Initial Report: Implementation of a Web Application Firewall for a high availability front end

Hernan Espinosa Reboredo
System Administration
SIGMA Gestió Universitaria
Sant Cugat del valles, Barcelona, Spain
hespinosar@gmail.com

I. Introduction

The aim of this project is to build a high availability web application firewall [1] that will monitor and analyse client's requests data with the aim of protecting the backend server's security. In order to do so, a web application architecture will be developed to carry out the monitorization, tests and conclusions.

A Web Application Firewall helps to protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. By deploying a WAF in front of a web application, a shield is placed between the web application and the Internet. While a proxy [2] server protects a client machine's identity by using an intermediary, a WAF is a type of reverse-proxy, increasing security, performance and reliability by having clients requests pass through the Web Application Firewall before reaching the servers. A Web Application Firewall operates through a set of rules often called policies. These policies aim to protect against vulnerabilities in the application by monitoring and filtering out malicious traffic.
Consequently, I am going to carry out an exhaustive analysis of the events that take place in the front end of the architecture, using different development tools to monitor the traffic.
The main goal that has motivated this work is to improve the security of any architecture's servers, to have a better control over the events of their activity, and to reduce the time of action before an incident. Therefore, this project is deployable to any web application architecture in order to improve its security.

II. Requirements analysis

A. Objectives

The main objective of this work is to improve backend server's security and to reduce security vulnerabilities from web application services, by analysing and monitoring the data-flow from different requests from the front end, all the way to the backend of the service. Firstly, a scalable client-server web application architecture is going to be built. Once the architecture is built, a Web Application Firewall will be deployed to the service, and after the configuration, we will be able to see all requests in real-time and establish rules to grant or deny access to the backend servers. That way we will be able to understand and monitor the requests that will be sent to the backend servers, and with the help from different development tools and techniques we will prevent any malicious requests from compromising the Web Application's service and we will learn how to avoid them.

III. Methodology

This project will be written using a Linux operating system, Ubuntu.

A. Docker: The development environment

Docker [3] is a tool designed to make it easier to create, deploy, and run applications by using containers [4]. Containers allow us to package up an application with all the parts it needs, such as libraries and other dependencies, and ship it all out as one package. Thanks to its environment management, the fact that Docker let us deploy server environments in containers rapidly and the ease to maintain different versions of, for example, a website using Apache [5], makes it a suitable environment for this project. We can have a separate container for testing, development, and production on the same cloud architecture and easily deploy to each one.
Separating the different components of our web application service into different containers will have security benefits: if one container is compromised the others remain unaffected, therefore increasing security in many ways and avoiding conflicts with dependencies between Docker Containers since Isolation in Docker is taken care of.

B. HAProxy: Building tool

Since in this work a high availability web application will be built, we are going to use HAProxy as a tool for managing load balancing traffic. HAProxy is an open source, *very* fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for web application services. It's mode of operation makes its integration into existing architectures very easy and riskless, while still offering the possibility not to expose fragile web servers to the net.

C. ModSecurity: Building tool

As well as building the web application service, and managing its traffic load, monitorisation is very important for analysis and to take conclusions. ModSecurity is a toolkit that will be used for real-time web application monitoring, logging, and access control.

D. Elastic Stack: Data Management tool

Elastic Stack [8] is a powerful search engine which allow us to process big volumes of data. Elastic Stack is a complete end-to-end log analysis solution which helps in deep

searching, analysing and visualizing the log generated from different machines. By using the Elastic Stack we won't only be able to process big volumes of data but also deep search, analyse and visualise the logs generated from different machines.
Elastic stack reliably and securely takes data from any source, in any format, and search, analyse, and visualize it in real time. Elastic Stack provides a strong mechanism to perform centralized logging which plays an important role in identifying the web server and/or application related problems. It lets us search through all the logs at a single place and identify the issues spanning through multiple servers by correlating their logs within a specific time frame found in IT environments including use cases for web analytics, business intelligence, compliance and security.

E. Security Onion: Testing tool

Security Onion is a free and open source Linux distribution for intrusion detection, enterprise security monitoring, and log management. It includes Elasticsearch, and many other security tools.
Security Onion collects many tools for forensic analysis, both networks and systems, so that we can guarantee the proper functioning of all of them and the absence of all kinds of intruders in the net. Security Onion comes with a wide variety of packages and default tools to audit the security of all types of networks.

F. Github: Versionj Control tool

A versioning tool will be used, in order to keep copies and record of the work done over the course of this project. A repository will be used in GitHub [6] where the project will be updated, and version controlled.

IV. Planning

This work is divided into three main phases, without keeping in mind the delivery of the reports and the presentation. The phases go in line with the tools and techniques to be used in this work. Phase 1, named WAF_v1, consists in building the running Web Application service and the testing of it's functionality to corroborate that it works. Phase 2, named WAF_v2, consists in developing the high availability Web Application service with traffic load balancing and data persistence taken into consideration. In Phase 3, named WAF_v3, the integration of the WAF to the Elastic Stack, in order to manage logging information and the deployment of ModSecurity and SecurityOnion to the WAF for real-time analysis and monitoring of traffic data will be developed.

A. Phases and processes

- Define objectives and requirements: these objectives and requirements are: the ones we want to also satisfy, and those new ones required while the project is in development
- The Development of the WAF_v1 architecture: A scalable client-server web application architecture, known as WAF_v1 in the project, is going to be built and tested. The WAF_v1 consists in 1 HAProxy node and 2 Apache servers responding to client http requests. No high availability will be developed.
- WAF_v2 Structure proposal: give a first proposal of the Web Application Firewall and the firewall's structure to achieve the objectives.
- The Development of the WAF_v2 architecture: In this version of the Web Application Firewall front-end high availability, data persistency, layer 4 load balancing, logging and http traffic rate limitation will be taken care of.
- Architecure justification: justify the structure proposed explaining the decisions made and the reasons why the structure is suitable.
- Initial project test: test that the project satisfies the objectives and that works for at least more than one product.
- The Development of the WAF_v3 architecture: integration of the WAF_v2 architecture to the Elastic Stack, in order to manage logging information.
- Data monitorization: Real-time analysis and monitoring of traffic data will be managed using ModSecurity and SecurityOnion.
- Final structure and architecture: correct all possible mistakes seen before, and make those modifications needed.
- Final project design and testing: test the project and verify all requirements are satisfied.
- Documentation and justification: write all the documentation required, also the manual guide for the future users, and the final architecture justification.

B. Schedule

The methodology that will be used to develop this project is the Gantt chart [11], which is a useful way of scheduling a project and defining the different dependencies between tasks. Here we have all the tasks, before mentioned, listed and with their start and end date, and the Gantt chart with its critical path. In the appendix there is a Gantt diagram, in which it you can see the tasks and the planning in weeks of the work.

References

[1] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[2] Peters, J. (2019). *What is a Proxy Server and How Does it Work?*. [online] Inside Out Security. Available at: https://www.varonis.com/blog/what-is-a-proxy-server/ [Accessed 17 Sep. 2019].

[3] Hat, R. (2019). *¿Qué es Docker?*. [online] Redhat.com. Available at: https://www.redhat.com/es/topics/containers/what-is-docker [Accessed 22 Sept. 2019].

[4] "What is a Container? | Docker", *Docker*, 2019. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed: 15 Sept- 2019]

[5] "Welcome to The Apache Software Foundation!," *The Apache Software Foundation*. [Online]. Available: http://apache.org/. [Accessed: 25-Sep-2019].

[6] Repository of Hernan's Espinosa Work. https://github.com/HernanEspinosa/Web_Application_Firewall.git. Last Acces: October 2, 2019.

ANNEX