Improving Web Application Firewalls through Anomaly Detection

Gustavo Betarte*†, Eduardo Giménez†, Rodrigo Martínez*† and Álvaro Pardo‡
*Instituto de Computación, Universidad de la República, Uruguay
†Tilsor SA, Uruguay
‡Departamento de Ingeniería Eléctrica, Universidad Católica del Uruguay, Uruguay
Email: [gustun,rodmart]@fing.edu.uy,[gbetarte,egimenez,rmartinez]@tilsor.com.uy, apardo@ucu.edu.uy

***Abstract*—Web applications are permanently being exposed to attacks that exploit their vulnerabilities. In this work we investigate the application of machine learning techniques to leverage Web Application Firewalls (WAF)s, a technology that is used to detect and prevent attacks. We put forward an approach of complementary machine learning models, based on one-class classification and n-gram analysis, to enhance the detection and accuracy capabilities of MODSECURITY, an open source and widely used WAF. The results are promising and outperform MODSECURITY when configured with the OWASP Core Rule Set, the baseline configuration setting of a widely deployed, rule-based WAF technology.**

***Index Terms*—Web Application Firewalls, Machine Learning, Anomaly Detection, One-class Classification, N-gram Analysis**

I. INTRODUCTION

A web application is a piece of software, based on a client-server architecture, that embodies a coordinated set of functions. The information flowing between the client, which runs on the user's web browser, and the application server is transmitted using the HTTP protocol. It is quite usual for the code of a web application to contain vulnerabilities like the ones listed and described in the OWASP TOP 10 [1].

A Web Application Firewall (WAF) is a piece of software that intercepts and inspects all the traffic between the web server and its clients, searching for attacks inside the HTTP packet contents. Once recognized, the suspicious packets may be processed in a different, secure way, for instance being logged, suppressed or derived for processing. MODSECURITY [2] is an open source, widely used WAF enabling real-time web application monitoring, logging and access control. The working of MODSECURITY is driven by rules that specify the contents of the HTTP packets to be spotted and it supports the configurations of both positive and negative security models. In this work we have focused in a negative model, that is, the firewall shall allow all traffic to pass except what is known to be malicious.

For tackling the most usual vulnerabilities MODSECURITY offers a default set of rules known as the OWASP Core Rule Set (OWASP CRS) [3]. An approach only based on rules though has some drawbacks: rules are static and rigid by nature, so the OWASP CRS usually produces a high rate of false positives, which in some cases may be close to 40% [4]. As the intended use of MODSECURITY is to block attacks, such a high false positive rate would potentially lead to a denial of service of the application. Rule tuning, however, is a time consuming and error prone task that has to be manually performed for each specific web application. In traditional networks firewalls and IDS, the approach based on rules has been successfully complemented with other machine learning-based tools which provide higher levels of flexibility and adaptability. Those approaches take advantage of sample data, from which the normal behavior of the web application can be learned, in order to spot suspicious situations which fall out of this nominal use (anomalies), and which could correspond to on-going attacks. The work we present in this paper contributes to improve the detection capabilities of MODSECURITY with such anomaly detection techniques.

The structure of the rest of the paper is as follows: Section II provides some background and puts forward our proposal to enhance MODSECURITY using machine-learning techniques. Section III presents the two complementary learning models that we use to ground a statistical WAF. Then, Section IV describes and discusses the outcomes of the experiments. Section V reviews related work. Further work and conclusions are presented in Section VI.

II. BACKGROUND AND PROBLEM STATEMENT

In order to enhance the detection and accuracy capabilities of MODSECURITY configured using the OWASP CRS rules out of the box we have first experimented with a mechanism that integrates one-class classification. This amount to combine two experts with the objective of classifying a request. When both experts agree (both say valid or attack) then the result is straightforward. In the case the one-class approach classifies a request as an attack we prioritize the answer of MODSECURITY given that the OWASP CRS rules embody the know-how on attacks. This integration decision also allows us to have a well known mechanism to tune our WAF in the case we find a false positive: we need to modify MODSECURITY rules that specify that this request is not an attack. This rule tuning is exactly the same as the one used with the OWASP CRS nowadays. Finally, in the case that one-class classifies the request as

This research was partially supported by a grant given to Rodrigo Martinez from the ICT4V center (http://ict4v.org) and was done in the context of projects WAFINTL and FMV_1_2017_136337 (ANII).

valid and MODSECURITY as an attack we prioritize one-class since OWASP CRS is known to have high false positive rates. This approach has shown to adapt quite well to the scenario where there is not available a specific training dataset for the application. We train the one-class classifier using several datasets and the resulting classification model can be used to protect different web applications. The main advantage is that it is easy to deploy and capable of adapting to changes in the web application.

However, the detection capabilities provided by the one-class approach do not adapt so well to prevent both zero-day attacks and attacks that exploit specific vulnerabilities of an application, in particular those involving suspicious input. This is why the second approach we have investigated focuses on characterizing which are the expected (valid) input values for a web application and classifying as a potential attack any anomalous input supplied to it. Furthermore, this positive characterization approach only requires a collection of unlabeled HTTP requests produced by friendly users in order to train the WAF.

For this second approach, we have experimented with analyzing the frequencies of n-grams in the application inputs up to some n. Our experiments show that such technique provides higher performance rates than the other tested ones. However, the price to pay for this higher flexibility and performance is that it requires to have an application specific dataset with valid request to train the n-gram model. A second drawback for the positive characterization of the web application is that in some cases there is no a priori expected behavior for some pieces of input [5]. For example, the values of a user password should not have any particular expected bias. In this cases it is necessary to adopt the symmetric approach, and search for attack signatures on passwords according to the current state of the art. This amounts to search for carefully selected tokens which have been extracted from a labeled training dataset and representing the knowledge of the security expert on the current attack vectors, as it is done in the one-class classification approach. This technique deems the HTTP request as valid by default when it cannot be recognized as an attack according to the previous training. This way of coping with unbiased input provides a second level of complementary between the one-class and the n-gram analysis approaches.

In summary, if we need to fast deploy a WAF to protect a web application without having a specific dataset or the web application changes constantly (e.g. public website powered by a content manager), we propose to use the one-class approach. In this case (Scenario I) we would like to address the following question: *Is it possible to build an attack detection system learning from training data collected from other web applications?* A second question in this scenario is related to MODSECURITY: *Can we improve the results of* MODSECURITY *using anomaly detection methods?* That is, can we reduce the number of false positives (FP) generated by MODSECURITY? In the scenario where we need to protect a critical web application where high levels of security are required, and the application's changes are controlled, we can deploy the n-gram approach. In this second case (Scenario II), it shall be required to have a software testing phase before each new deployment of the application, so specific training data will be available. In this latter scenario, *we would like to understand the attainable performance of anomaly detection methods against that of* MODSECURITY.

III. LEARNING MODELS

The proposed models for anomaly detection follow the standard sequence of pre-processing of the HTTP request, feature extraction and model learning using a training set $\mathscr{T}$.

The requests are first pre-processed to decode the information they contain. Then, a set of features, related to the occurrences of a collection of substrings or *tokens* in the request, are extracted. The model learns the distribution of token occurrences in order to classify new request as either valid or abnormal. Together with this classification the model provides a score of normality.

The rest of this section is devoted to describe the two different models: one-class classification and n-gram analysis.

A. One-Class Classification

In this approach, during the pre-processing phase decodification is carried out together with the filtering of request headers that are used to exchange contextual information between the user-agent and the server. All information contained in headers that are specific to the protocol, such as cookies, proxies and IP, which do not represent user behavior and should not be considered to infer application behavior, are filtered out.

Following [6] we have relied on the experience of a security expert to define the features that capture the properties of well- known web application attacks. The features defined includes symbols (p.e. $<, =, |$) and tokens (p.e. *select*, *passwd*). A complete list of the features used could be found in [7]. We apply a *bag-of-words* model where each document (in our case each request) is represented as a bag of those *words*. Since we are modeling the valid class, we expect that few of those words will be present in each request.

A well known method to build a one-class classifier is to estimate the *probability density function* (pdf) of the training data and to estimate a threshold to segment normal and abnormal instances. Among the existing methods to estimate the pdf, Gaussian Mixture Models (GMM) is one of the most common one. In our case, each component of the obtained GMM constitutes a cluster that captures the distribution of valid requests. We use the Expectation Maximization (EM) algorithm [8] to estimate the parameters of the GMM and the number of components (clusters). To measure the distance of a feature vector to each cluster (C_k) we use the Mahalanobis distance:

$$dist(x_i, C_k) = \sqrt{(x_i - \mu_k)\Sigma_k^{-1}(x_i - \mu_k)} \qquad (1)$$

If one of the dimensions has a standard deviation of 0, the Mahalanobis distance can not be calculated. To solve that, we regularized the covariance matrix adding $\epsilon * Id$, where ϵ is the smallest standard deviation in $diag(\Sigma_i)$ different from 0 and *Id* is the identity matrix. This regularization allows the observation of new values on requests not seen in $\mathscr{T}$.

Before being able to do request classification, we need to estimate the threshold of each cluster. The threshold of the

cluster is defined by analyzing the distribution of the distances of each instance of $\mathscr{T}$ that where assigned to the cluster. Given that all requests assigned to a cluster (as a result of using EM) can be represented by a component of the GMM, we approximate the distribution of the distances with a normal distribution. We define $\overline{dist}_k$ to denote the mean of all distance of x_i to C_k where $x_i \in C_k$ and std_k to denote the standard deviation of the distances of $x_i \in C_k$. Finally, the threshold t_{C_k} is defined as $\lambda[\overline{dist}_k + 10 * std_k]$, with $\lambda \in (0, 1]$. The use of $\overline{dist}$ and *std* makes it possible to have in the model a specific threshold for each cluster that depends directly on how the instances that belong to that cluster behave. The constant factor 10 was derived using different dataset in a way that when $\lambda = 1$ the distance for all $x_i \in C_k$ is less than t_{C_k}. Varying the threshold, by multiplying it by a constant λ, let us change the model's precision. This parameter allows us to have different operational points. The classification of a new request starts by calculating the Mahalanobis distance to each cluster. If any of the distances is lower than the corresponding threshold the request is classified as normal, otherwise is classified as an attack.

In real life applications, typically we only have valid requests. In this case, one way to define the operational point is to set the amount of false positives that we are willing to accept. When we do not have application specific dataset, a mixture of several datasets from different web applications can be used. In Section IV we will present the results of this approach (Scenario I) to show that good performance scores can be achieved.

B. Anomaly detection using n-grams

We now turn to present our second approach, which positively characterizes the normal behavior of each application using n-grams as tokens.

A well-known technique for identifying the language of a piece of text is to measure the relative frequency of each sequences of n consecutive tokens from the alphabet used in the text (n-grams), for example, ASCII characters, words, etc. This provides a tokenization procedure that can be uniformly applied independently from the language or text structure.

We try to exploit as much as we can from the HTTP structure, computing a specific n-gram frequency distribution for each CGI parameter and HTTP header (both referred as *fields* in the sequel). The n-gram frequency distribution provides the *language signature* of each field x. We consider the n-grams up to a given length n and count them after normalizing the text by uncapitalizing letters, removing accents and replacing digits by the capital letter "*N*". The model is enriched with an additional attribute, namely, the number of characters (length) of each field x. This additional attribute is also a good indicator for code injection attacks, as they are likely to increase the expected field length.

Let M be the distribution of n-gram frequencies for each field x, computed from the training set $\mathscr{T}$. In order to test a given HTTP request r, the WAF computes the frequency $f_r(x.z)$ of each n-gram z of field r_x. If x is not defined in model M, then r is rejected. Otherwise, the *score* $s_{r.x} = \sum_{i=1..n} dist_i^x(r, M)$ is computed for each field x using the following version of Mahalanobis distance under the assumption of n-gram independence:

$$dist_i^x(r, M) = \sqrt{\sum_{z \in \mathscr{T}_i^x} \frac{|\mu_M(x.z) - f_r(x.z)|^2}{\frac{1}{|\mathscr{T}_i^x|} + \sigma_M^2(x.z)}} \qquad (2)$$

where $|\ \mathscr{T}_i^x\ |$ is the number of i-grams in model field x. The constant $\frac{1}{|\mathscr{T}_i^x|}$ is added to the denominator to prevent a division by zero for those n-grams with constant frequency.

A score is considered acceptable when it is between the minimum and maximum values of the score distribution *ds* drawn from the training set $\mathscr{T}$ by computing the score of each individual training request. If there is a field x for which the obtained score is not inside the min-max interval, the whole request is deemed as anomalous (a potential attack).

Should a given n-gram of field x not be defined in M, we use a *prior distribution* of the contents of x to draw its expected frequency. By assigning priors to some CGI parameters we decrease the false positive rates. Specific priors, n-gram lengths and other model parameters are configured for each field x as part of the model tuning of each web application.

IV. Experimental results

We present the results of evaluating the proposed models on the CSIC2010 dataset [9] and on a dataset of our own which was obtained from the HTTP traffic to the web server of a University (DRUPAL dataset). We have also experimented the one-class approach on the dataset from the PKDD2007 challenge [10]. The results for this dataset for the one-class approach are presented in [7]. This dataset is not suitable for experimenting the n-gram approach, as its developpers performed a process of obfuscation to protect the data in it, replacing URLs, parameter names and values with random data.

Each datasets is made of a collection of complete HTTP requests (header and body) which have been partitioned into one training dataset and two testing ones with valid and anomalous traffic. In Table I, we present the details of each dataset. As explained before, our models only use normal traffic for training, labeled attacks were considered only for testing. The model was evaluated using true positive rate (TPR) and true negative rate (TNR). The TPR corresponds to the percentage of anomalous requests that are detected as attacks. The TNR corresponds to the percentage of valid request that are spotted as normal traffic.

The baseline to which we compare the behavior of our models are the TPR and TNR values obtained as the outcome of using MODSECURITY configured with the OWASP CRS version 2.2.9 out of the box.

In the one-class classification approach the main parameter of the classifier is the threshold that governs the size of the clusters. The size is adjusted by a parameter λ whose values range from 0 to 1. Each value of λ determines an operational point of the classifier. For each dataset we display two ROC curves. The blue thin line represents the curve obtained for the one-class classifier when training is performed using specific data for the web application and varying λ in $(0, 1]$. This

Table I
DATASETS COMPOSITION

	Train (Valid)	Test (Valid)	Test (Attack)
CSIC	36000	36000	25065
Drupal	45907	19675	1287

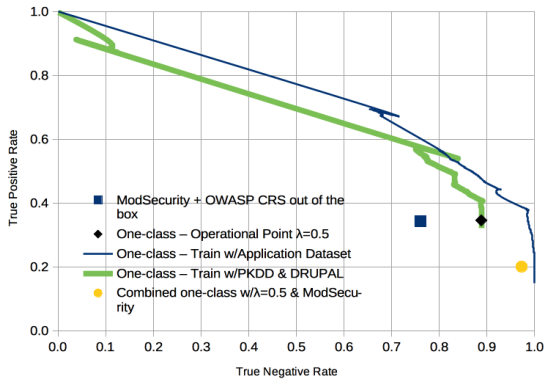

Figure 1. Results for the CSIC2010 dataset.

curve can be seen as the ideal result for the classifier since it uses an application-specific training dataset. The green thick line represents the curve produced with the one-class classifier trained with data from other web applications (Scenario I). A black diamond over the green curve indicates a default operational point obtained with $\lambda = 0.5$. In Scenario I we do not have specific training data, so we cannot fine tune the operational point. Therefore, this point was fixed based on the number of FP for the training set. A blue square indicates the result of MODSECURITY using the OWASP CRS out of the box. Finally, a yellow ball shows the performance of the model that combines one-class with MODSECURITY following the combination strategy discussed in Section II.

For the n-gram analysis approach, the model parameters are the tokenization method, prior distributions and n-gram length bound assigned to each specific field. As these are enumerated values, the results are displayed using a table instead of a curve. The table presents the results for the n-gram length bounds $n = 1 \ldots 5$ when an optimized configuration is assigned to the other model parameters.

A. CSIC2010 *dataset*

This dataset embodies a collection of normal and abnormal HTTP requests for a web application that provides functionalities to perform an on-line shopping. The dataset contains 25.000 abnormal test requests mixing attacks with valid requests containing infrequent characters in the parameter fields (typos). Unfortunately, the distribution between attacks and infrequent values is not specified. The attacks are concentrated on the web application parameters.

In Figure 1 we present the results in terms of TPR and TNR for the CSIC2010 dataset. The simplicity of the normal requests and the mixture of attack with just anomalous traffic is observed in the high TNR and the low TPR obtained by MODSECURITY (blue square). As can be noticed, the one-class classifier trained with data from other web applications (green curve), used to evaluate Scenario I, produced good performance scores (TNR and TPR). The ROC curve shows that there are several points that outperform MODSECURITY (blue square). Furthermore, if we compare the results obtained using an application specific training set (blue curve), we can see the performance it is not far from the ideal one (when we train the one-class classifier with data from the same web application). Finally, the yellow ball shows the performance of the model that combines one-class with MODSECURITY. The combination improves in terms of TNR but decreases in TPR. This is because our main objective is to decrease MODSECURITY false positives so the combination algorithm only marks a request to be an attack if both experts tag it as an attack. This means that some requests that were tagged as an attack by the one-class model where tagged as valid by MODSECURITY and viceversa.

Table II
TRUE NEGATIVE AND TRUE POSITIVE RATES (IN %) FOR EACH DATASET

		CSIC2010		DRUPAL	
Method		**TNR**	**TPR**	**TNR**	**TPR**
ModSecurity		76,1	34,3	61,1	72,2
One-class: $\lambda = 0,5$		88,9	34,6	93,3	86,2
Combined OC-MS		97,3	20,1	99,1	63,0
N-grams	n=1	99,9	93,0	93,9	95,9
	n=2	99,9	94,8	94,4	97,6
	n=3	99,5	96,1	92,0	97,5
	n=4	96,2	96,8	90,7	98,8
	n=5	90,9	97,5	89,4	98,9

If we compare the results with our baseline, e.g. for the same TNR, MODSECURITY detects 34% of the attacks, where the one-class approach detects 56%. In this particular dataset, the integration of MODSECURITY rules with the one-class approach does not improve the results of one-class by itself in terms of TPR but clearly reduces the number of false positives (i.e., TNR close to 1). See Table II for details on the results.

Let us turn now to the n-gram model for this dataset. We performed a fine tuning, configuring our n-gram analysis tool to perform some specific behaviors on certain fields. The URI, login identifier, customer's national identifier and password fields are restricted to monogram analysis. This is because the only biased aspect of those fields is the set of allowed characters, but almost any combination of them is possible in principle, so higher order n-gram analysis is prone to produce false positives. Finally, following the technique explained in section III-B, we specified a prior n-gram distribution to some web application parameters (customer's name, city and address in Spain) drawn from a collection of Wikipedia articles written in Spanish. The use of priors for these fields reduced the false positive rate in 3% for trigrams. The results are presented in Table II. The most significant impact on false positives is observed for $n = 3$, where up to 96% of the attacks are detected for less than 0,1% of false positives, clearly outperforming all the other methods.

B. DRUPAL *dataset*

The CSIC2010 dataset has been artificially conceived. In order to evaluate our approach on real life applications, based on actual requests and attacks, we crafted a dataset by registering three days of incoming traffic to the public website of a

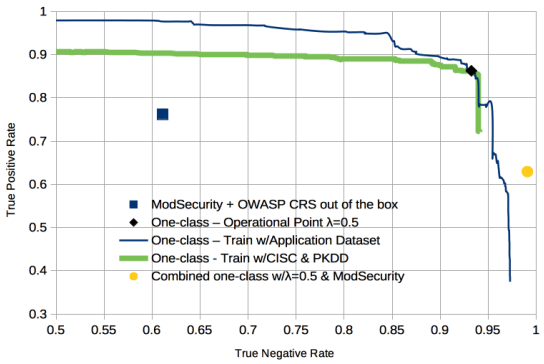

Figure 2. Results for the DRUPAL dataset.

University [1]. The only post-processing of this dataset consisted in blurring password values in the request.

Since the requests are from real traffic, this dataset is less balanced. One of the difficulties that raises is classifying the registered requests into valid ones and attacks. The web site of the University is protected by an instance of MODSECURITY featuring the OWASP CRS, which has been tuned for several years by a team of security experts. We therefore used MODSECURITY as the labeling tool: those requests that were accepted by MODSECURITY were considered as the valid ones, while those requests that MODSECURITY rejected were labeled as attacks. Figure 2 describes the results for the DRUPAL dataset. We observe that they are similar to the ones obtained for CSIC2010. The one-class classifier trained with data from other web applications (green curve), used to evaluate Scenario I, is very close to the blue curve which is the ROC generated with the one-class classifier trained with application specific data. The default operational point, black diamond, outperforms MODSECURITY, and several of the points in the previous curves clearly perform better than MODSECURITY. Finally, the results of the model that combines one-class and MODSECURITY (yellow ball) improves in terms of TNR but decreases in TPR. This is because our main objective is to decrease MODSECURITY FP so the combination algorithm only marks a request to be an attack if both systems tag it as an attack. This mean that some requests that where tagged as an attack by the one-class where tagged as valid by MODSECURITY and viceversa.

As for the n-gram model, the only tuning performed was excluding the URI from the analysis, as the high variance of this model field produced too many false positives. The results are presented in Table II. It provides a similar detection rate as the one-class approach for monograms and bigrams. Again, this approach outperforms the other ones respect to the TPR.

C. Discussion

Based on the previous sectionswe discuss the main questions presented in Section II.

a) Scenario I - Is it possible to build an attack detection system learning from training data collected from other web applications?: The results for datasets CSIC2010 and DRUPAL present evidence that it is possible to build a one-class classifier using generic training data, that is, using a dataset with requests not from the web application to protect. Furthermore, the degradation with respect to the same classifier trained with specific data for the web application is not critical.

b) Scenario I - Can we improve the results of ModSecurity using machine learning methods?: Looking at the results for the datasets CSIC2010 and DRUPAL we can conclude that it is possible to improve the results of MODSECURITY.

c) Scenario II - Attainable performance of machine learning methods against ModSecurity: Based on the obtained results we can draw the following conclusions. First, also in this scenario it is possible to improve the results of MODSECURITY using machine learning approaches such as on-class classification or n-gram analysis. Second, looking at the TNR and TPR scores computed for the n-gram anaysis, we can say that the n-gram approach is the most performant solution if we have an application specific dataset to train its model.

V. RELATED WORK

Several multiclass classification techniques have been previously proposed for web application protection [10, 11, 12, 13]. Those techniques require a training set containing both normal traffic and attacks. On the contrary, our proposal is based on one-class anomaly detection, which only requires normal traffic for the training phase.

Regarding anomaly detection, to the extent of our knowledge, there is no previous work on the application of one-class classification to web application protection. However, several previous works based on n-gram analysis have been proposed [14, 15, 16, 17, 18, 5]. Ingham and Inoue have compared them in [19]. Some of these works construct one single n-gram signature for the whole HTTP request [14, 15]. Our n-gram approach goes further, constructing a specific signature for each component of the HTTP request: URI, CGI parameters, headers and body. This fine grain model enables us to identify more specific bias for each one. We conjecture that mixing field contents is the reason of the poor perfomance reported for Stolfo and Wang's PAYL method in [19], as well as the rather high optimal length (n=6) proposed in that paper for n-gram analysis.

On the other hand, previous works performing a specific analysis for the URL and the web parameters restricts this analysis to monograms (n=1) [16, 17, 18]. However, short n-grams are vulnerable to mimicry attacks, in which the attacker carefully adds characters in order to get closer to the expected n-gram distribution [20, 21]. Mimicry attacks become much more challenging when using higher order n-grams, but frequencies become too sparse for long n-grams [15]. In our n-gram approach, we add the scores obtained for n-grams analysis of all lengths up to a given one, which is provided as a model parameter. The combination of several n-gram analyses and a fine grain parsing of the HTTP request both prevents mimicry attacks and keeps a relative low value for optimal n (n=3).

Normalizing text before computing n-grams simplifies and speeds up the analysis by reducing the combinatory of possible tokens. It also reduces the risk of overfitting. Our n-gram

[1]The dataset is not public but it is available on demand.

approach normalizes the text before computing the n-grams, removing accents, uncapitalizing it and replacing numbers by the capital leter N. Other authors propose more radical transformations, such as collapsing all alphabetic strings or numbers into single values [18, 17]. We think that the proposed transformation provides a better compromise. For example, it collapses all the possible variants of the *where* keyword used in SQL injection attacks (namely, Where, WhERe, etc.) but still enables to recognize that it does not match, for instance, the distribution of a web parameter expecting personal names in Spanish.

VI. Conclusion and Further Work

We put forward a one-class classification approach to learn the behavior of valid requests based on features typically present in the payload of an attack which were selected based on the knowledge of security experts. Hence, the defined model learns the distribution of these features for the valid requests and detects attacks as deviation from normality. This model provides better detection and false positive rates than ModSecurity configured with the OWASP CRS out of the box. We also proposed a simple integration of one-class and ModSecurity based on the output labels of both components (normal, attack). In future work we will study the combination of the scores provided by both of them.

We also presented an anomaly detection model based on the expected n-gram frequencies of each specific HTTP request component, which combines n-grams of multiple lengths. We have experimented with several variants of this general framework and described results obtained with some particular instances of the model parameters, such as the assignment of prior distributions Anomaly detection draw its attributes from the expected normal behavior of the application, and deems as an attack those HTTP requests which deviates from such behavior. This positive characterization of the normality makes the WAF more resilient to zero-day attacks. The results of this model clearly outperform the ones of ModSecurity. Despite this, the two approaches can be seen as complementary, the one-class can be used in Scenario I, when no specific training data is available, and the n-gram is suitable for Scenario II since it needs specific training data to learn from normal traffic.

References

[1] OWASP. Owasp top ten project. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[2] I. Trustwave Holdings, "Modsecurity: Open source web application firewall." [Online]. Available: http://www.modsecurity.org/

[3] OWASP. Owasp modsecurity core rule set project. [Online]. Available: https://www.owasp.org/index.php/

[4] C. Folini. (2016) Handling false positives with the owasp modsecurity core rule set. [Online]. Available: https://www.netnea.com/cms/apache-tutorial-8_handling-false-positives-modsecurity-core-rule-set/

[5] C. Wressnegger, G. Schwenk, D. Arp, and K. Rieck, "A close look on n-grams in intrusion detection: Anomaly detection vs. classification," in *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*, ser. AISec '13. New York, NY, USA: ACM, 2013, pp. 67–76.

[6] C. Raıssi, J. Brissaud, G. Dray, P. Poncelet, M. Roche, and M. Teisseire, "Web analyzing traffic challenge: description and results," in *Proceedings of the ECML/PKDD*, 2007, pp. 47–52.

[7] G. Betarte, E. Giménez, R. Martínez, and Á. Pardo, "Machine learning-assisted virtual patching of web applications," *arXiv preprint arXiv:1803.05529*, 2018.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://www.jstor.org/stable/2984875

[9] A. P. V. Carmen Torrano Giménez and G. Á. Marañón, "CISC2010 dataset," http://www.isi.csic.es/dataset/.

[10] "Analyzing web traffic: Ecml/pkdd 2007 discovery challenge," http://www.lirmm.fr/pkdd2007-challenge/.

[11] M. Exbrayat, "Ecml/pkdd challenge: analyzing web traffic a boundaries signature approach," 2007, p. 53.

[12] K. Pachopoulos, D. Valsamou, D. Mavroeidis, and M. Vazirgiannis, "Feature extraction from web traffic data for the application of data mining algorithms in attack identification." Citeseer, 2007.

[13] B. Gallagher and T. Eliassi-Rad, "Classification of http attacks: a study on the ecml/pkdd 2007 discovery challenge," Lawrence Livermore National Laboratory (LLNL), Livermore, CA, Tech. Rep., July 2009.

[14] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *RAID 2004*, vol. 3224. Springer, 2004, pp. 203–222.

[15] K. Wang, J. J. Parekh, and S. J. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," in *RAID 2006*, vol. 4219. Springer, 2006, pp. 226–248.

[16] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of CCS 2003*. ACM, 2003, pp. 251–261.

[17] C. Torrano-Gimenez, A. Perez-Villegas, G. Á. Marañón *et al.*, "An anomaly-based approach for intrusion detection in web traffic," *Journal of Information Assurance and Security*, vol. 5, no. 4, pp. 446–454, 2010.

[18] I. Corona, D. Ariu, and G. Giacinto, "Hmm-web: A framework for the detection of attacks against web applications," in *Proceedings of ICC 2009*, 2009, pp. 1–6.

[19] K. L. Ingham and H. Inoue, "Comparing anomaly detection techniques for http," in *RAID*, vol. 4637. Springer, 2007, pp. 42–62.

[20] O. Kolesnikov and W. Lee, "Advanced polymorphic worms: Evading ids by blending in with normal traffic," Georgia Tech, Tech. Rep., 2004.

[21] S. Pastrana, C. Torrano-Gimenez, H. T. Nguyen, and A. Orfila, *Anomalous Web Payload Detection: Evaluating the Resilience of 1-Grams Based Classifiers*. Springer International Publishing, 2015, pp. 195–200.