Anomaly-based Web Application Firewall using HTTP-specific features and One-Class SVM

Nico Epp[1], **Ralf Funk**[1], **Cristian Cappo**[1]

[1] Facultad Politécnica – Universidad Nacional de Asunción
San Lorenzo – Paraguay

{nicoeppfriesen,ralffunk0}@gmail.com, ccappo@pol.una.py

Abstract. *Vulnerabilities in web applications pose great risks because they can be exploited by malicious attackers through the Internet. Web Application Firewalls placed in front of these applications can help to minimize these risks. In this paper, we present such a firewall based on anomaly detection that aims to detect anomalous HTTP requests using One Class SVM classifier. Our work uses expert knowledge about the HTTP request structure to build feature extraction methods that improve the detection rates. We include a link to the online repository that contains the code of our implementation for the purpose of reproducibility and extensibility.*

1. Introduction

Web applications have become mainstream in the last decade. Their vulnerabilities pose a great risk because they can easily be exploited by malicious attackers through the Internet. Web Application Firewalls (WAF) can be placed in front of these applications to minimize the risk of attacks [Torrano-Giménez 2015].

According to the detection method, a WAF can be either *misuse-based*, when in looks for known attack signatures, or *anomaly-based*, when it aims to differentiate between normal and anomalous HTTP requests, where the anomalies can include malicious attacks [Torrano-Giménez 2015]. In order to accomplish this differentiation task, the WAF needs to recognize characteristics about the individual HTTP requests. This kind of WAF has two phases, namely training and detection. During the training phase, it extracts models from observed normal requests. During the detection phase, it compares the incoming requests to these models and flags them as anomalous if they deviate significantly from normal requests previously seen. A great strength of this approach lies with the fact that the WAF only needs to be retrained if there are changes in the applications it protects and not because of the appearance of new attacks [Kruegel and Vigna 2003].

For the anomaly detection process of the WAF, one possible strategy is to face it as a classification problem with a single positive class, denominated One Class Classification (OCC) problem [Khan and Madden 2014]. This way the normal HTTP requests belong to this single positive class and all the anomalies will be categorized as not belonging to this class. This OCC approach has the advantage of not needing labeled anomalous data for training, only normal requests are required. One methodology to solve OCC problems is to employ tools from the area of Machine Learning, one of those being the *One-Class* SVM, which has been used with great success for this kind of problems [Khan and Madden 2014]. The characteristics of HTTP requests need to be expressed as numeric feature vectors in order to be used by this classifier.

In this article, we present a WAF based on anomaly detection that uses features selected with expert knowledge about HTTP requests and also uses *One-Class* SVM classifier to detect anomalous requests. In Section 2 we present related works, in Section 3 we describe our proposed WAF, Section 4 contains our test results and we finish this paper with some conclusions and ideas for future work in Section 5.

2. Related works

In [Kruegel and Vigna 2003] and [Kruegel et al. 2005] the authors use expert knowledge about the structure of HTTP requests in their anomaly detection system. They describe how the query string of an HTTP request consists of an ordered list of n pairs of parameters and their corresponding values and how they use statistical methods to build anomaly models of the values for each of the parameters. Some of the models they use are the length of the values, their character distribution, structural inference, parameter presence, among others. For example, one of their models describes the string length of the different values that appear for the parameter *username* in the URL */login*. The approach of these authors differs from ours in that they use statistical methods to detect anomalous requests while we employ *One-Class* SVM. Similar HTTP features and statistical detection methods are used in [Torrano-Giménez 2015].

The *One-Class* SVM has been used successfully in many areas, including text classification, face recognition, spam detection, anomaly detection, among others [Khan and Madden 2014]. In [Tran et al. 2004], network traffic statistics are computed with *tcpstat* and fed to a *One-Class* SVM to detect anomalous network packets. In [Perdisci et al. 2006], byte *n*-grams are extracted of the payload of network packets and passed to an ensemble of *One-Class* SVMs to detect mimicry attacks. The authors in [Parhizkar and Abadi 2015] extract a fixed number of features from HTTP requests and use an ensemble of *One-Class* SVMs to detect attacks in web traffic. This last paper also uses the same data sets we employ, and later we compare our results with theirs. It is worth noting that the three cited works use a fixed number of features, while in our WAF this number depends on the quantity of parameters in the training data.

3. Proposal

In this section, we describe the four main parts that make up our WAF, namely a routing step, a data preprocessing step, a classification step and lastly a response step. Our implementation was done in the programming language *Python* using the Machine Learning library *scikit-learn* and the code is accessible in our online repository under `https://github.com/nico-ralf-ii-fpuna/paper`.

3.1. Routing step

Since requests to the same URL show a greater similarity to each other than to requests to other URLs, our routing step groups the incoming requests by URL and HTTP method. This way during the training phase the anomaly models can be built independently for each group, resulting in a more precise description of the normal requests within each group. Consequently, during the detection phase there are more accurate models of normal requests available, which help to identify requests that are anomalous within their corresponding groups, even though they would be considered normal in other groups.

3.2. Data preprocessing step

Our data preprocessing step extracts a vector of numeric features from each HTTP request, which is then passed to the classification step. We use feature extraction methods that yield a total of 10 numeric features. Four of our features indicate total length, number of digits, number of letters and number of non-alphanumerical characters. We based these on [Kruegel and Vigna 2003] and [Nguyen et al. 2011], adding some own extensions. Another five features, also based on [Kruegel and Vigna 2003], represent the bins of a method called character distribution. Our last feature represents the entropy according to Shannon [Dobrushin and Prelov 2011], an idea taken from [Nguyen et al. 2011].

Our feature extraction methods are applied to the whole request, including HTTP method, URL, query string, headers, and body. Additionally, we employ the previously mentioned approach used in [Kruegel and Vigna 2003] and apply our extraction methods on each of the parameter values in the query string. We extend this approach to also analyze the parameter values in the body of the requests if there are any.

The obtained feature vector that represents a request will have $m \times (1+n)$ features. Here m is the number of features returned by our feature extraction methods for each value. The 1 represents the whole request and n is the number of parameters that are present in the query string and body. Its important to note that n might be different for each group of URL and HTTP method. During the training phase, all the different parameters from a group of requests are listed and given a fixed order for building the feature vector. This way, during the detection phase our WAF extracts the features of the values whose parameters have been seen in the training phase, assuring that their position within the vector is preserved. For example, if during training our WAF gets POST request which all have only the parameters *username* and *password* in its body, then $m = 10$ and $n = 2$, resulting in each request being represented by a vector of 30 features.

We are aware that our analysis of parameter values during the detection phase is only applied to parameters observed in training. This gives way to the risk of an intruder including an attack inside the value of a previously unseen parameter. But since our feature extraction methods are also applied to the whole request, these attacks can still be detected and do not simply bypass our WAF.

3.3. Classification step

During the training phase, our WAF scales and normalizes the numeric feature vectors produced by the previous step and trains one *One-Class* SVM classifier for each group of URL and HTTP method. In this high dimensional vector space, the classifier tries to find boundaries to the regions in which the feature vectors of normal request reside, drawing these borders as tight as possible to exclude future vectors of anomalous requests, but loose enough to still include future normal requests not seen in the training phase [Perdisci et al. 2006]. This way the WAF obtains a model, a trained classifier, that generalizes the characteristics of normal HTTP requests within each group.

During the detection phase, the incoming requests are routed to their corresponding group, the feature extraction methods are applied and the trained classifier for that group checks if this new feature vector falls inside the established boundaries, marking it as normal if it does. All other requests will be denoted as anomalous.

The *One-Class* SVM takes a few parameters that influence its behavior. One of these parameters, denominated nu or ν, is used to set an upper bound to the fraction of training requests that may fall outside of the established boundaries. This gives the classifier some flexibility when drawing the borders in order to achieve higher generalization capabilities and also achieves robustness in case of some anomalies in the training data [Schölkopf et al. 2001]. Sometimes it can be difficult to find boundaries in the given vector space, so the classifier can use a so called kernel function to simulate a higher dimensionality in which it is easier to draw the separating borders. We tested three kernels, namely the linear, the polynomial and the *Radial Basis Function* (RBF) kernel. We obtained the best classification results with the last one. The RBF kernel also has a parameter called $gamma$ or γ that influences the shape of the boundaries [Tran et al. 2004].

3.4. Response step

Our WAF can be configured to respond in different ways to the classification results. If a request is considered normal, it is forwarded to its intended destination. Otherwise, the WAF logs the result and optionally can also block that request, preventing possible attacks from reaching the applications. Our implementation could be extended, for example, to send alarms about anomalies to the people responsible for the system.

4. Experiments and results

For the quantitative evaluation of the performance of our WAF, we used the public data sets CSIC 2010 [Torrano-Giménez et al. 2010] and CSIC TORPEDA 2012 [Torrano-Giménez et al. 2012], which contain labeled HTTP requests to an e-commerce web application. Given that our WAF groups the requests by URL and HTTP method, we grouped the requests from both data sets by these criteria and selected those groups that had more than 100 samples of each of the two categories, normal and anomalous. This left us with 18 groups, totaling 40,130 normal and 42,444 anomalous HTTP requests.

4.1. Detection effectiveness test

In a first test, we used the aforementioned data to measure the detection effectiveness of our WAF. To demonstrate the added value of analyzing the parameter values, we tested two different scenarios. Firstly, we applied our feature extraction methods only to the whole request, obtaining a fixed number of 10 features. Secondly, we included the analysis of parameter values, yielding different number of features for each group, as described in the previous section. Additionally, for the selection of ν and γ for each group, we tested values in the range $[0.0001 : 0.1]$ and chose those that gave the best result in each group. Table 1 shows average and standard deviation of true positive rate (TPR), false positive rate (FPR) and F_1 score of our results. F_1 score uses the number of true positives (TP), false positives (FP) and false negatives (FN), and it is expressed as $F_1 = \frac{2TP}{2TP+FP+FN}$, where values closer to 1 indicate better results.

Table 1. Results of detection effectiveness test for all 18 groups

Scenario	average TPR	average FPR	average F_1	best F_1
using only the whole request	0.91 ± 0.11	0.31 ± 0.34	0.82 ± 0.18	1.00
including parameter analysis	0.95 ± 0.05	0.09 ± 0.11	0.93 ± 0.07	1.00

Our results show that the second scenario improves the overall detection, with TPR and F_1 raising from 0.91 and 0.82 to 0.95 and 0.93 respectively and FPR lowering from 0.31 to 0.09, even though both scenarios have groups that achieve a perfect score. These results demonstrate the usefulness of including the analysis of parameter values.

Comparing our results with related works that use the same data sets that we employ, we find that in [Parhizkar and Abadi 2015] a TPR of 0.96 and a FPR of 0.03 is reported for the ensemble of *One-Class* SVMs. Two popular *misuse-based* WAFs, ModSecurity[1] and PHPIDS[2], were used with default rules on the first data set, CSIC 2010, obtaining a TPR of only 0.55 and 0.29 respectively [Giménez and Cappo 2015], significantly lower than the results we achieved with our WAF.

4.2. Response time analysis

Our second test aimed to quantify the effects our WAF has on the response time of the web applications it protects. To that end, we set up a simple application and measured the response time in three different scenarios; in one the requests went directly to the application, in the second the traffic was routed through our WAF but with detection disabled, and in the third scenario with enabled detection. The second and third scenarios show an increase of 2 and 4 ms respectively when compared to the scenario without the WAF. This delay is only a small increase compared to the overall roundtrip time of HTTP traffic, so our WAF should not have noticeable effects on the response time.

4.3. Training time analysis

In our third test, we analyzed how the training time of our WAF relates to the amount of requests used. Since groups may have different durations because of an unequal number of features, the highest numbers give an upper bound. We observed that the time per request stays close to 2 ms in our test setup with 10,000 requests. Our numbers show that the training time has an almost linear relationship to the amount of training data.

5. Conclusions

The WAF we present in this article uses the *One-Class* SVM classifier to tackle the task of detecting anomalous HTTP requests in order to protect web applications from possible attacks. Following and extending the ideas obtained from other authors, we built feature extraction methods that take advantage of the structure of HTTP requests, specifically the values of individual parameters, to represent these requests with vectors which contain more useful information. Our tests show a significant improvement in the detection results when including the analysis of parameter values. Furthermore, our implementation shows that it is fast enough to be used as a WAF, even though the minimization of the processing time was not our primary goal and further optimizations could be made.

One contribution of our paper is the adaptation and extension of the parameter value analysis presented in [Kruegel and Vigna 2003]. The anomaly models employed by the cited authors do not produce numeric vectors for each request, and so we adapted them to make feature extraction methods that are useful for Machine Learning tools, in our case specifically the *One-Class* SVM. As another contribution, we leave a link in Section 3 to

[1]`www.modsecurity.org`

[2]`https://github.com/PHPIDS/PHPIDS`

the online repository that contains the code of our implementation, which includes the WAF and the tests mentioned in this paper, so that other authors can reproduce our results and have a starting point for further research.

Future works could look into finding additional features to detect anomalous requests. Another research area is other classification tools to be used instead of the *One-Class* SVM to tackle the OCC task. Additionally, our work lacks automatic selection of the values for ν and γ of the classifier and further inquiries can be made in that direction.

References

Dobrushin, R. and Prelov, V. (2011). Entropy - encyclopedia of mathematics. `http://www.encyclopediaofmath.org/index.php?title=Entropy&oldid=15099`. Accessed: August-2017.

Giménez, J. and Cappo, C. (2015). Http-ws-ad: Anomaly detector oriented to web applications and services. In *Computing Conference (CLEI), 2015 Latin American*. IEEE.

Khan, S. and Madden, M. (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374.

Kruegel, C. and Vigna, G. (2003). Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*. ACM.

Kruegel, C., Vigna, G., and Robertson, W. (2005). A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5).

Nguyen, H. T., Torrano-Gimenez, C., Alvarez, G., Petrović, S., and Franke, K. (2011). Application of the generic feature selection measure in detection of web attacks. In *Computational Intelligence in Security for Information Systems*, pages 25–32. Springer, Berlin, Heidelberg.

Parhizkar, E. and Abadi, M. (2015). Oc-wad: A one-class classifier ensemble approach for anomaly detection in web traffic. In *2015 23rd Iranian Conference on Electrical Engineering (ICEE)*, pages 631–636. IEEE.

Perdisci, R., Gu, G., and Lee, W. (2006). Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 488–498. IEEE.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Torrano-Giménez, C. (2015). *Study of stochastic and machine learning techniques for anomaly-based web attack detection.* PhD thesis, Universidad Carlos III de Madrid.

Torrano-Giménez, C., Pérez, A., and Álvarez, G. (2012). Csic torpeda 2012 http data sets. `http://www.tic.itefi.csic.es/torpeda`. Accessed: July-2017.

Torrano-Giménez, C., Pérez Villegas, A., and Álvarez Marañón, G. (2010). Csic 2010 http data sets. `http://www.isi.csic.es/dataset/`. Accessed: July-2017.

Tran, Q.-A., Duan, H., and Li, X. (2004). One-class support vector machine for anomaly network traffic detection. *China Education and Research Network (CERNET), Tsinghua University, Main Building*, 310.