2018 17th IEEE International Conference on Machine Learning and Applications

Web Application Attacks Detection Using Machine Learning Techniques

Gustavo Betarte*, Rodrigo Martínez* and Álvaro Pardo†
*Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay
Email: [gustun,rodmart]@fing.edu.uy
†Departamento de Ingeniería Eléctrica, Facultad de Ingeniería y Tecnologías
Universidad Católica del Uruguay, Uruguay
Email: apardo@ucu.edu.uy

***Abstract*—Web applications are permanently being exposed to attacks that exploit their vulnerabilities. In this work we investigate the use of machine learning techniques to leverage the performance of Web Application Firewalls (WAFs), systems that are used to detect and prevent attacks. We propose a characterization of the problem by defining different scenarios depending if we have valid and/or attack data available for training. We also propose two solutions: first a multi-class approach for the scenario when valid and attack data is available; and second a one-class solution when only valid data is at hand. We present results using both approaches that outperform MODSECURITY configured with the OWASP Core Rule Set out of the box, which is the baseline configuration setting of a widely deployed WAF technology. We also propose a tagged dataset based on the DRUPAL content management framework.**

***Index Terms*—Web Application Firewall; Web Application Security; Machine Learning; Pattern Recognition.**

I. INTRODUCTION

It is a known fact that web applications are permanently being exposed to attacks that exploit their vulnerabilities. Initiatives like the OWASP Top 10 [1] have greatly contributed to rise awareness concerning the security of web applications but have not prevented the ever increasing amount of (successful) attacks. In this context, attack detection techniques become necessary. These techniques involve procedures that help distinguishing between the behavior of a valid user of the system and a malicious agent. The identification and determination of a user's behavior should consider whether each detected event is simply suspicious or actually it is an event that is part of an attack. These types of techniques assist in aspects such as preventing attackers to identify/verify successfully the existence of vulnerabilities in applications and to minimize the number of false positives (non-malicious activity identified as such).

A technological alternative for performing attack analysis is the use of a Web Application Firewall (WAF) [2]. A 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.

An implementation of an open source WAF that has become a *de facto* standard is MODSECURITY [3], which allows the analysis of the users requests and the application responses by enabling real-time web application monitoring, logging and access control. The actions MODSECURITY undertakes are driven by rules that specify, by means of regular expressions, the contents of the HTTP packets to be spotted.

MODSECURITY offers a default set of rules, known as the OWASP Core Rule Set (OWASP CRS) [4], for handling the most usual vulnerabilities included in [5]. However, an approach only based on rules also has some drawbacks: rules are static and rigid by nature, so the OWASP CRS usually produces a rather high rate of false positives, which in some cases may be close to 40% [6]. 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 (1 of 3 valid requests gets blocked), so rule tuning is required. Rule tuning is a time consuming and error prone task that has to be manually carried out for each specific web application. In traditional network firewalls and IDS, the approach based on rules has been successfully complemented with machine learning-based and anomaly detection tools and other statistical approaches 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. A primary objective of our work is to improve MODSECURITY with such techniques.

Our contributions: We apply machine learning techniques to improve the detection capabilities of the WAF MODSECURITY giving particular importance to the task of diminishing the *false positives* generated by this tool when it is set up to protect a web application without reducing the *true positive rate*. We provide a characterization of the problem by identifying different scenarios depending on the availability of data to train the learning models. We have defined a learning framework that allows us to experiment with different learning algorithms depending on the corresponding

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).

978-1-5386-6805-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICMLA.2018.00174

scenario. The experiments have been carried out over three different datasets. The obtained results outperform, in most of the cases, the execution of MODSECURITY configured with the OWASP Core Rule Set (OWASP CRS) out of the box.

The structure of the rest of the paper is as follows: Section II provides a primer on web application security and protection mechanisms. In Section III we present the learning framework we have used to carry out the experiments. The outcomes are described and discussed in Section IV. Related work is discussed in Section V. Further work and conclusions are presented in Section VI.

II. BACKGROUND

This section provides a concise primer on web application security and briefly discusses protection mechanisms.

A. Web application security

The most critical web application security risk defined by the OWASP Top 10 2017 [5] is *Injection*, which directly relates to input validation. Injection occurs when data sent by the user to the application is used as part of an instruction that an interpreter executes in the backend producing unexpected (by the designer of the application) results. There are different types of injections depending on the interpreter being exploited: SQL, LDAP, XPath, NoSQL queries, OS commands, XML parsers, SMTP headers, and others. Injection has been consistently occupied one of the first three security risk since the first publication of the OWASP Top 10 (year 2007). Nevertheless applications still suffer from this type of vulnerability. Thus, input validation is critical for software security. One such validation consist of verifying and filtering all data that flow to the system before they are effectively used. These procedures usually are not introduced at the design stage of applications leaving then unattended vulnerabilities that might be critical, specially to web applications. When data is processed without proper validation an attacker could lead the system to unpredictable states and exploit this for his own benefit. This data may also produce unexpected results that could be analyzed by the attacker to infer further information to proceed with his activity.

B. Protecting Web Application

The principle of *in-depth security* means that security mechanisms are layered around the system being protected so if an attack bypass one mechanism there exist other mechanisms to provide the needed security. When concerned with the security of a web application, the outer layer to be protected is the organization's network perimeter. Typically, network firewalls are deployed to protect in/out traffic. As to the network itself, it is common practice to use an Intrusion Prevention System (IPS), for instance, to analyze network traffic in order to detect potential threats. Additionally, it has become a good security practice to deploy a Web Application Firewall (WAF) to analyze the request/response flow through the communication channel to identify attacks that exploit vulnerabilities proper of web applications. In the general case, both traditional firewalls and IPS inspect traffic at the network layer. The research we present in this work is focused on WAF, which are the specific tools that are currently being widely deploy to prevent web application attacks. In contrast to traditional network firewalls, WAFs are designed to perform packet inspection at the application layer. Secure communications using SSL and TLS are done by the WAF so our work focus on the HTTP protocol analysis.

A WAF often supports different security model configurations: normally it makes it possible to enforce both positive and negative security models. A positive security model only allows to pass known good traffic, all other traffic is blocked. A negative one allows to pass all traffic except what is known to be malicious. MODSECURITY is a *de facto* standard open source technology. Large organizations like Verizon [7] are currently using this technology to protect large amount of applications. This WAF supports two working modes: detection and prevention. In the first mode, logs are generated for every potential attack detected. Normally this mode is used when adding new rules and monitoring for false positives. The second mode is when the WAF is really useful: by correctly configuring rules it is capable of blocking potential malicious Web traffic to and from applications. The core of MODSECURITY implements a flexible rule engine [8]. These rules can be applied in every application request/response.

To detect and prevent the exploitation of well-known and common vulnerabilites the Open Web Application Security Project [1] (OWASP) has defined a generic rule set that is known as the OWASP Core Rule Set [4] (OWASP CRS). The OWASP CRS is widely deployed in large organizations as Akamai, Azure, CloudFare, Fastly and Verizon. The goal of the OWASP CRS is to provide a set of generic attack detection rules that when fed to MODSECURITY provide a base level of protection for any web application. The OWASP CRS implements a negative model, where the rules are designed to detect known attacks patterns.

The last Gartner's Magic Quadrant for Web Application Firewalls [9], reviews and ranks several proprietary WAF, among others Akamai, F5 and Imperva. In that report it is remarked the rare and still unproven use of machine learning techniques to leverage the detection capabilities of those technologies.

III. ADAPTIVE APPLICATION SECURITY

MODSECURITY is usually configured to work using a negative model because defining the rules that describe normal behavior of the application in real life is an almost impossible task. The problem with that operational mode is the high number of false positives that the WAF generates and therefore the amount of tuning that is needed during the learning phase. One of the main objectives of the research reported here is to make MODSECURITY behavior more flexible by using machine learning techniques to better adapt its (defensive) behavior to that of the application that is protecting. Additionally, depending on the operational context of the application to protect one may consider alternative learning scenarios. The ideal situation of having available a labeled dataset of

application requests that represent valid and attack behavior of a specific application is not always possible so we have investigated different scenarios that we now proceed to discuss.

A. Learning Scenarios

The first scenario ($sc1$) corresponds to the ideal situation where real application traffic is available which has been tagged discriminating normal (valid) requests from attacks. In the scenario $sc2$ real valid traffic (obtained from valid requests to the application) is available and the requests classified as attacks are a set of requests known to be malicious but not specifically for the application to be protected (the requests could have been collected using a Honeypot [10], for example). Since tagged valid and attack requests are available in both cases it is possible to apply supervised multi-class techniques. In what follows we are going to present two different approaches of supervised multi-class techniques and the results obtained on $sc1$ and $sc2$.

In the scenario $sc3$ only valid requests are known, no requests tagged as attacks are available. We believe that this is a quite realistic approach, where valid traffic could be collected, for instance, from the result of performing functional testing of the application. We have pursued a one-class classification approach [11] to handle this scenario.

B. Datasets

The classification experiments have been performed on three different datasets containing labeled data. In addition to that, each logged request includes both its header and its body. As far as we know, there exists only three public datasets that comply to these requirements: 1998 DARPA Intrusion Detection Evaluation Data Set, PKDD2007 Challenge [12] and CSIC2010 [13]. The first dataset was discarded since it was generated recording network traffic, not only web application traffic. The other two datasets are quite old (given the evolution of web applications) and have been artificially crafted. In order to validate our approach with recent and real life requests, we generated an additional dataset, called here DRUPAL, based on the public Website of our School. The three datasets are briefly described in what follows.

a) PKDD2007*:* In 2007 the 18th European Conference on Machine Learning (ECML) and the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), put forward a challenge on Analyzing Web Traffic. As part of the challenge it was provided a dataset which contained valid traffic and requests classified in seven different types of attacks. The dataset contains 35.006 requests classified as normal and 15.110 requests classified as attacks. The PKDD2007 dataset was generated by recording real traffic which was then processed to sanitize the information. This masking process consisted in renaming every url, parameters names and values with randomly generated strings.

b) CSIC2010*:* The Spanish Research National Council (CSIC) developed in 2010 a dataset to test web application attack protection systems. The dataset is tagged in Normal Traffic and Anomalous Traffic. It was developed automatically

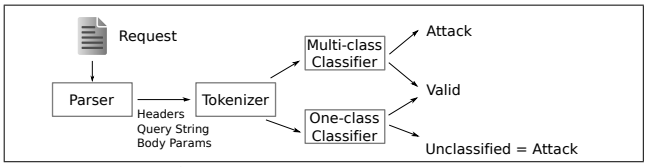

Figure 1: Learning architecture

based on real request to an e-commerce application. The dataset contains 36000 valid requests for training, other 36000 request for testing and 25000 requests of anomalous traffic. To generate the anomalous traffic there were used tools such as Paros (which later became OWASP Zed Attack Proxy (ZAP) [14]) and w3af [15]. In addition to that, some valid requests were modified with typo errors in parameters. Unfortunately, we do not know how real attacks and anomalous traffic distribute in this dataset. However, this dataset seems to be a reference in the MODSECURITY community, as may be understood from a note written by Christian Folini[1] who is the project leader of the OWASP CRS project.

c) DRUPAL*:* In order to experiment with a real life application, based on real requests and real attacks, we crafted a dataset by capturing six day incoming traffic to the public Website of our School. The only post-processing of this dataset consisted in blurring password values in the request.

This dataset was divided into two parts, the first three days were used to create a training and testing datasets and the last three days were used to create a validation dataset. This gives a validation dataset completely separated from the train/test datasets in order to validate our results using real traffic without any modifications. Since the requests are obtained from real traffic they are less balanced: 65600 and 41519 valid request and 1287 and 2226 real attacks for the train/test and validation datasets respectively.

The web site of the School is protected by an instance of MODSECURITY featuring the OWASP CRS, which has been tuned for several years by a team of security and infrastructure experts. We therefore used MODSECURITY as the labeling tool: those requests that were accepted by MODSECURITY were considered as valid traffic, while those requests that MODSECURITY rejected were tagged as attacks.

C. The learning architecture

We now briefly review the components of the proposed learning architecture depicted in Figure 1.

In this specific problem the samples are instances of the semi-structured text-based protocol HTTP *Request*, which is used to exchange information between the client and the server. The information is encoded into binary format. We have built a parser to decode the information which is in HTTP URL encoded [16] format so the learning algorithm can work with the real information. On the other hand, the information contained in headers that are specific to the request, and therefore should not be considered to infer application behavior, is

[1]https://github.com/SpiderLabs/owasp-modsecurity-crs/issues/1016#issuecomment-366602493

Table I
SELECTED FEATURES BY THE SECURITY SPECIALIST

<	../	alert	exec	password
<>	’	alter	from	path/child
<!–	“	and	href	script
=	(	bash_history	#include	select
>	)	between	insert	shell
—	$	/c	into	table
——	*	cmd	javascript:	union
-	*/	cn=	mail=	upper
–>	&	commit	objectclass	url=
;	+	count	onmouseover	User-Agent:
:	%00	-craw	or	where
/	%0a	document.cookie	order	winnt
/*	Accept:	etc/passwd	passwd	

filtered. Examples of these cases are the value of a cookie or the timestamp of the last time the web page was modified.

Several attacks make use of specially crafted input to make the server to execute an unexpected functionality. Most of these inputs use as part of the payload special characters, for instance . , ; < >, which are normally used in information retrieval to split the documents. In order to preserve those characters the parser analyzes the parameters from the query string, the body and headers and split them in name and value pair as different text string.

The tokenization process adopted was highly dependent of the learning scenario. In $sc1$ we followed the classic approach of *bag of words*. In this case, the tokenizer performs the splitting using only spaces preserving special characters that may be included in parameters values or headers. To preserve the special characters used in attacks, this tokenizer uses the following character to split the request: ␣\r \t \n. In the case of $sc2$ and $sc3$, the use of features that were blindly generated from the dataset, as in the case of $sc1$, did not work. The main problem was the large amount and sparsity of the extracted features, because few of them were active in each instance. In this feature space it was difficult to distinguish attacks from valid requests. To address this difficulty, we incorporated the experience of a security expert into the analysis. In this *expert-assisted* approach, a set of features that better characterize different web application attacks were defined. Validation of the proposed features was performed applying an information gain [17] algorithm on the three datasets. The results showed that all features have a positive information gain in at least one of the datasets. The CSIC2010, PKDD2007 and DRUPAL dataset have 43, 38 and 59 (respectively) features out of 64 with a positive information gain. In this case, the tokenizer counts for each request the number of appearances for each feature. Table I lists the defined features.

The last component of the learning architecture is the classifier. For the scenarios $sc1$ and $sc2$ we have followed a multi-class supervised approach. For the scenario $sc3$ we applied a one class classification approach.

The supervised approach requires a training set with labeled samples as valid or attack. We have carried out two variants of supervised multi-class learning, that we proceed to describe.

1) $sc1$ - multi-class information retrieval: After tokenization each request is transformed into a vector applying the classic Term Frequency Inverse Document Frequency (TF-IDF) [18] scheme in order to calculate the corresponding weights of each of the terms in the request. In this case, features correspond to terms in our vocabulary, which are generated as a result of the tokenization process. To reduce the amount of features, feature selection is performed using the information gain algorithm [19], keeping all features that have an information gain grater than 0. Finally, we have trained different classifiers using Weka: Support Vector Machine [20](SVM), K-nearest neighbours [21](K-NN) and Random Forest [22].

SVM is used to replicate and enhance the results reported by Gallagher et all [23]. For the experiments we used a polynomial kernel and trained the SVM using sequential minimal optimization (SMO). We include K-NN as its simplicity allows to categorize the complexity of the learning problem. Finally, we used Random Forest as it could be seen as a rule set generator (comparable to the OWASP CRS) and it has been reported to produce good results in problems related to fraud detection [24].

2) $sc2$ - multi-class expert-assisted: In this approach, the tokenization was performed using the features defined by the expert (see Table I). As was mentioned before, the resulting vector after tokenization contains in each position the number of occurrences of the feature in the requests. As in the classic information retrieval approach, after tokenization we have trained the K-NN and the Random Forest classifiers. One important difference with the multi-class information retrieval approach is that the tokenization in this case does not apply the TF-IDF scheme in order to encode the value of a feature depending on the frequency on the document and in the corpus. Without this encoding, these features are not suitable for the SVM classifier. For this reason no results using SVM are reported.

3) $sc3$ - anomaly detection expert-assisted: Scenario $sc3$ uses only requests that belong to the valid class, that we called the *target class*. We have investigated a one-class classification approach [11] where there are available instances of one class and none or very few samples of the other one. The proposed anomaly detection classifiers organizes samples of the target class into clusters and then uses the distance to these clusters as a measure of anomaly; samples away from the clusters are classified as anomalies.

Using the Expectation Maximization (EM) algorithm [25] we cluster the training set containing only target samples. The EM algorithm was used to estimate the parameters of a Gaussian Mixture Model (GMM) and the number of clusters (components in the GMM). In our case, each component of the GMM constitutes a cluster that captures the distribution of the target class. To capture the intra-cluster variability we use the distance between samples in the cluster and its centroid. The distance between a given x and the cluster *C* is computed using the Mahalanobis distance [26]:

$$dist(x, C) = \sqrt{(x-\mu)\Sigma^{-1}(x-\mu)} \quad (1)$$

where Σ represents the full covariance matrix calculated during the training phase. If one of the features is not seen during the training phase, the corresponding dimension will have a standard deviation of 0, and the Mahalanobis distance

cannot be calculated. For this reason, we adjust the covariance matrix by adding a regularization term to it $\epsilon * Id$, where ϵ is the smallest standard deviation in diag(Σ) different from 0 and Id is the identity matrix.

If the distance of a sample to a given component is within the observed intra-cluster variability during training, the sample will be labeled as valid. To apply this idea we need a distance threshold for each cluster. Therefore, for each cluster, we obtain the corresponding mean distance (μ_d) and the standard deviation of the distances (*std*) of the samples assigned to it. On this basis, the threshold is defined as shown in Eq 2.

$$t = \lambda[\mu_d + 10 * std], \lambda \in (0, 1] \quad (2)$$

The parameter λ allows us to change the size of the cluster from 0 (where only instances that correspond to the centroid of the cluster are classified) to 1 where almost all requests are classified as the target class (the factor 10 was empirically calculated in order to enforce this).

During classification we calculate the Mahalanobis distance of the requests to the clusters: if the distance is equal or less than the threshold the request is assigned to the cluster. Requests that are not assigned to any cluster are classified as attacks. The threshold of each cluster then might be defined by setting the number of false positives that we are willing to accept. This can be done observing the distribution of intra-cluster distances. All points above the threshold will constitute false positives. In future work we plan to model this distribution in order to obtain an estimation of the false positive rate given the selected threshold.

IV. Results

This section is devoted to present and discuss the outcomes of the experiments that have been carried out. The results will be presented in terms of Precision, Recall, True Positive Rate (TPR) and False Positive Rate (FPR). In our case, TPR and FPR indicate the ratio of requests correctly and incorrectly classified as attacks, respectively.

A. Baseline

One of the main objectives of this work is to reduce the amount of false positives generated by ModSecurity without decreasing the TPR. The entry *Baseline* of Table II presents the results of ModSecurity configured with the OWASP CRS out of the box for each dataset. The baseline was generated with an Apache HTTP Server 2.4, configured with ModSecurity 2.7 and the OWASP CRS 2.2.9 in collaborative detection mode with the standard configuration (no tuning of the rules were made).

B. Scenario 1

We have trained different multi-class classifiers after tokenization of the requests using a bag of words approach. Given that real valid and attack requests are available we have performed 10-fold cross validation to train and test the SVM, KNN (with K = 3) and Random Forest classifiers. The results are summarized in Table II in the section corresponding to $sc1$. It can be noticed that in all datasets the results show good performance in terms of precision, recall, TPR and FPR for all the classifiers evaluated. In particular, we can say that in an overall analysis, the Random Forest classifier has better performance. The main conclusion from these results is that the application of the multi-class approach in this case is feasible and with good performance scores. However, this approach has two important limitations. On the one hand, labeled data from both classes are needed in order to train the classifiers (see discussion in Section III-A). On the other hand, the classifier designed for one dataset cannot be used in other ones. Even using the same features we could not obtain good performance scores when applying, for instance, the model trained for PKDD2007 and then tested using DRUPAL. In other words, the models do not seem to generalize.

C. Scenario 2

Given the lack of datasets, to emulate this scenario we have split the PKDD2007 and DRUPAL datasets at 50% and use half for training and half for testing. In order to generate the training datasets, we use the valid requests from the dataset we are going to test and the attacks requests from the other two datasets (emulating to be generic attacks). The CSIC2010 dataset is already split into testing and training, but the attacks contained in the dataset not only include attacks but also valid requests with typos. For this reason, we did not use attack requests from the CSIC2010 dataset in the experiments of this scenario. After generating the datasets, we have applied the expert-assisted multi-class approach. We proceeded to experiment with the KNN (with K = 3) and Random Forest classifiers. In this case, instead of using 10-fold cross validation, we used the training and testing datasets generated in the previous step. In Table II, entry $sc2$, we present the results of applying this technique on the three datasets. As can be noticed, the performance in terms of TPR decreased comparing to the results of $sc1$. In case we compare the results with our baseline, we can notice that in general even if there exist decrements in the number of attacks detected (decrease of TPR), the FPR has a major increment. We also validated that in the case that more generic attacks are added to the training datasets the attack detection increased without increasing the false positives. As mentioned in section III-A in order to improve the results is required that the set of generic requests (in this scenario the attack requests) characterize as much as possible the universe they represent. Once again, we can say that the Random Forest is the classifier with better performance analyzing the overall results.

D. Scenario 3

The evaluation of this approach was performed, on each of the datasets, using 70% of the valid requests for training and the rest of the dataset (30% of valid and 100% of attack) for testing. The experiments include varying the λ value from 0 to 1 in order to show 500 different operational points. In entry $sc3$ of table II we present the results for the special case of λ=0.156. This value of λ showed to be the one in which

Table II
SUMMARIZED RESULTS OF THE MODSECURITY BASELINE AND THE THREE SCENARIOS PRESENTED

Scenario	Dataset	Algorithm	Precision	Recall	TPR	FPR
Baseline	DRUPAL	MODSECURITY W/OWASP CRS	0.11	0.76	76.22%	38.89%
	PKDD2007	MODSECURITY W/OWASP CRS	0.70	0.93	92.97%	57.21%
	CSIC2010	MODSECURITY W/OWASP CRS	0.50	0.34	34.32%	23.93%
*sc*1	DRUPAL	Random Forest	0.97	0.91	91.22%	0.05%
		KNN-3	0.97	0.89	88.97%	0.05%
		SVM	0.98	0.90	89.67%	0.04%
	PKDD2007	Random Forest	0.96	0.85	85.10%	1.67%
		KNN-3	0.96	0.70	70.39%	1.41%
		SVM	0.95	0.81	81.10%	1.91%
	CSIC2010	Random Forest	0.97	0.72	72.00%	1.47%
		KNN-3	0.95	0.65	65.03%	2.38%
		SVM	0.97	0.70	70.34%	1.26%
*sc*2	DRUPAL	Random Forest	0.95	0.48	47.57%	0.05%
		K-NN 3	0.90	0.07	6.99%	0.01%
	PKDD2007	Random Forest	0.96	0.23	22.56%	0.45%
		K-NN 3	1.00	0.12	11.70%	0.02%
	CSIC2010	Random Forest	0.91	0.32	32.06%	2.27%
		K-NN 3	0.66	0.50	50.43%	17.85%
*sc*3	DRUPAL	One-class w/λ=0.156 (test)	0.22	0.95	95.34%	22.15%
		One-class w/λ=0.156 (validation)	0.15	0.89	88.90%	26.83%
	PKDD2007	One-class w/λ=0.156 (test)	0.70	0.93	93.12%	58.30%
	CSIC2010	One-class w/λ=0.156 (test)	0.75	0.48	47.66%	11.15%

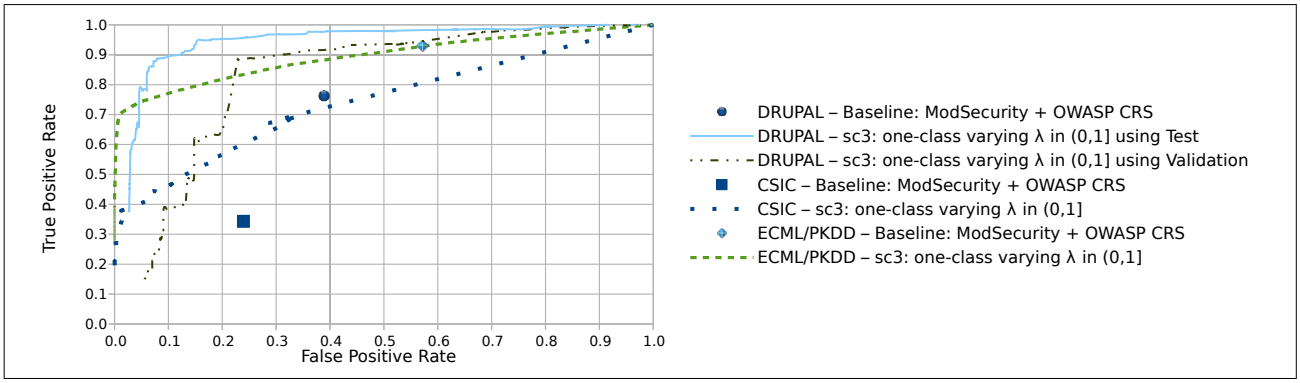

Figure 2: *sc*3: one-class ROC curve varying the λ value

for each dataset, the model behaves equal or better than the baseline.

In Figure 2 we present the results, in terms of a ROC curve, using the DRUPAL, PKDD2007 and CSIC2010 dataset, respectively. The line in the graph represents the different operation points of the one-class model. The dot represents the performance of our baseline MODSECURITY. In the three experiments there exist operational points where the one-class model behaves equal or better than the MODSECURITY baseline. For the case of the DRUPAL dataset, besides using the testing dataset, we also validate our model using the validation dataset. If we select the operational point with the same TPR than the baseline, the FPR will decrease from around 38% to 4% or 22% depending on the test or validation dataset used respectively. Using the operational point with the selected λ, the model tested on the test and validation dataset behaves improves the baseline in terms of TPR and FPR. In the case of the experiments on the PKDD2007 dataset, we can notice that our approach does not improve the baseline, but we have at least one operational point that behaves as good as MODSECURITY. As to the CSIC2010 dataset, like in the case of DRUPAL, we can observe several operational points that outperform MODSECURITY baseline. If we maintain the same TPR as MODSECURITY, the FPR decreases from 24% to 1%.

E. Discussion

As already pointed out, false positives of MODSECURITY might lead to a denial of service to valid users. We have presented three approaches for addressing the problem of attack detection in web applications.

The first approach, in which the multi-class paradigm is used, resulted in very good performance scores (see Section IV). The three tested classifiers provided similar results. The fact that K-NN provided good results, very close to the ones of Random Forest and SVM, indicates that if real samples of valid requests and attacks are available, the classification problem is not very complex. That is, the Bayes error (the theoretical lowest error attainable knowing the class distributions) is not large. However, this approach has two limitations: labeled data from both classes are needed in order to train the classifiers and a classifier designed for one dataset can not be used with another one.

The second approach is also based on a multi-class paradigm, but with attack traffic collected from attacks to

other applications. This approach has the advantage that only real valid traffic from the web application is needed for training. Although a low rate of false positive was obtained, the performance on attack detection decreased. We are convinced that this behavior relates to the complexity of constructing the generic attack dataset from the datasets that we had at hand. Random Forest produces more false negatives and a similar number of true negatives compared to scenario one. We believe that this indicates that the attack samples do not cover the whole universe of possible attacks and therefore the classification boundaries do not capture the optimal solution as can be done in scenario one. In other words, for this approach to work we need an attack training dataset that covers all possible attacks to the given application.

Finally, the third approach, uses only valid requests to construct a detection model. In this case, the outcome is quite promising, since the results outperform the ones obtained with the classic rule based MODSECURITY solution. As we have seen in Section IV-D this one-class approach reduced the number of false positives while not decreasing the true positives. Furthermore, this approach has a threshold that can be tuned depending on the number of false positives that we are willing to accept (see Figure 2). If we take the number of false positives produced by MODSECURITY as a reference, we can see that the proposed solution clearly increases the number of true positives for two of the datasets (CSIC2010 and DRUPAL) and keeps the third one without modification (PKDD2007). Therefore, if we keep the same number of false positives of MODSECURITY, we reduce the number of undetected attacks. On the other hand, if we consider the number of true positives generated by MODSECURITY as a reference, in two of the datasets we reduce the number of false positives (CSIC2010 and DRUPAL) and in the third one there are no improvements (PKDD2007).

V. RELATED WORK

In the ECML/PKDD2007 challenge [12], the main objective was to classify web application requests using a multi-class approach. Two solutions ([27], [28]) were presented. In the solution reported by Pachopoulos et al [28], the authors identify a set of tokens that describe attack patterns and transform the requests into a feature vector of true/false, where each position indicates if the token is present or not. Then they train a C4 classifier using this information. In our supervised multi-class approach, we identify basic tokens that describe attack patterns and we count the number of appearances. Then we train different classifiers using these feature vectors.

Gallagher et al [23] analyzed the results of the ECML/PKDD2007 challenge and introduce a supervised multi-class classification of web attacks by applying classic techniques of information retrieval. In our work, the multi-class information retrieval approach is a modified version of this approach to tackle the $sc1$ by adding a pre-processing phase and training using different classifiers. The main conclusion of our experiments is that we agree with Gallagher's et al regarding the precision of the approach. However, we have studied this approach in depth from a real application perspective and conclude that the classifiers obtained in this way do not generalize. This supports the idea that even if the performance scores are good the $sc1$ is more a lab scenario than a real life application.

In [29], Raïssi et al conclude, based on the results of the ECML/PKDD2007 challenge and a feedback survey written by the challengers, that using machine learning techniques to detect web application attacks requires to involve the security experts earlier in the knowledge discovery process. In our approaches, the feature selection phase uses the knowledge of the security expert to identify the tokens to be considered in the model construction. In this way we were able to obtain classifiers with generalization capabilities.

In [30], [31] Kar et al present a solution to detect SQL Injection attacks by modeling SQL queries to train a Support Vector Machine (SVM). This supervised multi-class approach focuses in detecting SQL Injection attacks through the analysis of the traffic between the application and the database. In our supervised multi-class approach we analyze the traffic between the user and the application focusing on different types of Injection attacks.

Kruegel and Vigna in [32], propose an anomaly detection approach where they model specific characteristics of the URL parameters. They focus on parameter length, order and even generate a probabilistic grammar of each parameter. In our one-class approach we work using the whole requests, not only the URL parameters, capturing the normal behavior by modeling the occurrence of a specific set of tokens defined by a security specialist. This allows us to capture the behavior of the data sent in the normal use of the application, focusing particularly on the behavior of special tokens that are highly related to attacks patterns. Since the dataset used not only have attacks in the URL parameters, but also in the body and headers, these approach can not be compared.

Several authors propose anomaly detection techniques that work over simplification of the application's parameter values. In [33], Corona et al. abstract away numbers and alphanumeric sequences, representing each category with a single symbol. Torrano, Perez and Marañón [34] present an anomaly detection technique where instead of using the tokens themselves, they use a simplification that only considers the frequencies of three sets of symbols: characters, numbers and special symbol. As was mentioned, in our one-class approach we analyze not only the parameters values but the whole request without any further simplification.

VI. CONCLUSION AND FURTHER WORK

We showed that machine learning techniques can improve the detection capabilities of MODSECURITY in terms of the reduction of *false positives* and the increment of *true positives*. We also provided a characterization of the problem by identifying different scenarios depending on the availability of training data. The scenarios vary from the rare, but best case, where we have a dataset with real application traffic to more practical scenarios where we have only valid requests to an application that could be collected, for instance, during the functional testing phase.

If we consider the results of this work, we can conclude that the techniques we have applied can improve the performance of MODSECURITY. Although the task of writing rules to detect attacks may be complex for a human expert, the results of $sc1$ show that machine learning algorithms can easily learn the decision boundary from training data. The results of $sc2$ complement the previous one and show the importance of having representative attack training samples. Although a low rate of false positives was obtained, the performance on attack detection decreased. In future work we will address this issue using synthetic attacks generated with automatic tools.

If only valid request are at hand, the results of the last scenario ($sc3$) show that a one-class solution provides many operational points that outperform MODSECURITY. In future work we will study the automatic selection of the operational point using either sampled attacks, as in $sc2$, or synthetic attacks. We believe that the results of the three scenarios here presented show the potential of machine learning for the construction of a WAF. We plan to experiment with one-class algorithms like SVM, instead of using classic distances, and to study the special one-class scenario where we only would have available attacks requests.

One of the biggest challenges we faced was the lack of publicly available labeled datasets with complete HTTP requests. We were able to find only three datasets, two of them were used in our experiments (see section III-B) but the third one (1998 DARPA Intrusion Detection Evaluation Data Set) was discarded since it was constructed based on network traffic, not only web application traffic. Additionally, these datasets are at least 10 years old. We think those datasets no longer represent the current state of the involved technologies. We plan to continue working on the construction of new datasets.

REFERENCES

[1] OWASP, "Open web application security project." [Online]. Available: https://www.owasp.org

[2] A. J. Hacker and I. CISSP, "Importance of web application firewall technology for protecting web-based resources," *ICSA Labs an Independent Verizon Business*, 2008.

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

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

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

[6] 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/

[7] Verizon. Podcast: Advantages of web application firewalls and open-source waf. [Online]. Available: https://www.verizondigitalmedia.com/blog/2017/11/advantages-of-web-application-firewalls-and-open-source-waf/

[8] SpiderLabs/ModSecurity, "Reference manual." [Online]. Available: https://github.com/spiderlabs/modsecurity/wiki/reference-manual

[9] M. Quadrant, "Magic quadrant for web application firewalls," *Analyst (s)*, p. G00314552, 2017.

[10] Owasp wasc distributed web honeypots project. [Online]. Available: https://www.owasp.org/index.php/OWASP_WASC_Distributed_Web_Honeypots_Project

[11] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," in *Irish conference on Artificial Intelligence and Cognitive Science*. Springer, 2009, pp. 188–197.

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

[13] Http dataset csic 2010. [Online]. Available: http://www.isi.csic.es/dataset/

[14] OWASP. OWASP Zed Attack Proxy (ZAP). [Online]. Available: https://www.owasp.org

[15] A. Riancho, "w3af-web application attack and audit framework," *World Wide Web electronic publication*, p. 21, 2011.

[16] Html url encoding reference. [Online]. Available: https://www.w3schools.com/tags/ref_urlencode.asp

[17] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[18] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.

[19] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, 1997, pp. 412–420.

[20] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998. [Online]. Available: http://research.microsoft.com/\~jplatt/smo.html

[21] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] 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.

[24] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[25] 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

[26] P. C. Mahalanobis, "On the generalized distance in statistics." National Institute of Science of India, 1936.

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

[28] 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.

[29] 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.

[30] D. Kar, S. Panigrahi, and S. Sundararajan, "Sqlidds: SQL injection detection using document similarity measure," *Journal of Computer Security*, vol. 24, no. 4, pp. 507–539, 2016.

[31] ——, "Sqligot: Detecting SQL injection attacks using graph of tokens and SVM," *Computers & Security*, vol. 60, pp. 206–225, 2016.

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

[33] 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.

[34] 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.