP requests. Thus, each vector is composed of these 89 features.

We have conducted our experiments with IDSs using four **classification algorithms**: C4.5 and CART decision trees, SVM, and MLP. First, each IDS is trained to classify HTTP packets using labeled data with both normal and intrusive packets, using one third of the dataset. Second, the IDS is tested with a second third of the dataset (the final third of the dataset is used to test the reverse engineering attack). Table 1 shows the effectiveness of the different IDSs studied over test data in terms of the hit rate (H), i.e., the ratio of attacks detected by the system, and the false positive rate (F), i.e., the ratio of normal payloads wrongly classified as intrusions. It can be observed that they all obtain high detection rates and acceptable false positive rates, which makes them a suitable solution to detect malicious payloads.

Table 1 Detection rate (H) and false alarm rate (F) of the classification algorithms studied.

	C4.5	CART	SVM	MLP
H	0.97	0.97	0.95	0.96
F	0.04	0.07	0.06	0.12

3 Description of the attacks

We adopt an **adversarial model** where the attacker has knowledge about the distribution of the training data used by the IDSs and the feature construction method (FC). Accordingly, the adversary can generate training samples that are similar to those used by the IDSs. Both the distribution and feature construction method may be kept secret in many scenarios. In this work we do not tackle the problem of how to get this information. Indeed, many authors have assumed that this information is known by an adversary [3] or inferable by a query-response analysis [2].

Our **reverse engineering attack** aims to approximate the classifier with models that are easy to process. Such models are later used to perform evasion attacks. The adversary knows the training distribution so she is able to generate training samples and build models from them that are good approximations, in terms of the decision surface, to the IDS classifier she wants to evade. Concretely, we use Genetic Programming (GP) to obtain an approximation of the decision surface of the actual detection model. We choose GP because it outputs tree-based expressions that are easy to understand and can be evaluated with a recursive function, where the root and intermediate nodes are mathematical and logic functions, and the leaves are terminal features. The final output of each program or individual indicates whether the payload is considered anomalous or normal. To evaluate the GP individuals, we use the fitness function shown in Equation 1, which considers both the classification error (ratio of incorrectly classified payloads), and the C_{id}, which is a modern metric used to assess the efficacy of IDSs (see details in [4]).

$$fitness = \frac{E\_class + (1 - C_{id})}{2} \tag{1}$$

The **evasion attack** uses the GP model to look for strategies that transform a malicious payload that would be classified as anomalous by the IDSs into one that is classified as normal. The evasion search is done by analyzing the features and operators from the models to look for potential vectors that evade the classifier. Then, these vectors are mapped into real payloads to evade the system using the inverse process of the feature construction. We perform the evasion search by using a top-down tree-traversal searching algorithm over the model. The final goal is to make the root node change the output from non-zero values (meaning intrusion) to zero (meaning normal). The search over the tree models provides the adversary with a set of evasion strategies, which indicate which features should be modified in order to evade the IDSs. The adversary obtains a set of modified vectors which are given as input to the studied IDS. Each of these feature vectors that passes undetected by the IDS is considered as an evasion candidate. Then, these candidates must be mapped into real payloads by inverting the feature construction algorithm. In the concrete example of 1-grams, each feature in the vector specifies the number of 1-grams in the payload. Accordingly, the adversary only has to remove or insert 1-grams (i.e. bytes) in the payload to get the desired numbers.

4 Results

In this section we first present a sample model obtained in the reverse engineering attack, which is later used to explain how the evasion is conducted. Then, we show a malicious payload that actually evades the four IDSs studied using two different evasion strategies.

In order to control the bloat of GP individuals, we have settled a maximum depth of the trees. Concretely, we have experimented with values from 1 to 10. As an example, Figure 1 shows a GP model obtained with a maximum depth of 4. The model shows in the leaves the relevant features considered for detection, like the feature F37 (the number of 'i' characters in the payloads) or the feature F39 (number of '-' characters). We next use this individual to illustrate how the search of evasion attacks is done.

The evasion search conducted over the GP models suggests possible modifications of features to evade the classifiers. For instance, the rule in Table 2 has been obtained from a search over the model shown in Figure 1. In order to evade the GP model, the rule expression must be evaluated to one. Thus, the rule provides different evasion strategies. For example, setting to zero the features F39 (number of '-'),

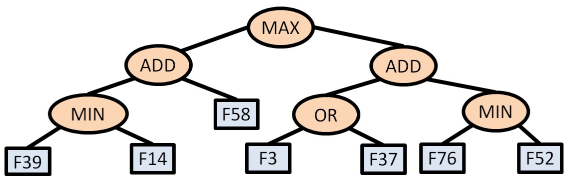


Fig. 1 Reverse Engineering attack. GP model obtained.

F58 (number of ’A’), F37 (number of ’i’), and F76 (number of ’U’), results in vectors that evade the model. The next step is to obtain real payloads from the vectors obtained to evade the IDSs. As discussed below, this is simple because inverting the feature construction process is straightforward for the adversary.

Table 2 Sample evasion rule suggested after performing the evasion search.

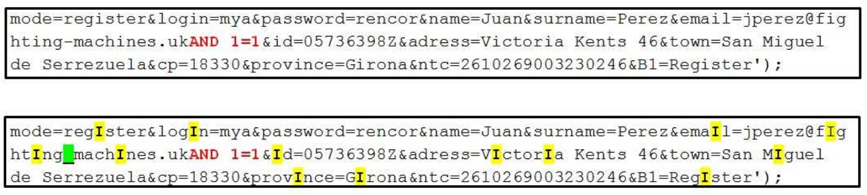
[(F39=0 OR F14=0) AND F58=0] AND [(F3=0 OR F37=0) AND (F76=0 OR F52=0)]

We analyze all the models generated with different tree depths, and obtain all the rule expressions. From the analysis of these rules, we next provide two examples of how an evasion could succeed. Figure 2 shows an example of malicious payload (concretely, performing an SQL Injection Attack) which was originally detected by the detectors, and after modification, evades them all. The suggested modifications, highlighted in colors in the modified payload, are:

1. **Setting the feature F37 (number of ’i’) to zero**. We replace the character ’i’ by its upper-case representation (’I’) to generate payloads free of ’i’ characters (highlighted in yellow in Figure 2). It can be observed that it is a real-world evasion because the HTTP protocol is not case-sensitive and thus the attack still succeeds. However, from the ML perspective, the corresponding feature vector of the testing data changes and so does the detection output.
2. **Setting the feature F38 (number of ’-’) to zero**. This requires removing the hyphens (’-’) characters from the payload, which can be performed by changing these characters by underscores (’_’). In the example highlighted in green in Figure 2, the argument “email” had the value “*jperez@fighting-machines.log*”. If the domain of this email is changed to “*fighting_machines.log*”, the evasion succeeds. However, in real settings, the HTTP request has a different semantic, i.e., the domain of the email may not exist, and the response to this request may lead to some error message, like “invalid email”. Thus, in our approach, it is required to have a knowledge of which the valid modifications are in order to stealthy bypass the detection.

5 Conclusions

In this work, we present reverse engineering and evasion attacks against anomaly-based IDSs based on 1-grams and conventional machine learning classifiers. The



```
mode=register&login=mya&password=rencor&name=Juan&surname=Perez&email=jperez@fig
hting-machines.ukAND 1=1&id=05736398Z&adress=Victoria Kents 46&town=San Miguel
de Serrezuela&cp=18330&province=Girona&ntc=2610269003230246&B1=Register');
```

```
mode=regIster&logIn=mya&password=rencor&name=Juan&surname=Perez&emaIl=jperez@fIg
htIng_machInes.ukAND 1=1&Id=05736398Z&adress=VIctorIa Kents 46&town=San MIguel
de Serrezuela&cp=18330&provInce=GIrona&ntc=2610269003230246&B1=RegIster');
```

Fig. 2 Evasion Attack. Original payload (above) and modified payload (below) which are classified as normal and intrusive respectively by the IDSs.

reverse engineering attack allows discovering the patterns the detectors have learned from the training dataset. Particularly, it uses Genetic Programming to derive tree-based models. Then, the proposed evasion attack analyses these models and suggests modifications of the payload that evade the classifiers. In this case, the modifications are possible as features are easily manipulable by an adversary to get real-world evasions, due to the use of 1-grams. The premises for the attack are realistic, as both the traffic distribution of the protected network and the feature construction method are generally either public or easy to infer.

References

1. The HTTP dataset CSIC 2010 (2010), `http://www.isi.csic.es/dataset/`
2. Ateniese, G., Felici, G., Mancini, L.V., Spognardi, A., Villani, A., Vitali, D.: Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. arXiv:1306.4447 (2013)
3. Biggio, B., Fumera, G., Roli, F.: Security evaluation of pattern classifiers under attack. IEEE Transactions on Knowledge and Data Engineering 99(PrePrints), 1 (2013)
4. Gu, G., Fogla, P., Dagon, D., Lee, W., Skorić, B.: Measuring Intrusion Detection Capability: an Information-theoretic Approach. In: ACM Symposium on Information, Computer and Communications Security. pp. 90–101. ACM, New York, NY, USA (2006)
5. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D.: Adversarial machine learning. In: Workshop on Security and Artificial Intelligence. pp. 43–58. ACM, NY, USA (2011)
6. Nguyen, H.T., Torrano-Gimenez, C., Alvarez, G., Franke, K., Petrović, S.: Enhancing the effectiveness of web application firewalls by generic feature selection. Logic Journal of IGPL 21(4), 560–570 (2013)
7. Pastrana, S., Mitrokotsa, A., Orfila, A., Peris-Lopez, P.: Evaluation of classification algorithms for intrusion detection in MANETs. Knowledge-Based Systems 36, 217–225 (2012)
8. Torrano-Gimenez, C., Nguyen, H.T., Alvarez, G., Franke, K.: Combining expert knowledge with automatic feature extraction for reliable web attack detection. Security and Communication Networks (2012)
9. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: A review. Expert Systems with Applications 36(10), 11994–12000 (2009)
10. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A content anomaly detector resistant to mimicry attack. In: International Symposium on Recent Advances in Intrusion Detection. Lecture Notes in Computer Science, vol. 4219, pp. 226–248. Springer, Germany (2006)