Enhancing web application attack detection using machine learning

Rodrigo Martínez
Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Uruguay
Email: rodmart@fing.edu.uy

***Abstract*—The exploit of vulnerabilities present in Web applications has been the main attack vector in the last decade biggest data breaches. In this work we put forward a framework to leverage the performance of Web Application Firewalls (WAFs) using machine learning techniques. We propose the use of two types of machine learning models: a multi-class approach for the scenario when valid and attack data is available and alternatively a one-class model when only valid data is at hand. The use of both models to predict potential malicious traffic has shown to outperform MODSECURITY, a widely deployed WAF technology, configured with the OWASP Core Rule Set out of the box. We also present a prototype that integrates the one-class model with MODSECURITY.**

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

I. INTRODUCTION

The ever increasing advance in communications and globalization forces organization to take fast decisions, in particular in the way they make available new services through the Internet. Due to the potential of web applications in supporting this new business rush the use of web technologies has spread significantly, usually inadvertently making also available paths to access critical or sensible data.

In the recent years several data breaches have seriously affected many companies inflicting not only an economic loss but also causing damages to their image and reputation. Some examples of the most popular cases are the Equifax and Adult Friend Finder data breaches. In both cases the attack vector essentially consisted in the exploitation of web applications vulnerabilities. According to Verizon [1] this was the case in almost 20% of the last 10 years most significant data breaches. Within this category, the use of stolen credentials is the first vector of attack followed by SQL Injection.

New regulations about privacy and personal information force organizations to improve data protection. Following an in-depth security approach to protect web applications, several layers of security activities and controls need to be implemented in order to mitigate this type of threats. One tool that is typically used as a first line of defense for web applications is the Web Application Firewall (WAF). A WAF is a piece of software that intercepts and inspects all the traffic between users and web servers, 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 [2], 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) [3], for handling the most usual vulnerabilities included in [4]. 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% [5]. 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.

The structure of the rest of the paper is as follows: Section II provides a primer on web application security and protection. In Section III we present the learning framework we have conceived and developed to integrate machine learning techniques to MODSECURITY. Further and current work is described and discussed in Section IV. Conclusions are presented in Section V.

II. WEB APPLICATION SECURITY

OWASP [6] is a worldwide not-for-profit organization focused on improving software security. One of the OWASP flagship projects is the Top 10 most critical risk in web application security [4]. Since its first publication in 2007, *Injection* has been in the top 3 positions. 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, among others. A SQL injection takes place when an attacker sends SQL code as input to the application and it results in the execution of a SQL sentence that behaves as not expected by the developer.

Even if the research and industrial communities have identified injection as a critical issue already 15 years ago, 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.

The research we present in this work is focused on WAF, which are the specific tools that are currently being widely deployed to prevent web application attacks. In contrast to traditional network firewalls and Intrusion Prevention System (IPS), WAFs are designed to perform packet inspection at the HTTP layer analyzing the request/response flow through the communication channel to identify attacks that exploit vulnerabilities proper of web applications.

A WAF usually 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. It is open source, flexible and extensible. It has 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 OWASP has defined a generic rule set that is known as the OWASP Core Rule Set [3] (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. LEVERAGING MODSECURITY AND THE OWASP CRS WITH MACHINE LEARNING TECHNIQUES

When fed with the rules of the OWASP CRS the WAF MODSECURITY implements a negative model: the rules are designed to identify attacks. This approach has two major disadvantages: i) the application of a static and generic set of rules usually generates a high amount of *False Positives*, and ii) only the known attacks characterized by the rules are the ones detected. One of the main objectives of our research is to enhance with machine supported learned knowledge this rule-based decision process. For doing that we have conceived a framework whose architecture is depicted in Figure 1. The main idea is to combine the flexibility provided by classification procedures obtained from the definition of machine learning models with the hard-coded knowledge embedded in the specification of the rules. One important challenge is to be able to provide the possibility to integrate user defined learning models with the rule decision engine of MODSECURITY. Thus, the framework embodies a model development environment and a classifier module called the Web Attack Classification Engine (WACE).

The core of the model development environment is a set of tools that implement several machine learning algorithms, preprocessing and training tasks. In order to maximize the expert's knowledge the framework allows the expert to work using his preferred tool and then exporting the resulting model into PMML [10]. PMML is an XML-style language that has been conceived and designed to describe trained models. The model described in PMML is then fed to WACE in order to classify new requests. The tool provides support for carrying out a statistical pattern recognition approach, like the one presented in [11], where the experts design the models based on two steps: first a model is trained using testing data and then new instances are processed with the help of the trained model.

The module WACE, on the other side, embodies two major components: i) an API that encapsulates all the interactions of this module with MODSECURITY. It provides a set of services to be used by the client, namely information available of the models, classification of a HTTP requests/response and module configuration, and ii) a model evaluator, which is a component that allows to load and process different machine learning models described using PMML. It is composed of the PMML processor and the evaluator it self. The PMML processor allows to import an already trained model described in the PMML language. After the model is loaded, the evaluator can extract the features, pre-process and classify an HTTP

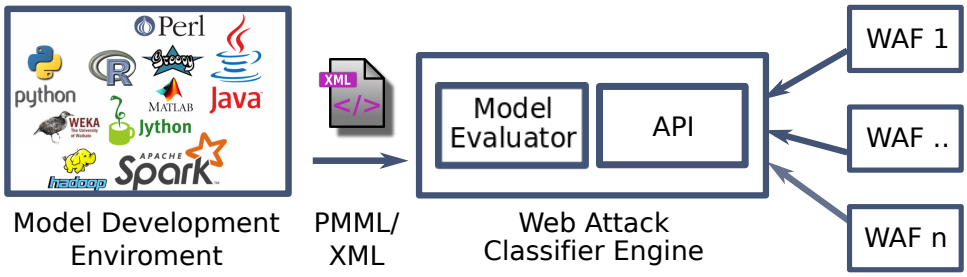

Figure 1. Framework for machine learning leveraged attack detection: high level architecture

requests/response using the defined model. The evaluator will provide the implementation of common machine learning algorithms and it will allow to be extended with custom implementations.

A. Machine Learning Models

In order to validate the chosen approach, the first step was to define and train different models in order to understand the impact that machine learning could have when classifying web applications attacks. We have followed two different approaches: a multi-class approach for the scenario when valid and attack data is available and a one-class solution when only valid data is at hand.

The multi-class approach is an extension of the work presented by Gallagher et al [12]. They analyzed the results of the ECML/PKDD2007 challenge [13] and introduced a supervised multi-class classification of web attacks by applying classic information retrieval techniques. They process the dataset as a corpus of documents and extract features by splitting the request into tokens and then train a vector space model. In our work, we add a pre-processing phase using the knowledge of the HTTP structure to improve the feature extraction. And we train using different classifiers, we include K-nearest neighbours [14](K-NN) as its simplicity allows to categorize the complexity of the learning problem and the Random Forest [15] 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 [16]. 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, by training the model using generic requests and then test it on each application dataset, and conclude that the classifiers obtained in this way do not generalize. This means that a model that has been trained for one application can not be used to protect a different one.

In scenarios where we only have requests that belong to one of the classes, valid or attack, we have investigated a one-class classification approach [17] where there are available instances of one class and none or very few samples of the other one. In [18] we present the one-class approach when only valid requests are available. In this approach we process the requests by counting the amount of times that a specific feature appears. The features that better characterize different web application attacks were defined making use of a security expert's knowledge. Then we measure the distance of the requests to the clusters defined during the learning phase. One major feature of this approach is the threshold that allows to change the classification tolerance. Each possible value of this threshold is a model operational point, allowing the expert to change the attack detection or false positive rate by only changing this value. We present several operational points in which the one-class approach outperforms MODSECURITY configured with the OWASP CRS. We also present one way of integrating this approach with the OWASP CRS.

B. Prototyping the Web Attack Classification Engine

We have developed a proof of concept of the WACE discussed above. It has been implemented using the LUA extensions provided by MODSECURITY. The machine learning model evaluator is implemented in Java.

The prototype allows us to test the whole process: from adding a rule in MODSECURITY to invoke the LUA ML module, which in turns invoke the Java one-class evaluator, to getting the result back to MODSECURITY in order to take an action. The prototype implementation and the Java one-class evaluator are available at [19].

IV. WORK IN PROGRESS

This section discusses current and further work.

Machine Learning Models: Concerning the one-class approach model we shall study the automatic selection of the optimal operational point using either sampled or synthetic attacks. We plan to experiment with one-class algorithms like SVM, instead of using classic distances, and to study the special scenario where we only would have available attacks requests for training.

We also plan to continue working on the construction of new datasets in order to produce new examples of attacks and valid application traffic that represents nowadays applications. We intend to use these new datasets to train specialized models for an specific kind of application. For instance, we want to experiment with the generation of a specialized learning model that captures the behavior of Magento, a widely used e-commerce application.

Design and Implementation of the Web Application Classifier Engine: As an of improvement of the prototype we described in Section III we are currently working on the design

and full implementation of WACE as a library package. This library could be used by different WAF or any other application that needs to classify HTTP requests. To improve the prototype described in Section III, we are currently working on the design and full implementation of WACE as a library package. This library could be used by different WAF or any other application that needs to classify HTTP requests.

Integration with MODSECURITY*:* The implemented prototype uses the LUA bindings provided by MODSECURITY to call the evaluator. The integration experiment showed to work alright, however the exchange of information (the requests and the scores) between the evaluator and MODSECURITY is not transparent enough. In order to integrate the ML Module with MODSECURITY we plan to extend the MODSECURITY language with a new rule directive specifically design to call the WACE API.

Integration with OWASP CRS*:* In [18] we have proposed a concrete integration mechanism where the scores produced by both the rules and machine learning models were combined in a fixed voting manner. When both agree (both say valid or both say attack) then the result is straightforward. In the case the one-class approach classifies a request as an attack, given that the OWASP CRS have the know-how on attacks, we prioritize its answer (so if OWASP CRS classifies the request as valid then is valid). The OWASP CRS when configured to block attacks, has two working modes: traditional and anomaly scoring. In the anomaly scoring mode, all rules get evaluated, and each matching rule will add to a global score depending on the rule severity. After all rules gets evaluated, the final score is compared with a defined threshold and takes the action depending on the result, if the score is higher than the threshold, the request is rejected. In this line of work, we plan to study the case where we use the machine learning results in order to adjust the value of the threshold per requests when the OWASP CRS is configured in anomaly scoring mode. This will allow the WAF to be more strict o tolerant with the scores of the rules, depending on the machine learning model result.

V. FINAL CONSIDERATIONS

In the last decade web applications have been used as the major attack vector in the most critical data breaches. New data privacy regulations, high fines and image reputation have became relevant for organizations which are forced to take measures in order to mitigate web applications risks. Nowadays, the first line of defense that an organization has to protect the web applications it exposes to the Internet is the use of Web Application Firewalls. According to Gartner [9], *"by 2020, more than 50% of public-facing web applications will be protected by cloud-based* WAF *service platforms"*. One of the major challenges to face when deploying a WAF is to keep the rules correctly tuned while Web Applications evolve and new attacks are employed. In this work we have presented a framework that allows to use machine learning models in order to improve WAF capabilities.

We have designed, trained and tested two machine learning models: one for the rare, but best case, where we have a labeled dataset with real application traffic and a more practical case where we have only valid requests. Both models outperform the detection capabilities of MODSECURITY configured with the OWASP CRS out of the box.

We have also implemented a prototype to integrate the one-class model with MODSECURITY. This allows to validate the whole process, from the generation of the machine learning model to real-time web application attack detection using a widely deployed tool.

In the short term we plan to focus our work in the design and implementation of the web attack classification engine, as it will facilitate both the research of new machine learning models and the testing of different integration mechanisms with the OWASP CRS.

REFERENCES

[1] Verizon, "2018 data breach investigations report 11th edition research report," Verizon, Tech. Rep., 2018.

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

[5] 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/

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

[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] R. Pechter, "What's pmml and what's new in pmml 4.0?" *Acm Sigkdd Explorations Newsletter*, vol. 11, no. 1, pp. 19–25, 2009.

[11] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, Jan 2000.

[12] 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.

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

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

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

[16] 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.

[17] 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.

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

[19] Modsecurity and machine learning module. [Online]. Available: https://gitlab.fing.edu.uy/gsi/modsec-ml