An Adaptive Web Application Firewall

Miguel Calvo and Marta Beltrán
Department of Computing, ETSII, Universidad Rey Juan Carlos, Madrid, Spain

Keywords: Adaptive Controls, Risk-based Security, Web Application Firewall.

Abstract: Web Application Firewalls (WAFs) are security products responsible for protecting web applications with minimal cost and effort; by filtering, monitoring, and blocking HTTP traffic. Traditional WAFs work with a rule-based approach, applying predetermined rules when the signatures of known attack patterns or traffic anomalies are identified. This kind of design has suffered significant limitations in specific contexts since it is impossible to configure the WAF the first time and rely on that configuration over time. This paper proposes an adaptive WAF capable of context-aware risk-based adaptation, changing its configuration to every specific scenario, depending on the current value of risk indicators and on the level of risk tolerated at any given time. The proposed solution is implemented, validated and evaluated in a real use case.

1 INTRODUCTION

Traditional Web Application Firewalls or WAFs work the same way as traditional network firewalls: they are rule-based security solutions, relying on a predefined set of rules that enable proper response when the signature of a known attack pattern is detected, or a traffic anomaly is observed.

There are no one-rule-fits-all scenarios in current web applications security; no static configuration is general enough to cope with risk within all possible situations. The variety and complexity of novel attack patterns and threat agents lead organisations to develop significant efforts, therefore incurring high labour costs, maintaining the WAFs rule-sets, and updating attack signatures or legitimate traffic patterns (normal behaviour).

This paper proposes an intelligent and automated approach to WAF adaptation, an adaptive WAF. With a risk-based solution, security mitigations are adjusted to the asset's environment and state and the risk quantified or predicted at a given moment. The goal is to change, autonomously, the behaviour of security countermeasures by monitoring them (the WAF, in our case), the protected asset (a web application) and their operation context, quantifying the risk and keeping it at the desired level.

The contributions of this paper are 1) The architecture of an Adaptation Tool based on a MAPE-K feedback loop (Monitor-Analyze-Plan-Execute over a shared Knowledge) capable of deciding if an adaptation of the WAF is required and performing this adaptation when required. 2) The semantics of the policies and rules that allows security managers and administrators to automate adaptation. 3) A prototype of this Adaptation Tool that enables its validation and evaluation within a real use case.

The rest of this paper is organised as follows. Section 2 provides an overview of the background on underlying concepts and the related work. Section 3 discusses the primary motivations for this work, with some research challenges and opportunities. Section 4 describes the proposed architecture to add automated risk-based adaptation to Web Application Firewalls. Section 5 details the proposed solution implementation, validation and evaluation. And finally, Section 6 summarises our main conclusions and the most interesting lines for future work.

2 BACKGROUND AND RELATED WORK

2.1 On Web Application Firewalls

Security tools such as WAFs have been proposed to protect websites, web applications, and web services. These defensive solutions, like traditional firewalls, offer an extra layer of protection by inspecting traffic. WAFs typically operate at the application layer of the OSI (Open Systems Interconnection) model, inspect-

Calvo, M. and Beltrán, M.
An Adaptive Web Application Firewall.
DOI: 10.5220/0011146900003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 96-107
ISBN: 978-989-758-590-6; ISSN: 2184-7711

ing, mainly, HTTP traffic. They are deployed between the web server serving the website/application/service and the Internet, acting as a shield and analysing requests before responding to them and before traffic enters or leaves the web server. In this way, when a potential attack is detected, the WAF itself apply some kind of rule, for example, to block the request so that it does not impact the protected asset (Garn et al., 2021), (IŞiker and SoĞukpinar, 2021).

There are mainly three types of WAF solutions: signature-based WAFs, anomaly-based WAFs, and Machine Learning-based WAFs. Signature-based WAFs, the most common currently, require that the attacks have previously occurred since they check the traffic searching for signatures associated with known attack patterns. Therefore, they require a continuous update of the signatures. And they are not capable of identifying zero-day attacks. On the other hand, anomaly-based WAFs require prior knowledge about "normal" or "legitimate" traffic to/from the protected web server. In this case, rules are applied, not when the signature of a known attack is found but when an anomaly is detected. Again, a continuous update of normal or legitimate patterns is essential since the Internet and its users' behaviour evolve fast these days. But if it is performed correctly, this kind of WAFs are able to handle zero-day attacks. Finally, Machine Learning-based WAFs use Machine Learning (ML) to rely on models trained to overcome the limitations of the rest of WAFs. A Machine Learning approach allows a WAF to classify a request as malicious without applying signature-matching or anomaly detection techniques. The goal is to decrease false positive and false negative rates while making it difficult for adversaries to cheat the WAF and reducing the burden related to the WAF updates (signatures or regular traffic). But a Machine Learning model must be trained and retrained (Applebaum et al., 2021), (Babiker et al., 2018).

In recent years, research on WAF tools and their improvement for the protection of web assets has been focused mainly on applying Machine Learning or Deep Learning to these solutions. In this sense, to mention just some significant examples, (Valenza et al., 2020) proposes a mechanism that replaces policies with Machine Learning. (Domingues Junior and Ebecken, 2021), on the other hand, proposes the evaluation of headers and URIs at a high level using Machine Learning and classifiers applied to rule engines such as ModSecurity; (Betarte et al., 2018), and (Moradi Vartouni et al., 2015) proposals focus on applying Machine Learning techniques based on expected n-gram frequencies to exploit WAFs, (Gogoi et al., 2021) focuses on the detection of XSS attacks through a WAF that uses Machine Learning. There are also approaches, such as (Tekerek and Bay, 2019), based on learning too but using Artificial Neural Networks. Another exciting research is shown in (Ito and Iyatomi, 2018), which focuses on a WAF system to identify malicious HTTP requests using a character-level Convolutional Neural Network. The framework proposed in (Shahid et al., 2022) combines a classifier based on Deep Learning and a cookie analysis engine; (Tan and Hoai, 2021) proposes an original mechanism, applicable to WAFs, in which HTTP requests are transformed into anomalous univariate data points.

2.2 On Risk-based Security

The rise of sophisticated web attacks and the wide variety of new technologies and paradigms in recent years (such as Industry 4.0, Cloud Computing or the Internet of Things) have made conventional security mechanisms insufficient. It is necessary to explore more intelligent and adaptive ones that protect different types of assets in current infrastructures (Nafea and Amin Almaiah, 2021), (Ande et al., 2020). It is common to find the term dynamic security among these new solutions. This paradigm opens the door to technologies capable of automatically reconfiguring and adapting when changes are detected in the environment in which they operate, when specific threats are detected, when the risk tolerated by the organisation changes, etc.

Two main categories can be found in previous works in this research area. First, adaptive security, which, depending on the environment or state, is capable of modifying the architecture or behaviour (Elkhodary and Whittle, 2007), (Djoudi et al., 2014), (Boudko and Abie, 2019) of security controls and mitigations. Second, risk-based security, which focuses on the analysis of the different risks that an organisation's assets may suffer and prioritising different factors (such as the cost of mitigation, functionality, quality of experience, etc.), can make the pertinent changes to reduce that risk to an acceptable level.

In this way, risk-based security requires adequate and constant monitoring and evaluation of risks and the existence of prior planning. It is necessary to identify the assets and the threats that may arise to apply this approach. It is also required to find the existing vulnerabilities and create the different risk profiles of each asset or group, assigning them a score. Also, it is necessary to decide how this risk will be handled when it arises. The most significant difficulty in these tasks is usually quantifying the risk and translating the score obtained into a balanced decision. In dynamic

risk-based security solutions, decision-making is frequently based on different metrics and indicators such as IoCs (Indicators of Compromise), IoAs (Indicators of Attack) or KRIs (Key Risk Indicator).

Focusing on current research on risk-based security, we can see many proposals focused on identity and access management for emerging technologies and environments; for example, (Chen et al., 2016) for Cloud, (Steinegger et al., 2016), for Web Applications, and (Martinelli et al., 2018) for IoT.

3 MOTIVATION, CHALLENGES AND OPPORTUNITIES

As introduced in the previous section, Machine Learning offers good results when applied to traditional WAFs, detecting zero-day attacks and avoiding the requirement of continuous updates of signatures and traffic patterns. However, these solutions also pose new challenges and difficulties, in many cases hard to solve or requiring a high investment of human and financial resources and knowledge.

Among these difficulties, we can highlight the large-scale optimisation problem. This setback is largely observed when training some mathematical models used in Machine Learning. In this sense, training data poisoning is one of the biggest concerns in the WAF detection line. It is tough to identify if the data or information we use to train ML algorithms has been manipulated. Poisoned or unreliable data may cause both false positives and false negatives (Gambella et al., 2021), (Sun et al., 2020).

Another significant difficulty with ML applied to WAFs is the need to retrain the models from time to time and the computational cost required by many of the techniques used for this task. In addition, it must be taken into account that this retraining requires an investment of time and human resources, as well as the need to have datasets that meet the necessary expectations (Kumeno, 2019), (Schelter et al., 2018).

This research proposes an alternative approach, an adaptive WAF capable of overcoming some limitations observed in the traditional solutions without adding ML and, therefore, avoiding all these difficulties. Approaches such as MAPE-K (Lara et al., 2019) or (Calvo and Beltrán, 2022) offer a new opportunity that is able to substantially reduce the necessary resources and costs, with effective results in terms of dynamicity and adaptation capability. The rest of this paper proposes the architecture of a new category of WAF, the adaptive WAF or risk-based WAF.

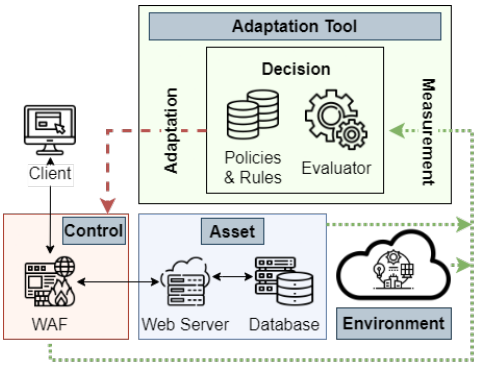

Figure 1: Proposed architecture for the risk-based WAF.

4 PROPOSED ARCHITECTURE

4.1 Overview

The proposed approach to design a risk-based WAF is to add the capability of adaptation from outside the WAF, with a separation of concerns design strategy. The WAF is still a WAF; the Adaptation Tool is designed explicitly to perform adaptation. In this way, the adaptation capabilities are generic and can be added to any WAF-type product with which integration is possible. In addition, it avoids redesigning existing WAFs by adding complexity and cost.

As it can be seen in figure 1, the proposed solution works as follows:

1. The WAF administrator defines the different Measurements that will enable the risk-based adaptation of the WAF. These measurements (basic or extended metrics and indicators or events) will be sent from different sources (internal or external) to the Decision element. These sources may be:
   (a) The Environment. For example, external sources such as social networks, third-party threat intelligence, the status of other services or servers outside the infrastructure, etc.
   (b) The Asset (or assets) to protect. In this case, the web server and the rest of the servers are running the protected application or service.
   (c) The Control. The WAF and any other additional layers added to the WAF to increase or decrease performance or security. For example, a load balancer or an anti-Distributed Denial of Service solution (anti-DDoS).
2. When these measurements are received, they are analysed to decide if a policy or rule should be triggered.
3. If this is the case, these policies and rules determine the adaptation that must be carried out;

therefore, the pertinent changes or modifications to the "Control" (the WAF and/or the tools that complement it) that must be performed. For example, changing the WAF configuration to start working through allow-lists, adding a new security layer such as anti-DDoS (anti-Distributed Denial of Service), etc. Integration with the adapted WAF is required to perform these changes automatically (via scripting, plugins, etc.).

4.2 Measurement

The adaptive WAF needs to quantify the risk to make the right adaptation decisions at the right time. Comparing the risk taken or expected to be taken in the future with the risk tolerance of the protected application allows security measures to be relaxed or reinforced at any point in time. Several questions must be asked: what we should measure, when should we make the measurement, and which data sources are available. It has to be considered that without a good measurement strategy, the proposed solution loses effectiveness and could even become counterproductive (false positives, false negatives, unnecessary adaptations consuming resources, temporary blocks of architecture, etc.).

These measurements, as already mentioned, can be done in the assets to be protected (the application and servers), in the control itself (in the WAF or any other additional security layer) or in the environment. In this way, and just to mention a few examples for the collection and extraction of measurements, common data sources for the adaptive WAF will be logs (at the application, servers, the WAF), traffic analysers, IDPS (Intrusion Detection and Prevention System) or SIEM (Security Information and Event Management), anti-malware solutions and endpoint protection systems, open sources (social networks, government data, press, information shared by other organisations) or threat hunting and intelligence products.

The specific measures to be performed by the adaptive WAF will depend on each organisation's specific objectives in securing the application or web service with the WAF and its strategies for managing risk. But some specific measures can be mentioned in the context of WAFs:

- The number of HTTP requests received per period.
- The number of HTTP errors (404, 500, etc.) per period.
- The number of requests to malformed URIs and/or unexpected URIs that are requested in a period.
- Requests from IP addresses in a block-list and/or from addresses that are not in the allow-list.
- Number of database queries per period.
- Number of UPDATE and/or DELETE functions on the database per period.
- Number of simultaneously connected database users.
- Volume of Tweets or other social network interactions that mention a specific topic at a given time.
- Terrorist alert level of a country at a given time.
- Number of Distributed Denial of Service (DDoS) attacks in a recent period.

How to quantify risk from these measures with risk scores is beyond the scope of this paper, as are the methods for deciding whether that risk is acceptable or has changed sufficiently to justify an adaptation of the WAF.

4.3 Decision

When risk-related metrics are received in the Decision element, they are analysed to know if their values must trigger any action. Therefore, although the measurements may be frequent, the execution of policies and rules, i.e., verifying whether it is necessary to carry out a WAF adaptation, is not performed with the same assiduity. In this way, excessive consumption of resources (CPU, memory) is avoided.

To understand decision-making, first, it is necessary to know the structure of the policies and the rules defined for the proposed solution.

- In the case of **policies** they state, mainly, the "Adaptation conditions". These break down into different predicates (as many as required) composed of one or more observable triggers and states (as many as required with AND-OR operators) and one or more rules. A Trigger occurs when a risk score or RS exceeds a certain threshold or changes a certain degree, a State corresponds with a situation, condition, the situation of the configuration of the asset or the control. For example, you can see in table 1 a policy containing adaptation conditions from 1 to N, each one expressed as a predicate. Predicate 1 states that the observation of the Trigger1 ($RiskScore1 > X$) leads to the execution of the adaptation rule A. In the same way, the Predicate 2 states that the observation of the Trigger2 ($RiskScore2 < Y$) leads to the execution of the adaptation rule B and the Predicate N states that the observation of the StateN for the control leads to the execution of the adaptation rule N

Table 1: Policy definition.

Adaptation conditions:
Predicate 1: *observed*(Trigger1 when RS1 greaterThan X) → Rule A.
Predicate 2: *observed*(Trigger2 when RS2 smallerThan Y) → Rule B.
Predicate N: *observed*(StateN) → Rule N.

- On the other hand, the **rules** (see table 2) mainly contain timing and certain controls.
  - Timing. Rules can be periodic, event-driven or on-demand. Periodic rules activate with a timer (periodically); event-driven activate only when a specific event is observed; and, on-demand, leave the rule on standby until an administrator activates the control manually.
  - Controls. They specify the action that is performed to apply the adaptation. In addition, they determine the artefact or mechanism that can be used to connect the Adaptation Tool to the WAF. For example, connection via SSH to a control panel, connection via HTTP to the service that increases security by adding an extra layer, etc.

In this way, when a trigger is activated, it starts the execution of the associated policies and checks the rules linked to it. Once located, they are analysed:

- If a timer exists (periodic rule), the rule will continue its flow when its period finishes.
- If the rule is event-driven, the rule will wait until the event occurs. This event can be the trigger that executed the policy in the first place, but it can also be a different one.
- If the rule is on-demand, the rule will wait until the WAF administrator (who will be notified by email, SMS or the decided way) intervenes to continue with the flow and carry out the actions manually.

Regardless of the rule timing, when the rule is executed, the specific actions to be carried out and the required artefacts will be extracted from the "Controls" section. The actions summarise the required adaptation and the script, configuration, tool, etc., that must be modified. The artefacts will be used to connect to the WAF and perform the actions.

4.4 Adaptation

The adaptation of the WAF is achieved by performing different actions through components, plugins, scripts, or pieces of code that will be executed in the different elements of the WAF from the Adaptation Tool. These elements enable the proper modifications of the adapted control, adding or removing security layers, modifying the configuration, etc. They depend on the control to be adapted, i.e., on the specific type/model of WAF.

Table 2: Rule definition.

Timing:
Event: observed(TriggerM); or
On Demand: true/false; or
Timer: Adaptation period value (in seconds).
Controls:
Action: Script that will apply the adaptation.
Artefact: Mechanism used to connect to the control.

Some examples of adaptations can be the activation or deactivation of the WAF, the addition or removal of rules, the selection of allow-lists or block-lists, the migration of a local WAF to a WAF as a Service solution (or vice versa), the addition of an anti-DDOS layer to the WAF, etc.

As mentioned before, WAF adaptations, once decided, are performed through artefacts. These artefacts are responsible for communicating the Adaptation Tool to the different elements of the WAF and facilitating reuse. They are connectors or generic connection mechanisms such as SSH connections, HTTP connections, specific APIs, etc.

5 PROTOTYPE, VALIDATION AND EVALUATION

A prototype of the proposed adaptive WAF has been implemented to protect a web asset. The protected website consists of an affiliate store of products sold by other stores. Administrators register different products (with their associated information: title, description, price, characteristics, photographs, etc.) and a personal evaluation of the product (highlighting its qualities and drawbacks). In addition, users who visit the web can buy the product through an affiliate link, which redirects them to the stores that sell the product. This redirection, generated with a particular link, reports a small benefit for each sale.

The web application is developed using Vue.js, NodeJS and MySQL technologies. It is deployed on two different servers (each with Ubuntu Server 20.04 operating system, 2 vCores, 2 GB of RAM and 80 GB of hard drive). The first (with a Nginx web server installed) is responsible for hosting both the frontend and the backend of the application and, therefore, is the user's gateway to the application; the second (with MySQL Community Server installed) contains

the database server where, among other things, all the information associated with the different products displayed in the affiliate store is stored.

The store is made up of three main pages:

- A list of all the existing products in the database (accessible through https://example.com/products; includes pagination, to avoid returning all the products at once).
- The one that shows the information of each product (accessible through https://example.com/product/ID, where each of the product identifiers replaces ID).
- A last one to facilitate the search for users who use the web (accessible through https://example.com/search; the different search parameters are sent by a POST method).

The WAF protecting this site is ModSecurity (SpiderLabs, 2022), an open source web application firewall. The choice has been supported by a large amount of existing documentation and the ease of use, the flexibility offered by this WAF and, of course, the possibility of integrating it with the application to be protected and the web server that serves it (the v3 is used to have the standalone engine built from scratch in C++ easily integrated into Nginx). This solution provides real-time monitoring, logging, and filtering of HTTP requests based on user-defined rules written in a rule configuration language named SecRules.

The WAF is installed and configured on the same server that hosts the backend and frontend of the application, relying on signature-based detection. The limitations of this WAF in the considered scenario are that with a relaxed configuration of the rules, few attacks are blocked while with a restrictive configuration, this blocking rate increases at the cost of blocking many of the legitimate requests made by users.

Finally, the proposed Adaptation Tool prototype is hosted on a separate server with Ubuntu Server 20.04 Operating System, 2 vCores, 2 GB of RAM, and 80 GB of hard drive. This server is on the same internal network as the web server and database server.

5.1 Implementation

The prototype implementation is illustrated in figure 2. First, it is worth highlighting the measurements selected to decide about the WAF adaptation. These measurements come from three different data sources (regarding the asset and the environment), specifically:

- On the one hand, HTTP requests of type 400 (internal measurement). This information is extracted from the Nginx server logs and sent to the decision element every minute. The metric consists of a numerical value corresponding to the number of new HTTP requests of type 400 per minute. A small Python script has been developed that reads the information from the Nginx log in real-time and sends it via HTTP to the server hosting the Adaptation Tool.

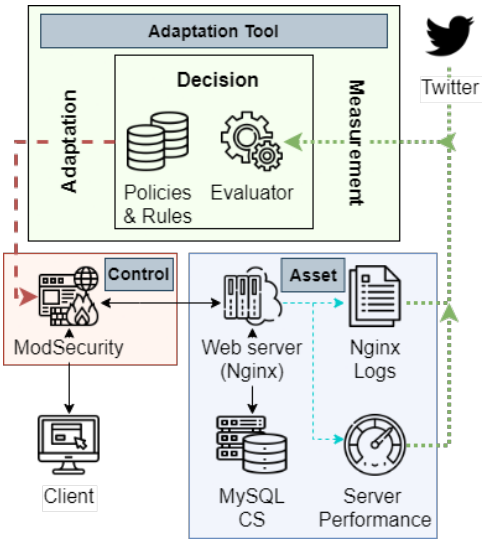

Figure 2: Prototype implementation.

- Another required measurement is the resource consumption (CPU and RAM memory) at the web server. The information is extracted from the server itself and consists of two numerical values corresponding to the percentage of CPU and the percentage of RAM memory that has been consumed, on average, in the last hour. To extract these metrics, a Python script is executed at the web server, making periodic measurements and sending the information via HTTP to the server hosting the Adaptation Tool.
- On the other hand, tweets that contain a shop URL (external measurement). This information is extracted by a script that relies on the Twitter API. It analyses the information, in real-time, in search of Tweets that contain a URL belonging to the protected website. The metric submitted to the decision element consists, in this case, of a numerical value that corresponds to the number of new appearances on Twitter per hour; it is sent via HTTP to the server hosting the Adaptation Tool.

In this way, the Decision element receives the number of HTTP requests of type 400 per minute, the resources average consumption in the last hour (CPU and RAM) and the number of tweets that contain a shop URL per hour. This information is re-

ceived through a web service (created in NodeJS) and stored in a MySQL database (deployed on the server where the Adaptation Tool is located).

The service itself, each time it receives a metric obtained from new measurements, checks to see if they are within the thresholds set by the WAF administrator and the security manager. A 400 "Bad Request" is an HTTP response status code that indicates that the server is unable to process the request sent by the client due to invalid syntax. If an adversary is trying to perform an SQL injection attack or is trying to scrape data from the store, all the IDs are likely gone through, leading to numerous errors. Therefore, it is considered in this use case as a risk indicator. The number of URLs linked from Twitter may work, on the other hand, as a risk indicator of a Distributed Denial of Service attack (DDoS), qualified by the consumption of resources at the web server. Therefore:

- For HTTP requests of type 400, a trigger is launched (and policies will be analysed) when 50 requests per minute are exceeded. The opposite trigger is launched when this value ins below 50. In this way, the WAF can be adapted to apply more or less restrictive configurations depending on this measurement (risk indicator).
- For Tweets with URLs, a trigger is launched when the number of tweets per hour exceeds 100. But the adaptation to a restricting configuration can be relaxed if the resource consumption at the server is not of concern.

In this prototype, two policies and four rules have been defined:

- Policy 1. This policy is activated when: (1) It is detected that HTTP requests of type 400 exceed the configured threshold and the WAF is not working in "allow-list" mode; (2) It is detected that the type 400 HTTP requests fall below the established threshold for more than 30 measurement periods (over 30 minutes) and, in addition, the WAF is working in the allow-list mode. As a result of the first activation, rule 1 applies, and because of the second, rule 2 applies. Table 3 shows this policy with the format proposed in table 1, where Trigger1 (HIGH-HTTP_400_Cod) is observed each time HTTP requests of type 400 per minute $>$ 50, State1 (OFF-WAF-ALLOWLIST) is observed when the WAF is not in the "allow-list" mode, Trigger2 (LOW-HTTP_400_Cod) is observed each time HTTP requests of type 400 per minute $<$ 50 during 30 minutes and State2 (ON-WAF-ALLOWLIST) is observed when the WAF is in the "allow-list" mode.
  - Rule 1. The first rule associated with policy 1 is executed, following the timing configured by the administrator ("Timing" section of the rule), when the same event that triggered the related policy occurs (exceeding the threshold for HTTP requests type 400). Therefore, it is done as soon as it is analysed. Applying this rule runs the script "waf-allowlist.py" on the WAF, using the artefact "ssh-connection.py" to connect to the ModSecurity WAF.
  - Rule 2. In this case, the rule, when triggered, waits until the o'clock hours. Once that time arrives, the "waf-normal.py" script is executed using the "ssh-connection.py" artefact to connect to the WAF.
- Policy 2). The activation of this policy occurs when: (1) It is detected that the number of Tweets with URLs of the store exceeds the established threshold and the WAF is not working with access restrictions by country; (2) It is detected that the resource consumption at the web server is below the configured threshold and, in addition, the WAF is working with access restrictions by country. As a result of the first activation, rule 3 applies and, by the second, rule 4. Table 4 shows this policy with the format proposed in table 1, where Trigger3 (HIGH-URL_on_Tweets) is observed each time tweets per hour $>$ 100, State3 (OFF-WAF-COUNTRY) is observed when the WAF is not working with access restrictions by country, Trigger4 (LOW-WS_Consumption) is observed each time CPU consumption $<$ 70% & RAM consumption $<$ 80% during 60 minutes and State4 (ON-WAF-COUNTRY) is observed when the WAF is working with access restrictions by country.
  - Rule 3. The first rule associated with policy 2 is executed when the event that triggered the related policy occurs (exceeding the threshold number of Tweets containing URLs to the store), therefore, it is analysed automatically. This rule runs the "waf-restricted.py" script using the "ssh-connection.py" artefact.
  - Rule 4. The fourth and last rule, also associated with policy 2, is executed when the web server has kept average CPU and RAM consumption below 55%. When this condition is met, it is the "waf-not-restricted.py" script using the "ssh-connection.py" artefact that is responsible for applying the adaptation.

Table 5 shows these two policies in JSON (JavaScript Object Notation). This standard file format has been selected to implement the prototype of the adaptive WAF because it enables data objects in-

Table 3: Policy 1 for the prototype.

Adaptation conditions:
Predicate 1: *observed*(Trigger1 and State1) → Rule 1.
Predicate 2: *observed*(Trigger2 and State2) → Rule 2.

terchange (attribute–value pairs and arrays, or other serializable values) using human-readable text. It is a language-independent data format that can be generated and parsed by humans, applications and servers.

Different code snippets enable the execution of the Adaptation, running the actions expressed in the four defined rules:

- ssh-connection.py. It is responsible for establishing the connection, via SSH, between the server that hosts the Adaptation Tool and the web server (where the WAF is running).
- waf-allowlist.py. This second piece of code, associated with rule number 1, connects to the server with the WAF (taking advantage of the functions contained in "ssh-connection.py") and (1) Queries the database to extract all existing product identifiers (IDs); (2) Modifies the WAF configuration file, activating an allow-list and including in it all the URLs to products previously extracted from the database; (3) Restarts the WAF for the changes to take effect.
- waf-normal.py. It is associated with rule 2, and after making the connection with the WAF (using "ssh-connection.py"): (1) Modifies the WAF configuration file, deactivating the allow-list for products; (2) Restart the WAF service for the changes to take effect.
- waf-COUNT-restricted.py. Associated with rule 3, connects via SSH with the WAF and: (1) Modifies the WAF configuration file to restrict access to the web application only from some specific countries; (2) Restarts the WAF service for the changes to take effect.
- waf-not-restricted.py. It is associated with rule 4. After making the SSH connection with the WAF: (1) Modifies the WAF configuration file, deactivating the access restriction by country; (2) Restarts the WAF service for the changes to take effect.

Table 6 shows the four proposed rules, again written in JSON. Each rule requires a name, the specification of a timing and the controls. In this prototype, rules 1, 3 and 4 are event-driven, therefore, governed by a trigger. Rule 2 is periodic, therefore, governed by a *cron* expressed with five digits: minute, hour, day, month, day of week (the $*$ represents any value, the 0 is to perform the adaptation only at the o'clock hours). The four rules use as an "Artefact" the code allowing the connection via SSH, each one of them performs a different "Action" thanks to the already introduced code snippets.

Table 4: Policy 2 for the prototype.

Adaptation conditions:
Predicate 1: *observed*(Trigger3 and State3) → Rule 3.
Predicate 2: *observed*(Trigger4 and State4) → Rule 4.

Table 5: Policies 1 and 2 for the prototype written in JSON.

```
[ {
    "name": "Policy1",
    "conditions": [{
        "antecedent": [
            "HIGH-HTTP_400_Cod",
            "OFF-WAF-ALLOWLIST"
        ],
        "consequent": ["Rule1"]
    },{
        "antecedent": [
            "LOW-HTTP_400_Cod",
            "ON-WAF-ALLOWLIST"
        ],
        "consequent": ["Rule2"]
    }]
},{
    "name": "Policy2",
    "conditions": [{
        "antecedent": [
            "HIGH-URL_on_Tweets",
            "OFF-WAF-COUNTRY"
        ],
        "consequent": ["Rule3"]
    },{
        "antecedent": [
            "LOW-WS_Consumption",
            "ON-WAF-COUNTRY"
        ],
        "consequent": ["Rule4"]
    }]
}]
```

5.2 Experimental Results

Different experiments have been performed to validate and evaluate the proposed approach (see table 7). The performed experiments have covered the entire

Table 6: Rules 1 to 4 for the prototype written in JSON.

```
[ {
    "name": "Rule1",
    "timing": {
        "period": null,
        "on-demand": false,
        "trigger": "HIGH-HTTP_400_Cod",
    },
    "controls": [ {
        "action": "waf-allowlist.py",
        "artefact": "ssh-connection.py",
    }],
}, {
    "name": "Rule2",
    "timing": {
        "period": "0 * * * *",
        "on-demand": false,
        "trigger": null
       },
    "controls": [ {
        "action": "waf-normal.py",
        "artefact": "ssh-connection.py",
    }],
}, {
    "name": "Rule3",
    "timing": {
        "period": null,
        "on-demand": false,
        "trigger": "HIGH-URL_on_Tweets"
    },
    "controls": [ {
        "action": "waf-COUNT-restricted.py",
        "artefact": "ssh-connection.py",
    }],
}, {
    "name": "Rule4",
    "timing": {
        "period": null,
        "on-demand": false,
        "trigger": "LOW-WS_Consumption"
    },
    "controls": [ {
        "action": "waf-not-restricted.py",
        "artefact": "ssh-connection.py",
    }],
}]
```

application for one week, although, in this section, we focus on the results obtained when protecting the "https://example.com/product/ID" page. As a baseline for performance, two ModSecurity deployments have been considered with two different levels of paranoia: one with a "relaxed" configuration working with a block-list and allowing requests from different countries, and one with a "restricting" one working with an allow-list and accepting only requests from IPs at selected countries. The Adaptive ModSecurity is our prototype, implemented as described in the previous section.

During this week, 50 legitimate users of the application (sending 400 legitimate requests per day each to the protected application) and 400 potential attacks per day have been simulated, including SQL injection and DDoS attacks, because the rules and policies included in the evaluated prototype are intended to block this type of attack. These attacks are carried out with the OWASP ZAP and Burp Suite tools and different known payloads.

Table 7 shows the average CPU and RAM consumption due to the WAF on the web server, the average CPU and RAM consumption due to the execution of additional tools on a different server (the server to execute the Adaptation Tool in the case of our prototype), the number of blocked and successful attacks during the week, the number of blocked legitimate requests and the type of task that needs to be performed by the WAF administrator prior to this week of experiments in order to achieve the obtained results.

As demonstrated by the obtained results, the relaxed WAF configuration consumes few resources on the web server but allows a higher number of attacks to succeed than the restrictive configuration. In return, this restrictive configuration consumes much more resources (CPU and RAM). The adaptive approach proposed in this work manages to block almost the same number of attacks as the restrictive configuration. The figures are slightly worse because the adaptation that takes the WAF from the relaxed to the restrictive configuration is not performed until it is observed that the risk is above the established threshold. But there is hardly any difference with the more restrictive static configuration. Other adaptive policies or rules could be explored to bring the results of the adaptive WAF even closer to those of the static WAF with the restricting configuration in terms of rate of blocked attacks (from the current 79% to 85%).

And yet, it consumes fewer resources than if it were applied, statically, all the time. An additional advantage shown in the table is not related to the reduction in resource consumption, but the improved accessibility of the application. When the restrictive configuration is continually applied, legitimate users in certain countries, for example, may be blocked by the WAF. However, with the adaptive WAF, restrictive settings are only applied when strictly necessary (using rule 3), so the rest of the time, these users will be able to use the application without restrictions.

In addition, the average time it takes for the WAF to change its configuration has been measured when this change is automatic (rules 1, 3 and 4 in our prototype, in rule 2 the change is periodic). In other words, once a specific policy is evaluated, and it is decided

Table 7: Experimental results.

	WAF relaxed	WAF restricting	Adaptive WAF
CPU consumption web server (%)	16.48	17.01	16.73
RAM consumption web server (%)	40.51	49.98	46.3
CPU consumption additional server (%)	-	-	8.21
RAM consumption additional server (%)	-	-	24.18
Number of blocked attacks	1792	2381	2214
Rate of blocked attacks – true positives (%)	64	85	79
Number of successful attacks	1008	419	586
Number of blocked legitimate requests	134	1567	359
Rate of blocked legitimate requests – false positives (%)	4.8	56	12.8
Workload for WAF admin.	WAF static configuration and signatures update	WAF static configuration and signatures update	Rules, policies, and signatures update

that the WAF needs to be adapted, the time it takes to evaluate the corresponding rule and to make the adaptation is measured, and therefore, to make the changes that the administrator wishes to make to the WAF. An average time of 16.1 s with a standard deviation of 8.4 s has been measured. It has to be pointed that no HTTP request suffers this latency, this time is required to decide about the WAF adaptation and to perform it (to change the WAF configuration). The runtime performance of the Adaptive ModSecurity is exactly the same as the performance of ModSecurity, each received request is analysed in 1 to 15 ms.

There is no version of ModSecurity that uses Machine Learning to be included in the table comparison. But we can deduce that the CPU and RAM consumption on the web server would be high (due to the complexity of the models used), that an additional server would be needed to train and retrain these models offline (the adaptive WAF requires this additional server to execute the Adaptation Tool instead, but resource consumption is limited), that the burden for the WAF administrator (or the data science specialist supporting him/her) would be related to building the suitable dataset and training and retraining the models. The latency added to each request would be higher than a few ms due to the complexity to the models applied to classify them as legitimate or malicious. And the unknown, in this case, is how much it would improve the WAF's ability to block attacks (especially zero-day attacks that signatures cannot block) and to avoid blocking legitimate requests. All the observed improvements should be in these two last respects.

6 CONCLUSIONS

Web Application Firewalls (WAFs) are used to protect websites, applications and services. They may offer in-depth security as long as they are frequently configured correctly and updated (signatures, standard traffic patterns).

Machine Learning can be used to avoid this workload because WAF solutions can learn by themselves, relying on these techniques. But they require the availability of the appropriate datasets and a new kind of workload related to the training and retraining of the learning models.

This paper proposes a new approach, adaptive WAFs, based on adding an external Adaptation Tool to WAFs, capable of performing measurement, applying policies and adaptation rules defined by administrators and managers, and adjusting the way the WAF works to the risk that is being taken (and that one wishes to take) at any given moment. A prototype of the proposed approach has been implemented to protect a web application, allowing us to validate the Adaptation Tool and evaluate it with the ModSecurity WAF.

The proposed adaptive WAF has shown significant improvements in resource consumption, blocked attacks, ease of configuration, and flexibility in risk management compared to traditional WAFs. All this without requiring large datasets and complex modelling processes (training and retraining) as in the case of emerging ML approaches.

ACKNOWLEDGMENT

This research has been partially supported by the Madrid region (EdgeData, Grant Ref. P2018/TCS-4499). Miguel Calvo is supported by grants from the Rey Juan Carlos University (ref. C-PREDOC21-007).

REFERENCES

Ande, R., Adebisi, B., Hammoudeh, M., and Saleem, J. (2020). Internet of things: Evolution and technologies from a security perspective. *Sustainable Cities and Society*, 54:101728.

Applebaum, S., Gaber, T., and Ahmed, A. (2021). Signature-based and machine-learning-based web application firewalls: A short survey. *Procedia Computer Science*, 189:359–367. AI in Computational Linguistics.

Babiker, M., Karaarslan, E., and Hoscan, Y. (2018). Web application attack detection and forensics: A survey. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–6.

Betarte, G., Gimenez, E., Martinez, R., and Pardo, A. (2018). Improving web application firewalls through anomaly detection. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 779–784.

Boudko, S. and Abie, H. (2019). Adaptive cybersecurity framework for healthcare internet of things. In *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*, pages 1–6.

Calvo, M. and Beltrán, M. (2022). A model for risk-based adaptive security controls. *Computers & Security*, 115:102612.

Chen, A., Xing, H., She, K., and Duan, G. (2016). A dynamic risk-based access control model for cloud computing. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pages 579–584.

Djoudi, B., Bouanaka, C., and Zeghib, N. (2014). Model checking pervasive context-aware systems. In *2014 IEEE 23rd International WETICE Conference*, pages 92–97.

Domingues Junior, M. and Ebecken, N. F. (2021). A new waf architecture with machine learning for resource-efficient use. *Computers & Security*, 106:102290.

Elkhodary, A. and Whittle, J. (2007). A survey of approaches to adaptive application security. In *International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '07)*, pages 16–16.

Gambella, C., Ghaddar, B., and Naoum-Sawaya, J. (2021). Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828.

Garn, B., Sebastian Lang, D., Leithner, M., Richard Kuhn, D., Kacker, R., and Simos, D. E. (2021). Combinatorially xssing web application firewalls. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 85–94.

Gogoi, B., Ahmed, T., and Saikia, H. K. (2021). Detection of xss attacks in web applications: A machine learning approach. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*, 9(issue-1):2347–5552.

IŞiker, B. and SoĞukpinar, I. (2021). Machine learning based web application firewall. In *2021 2nd International Informatics and Software Engineering Conference (IISEC)*, pages 1–6.

Ito, M. and Iyatomi, H. (2018). Web application firewall using character-level convolutional neural network. In *2018 IEEE 14th International Colloquium on Signal Processing Its Applications (CSPA)*, pages 103–106.

Kumeno, F. (2019). Sofware engneering challenges for machine learning applications: A literature review. *Intelligent Decision Technologies*, 13:463–476. 4.

Lara, E., Aguilar, L., Sanchez, M. A., and García, J. A. (2019). *Adaptive Security Based on MAPE-K: A Survey*, pages 157–183. Springer International Publishing, Cham.

Martinelli, F., Michailidou, C., Mori, P., and Saracino, A. (2018). Too long, did not enforce: A qualitative hierarchical risk-aware data usage control model for complex policies in distributed environments. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, CPSS '18, page 27–37, New York, NY, USA. Association for Computing Machinery.

Moradi Vartouni, A., Shokri, M., and Teshnehlab, M. (2015). Auto-threshold deep svdd for anomaly-based web application firewall.

Nafea, R. A. and Amin Almaiah, M. (2021). Cyber security threats in cloud: Literature review. In *2021 International Conference on Information Technology (ICIT)*, pages 779–786.

Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., and Szarvas, G. (2018). On challenges in machine learning model management. *IEEE Data Eng. Bull.*, 41:5–15.

Shahid, W. B., Aslam, B., Abbas, H., Khalid, S. B., and Afzal, H. (2022). An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling. *Journal of Network and Computer Applications*, 198:103270.

SpiderLabs (2022). Modsecurity waf.

Steinegger, R. H., Deckers, D., Giessler, P., and Abeck, S. (2016). Risk-based authenticator for web applications. In *Proceedings of the 21st European Conference on Pattern Languages of Programs*, EuroPlop '16, New York, NY, USA. Association for Computing Machinery.

Sun, S., Cao, Z., Zhu, H., and Zhao, J. (2020). A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681.

Tan, H. H. and Hoai, T. V. (2021). Web application anomaly detection based on converting http request parameters to numeric. In *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, pages 93–97.

Tekerek, A. and Bay, O. (2019). Design and implementation of an artificial intelligence-based web application firewall model. *Neural Network World*, 29(4):189–206.

Valenza, A., Demetrio, L., Costa, G., and Lagorio, G. (2020). Waf-a-mole: An adversarial tool for assessing ml-based wafs. *SoftwareX*, 11:100367.