Maestría en Ciberseguridad

Trabajo previo a la obtención de título de
Magister en Ciberseguridad


**AUTORES:**

Brayan Santiago Altamirano Rodríguez

Carmen Leonor Cepeda Altamirano

Catherine Alejandra Cadena Paillacho

Grace Susana Flores Cruz

**TUTORES:**

Iván Reyes Chacón

Alejandro Cortés López


**TEMA:**

# Diseño e implementación de un entorno de pruebas para evaluar la efectividad de un Web Application Firewall (WAF) en la detección y mitigación de ataques de API REST

diciembre – 2025

REINVENTEMOS EL FUTURO

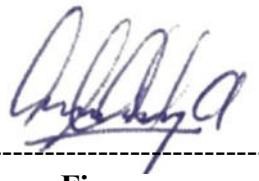
## Certificación de autoría

Nosotros, **Brayan Santiago Altamirano Rodríguez, Carmen Leonor Cepeda Altamirano, Catherine Alejandra Cadena Paillacho, Grace Susana Flores Cruz,** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

---

**Firma**
**Brayan Santiago Altamirano Rodríguez**

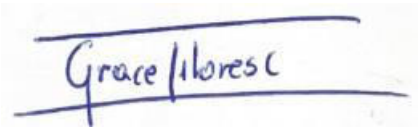


---

**Firma**
**Carmen Leonor Cepeda Altamirano**

---

**Firma**
**Catherine Alejandra Cadena Paillacho**



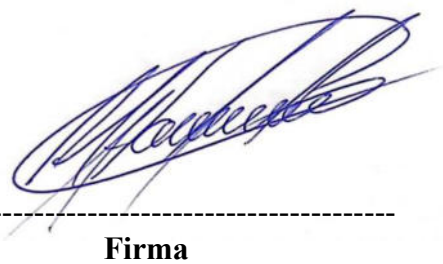

---

**Firma**
**Grace Susana Flores Cruz**

**Autorización de Derechos de Propiedad Intelectual**

Nosotros, **Brayan Santiago Altamirano Rodríguez, Carmen Leonor Cepeda Altamirano, Catherine Alejandra Cadena Paillacho, Grace Susana Flores Cruz,** en calidad de autores del trabajo de investigación titulado ***Diseño e implementación de un entorno de pruebas para evaluar la efectividad de un Web Application Firewall (WAF) en la detección y mitigación de ataques de API REST***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

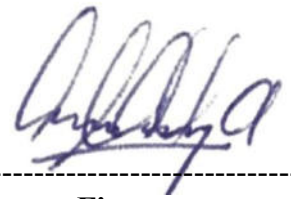
D. M. Quito, (mes año)



---

**Firma**

**Brayan Santiago Altamirano Rodríguez**



---

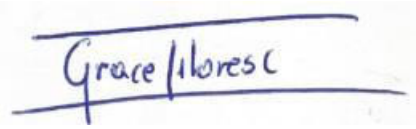
**Firma**

**Carmen Leonor Cepeda Altamirano**

---

**Firma**

**Catherine Alejandra Cadena Paillacho**




---

**Firma**

**Grace Susana Flores Cruz**

**Aprobación de dirección y coordinación del programa**

Nosotros, **Alejandro Cortes López Director EIG e Iván Reyes Chacón Coordinador UIDE,** declaramos que: **Brayan Santiago Altamirano Rodríguez, Carmen Leonor Cepeda Altamirano, Catherine Alejandra Cadena Paillacho, Grace Susana Flores Cruz** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.

Alejandro Cortés L.

Maestría en Ciberseguridad

Iván Reyes Ch.

Maestría en Ciberseguridad

## Dedicatoria

Este proyecto está dedicado a nuestras familias, por su apoyo, comprensión y paciencia en esta trayectoria. Su confianza fue fundamental para alcanzar esta meta planteada en nuestras vidas.

Finalmente dedicamos este trabajo a quienes creyeron en nosotros, recordándonos que los sueños se alcanzan con disciplina y fe.

## Agradecimiento

Un profundo agradecimiento a nuestros docentes por guiarnos en el camino del conocimiento, por compartir sus experiencias y motivarnos al crecimiento académico y profesional.

A nuestros compañeros y amigos, por el apoyo, su contribución con sus ideas y la colaboración brindada durante este proceso.

A la Institución, por impulsar el desarrollo de la innovación y fomentar la formación integral, permitiéndonos generar conocimientos y propuestas que contribuyan al progreso y desarrollo de nuestro país.

## Resumen

La implementación de un WAF (Web Application Firewall) constituye solo un componente dentro del ecosistema de seguridad necesario para proteger una API en un entorno digital caracterizado por amenazas y vulnerabilidades crecientes. Este proyecto presenta un marco teórico que aborda conceptos clave relacionados con API REST, WAF, herramientas y normativas aplicables, complementado con un laboratorio práctico para evaluar y comparar la efectividad de dos soluciones: Cloudflare y Safeline. El ejercicio se realizó mediante el envío de solicitudes web legítimas y maliciosas a una API REST vulnerable, con el objetivo de analizar la capacidad de detección y mitigación de ataques por parte de cada WAF. Los resultados obtenidos buscan aportar conocimiento a profesionales y organizaciones interesadas en fortalecer la seguridad de sus aplicaciones, promoviendo la mejora continua. Finalmente, se concluye que la mejor estrategia para proteger aplicaciones web consiste en integrar soluciones basadas en inteligencia artificial, aprendizaje automático, incorporar controles en los desarrollos, actualizaciones constantes y cumplimiento normativo, fortaleciendo la seguridad en las aplicaciones, así como la resiliencia frente a amenazas e incidentes.

***Palabras Claves:*** API REST, Firewall de Aplicación Web, detección de ataques, mitigación de ataques, seguridad web

## Abstract

The implementation of a WAF (Web Application Firewall) is only one component within the security ecosystem necessary to protect an API in a digital environment characterized by growing threats and vulnerabilities. This project presents a theoretical framework that addresses key concepts related to REST API, WAF, tools and applicable regulations, complemented by a practical laboratory to evaluate and compare the effectiveness of two solutions: Cloudflare and Safeline. The exercise was carried out by sending legitimate and malicious web requests to a vulnerable REST API, with the aim of analyzing the ability of each WAF to detect and mitigate attacks. The results obtained seek to provide knowledge to professionals and organizations interested in strengthening the security of their applications, promoting continuous improvement. Finally, it is concluded that the best strategy to protect web applications consists of integrating solutions based on artificial intelligence, machine learning, incorporating controls in developments, constant updates and regulatory compliance, strengthening security in applications as well as resilience against threats and incidents.


***Keywords:*** *API REST, Attack detection, Attack mitigation, Web Application Firewall, Web security*

# TABLA DE CONTENIDOS

# LISTA DE TABLAS

# LISTA DE FIGURAS

# Capítulo 1

## 1. Introducción

### *1.1. Definición del Proyecto*

El presente proyecto tiene como objetivo el diseño e implementación de un entorno de pruebas controlado que permita evaluar la efectividad de un Web Application Firewall (WAF) en la prevención de ataques de tipo API Injection dirigidos a servicios web que implementan la arquitectura REST.

En un contexto en el que las APIs REST se han convertido en componentes críticos de las arquitecturas modernas como microservicios, aplicaciones móviles y sistemas distribuidos, la seguridad en la capa de exposición adquiere una relevancia fundamental. Uno de los vectores de ataque más comunes contra este tipo de APIs es la inyección de código malicioso en los parámetros de entrada, la cual puede manifestarse en diversas formas, tales como SQL Injection, Command Injection, JSON Injection, Header Injection, entre otras. Estos ataques buscan explotar debilidades en la validación de entradas con el fin de comprometer la lógica del servidor, acceder a información confidencial o alterar el comportamiento del sistema.

Para abordar esta problemática, el entorno de pruebas incluirá el desarrollo de una API REST funcional que simule un caso de uso real, así como la configuración de dos WAFs que actúen como sistema de defensa perimetral frente a peticiones maliciosas.

En definitiva, la finalidad de este proyecto es generar un análisis objetivo sobre la eficacia de las soluciones WAF en la protección de APIs REST ante amenazas basadas en inyección, y ofrecer recomendaciones para su implementación como parte de una estrategia integral de seguridad en el desarrollo y despliegue de servicios web.

### *1.2. Justificación e Importancia del Trabajo de Investigación.*

La relevancia de este trabajo se fundamenta en la creciente exposición de datos e infraestructuras críticas a través de APIs REST. Organizaciones de todos los sectores utilizan APIs para habilitar servicios digitales; sin embargo, en muchos casos lo hacen sin aplicar mecanismos robustos de seguridad, lo que incrementa el riesgo de ataques de inyección. De acuerdo con reportes de seguridad como el OWASP API Security Top 10, la inyección continúa siendo una de las principales amenazas para las aplicaciones modernas, generando incidentes de filtración de datos, fraude y compromisos de sistemas.

En este contexto, el WAF actúa como una capa de defensa que intercepta tráfico malicioso antes de que alcance la aplicación, lo cual resulta particularmente útil en escenarios donde no siempre es posible modificar el código fuente de la API.

Asimismo, esta investigación busca ofrecer un aporte académico y profesional al área de la ciberseguridad, sirviendo como guía para desarrolladores, arquitectos de software y especialistas en seguridad que requieren implementar medidas prácticas y costo-efectivas contra ataques de inyección. Además, se pretende generar un marco de referencia replicable en distintos contextos empresariales, contribuyendo a la disminución de vulnerabilidades comunes en las APIs REST.

### *1.3. Alcance*

El presente proyecto abarca el diseño, implementación y validación de un entorno de laboratorio virtualizado, orientado a la evaluación de la efectividad de dos Web Application Firewall (WAF) en la protección de APIs REST frente a diferentes vectores de ataque. El alcance

se limita a un ambiente controlado de pruebas, sin despliegue en entornos productivos, garantizando así condiciones reproducibles y seguras.

El laboratorio estará conformado por los siguientes componentes:

- Un servidor con API REST vulnerable
- Dos Web Application Firewall (Comercial y Código Abierto)
- Una máquina atacante equipada con herramientas de pentesting

Las pruebas que se ejecutarán en el laboratorio incluirán, de manera controlada, los siguientes vectores de ataque:

- Inyecciones SQL y NoSQL en parámetros de entrada.
- Fuzzing de parámetros y manipulación de payloads.
- Cross-Site Scripting (XSS) en entornos de API REST.
- Payload

Los resultados se medirán en función de las siguientes variables de interés:

- Capacidad de detección y mitigación de ataques por parte del WAF.
- Tasa de falsos positivos y falsos negativos, tasa de bloqueo con el fin de analizar la precisión del WAF.
- Tasa de Negativos verdaderos de los WAF.
- Precisión equilibrada.

## *1.4. Objetivos*

### Objetivo general

Diseñar e implementar un entorno de pruebas controlado para evaluar la efectividad de dos Web Application Firewall (WAF) en la detección y mitigación de ataques dirigidos a servicios API REST, con el propósito de obtener métricas de desempeño que contribuyan al fortalecimiento de la seguridad en aplicaciones web.

### Objetivo específico

- Implementar un entorno controlado con una API REST vulnerable, un WAF y herramientas de ataque.
- Configurar los WAF con diferentes reglas de seguridad para la protección de APIs REST.
- Ejecutar pruebas de ataque controladas de acuerdo con el alcance establecido contra la API.
- Analizar los resultados obtenidos de detección, bloqueo y falsos positivos.
- Proponer recomendaciones de configuración y buenas prácticas para el uso de WAF en entornos con APIs REST.

# Capítulo 2

## 2. Marco Teórico

### *2.1 Evolución de las aplicaciones web hacia arquitecturas basadas en APIs.*

A lo largo de los últimos treinta años, las aplicaciones web han experimentado cambios importantes en su diseño, funcionamiento e integración con otros sistemas. Estos cambios han llevado a que las arquitecturas modernas estén construidas alrededor de APIs, que hoy son el eje central del desarrollo de software, la interoperabilidad y la escalabilidad empresarial. Esta evolución no ocurrió de la noche a la mañana; más bien, fue el resultado de transformaciones motivadas por las necesidades tecnológicas y de negocio.

Para mejorar la distribución de recursos se comenzó a experimentar métodos más simples, ligeros y flexibles, lo que finalmente abrió el camino hacia las APIs modernas que se conocen hoy en día. Este cambio transformó por completo la forma en que las aplicaciones se comunican entre sí (Fielding, 2000).

En la Tabla 1 se puede observar la evolución que han tenido las API en los últimos 30 años, desde las primeras aplicaciones hasta las integraciones actuales y que aún se están utilizando a nivel de desarrollo y corporativo.

**Tabla 1**

*Evolución Histórica de las Arquitecturas Web hacia Modelos Basados en APIs*

| Etapa histórica | Periodo aproximado | Características principales | Limitaciones |
|---|---|---|---|
| Aplicaciones monolíticas (cliente-servidor) | 1990–2005 | Código integrado en un solo bloque; el servidor generaba páginas completas HTML; poco modularidad. | Baja escalabilidad, difícil mantenimiento, poca integración con otros sistemas. |
| Servicios Web SOAP | 2000–2010 | Uso de XML, WSDL y protocolos estrictos; integración empresarial estándar. | Complejidad elevada, bajo rendimiento, difícil adopción en web ágil. |
| AJAX y Web dinámica | 2005–2012 | Intercambio de datos sin recargar páginas; uso de JSON; primeras separaciones frontend-backend. | Comunicación aún acoplada y sin arquitectura uniforme. |
| REST y APIs modernas | 2010–presente | Comunicación ligera usando HTTP; JSON; endpoints independientes; auge móvil y microservicios. | Requiere diseño apropiado para evitar inconsistencias. |
| Microservicios impulsados por APIs | 2015–presente | Sistemas distribuidos, escalables y desacoplados; APIs como contrato entre servicios. | Complejidad operativa en despliegue y monitoreo. |
| APIs emergentes: GraphQL, gRPC, WebSockets | 2018–presente | Consultas flexibles (GraphQL), alto rendimiento (gRPC), tiempo real (WebSockets). | Requieren nuevos enfoques de diseño y monitoreo. |

### Aplicaciones monolíticas y el modelo cliente-servidor

Entre los años 1990 – 2005 se crearon las primeras aplicaciones web, que tenían un enfoque monolítico donde toda la lógica de progresión, usuarios y accesos a datos se integraban en un único sistema. Estas aplicaciones transferían páginas completas al navegador, utilizando tecnologías como CGI, PHP, ASP o JSP.

**Características principales:**

- Arquitectura exacta y poco escalable.
- Comunicación centrada en HTML transmitido desde el servidor.
- Integración limitada entre sistemas y generalmente manual.
- Poca reutilización de componentes.

Estas aplicaciones resolvían necesidades básicas, pero no estaban preparadas para integrarse fácilmente con sistemas externos ni para soportar cargas crecientes.

### Web dinámica y servicios web SOAP

Ente los años 2000 – 2010 la siguiente etapa en la evolución fue la aparición de servicios web basados en SOAP (Simple Object Access Protocol). Estos servicios permitieron por primera vez la comunicación estandarizada entre aplicaciones a través de Internet.

**Aportes clave de SOAP:**

- Implementación de XML como lenguaje de intercambio de datos.
- Definición formal de interfaces mediante WSDL.
- Estándares de seguridad, transaccionalidad y mensajería confiable.

Aunque robustos, los servicios SOAP eran complejos de implementar y poco eficientes para aplicaciones web que necesitaban fluidez y velocidad.

### AJAX y la web dinámica

Entre los años 2005 – 2012 Ajax marcó un cambio radical en la experiencia del usuario. Dio la posibilidad de realizar actualizaciones por partes sin la necesidad de recargar la página completamente. Por lo cual se vio una diferencia con interfaces rápidas más asemejadas a lo que tenemos actualmente.

**Aportes clave de AJAX y la web dinámica:**

- Cambio s XML a JSON.
- Primeros inicios en la separación del frontend y backend
- Aplicaciones interactivas.

A pesar de los avances, todavía no existía una arquitectura uniforme. La comunicación seguía siendo relativamente acoplada y los servicios no estaban definidos como componentes independientes, lo que impedía la evolución hacia sistemas más escalables.

### REST y surgimiento de las APIs modernas

Desde el 2010 hasta la actualidad, la arquitectura REST no fue un cambio tecnológico fue una nueva manera de comunicación entre sistemas, exponer recursos mediante endpoints, usando HTTP de forma nativa y con datos en JSON, simplificó enormemente la integración.

**Aportes de la arquitectura REST y surgimiento de las APIs modernas**

- Incorporación con aplicaciones móviles.
- Comunicación directa con el backend sin depender de una lógica de presentación
- Escalable para la comunicación entre ecosistema como web, IoT, aplicaciones

móviles.

**Microservicios impulsados por APIs**

Desde el año 2015 - hasta la actualidad muchas organizaciones comenzaron a adoptar los microservicios como respuesta a la necesidad de contar con sistemas que pudieran crecer sin tener que ser reconstruidos por completo. Este principio divide una aplicación en varios componentes pequeños que se comunican entre si a través de APIs. gracias a ello, es posible actualizar, escalar o incluso reemplazar partes específicas del sistema sin afectar su funcionamiento general.

**Aportes de los Microservicios**

- Independencias de sistemas.
- Permite actualizar, escalar o reemplazar servicios sin afectar al resto.
- Implementado en empresas con altos volúmenes de tráfico.
- Monitoreo avanzado necesario gestionar redes internas y balanceadores de carga.

**Generaciones de APIs: GraphQL, gRPC y WebSockets.**

Desde el 2018 – hasta la actualidad por necesidades más específicas, comenzaron a surgir alternativas al modelo REST tradicional:

- GraphQL permite consultas precisas, solicitando exactamente los datos que se necesitan, lo que resulta ideal para interfaces complejas.
- gRPC, basado en HTTP/2, ofrece gran rendimiento y es adecuado para comunicación entre microservicios o dispositivos con baja latencia.
- WebSockets facilita la transmisión en tiempo real, una funcionalidad clave en

aplicaciones de mensajería, paneles en vivo o juegos online.

Aunque estas tecnologías abren nuevas posibilidades, también requieren nuevas prácticas de diseño y monitoreo, lo que implica un esfuerzo adicional para las organizaciones.

### *2.2 Diferencias entre aplicaciones tradicionales y APIs REST.*

Las aplicaciones tradicionales (web clásicas) suelen estar construidas como monolitos donde la lógica de presentación, negocio y acceso a datos conviven en el mismo bloque o despliegue, muchas veces acopladas al servidor que genera y envía páginas al cliente.
Por contraste, una API REST es un estilo arquitectónico que permite que clientes (web, móviles) consuman recursos expuestos por un servidor de forma ligera, mediante HTTP, con comunicación desacoplada, stateless y usualmente en formatos como JSON. Por ejemplo: las API REST identifican recursos mediante URI, usan verbos HTTP (GET, POST, PUT, DELETE) y no mantienen estado del cliente en el servidor

Entre las diferencias más importantes:

- Coordinación entre el cliente y el servidor: Las API REST tienen una arquitectura poco acoplada, que permite el desarrollo independiente en el lado del cliente y del servidor. Con las API Web, los cambios entre el cliente y el servidor se coordinan de forma más precisa.

- Interfaz: Dependiendo de los detalles de implementación, las API REST tienden a utilizar interfaces estándar del sector. Las API Web utilizan interfaces personalizadas, dependiendo del proveedor de la API.

- Comunicación: Las API Web son lo suficientemente flexibles como para aprovechar cualquier estilo de comunicación, mientras que las API REST se utilizan principalmente

con JSON, XML y texto sin formato. Estas opciones significan que las API REST funcionan bien para la transmisión de datos textuales, como las operaciones de creación, lectura, actualización y eliminación (CRUD) contra una base de datos, pero son más restrictivas cuando se trata de datos binarios. Las API Web ofrecen una experiencia mucho mejor para los servicios que requieren datos binarios — como servicios de streaming de vídeo o música — ya que admiten más formatos de mensaje.

### *2.3 Definición de API REST.*

Una API REST (Application Programming Interface - Representational State Transfer) es una implementación de la arquitectura de software REST que permite que las aplicaciones accedan y manipulen los recursos a través de una serie de operaciones estándar, como GET, POST, PUT y DELETE.

La API REST se utiliza para crear servicios web que pueden ser accedidos a través de Internet y que siguen los principios de la arquitectura REST. La API REST permite que las aplicaciones se comuniquen con los servidores web mediante solicitudes HTTP a recursos específicos y que se identifican mediante URIs (Uniform Resource Identifiers).

La API REST utiliza formatos de intercambio de datos comunes como JSON (JavaScript Object Notation) y XML (Extensible Markup Language) para la transferencia de datos entre aplicaciones. Esto permite que diferentes sistemas y lenguajes de programación puedan interactuar entre sí de forma sencilla y consistente.

Una de las características clave de una API REST es que es stateless, lo que significa que no mantiene información sobre el estado de las conexiones entre el cliente y el servidor. Cada

solicitud que se realiza al servidor contiene toda la información necesaria para completar la operación, lo que hace que los servicios web sean altamente escalables y tolerantes a fallos.

La API REST también se basa en el concepto de recursos, que son objetos o datos que pueden ser manipulados a través de una interfaz uniforme. Cada recurso tiene una identificación única (URI) y puede ser accedido mediante operaciones estándar de HTTP. Las operaciones CRUD (Crear, Leer, Actualizar y Borrar) son las principales operaciones que se utilizan para manipular los recursos en una API REST.

La API REST es considerada una de las más utilizadas y populares debido a su simplicidad, flexibilidad y capacidad de adaptación a diferentes situaciones. Es compatible con una amplia variedad de plataformas y lenguajes de programación, lo que la convierte en una opción atractiva para desarrolladores y empresas que buscan crear servicios web escalables y eficientes.

### *2.4 OWASP API Security Top 10*

El proyecto OWASP dedica un listado “Top 10” específico para riesgos de seguridad en APIs, que identifica las amenazas más críticas.

Los 10 principales riesgos de seguridad de las APIs de OWASP – 2023:

- **"API1:2023 - Autorización a nivel de objeto rota:** Las APIs tienden a exponer endpoints que gestionan identificadores de objetos, creando una amplia superficie de ataque de problemas de Control de Acceso a Nivel de Objeto. Las comprobaciones de autorización a nivel de objeto deben considerarse en cada función que accede a una fuente de datos usando un ID del usuario.

- **API2:2023 - Autenticación rota:** Los mecanismos de autenticación suelen implementarse de forma incorrecta, permitiendo a los atacantes comprometer tokens de autenticación o explotar fallos de implementación para asumir temporal o permanentemente la identidad de otros usuarios. Comprometer la capacidad de un sistema para identificar al cliente/usuario compromete la seguridad de la API en general.
- **API3:2023 - Autorización de nivel de propiedad de objeto roto:** Esta categoría combina API3:2019 Exposición Excesiva de Datos y API6:2019 - Asignación Masiva, centrándose en la causa raíz: la falta o una validación incorrecta de autorización a nivel de propiedad del objeto. Esto conduce a la exposición o manipulación de información por parte de partes no autorizadas.
- **API4:2023 - Consumo de Recursos Sin Restricciones:** Satisfacer las solicitudes de API requiere recursos como ancho de banda de red, CPU, memoria y almacenamiento. Otros recursos como correos electrónicos, SMS, llamadas telefónicas o validación biométrica están disponibles por los proveedores de servicios mediante integraciones API y se pagan por solicitud. Los ataques exitosos pueden llevar a la denegación de servicio o a un aumento de los costes operativos.
- **API5:2023 - Autorización de nivel de función rota:** Las políticas complejas de control de acceso con diferentes jerarquías, grupos y roles, y una separación poco clara entre funciones administrativas y regulares, tienden a provocar fallos en la autorización. Al explotar estos problemas, los atacantes pueden acceder a los recursos y/o funciones administrativas de otros usuarios.

- **API6:2023 - Acceso Irrestricto a Flujos Empresariales Sensibles:** Las APIs vulnerables a este riesgo exponen un flujo de negocio —como comprar un ticket o publicar un comentario— sin compensar cómo la funcionalidad podría perjudicar al negocio si se usa de forma excesiva de forma automatizada. Esto no viene necesariamente por errores de implementación.
- **API7:2023 - Falsificación de peticiones en el lado del servidor:** Las fallas de falsificación de solicitudes en el lado del servidor (SSRF) pueden aparecer cuando una API recupera un recurso remoto sin validar el URI suministrado por el usuario. Esto permite a un atacante coaccionar a la aplicación para que envíe una solicitud elaborada a un destino inesperado, incluso cuando está protegida por un cortafuegos o una VPN.
- **API8:2023 - Error de configuración de seguridad:** Las APIs y los sistemas que las soportan suelen contener configuraciones complejas, diseñadas para hacer que las APIs sean más personalizables. Los ingenieros de software y DevOps pueden pasar por alto estas configuraciones o no seguir las mejores prácticas de seguridad en cuanto a la configuración, abriendo la puerta a diferentes tipos de ataques.
- **API9:2023 - Gestión incorrecta de inventario:** Las APIs tienden a exponer más endpoints que las aplicaciones web tradicionales, lo que hace que la documentación adecuada y actualizada sea muy importante. Un inventario adecuado de los hosts y versiones de API desplegadas también es importante para mitigar problemas como versiones obsoletas de la API y puntos finales de depuración expuestos.
- **API10:2023 - Consumo inseguro de APIs:** Los desarrolladores tienden a confiar más en los datos recibidos de APIs de terceros que en la entrada del usuario, por lo que tienden a

adoptar estándares de seguridad más débiles. Para comprometer APIs, los atacantes van a por servicios integrados de terceros en lugar de intentar comprometer directamente la API objetivo. " (OWASP Foundation, 2023)

### *2.5 Web Application Firewall (WAF)*

**Definición y características.**

Firewall de aplicaciones web WAF es una solución fundamental en la seguridad web, actúa como una barrera protectora supervisando, filtrando y bloqueando el tráfico entre Internet y las aplicaciones web, APIs o aplicaciones móviles para protegerlas en tiempo real contra amenazas, contribuyendo al rendimiento y a optimizar el procesamiento de solicitudes.

Funciona en la capa 7 del modelo OSI, y protege contra ataques de capa de aplicación, como: inyección de SQL, inclusión de archivos, scripts entre sitios (XSS), envenenamiento de cookies, etc.

Uno de los principales beneficios del WAF en la seguridad web es su capacidad para detectar y mitigar ataques en tiempo real, como exploits de día cero, vulnerabilidades comunes, bots maliciosos, y malware.

El WAF inspecciona la capa de aplicación de la red y puede eludir muchos ataques sospechosos (que son invisibles para otros tipos de firewalls) y los bloquea automáticamente, reduciendo el riesgo de brechas de datos para proteger la integridad del sitio. Los WAF pueden bloquear los intentos maliciosos de acceder a datos sensibles y prevenir ataques que conduzcan a violaciones de seguridad.

Un WAF complementa el modelo de confianza cero al inspeccionar el tráfico de la capa de aplicaciones, identificando patrones e integrándose con otros sistemas de seguridad e inteligencia sobre amenazas para ofrecer protección en tiempo real.

WAF mejora el rendimiento web al optimizar el flujo de datos y reducir la carga en el servidor. Al filtrar tráfico no deseado, libera recursos y acelera los tiempos de respuesta, dando como resultado una experiencia de usuario más fluida.

**Arquitectura y Funcionamiento.**

Un firewall de aplicaciones web (WAF) inspecciona las solicitudes de tráfico y opera por medio de un conjunto de reglas, normalmente denominadas directivas o políticas de seguridad. Las reglas del WAF se utilizan para definir cómo inspeccionar el tráfico web HTTP/HTTPS (solicitudes) a una aplicación, dónde y qué parámetros y condiciones buscar en la solicitud y qué acción debe tomar cuando una solicitud coincide con esas definiciones. Las directivas tienen el propósito de proteger al servidor web, contra vulnerabilidades en la aplicación al filtrar y bloquear el tráfico malicioso. El valor de un WAF procede, en parte, de la velocidad y facilidad con que se pueden aplicar modificaciones en las directivas que permiten una respuesta más rápida ante diversos vectores de ataque.

El WAF revisa los encabezados, las cadenas de consulta y elementos claves para identificar patrones sospechosos. Elementos clave como: GET (para obtener o recuperar datos de un servido), PUT (para crear nuevos recursos o actualizar datos existentes en el servidor), POST (para enviar datos a un servidor, lo que a menudo da lugar a un cambio de estado) y DELETE (para eliminar datos),

WAF comúnmente se implementa como un proxy inverso, inspeccionando las solicitudes entrantes antes de que lleguen al servidor, se implementa en línea entre los clientes y las aplicaciones web para filtrar el tráfico no seguro, al tiempo que permite el paso de solicitudes legítimas. WAF se puede implementar como software, hardware o servicio en la nube.

La evolución constante de aplicaciones, servicios y amenazas hace necesario que las políticas de seguridad se actualicen periódicamente de forma manual o utilizando aprendizaje automático. Actualizar una política de seguridad significa actualizar y ajustar el conjunto de reglas que componen dicha política para responder a las amenazas y necesidades de seguridad además de brindar protección automática.

Las soluciones de WAF avanzado deben ser capaces de utilizar la automatización para reducir el coste de propiedad y evitar los errores humanos asociados a procesos manuales de definición de políticas y reglas de seguridad. La generación automática de políticas suele basarse en la capacidad de identificar y perfilar las transacciones legítimas de las aplicaciones

Cuanto mejor sea el WAF, más amplio, robusto y relevante será el conjunto de reglas predefinido para responder a las amenazas de ciberseguridad más actuales. Aunque estas reglas están predefinidas, los proveedores de WAF deben actualizarlas constante y continuamente para garantizar que sus clientes estén protegidos contra las amenazas más recientes.

**Ventajas y Desafíos de Implementar un WAF.**

La implementación de un Web Application Firewall (WAF) ofrece múltiples beneficios para la protección de aplicaciones web y APIs, fortaleciendo la postura de seguridad organizacional. Para elegir la solución de seguridad adecuada, se debe asegurar que ofrezca capacidades como:

- Integración con la infraestructura existente en una organización. Los WAF se pueden configurar a nivel de red con cambios mínimos o nulos en las aplicaciones preexistentes
- Protección del borde contra una amplia gama de amenazas como ataques DDoS y actores maliciosos ubicados en el perímetro de red o en el perímetro de la nube.
- Protección frente a amenazas comunes, seguridad mejorada y mitigación de amenazas: los WAF detectan y bloquean ataques basados en las vulnerabilidades de OWASP Top 10. Esto reduce de manera significativa la superficie de ataque ofreciendo prevención frente a ataques de día cero.
- Seguridad adaptada al tráfico de aplicaciones y APIs

  Los WAF modernos analizan el tráfico en tiempo real, permitiendo identificar patrones anómalos y ajustar las políticas de defensa según el comportamiento del usuario y del tráfico legítimo.
- Mitigación de ataques automatizados y de bots

  Un WAF avanzado puede diferenciar entre usuarios reales y tráfico automatizado, limitando intentos de fuerza bruta, scraping o abuso de recursos mediante herramientas de bot management y rate limiting, bloquean bots y restringen el acceso a regiones conocidas por lanzar ataques.
- Visibilidad, monitoreo y control centralizado: Los equipos de seguridad pueden gestionar reglas y aplicar políticas desde una plataforma unificada.
- notificar en tiempo real los ataques para que al equipo de seguridad permite tomar

decisiones basadas en datos, detectar comportamientos anómalos, e implementar medidas proactivas de mitigación.

- Cumplimiento de requisitos normativo y regulatorio: Facilita el cumplimiento de estándares de seguridad como PCI DSS, ISO 27001 o Protección de datos personales, al proporcionar controles de protección de datos y registro de eventos de seguridad.
- Reducción de la latencia y optimización del rendimiento de las aplicaciones. Los cortafuegos gestionados de aplicaciones web actualizan continuamente las reglas para proteger frente a vulnerabilidades y exploits de día cero antes de que los parches/actualizaciones estén disponibles,
- Implementación, gestión y escalabilidad rentables. los WAF permiten escalar las operaciones rápidamente a medida que cambian las necesidades de seguridad. Puede desplegarse en entornos on-premise, cloud o híbridos, integrándose con servicios como CDN, balanceadores de carga y API Gateways, sin afectar la disponibilidad o el rendimiento.

La implementación de un WAF presenta ciertos retos que deben considerarse para evitar brechas o problemas operativos:

- Configuración inicial compleja: Requiere una fase de ajuste y pruebas exhaustiva para evitar falsos positivos (bloquear tráfico legítimo) o falsos negativos (dejar pasar tráfico malicioso). La calibración incorrecta puede afectar la disponibilidad del servicio.
- Mantenimiento y actualización constante: Las amenazas están en constante evolución con patrones de ataque nuevos y avanzados. Si el WAF y sus reglas no se actualizan

de forma continua, puede volverse ineficaz frente a nuevas vulnerabilidades o técnicas de evasión.

- Impacto en el rendimiento: El análisis profundo de solicitudes HTTP/HTTPS puede aumentar la latencia, especialmente en aplicaciones con alto volumen de tráfico o payloads grandes (como APIs REST o GraphQL). Descifrar e inspeccionar el tráfico SSL (capa de sockets seguros) y TLS (Transport Layer Security) sin afectar al rendimiento es todo un reto, especialmente a medida que aumenta el tráfico.
- Dependencia del proveedor o del entorno: Algunas soluciones WAF son propietarias o dependen de infraestructuras específicas (por ejemplo, un WAF administrado en la nube), lo que puede limitar la portabilidad o la personalización de políticas.
- Costo operativo: Los costos pueden aumentar según el volumen de tráfico inspeccionado, el número de reglas activas y los servicios adicionales como mitigación DDoS, análisis avanzado de comportamiento o incorporación de inteligencia artificial
- Contexto restringido: Los WAF solo analizan el contenido de las solicitudes, lo que los hace vulnerables a ataques que explotan la lógica de las aplicaciones.
- Minimizar los falsos positivos: Bloquear el tráfico legítimo puede frustrar a los usuarios y a los equipos de seguridad. Esto hace que sea difícil mantener la confianza y el tiempo de actividad. En ocasiones, la detección basada en firmas puede generar falsos positivos, bloqueando tráfico legítimo
- Mantener el tiempo de actividad en entornos de alta disponibilidad: Un único punto de fallo en la implementación de WAF puede afectar a las operaciones comerciales.

- Brechas en la integración del centro de operaciones de seguridad (SOC): Una configuración incorrecta puede impedir que los WAF envíen datos detallados a nivel de aplicación al SIEM o plataformas de monitoreo limitando la gestión y monitoreo especializado
- Integración con DevOps y flujo de CI/CD: Esta integración puede ralentizar la velocidad de entrega.

**Tipos de WAF.**

Los WAF se pueden clasificar según su método de implementación, dependiendo del entorno, objetivos de seguridad y recursos disponibles, basado en la nube o en dispositivos, cada uno diseñado para proteger el tráfico HTTP/HTTPS. Los WAF basados en dispositivos pueden ser basados en software o hardware. Cada tipo de WAF tiene sus propias ventajas y desventajas.

**WAF basado en la nube (Cloud-based WAF).**

Los WAF basados en la nube son ideales para organizaciones de todos los tamaños que necesitan una protección sólida sin tener que gestionar una infraestructura local. Los WAF de nube tienen costos iniciales mínimos y suelen estar basados en suscripciones, basadas en la seguridad como servicio. Están alojadas en proveedores externos y el tráfico entrante se enruta a través de la red para su inspección antes de llegar al servidor, lo que significa que las amenazas se filtran de forma temprana, sin consumir recursos internos.

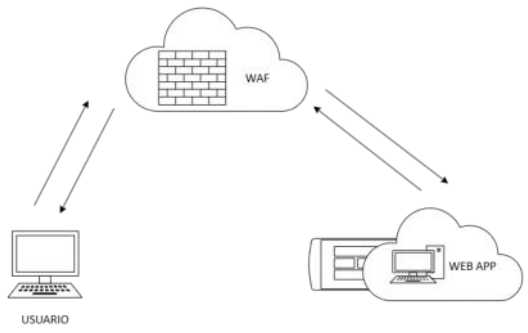
Además de ser altamente escalable, es una opción de implementación asequible y fácil, que se puede desplegar rápidamente. Según Coherent Market Insights, la implementación de WAF basada en la nube captará alrededor del 61,3 % de la cuota de mercado mundial en 2025. (Coherent Market Insights, 2025)

Varios proveedores incluyen servicios adicionales como acceso a inteligencia sobre amenazas constantemente actualizada y pueden ofrecer servicios gestionados para ayudar a definir reglas de seguridad y responder a los ataques a medida que se producen, sin embargo, dado que el WAF opera fuera del entorno, puede limitar la visibilidad y el control de la organización.

Hay dos tipos de implementaciones de WAF basado en la nube: en línea y fuera de ruta. El WAF basado en la nube ofrece la opción de implementarse en línea o como un servicio basado en API, fuera de la ruta (OOP). Una implementación basada en API y OOP ofrece varias ventajas únicas que permiten optimizarla para entornos multi-cloud, entornos locales, entornos híbridos, etc.

**Figura 1**

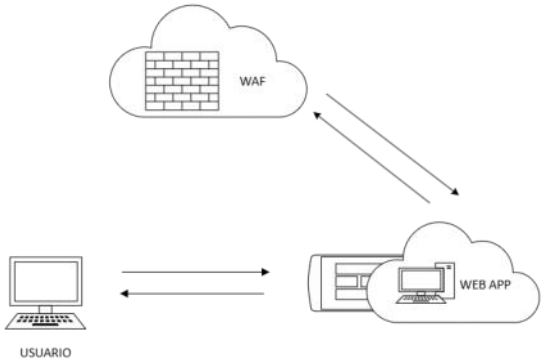
*WAF basado en nube, en línea*




Implementación de WAF basado en la nube en línea

**Figura 2**

*WAF en nube fuera de ruta y basado en API*




Implementación de WAF en la nube fuera de la nube y basado en API

**WAF basado en dispositivos.**

Los WAF basados en dispositivos son una buena opción para equipos que deben cumplir requisitos de cumplimiento estrictos u objetivos de rendimiento específicos. se utilizan habitualmente en entornos empresariales en los que la seguridad, rendimiento y mitigación de amenazas en tiempo real son fundamentales. Esta arquitectura garantiza que el tráfico se analice y filtre de forma proactiva. Esto lo hace ideal para redes de procesamiento elevado y baja latencia. Son útiles para organizaciones que desean tener un control total sobre cómo se filtra el tráfico y se gestiona la seguridad.

El WAF se Implementa en las instalaciones colocándolo directamente entre el cliente y la aplicación web, ya sea como dispositivo físico o como dispositivos virtuales, permite la integración con herramientas existentes y la personalización completa de las reglas y

políticas. Sin embargo, conlleva una sobrecarga operativa, que debe tener en cuenta el coste del mantenimiento periódico, las actualizaciones manuales y la escalabilidad limitada.

Pueden Introducir un posible punto de fallo, a menudo son necesarias configuraciones redundantes como las configuraciones activas-activas o activas-pasivas. Si el WAF deja de funcionar, el tráfico podría verse interrumpido. Por lo tanto, los sistemas de respaldo son cruciales para garantizar una alta disponibilidad y evitar tiempos de inactividad.

Pueden ser de dos tipos:

WAF basado en red (Network-based WAF): implementa en la capa de red, normalmente como un dispositivo físico o virtual

WAF basado en host (Host-based WAF); Se instala como software en el mismo servidor que aloja la aplicación web o API.

**Protección de API y Aplicaciones Web.**

WAAP - Protección de Aplicaciones Web y API por sus siglas en inglés, es una evolución del tradicional WAF, es una tecnología de seguridad diseñada para proteger de una manera inteligente le integral as API y las aplicaciones web de una amplia gama de los ciberataques cada vez más sofisticados.

Según Gartner: “Los servicios de protección de aplicaciones web y API en la nube son la evolución de los servicios de firewall de aplicaciones web en la nube, ampliando su alcance y la profundidad de la seguridad. Los líderes en seguridad y gestión de riesgos deben determinar cómo los servicios WAAP en la nube mejorarán su postura de seguridad.” (Gartner Research, 2019)

WAAP se compone de cuatro características principales:

- WAF
- Protección contra DDoS para capa de aplicación (L7)
- Gestión de bots
- Protección de API

Con el auge de las APIs y las aplicaciones basadas en microservicios, los WAF y las soluciones de seguridad tradicionales no son efectivas para protegerlas, lo que hace que WAAP sea una necesidad al ofrecer una protección unificada adaptada a entornos modernos (nube, contenedores, etc.), contra una amplia gama de ataques multivectoriales.

Mientras que un WAF se centra en proteger las aplicaciones web frente a ataques como inyección SQL, XSS, etc., un WAAP amplía esa protección incluyendo API, mitigación de bots, protección DDoS, y otras capacidades más modernas.

En el mundo interconectado actual, las aplicaciones web son más complejas y evolucionan constantemente, lo que las convierte en el principal objetivo de los ciberataques, cada vez más sofisticados. En este entorno, la gestión de aplicaciones web y la protección de API requiere una funcionalidad más potente y soluciones automatizadas que puedan aumentar la seguridad y minimizar las tareas de gestión para mitigar los ataques

- El auge de las metodologías de desarrollo ágiles y DevOps significa que las aplicaciones web y las API modernas se encuentran en constante cambio,
- Las aplicaciones web están constantemente bajo ataque y estas amenazas cambian

periódicamente. Intentar protegerse contra ellos con soluciones de detección tradicionales basadas en firmas es un enfoque poco escalable. Las soluciones WAAP, con autoaprendizaje continuo, pueden ayudar a una organización a mantenerse al tanto del panorama de amenazas a la seguridad de las aplicaciones en rápida evolución.

- Más de la mitad de todo el tráfico web utiliza actualmente el cifrado TLS, lo que es bueno para la privacidad, pero malo para detectar malware y otro contenido malicioso.

**Capacidades de protección de API y Aplicaciones Web.**

WAAP protege la aplicación web contra una amplia gama de ataques sin requerir mucha supervisión y gestión. Las principales capacidades que una solución WAAP mantiene son las siguientes:

- Automatización e Inteligencia: Las soluciones WAAP aprenden por sí solas a adaptarse a los cambios en la aplicación que protegen y las amenazas a las que se enfrentan, lo que mejora los resultados de seguridad y minimiza la sobrecarga administrativa y la fricción operativa. Para lo cual se requiere automatización e inteligencia integradas
- Protección para API y microservicio: las API y microservicios son un objetivo de ataque cada vez mayor. Una solución WAAP debe proporcionar protección integral. Una mayor visibilidad ayuda a defenderse de los ataques ocultos, a la vez que revela cambios y errores inesperados.
- Firewall de aplicaciones web de próxima generación (NGWAF): los WAF tradicionales basados en firmas son ciegos a los ataques de día cero. Un NGWAF

integra capacidades de seguridad adicionales para ayudar a proteger contra una gama más amplia de amenazas.

- Lograr una amplia protección. protección completa para todos los sitios web, aplicaciones y API frente a una amplia gama de ciberamenazas, incluidas botnets automatizadas, ataques basados en API, inyecciones y ataques de denegación de servicio volumétricos, entre otros.
- Minimizar la superficie de ataque de API. Detectar y proteger automáticamente las API de vulnerabilidades como las 10 amenazas principales de OWASP, las 10 amenazas principales de API de OWASP y mucho más
- Autoprotección de aplicaciones en tiempo de ejecución (RASP Runtime Application Self Protection): RASP proporciona protección personalizada a la aplicación, monitoreando sus entradas, salidas y comportamiento en busca de anomalías. Lo que permite detectar incluso ataques de día cero contra una aplicación web o API.
- Protección contra bots maliciosos: los bots maliciosos ejecutan ataques automatizados contra aplicaciones web, a escala. La visibilidad en tiempo real del tráfico de bots permite a los equipos de seguridad investigar los análisis web sesgados, evitar la sobrecarga del origen y añadir definiciones de bots personalizadas La capacidad de diferenciar entre bots maliciosos y usuarios humanos es esencial para equilibrar la usabilidad y la seguridad de la aplicación.
- Protección de denegación de servicio distribuido (DDoS): Los ataques DDoS son una amenaza cada vez mayor a medida que el crecimiento de la computación en la nube

brinda a los ciberdelincuentes acceso a gran volumen de informática barata. La protección DDoS es esencial para garantizar la disponibilidad de la aplicación web y las API, al tiempo que mitiga los ataques a la capa de aplicación en cuestión de segundos.

- Limitación avanzada de velocidad: Limitar la velocidad para evitar que los usuarios malintencionados consuman recursos, permiten tomar medidas eficaces contra usuarios malintencionados sin afectar el uso legítimo de las aplicaciones.
- Mantenimiento sin fricciones. Reducir alertas con controles de seguridad simplificados, incluido el ajuste automático, que permite a los equipos de seguridad centrarse en los ataques reales reduciendo falsos positivos.

**Recomendaciones y buenas prácticas para implementar un WAF.**

Es importante elegir, configurar y personalizar el WAF de acuerdo con el servicio a proteger.

- Elegir el WAF que se va a implementar: en el mercado existen disponibles varias opciones gratuitas y de pago, de acuerdo con la necesidad de protección y las diferentes infraestructuras a proteger
- Personalización de reglas y políticas: como bloquear cierto tipo de tráfico malicioso o establecer límites de acceso en función de la ubicación.
- Uso de whitelisting y blacklisting: La mejor cobertura de seguridad con impacto mínimo en el tráfico legítimo es combinar modelos de seguridad negativa (definiendo lo que está prohibido y aceptando el resto) y positivos (definiendo lo que está

permitido y rechazando el resto). La combinación de ambos modelos permite definir las políticas de forma granular y precisa, evitando así los falsos positivos y los falsos negativos.  para que la seguridad no se interponga en el tráfico legítimo

- Ajuste de sensibilidad en detección de patrones.
- Actualización continua de reglas y firma de ataques. optimización y gestión de políticas adaptativa basada en inteligencia sobre amenazas e inteligencia artificial.
- Pruebas de regresión para validar que nuevas reglas no rompan la aplicación.
- Ocultar la ruta predeterminada de acceso y Usar una buena CDN: Una CDN (Content Delivery Network) no solo mejorará la velocidad de su web, sino que también proporciona un nivel adicional de seguridad
- Despliegue en modo de prueba (learning mode) previo al bloqueo activo de tráfico
- Implementaciones de acuerdo con la legislación y normas de cumplimiento locales: Cumplir con las normas reglamentarias es fundamental para las organizaciones.
- Monitoreo de alertas y notificaciones, se recomienda integrar con soluciones como SIEM y dashboard: contar con un panel centralizado permite a los equipos realizar un seguimiento del tráfico de la capa de aplicación junto con otros eventos de red, detectar patrones, responder rápidamente y mantener la visibilidad.
- Enfoque unificado: combinar el WAF con otras herramientas de seguridad proporciona un contexto más profundo sobre las actividades sospechosas dentro de la red y mejora la seguridad. La incorporación de plataformas refuerza la defensa integral

contra una amplia gama de vectores de ataque y permite crear una estrategia de defensa por capas para proteger las aplicaciones frente a amenazas complejas y a gran escala. Además, de reducir la complejidad de gestionar múltiples herramientas, sin comprometer la seguridad.

**Figura 3**

*10 mejores WAF en 2025*


Nota: Adaptado de Top 10 Best Web Application Firewall (WAF) in 2025, Cyber Security News, Cyber Writes Team, https://cybersecuritynews.com/best-web-application-firewall-waf

### *2.6 Herramientas y metodologías de ataque a APIs REST.*

Una API REST exhibe recursos y operaciones mediante HTTP, por ende, son vulnerables a diversos tipos de ataques, por ello existen herramientas y metodologías para probar y explotar debilidades en los problemas de infraestructura.

Se detalla las metodologías y herramientas más utilizadas:

**Herramientas:**

En el momento que se genere un ataque exitoso a una API se llega a comprometer la integridad, disponibilidad de la información y confidencialidad, afectando tanto a organización, clientes y socios.

Para considerar las herramientas de seguridad de API más seguras suelen concentrar una composición de las siguientes características:

- Descubrimiento e Inventario de AP, trata de la identificación automática de todas las API exhibidas, que incluye versiones y puntos finales, asegurando que no exista API "fantasma" o no documentadas.
- Protección en Tiempo de Ejecución, se trata del monitoreo continuo del tráfico API para la detección y bloqueo de ataques en tiempo real, incluyendo inyecciones, ataques de fuerza bruta y abuso de lógica de negocio.
- Gestión de Postura de Seguridad y Cumplimiento, valoración de configuración de seguridad de las API y su aprobación con estándares y regulaciones.
- Detección y Respuesta a Amenazas, uso de inteligencia artificial y aprendizaje automático para validar patrones de ataque sofisticados y respuesta automática.
- Autenticación y Autorización Robustas**,** ejecución de OAuth, OpenID Connect, JWT y otros estándares para afirmar que solo los usuarios y servicios autorizados puedan ingresar a las API.
- Limitación de Tasa y Throttling, control del número de casos que un cliente puede hacer en una etapa de tiempo definido para prevenir ataques de denegación de servicio

y abuso.

- Validación de Datos y Esquema, validar que las solicitudes y respuestas de las API se efectúen con los esquemas definidos, negando datos maliciosos.
- Protección contra Bots, verificación y mitigación de tráfico automatizado malicioso.
- Escaneo de Vulnerabilidades, pruebas de seguridad estáticas y dinámicas para verificar debilidades en código y configuración de las API.

Por lo cual, la seguridad de las APIs es un aspecto crítico en la ejecución de cualquier API, ya que sin la debida protección puede producir peligrosos resultados, entre ellos está la filtración masiva de datos sensibles hasta interrupciones operativas y pérdidas en las finanzas, considerando estos puntos, se enfatizan los siguientes tipo de herramientas que son más comunes de uso en pentesting, análisis de seguridad y auditorias de seguridad.

**Imperva API Security:** Parte de una suite de seguridad de aplicaciones más complejas con un descubrimiento continuo en APIs donde identifica todas las APIs que se encuentran en uso o en desuso, ofrece protección de API a través de su WAF, API Gateway, defensa contra bots y capacidades de descubrimiento y protección en tiempo de ejecución.

**Wallarm:** Una plataforma de seguridad de API que combina WAF que descubre y trata todas las APIs en uso, ocultas y no documentadas, garantiza protección contra bots y seguridad de API en una única solución, además, ofrece detección basada en IA/ML que usa algoritmos autorizados de inteligencia artificial y machine learning para verificar abusos, ataques de lógica de negocio y anomalías; y bloqueo en tiempo real.

**Postman:** Facilita el proceso de prueba, desarrollo y documentación de servicios web, optimizando el flujo de trabajo y favoreciendo la colaboración entre equipos, lo cual permite el envío manual de peticiones para pruebas funcionales y de seguridad.

**Cloudflare API Gateway:** Brinda gestión de API, seguridad avanzada en tiempo de real en la validación de JWT, autenticación a través de mutual TLS y optimización del rendimiento, además ofrece características como limitación de tasa, autenticación, autorización y protección contra ataques DDoS.

**Burp Suite:** Es una plataforma integral diseñada para la realización de pruebas de penetración y análisis de seguridad en las aplicaciones web, debido a que intercepta y modifica tráfico HTTP/HTTPS que es ideal para fuzzing e inyecciones.

**OWASP ZAP:** Es una herramienta de código abierto adoptada para ejecutar auditorías de seguridad en aplicaciones web, en donde facilita la detección mediante un escaneo automático de vulnerabilidades en APIs REST.

**Nikto:** Es un escáner de servidor web de código abierto (GPL) que ejecuta un escaneo de vulnerabilidades en servidores web, detectando configuraciones inseguras y desactualizadas del software del servidor web.

**Fuzzapi:** Es una herramienta de prueba de penetración de API REST automatizada en la comprobación de vulnerabilidades en los APIs tanto en aplicaciones web como móviles, además, brinda una interfaz práctica para gestionar pruebas de fuzzing sin necesidad de usar línea de comandos, posee integración con API_Fuzzer gem y admite simulación de ataques comunes contra APIs REST, como son inyecciones, manipulación de parámetros y abuso de endpoints.

**JWT Tool:** Se emplea en la autenticación, autorización en aplicaciones web, y análisis de tokens JWT para detectar fallos de autenticación, al ser un script en Python permite la decodificación y modificación de JWTs, detección de configuraciones inseguras en servidores que aceptan tokens manipulados y la automatización de ataques comunes contra JWT.

**sqlmap, NoSQLMap:** Es una herramienta diseñada para automatizar ataques de inyección y explotar las deficiencias de configuración en bases de datos.

**Técnicas:**

Las técnicas se basan en la identificación de vulnerabilidades en el diseño, implementación o configuración de la API, entre las más comunes están:

**Fuzzing:** Su técnica consiste en la inyección de datos aleatorios en un sistema informático o malformados para detectar fallos en la validación y observar cómo reacciona, lo que permite detectar problemas de seguridad y rendimiento.

Entre las técnicas de fuzzing se tiene fuzzing de caja negra, caja blanca y caja gris según la comprensión del programa objetivo, fuzzing apoyado en mutaciones y basado en generación según su tipo de generación de datos; y en fuzzing con o sin retroalimentación según el tipo de retroalimentación.

El fuzzing brinda la detección temprana de vulnerabilidades antes de que sean explotadas, también ofrece ejecutar varios casos en poco tiempo, con una amplia cobertura en el descubrimiento de errores y mejora en la reducción de riesgos de seguridad.

**Inyección de parámetros:** Se genera cuando una API admite datos de entrada como parámetros en URL, cuerpo JSON, encabezados, cookies, entre otros, y esos datos son asociados a un intérprete sin la previa validación o escape adecuados.

Entre los tipos de inyección se tiene:

- Inyección SQL (SQLi) que trata de la manipulación de consultas SQL en el acceso, modificación o eliminación de datos en la base de datos.
- Inyección de comandos del sistema, en donde el atacante inyecta comandos en parámetros que luego son ejecutados por el sistema operativo.
- Inyección de scripts (XSS) se inyecta código JavaScript en parámetros visibles en la aplicación, que permite el robo de cookies o redirección maliciosa.
- Inyección en cabeceras HTTP, trata de la manipulación de cabeceras como User-Agent o Cookie para alterar la lógica de la aplicación.

**Bypass de autenticación:** Son mecanismos que detectan y verifican a un usuario o tipo de cliente que son insuficientes o mal implementados, otorgando a actores no autorizados obtener acceso a recursos protegidos. En APIs, esto incluye fallos en tokens, sesiones, controles de rate limiting, y validación de flujos auth.

Entre las técnicas más comunes de bypass de autenticación se destacan las siguientes:

- Manipulación de parámetros indica la alteración de valores en la URL, cookies o cabeceras para engañar al sistema.
- Inyección SQL modificar consultas para que devuelvan acceso sin validar usuario/contraseña.
- Uso de tokens inseguros (JWT, sesiones) trata del aprovechamiento de claves débiles, algoritmos inseguros o tokens caducados que aún son admitidos.

- Explotación de endpoints ocultos permiten el acceso a rutas no protegidas o mal configuradas.
- Fuerza bruta combinada con fallos de lógica, permite probar múltiples credenciales hasta que el sistema responda de forma inesperada.

**Abuso de métodos HTTP:** Se genera cuando un API o servidor expone o acepta métodos HTTP como PUT, DELETE, OPTIONS, PATCH, entre otras, sin el control o la restricción adecuados, permitiendo a un actor realizar operaciones inesperadas (subir/modificar/eliminar recursos, provocar respuestas de CORS, etc.).

Entre las medidas de prevención de este método se destacan las siguientes:

- Restringir métodos HTTP permitiendo habilitar solo los necesarios (GET, POST).
- Configurar el servidor web para bloquear PUT, DELETE, TRACE, OPTIONS si no son requeridos.
- Validar y autenticar todas las solicitudes que usen métodos sensibles.
- Monitorear logs para detectar intentos de abuso.
- Pruebas de seguridad periódicas (pentesting, fuzzing) para identificar configuraciones inseguras.

### *2.7 Definición de métricas de evaluación*

Para evaluar la efectividad de un WAF se usan métricas derivadas de la matriz de confusión (peticiones maliciosas vs peticiones legítimas). Entre las más importantes están:

**Tasa de Verdadero Positivo (TPR):**

También conocida como calidad de seguridad, la Tasa de Verdadero Positivo (TPR) evalúa la capacidad de un WAF para identificar y responder con precisión a solicitudes maliciosas. En este contexto, un verdadero positivo se refiere a que la WAF respondió correctamente a un ataque. Esto incluye tanto ataques conocidos como ataques de día cero que aún no se comprenden bien y carecen de firmas específicas que monitorizar.

La forma en que se calcula el TPR es la siguiente: es el número de veces que se activa el WAF dividido por el número de ataques, o el número de verdaderos positivos (TP) dividido por el total de verdaderos positivos (ataques capturados) y falsos negativos (FN) (ataques fallidos):

$$\text{TPR} = \frac{TP}{TP + FN}$$

**Tasa de Falsos Positivos (FPR):**

También conocida como: calidad de detección, mide con qué frecuencia las solicitudes legítimas se marcan incorrectamente como maliciosas. La Tasa de Falsos Positivos (FPR) se calcula como el número de falsos positivos (FP) (tráfico malicioso marcado incorrectamente) dividido por el número total de falsos positivos y verdaderos negativos (TN) (tráfico correctamente no marcado).

$$\text{FPR} = \frac{FP}{FP + TN}$$

**Tasa de Negativos Verdaderos (TNR)**

Proporción de peticiones legítimas reconocidas como legítimas. Se calcula como: (1 − FPR).

$$\text{TNR} = \frac{TN}{TN + FP} = 1 - \text{FPR}$$

**Precisión equilibrada**

La precisión equilibrada es una métrica global que combina TPR (detección de ataques) y TNR (no bloquear tráfico legítimo). Se calcula como el promedio de TPR y TNR. (Check Point Software Technologies Ltd., s.f.)

$$\text{Precisión equilibrada} = \frac{TPR + TNR}{2}$$

### *2.8 Marco Normativo*

**Referencias a ISO/IEC 27001, Datos Personales, PCI, OWASP.**

En el mundo digital, las APIs (Interfaces de Programación de Aplicaciones) son fundamentales en el ecosistema digital para la operatividad empresarial, permiten la interacción entre aplicaciones, servicios y sistemas, facilitando la innovación y la eficiencia. Sin embargo, esta dependencia las convierte en un objetivo atractivo para los atacantes., exponiendo a las organizaciones a riesgos como accesos no autorizados, abuso de credenciales, tráfico malicioso y brechas de seguridad. Por lo tanto, adoptar e implementar un Marco normativo de Seguridad para

APIs no es solo una opción, sino una necesidad estratégica
que garantice su seguridad y cumplimiento

El cumplimiento de regulaciones y estándares asegura que las prácticas de seguridad estén alineadas con las normativas aplicables en cada industria.
Implementar un marco normativo de seguridad para APIs permite proteger los datos sensibles, pr omueve la confianza de los clientes y el éxito empresarial a largo plazo.

Un enfoque integral de seguridad aborda múltiples capas críticas: protección perimetral, monitoreo en tiempo real, cumplimiento normativo, entre otros. Los requisitos se centran claramente en proteger las aplicaciones y la infraestructura dentro de las cuales operan las API. para lo cual se abordarán las siguientes normativas y estándares relevante.

- ISO/IEC 27001:2022
- Regulaciones de protección de datos personales
- PCI-DSS

**ISO/IEC 27001:2022**

Norma Internacional ISO/IEC 27001:2022, “Seguridad de la Información – Ciberseguridad y Protección de la Privacidad – Sistemas de Gestión de Seguridad de la Información”. Proporciona directrices y requisitos esenciales para establecer, implementar y mejorar continuamente un sistema de gestión de seguridad de la información (SGSI), que salvaguarde la confidencialidad, integridad y disponibilidad de la información en una organización.

La importancia de contar con un SGSI radica en proporcionar un marco estructurado para gestionar de manera efectiva la seguridad de la información, identificando y mitigando los

riesgos relacionados a la confidencialidad e integridad de los datos, lo que promueve una cultura de seguridad, aumento en la confianza de las partes interesadas,
cumplimiento de requisitos legales y regulatorios relacionados con la protección de datos. Contribuye con la mejora continua en los procesos en un entorno empresarial cada vez más complejo y digitalizado.

La Norma cuenta con una estructura establecida, para demostrar su cumplimiento y guiar la implementación de medidas de seguridad adecuadas, se debe desarrollar la Declaración de Aplicabilidad (SoA) donde se detalla los controles que aplican y su justificación. La norma incluye 93 controles para su cumplimiento, los principales controles que aplican a APIs son:

Si la organización proporciona servicios de IT a otras empresas, generalmente los clientes requieren certificaciones de seguridad entre ellas ISO 27001, aunque no es una normativa obligatoria de cumplimiento, a menudo es la forma más fácil de cumplir normas o reglamentos de protección de datos personales entre otras.

Al implementar un SGSI, las organizaciones demuestran su compromiso con la ciberseguridad y protección de sus datos, lo que les ayuda a diferenciarse de sus competidores, aumentar la confianza de los clientes y abrir nuevas oportunidades de negocio al facilitar la participación en licitaciones que requieran estándares de seguridad y certificaciones más específicas.

**Regulaciones de protección de datos sensibles -personales**

La protección de datos sensibles es uno de los pilares fundamentales en la seguridad de un API REST. Este tipo de datos incluye contraseñas, información personal identificable (PII), datos financieros, tokens de acceso, claves API, entre otros. Cualquier vulnerabilidad en su

manejo puede resultar en brechas de seguridad, pérdida de confianza de los usuarios e incluso sanciones legales según normativas como Ley Orgánica de Protección de Datos Personales o GDPR (legislación de la Unión Europea cuyo objetivo es reforzar y unificar la protección de datos de las personas dentro de la UE).

Las APIs que constituyen la base de aplicaciones o microservicios pueden intercambiar datos regulados, por lo tanto, las organizaciones que desarrollan API accesibles a través de Internet deben incluir la protección de datos en su diseño desde su desarrollo.

El cumplimiento de estas normativas requiere que:

- Los desarrolladores de API deben implementar controles de autenticación y autorización de usuarios para proteger los datos confidenciales utilizados.
- Los equipos de desarrollo de API deben garantizar la confidencialidad de los datos en tránsito mediante el uso de protocolos de comunicación seguros que cifren el intercambio de información entre el cliente y el servidor
- Se debe seguir el principio de privilegio mínimo, implementando medidas técnicas y administrativas apropiadas para garantizar que sean objeto de tratamiento únicamente los datos personales necesarios para cada uno de los fines específicos. Evaluar sus factores de riesgo (por ejemplo, los tipos de datos que han estado intercambiando y quién o qué puede acceder a esos datos).

**Estándar de seguridad de datos de la industria de tarjetas de pago- PCI DSS**

Norma creada por Payment Card Industry Data Security Council, es un estándar global para la protección de datos de los titulares de tarjetas de pago y garantizar la seguridad en el procesamiento, almacenamiento y transmisión de dicha información.

PCI DSS estándar de cumplimiento para todas las organizaciones que almacenan, procesan y transmiten datos de tarjetas de pago, consta de 12 requisitos agrupados en seis objetivos de control, garantizando que las organizaciones mantengan entornos seguros para el tratamiento de dichos datos. El cumplimiento ayuda a evitar sanciones y a proteger la información del titular de la tarjeta, fortalecer las medidas de seguridad generales y aumentar la confianza y la seguridad de los consumidores.

En general, la norma PCI DSS v4.0 se centra en cuatro objetivos clave:

1. Satisfacer las necesidades de seguridad del sector de pagos.
2. Establecer la seguridad como un proceso continuo.
3. Ofrecer a las empresas flexibilidad (por ejemplo, nuevas herramientas o nuevos controles) para cumplir los requisitos.
4. Mejorar los métodos y procesos de validación.

El requisito 6 de la norma PCI-DSS v4.0 incluye la obligatoriedad de revisión de riesgos y vulnerabilidades de las API, en las siguientes secciones:

- Desarrollo de software de forma segura.
- Identificar y corregir vulnerabilidades de seguridad.
- Aplicaciones web expuestas al público deben estar protegidas contra ataques.
- Los cambios realizados en todos los componentes del sistema deben ser gestionados de forma segura.
- Implementar controles para bloquear a los agentes sospechosos de manera que no se vulneren los sistemas.

# Capítulo 3

## 3. Desarrollo

### *3.1. Arquitectura de Implementación*

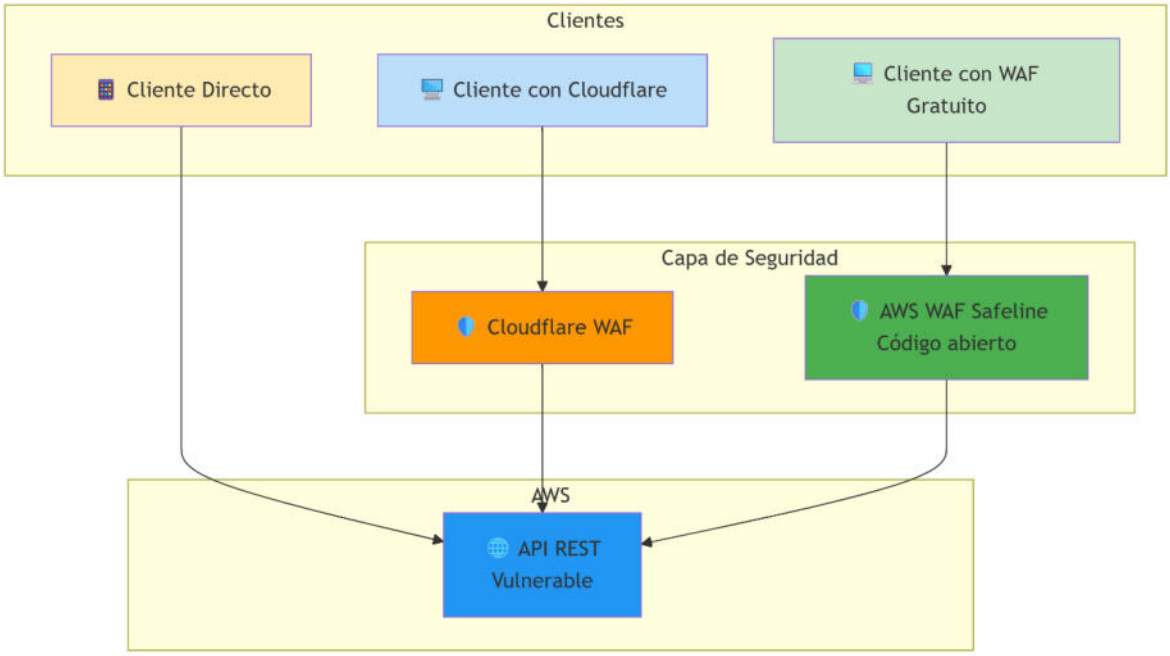
Para llevar a cabo el escenario de pruebas, se plantea la creación de una API REST con vulnerabilidades controladas, la cual será desplegada en la infraestructura de Amazon Web Services (AWS). El objetivo es contar con un entorno realista que nos permita observar cómo se comporta una aplicación expuesta a Internet cuando distintos mecanismos de protección intervienen o se omiten.

Como parte del diseño, se incorporarán dos tipos de Web Application Firewall (WAF): uno de carácter de código abierto y otro comercial, en este caso proporcionado por Cloudflare. Esta combinación permitirá comparar la capacidad de detección, reacción y filtrado de cada solución frente a intentos de explotación comunes, así como evaluar diferencias en su facilidad de uso, implementación y nivel de protección.

La idea central es reproducir condiciones similares a las que enfrentaría un servicio en producción, pero de manera controlada, con el fin de entender mejor cómo interactúan estos componentes y qué tan efectiva puede resultar cada alternativa dentro de un entorno de seguridad orientado a APIs.

**Figura 4**

*Arquitectura para implementación de un WAF en una API REST*




### *3.2. Configuración del entorno virtualizado en AWS*

Para disponer de un entorno virtualizado, estable y sencillo de administrar se ha decidido utilizar el servicio de Amazon Web Services (AWS) en la nube, en el cual se puede elegir una variedad de sistemas operativos para el alojamiento de servicios, programas o sistemas. En este caso se seleccionó Ubuntu Server como plataforma para la ejecución de la API creada (Amazon Web Services, 2024).

La decisión de trabajar con Ubuntu como sistema operativo no fue casual. Este sistema es reconocido por su estabilidad y por contar con versiones de soporte a largo plazo (LTS), lo que

garantiza actualizaciones y mantenimiento durante varios años. A esto se suma que Ubuntu se adapta muy bien a proyectos desarrollados en Python y a frameworks actuales como FastAPI.

La instancia implementada en AWS es una máquina virtual tipo EC2 – t2.micro. Que tiene los recursos necesarios para pruebas pequeñas como implementaciones de API. Las características técnicas se pueden notar en la Tabla 2.

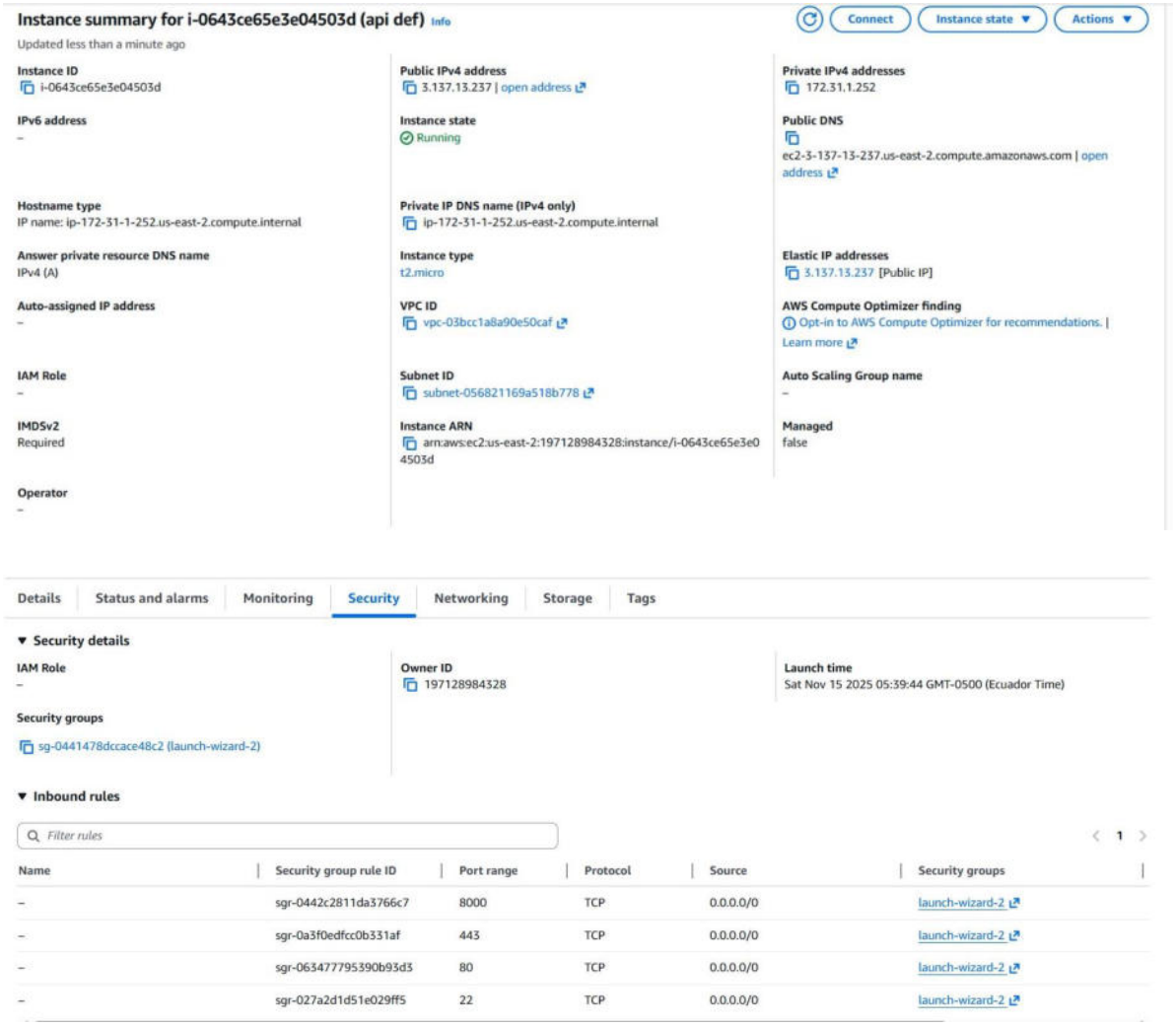
***Tabla 2***

*Parámetros técnicos de la instancia EC2*

| **Características Máquina Virtual** | |
|---|---|
| *Información General* | |
| Tipo de instancia | t2.micro |
| ID Instancia | i-0643ce65e3e04503d |
| Región | us-east-2 (Ohio) |
| *Red y Direccionamiento* | |
| IPv4 pública | 3.137.13.237 |
| IPv4 privada | 172.31.1.252 |
| DNS público: | ec2-3-137-13-237.us-east-2.compute.amazonaws.com |
| DNS privado: | ip-172-31-1-252.us-east-2.compute.internal |
| VPC: | vpc-03bcc1a8a90e50caf |
| Subnet: | subnet-056821169a518b778 |

**Figura 5**

*Configuración de la instancia EC2 en AWS*



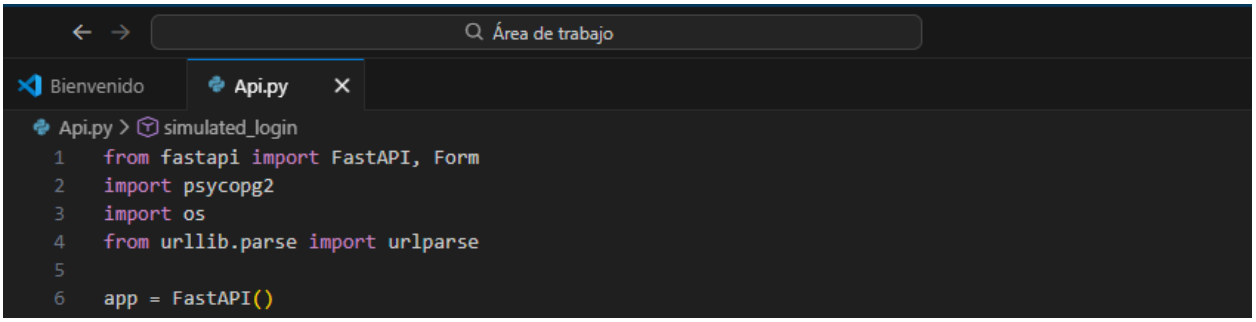

### *3.3. Desarrollo de la API Rest*

La API Rest desarrollada para este proyecto fue diseñada intencionalmente con vulnerabilidades de seguridad, con el fin de que puedan ser explotadas desde la máquina atacante y así evaluar el comportamiento del sistema, la efectividad de los mecanismos de defensa y el desempeño de los WAFs implementados.

En la implementación se utilizó el framework FastAPI por su facilidad en incorporarse con solicitudes HTTP y validación de datos. Adicionalmente se incorporó PostgreSQL para la gestión de la Base de Datos.

**Figura 6**

*Configuración Inicial de la API integrada a la Base de Datos PostgreSQL*




La API desarrollada permite gestionar usuarios almacenados en PostgreSQL utilizando el framework FastAPI. Se incluyeron varias funciones sencillas: permite listar, buscar, modificar datos o eliminar cuando es necesario. Además, se añadió un endpoint pensado específicamente para realizar pruebas de XSS y un módulo de inicio de sesión simulado. En la Tabla 3 se puede visualizar el endpoint y las vulnerabilidades que cada una expone. En la sección de los anexos se encontrará disponible el código de la API.

**Figura 7**

*Método GET para listar usuarios*

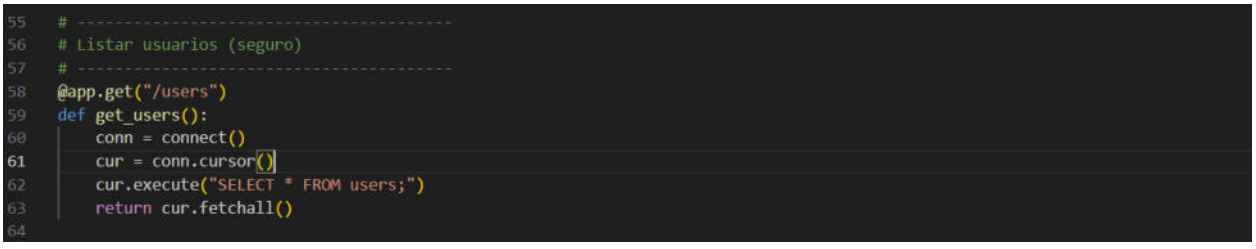


**Figura 8**

*Método GET para buscar usuarios*

```python
# ----------------------------------------
# Endpoint vulnerable: búsqueda SQL Injection
# ----------------------------------------
@app.get("/users/search")
def search_user(username: str):
    conn = connect()
    cur = conn.cursor()
    query = f"SELECT id, username FROM users WHERE username LIKE '%{username}%';"  #
    cur.execute(query)
    return cur.fetchall()

```

**Figura 9**

*Método PUT para modificar usuarios*

```python
# ----------------------------------------
# Endpoint vulnerable: actualización SQL Injection
# ----------------------------------------
@app.put("/users/{id}")
def update_user(id: int, username: str):
    conn = connect()
    cur = conn.cursor()
    query = f"UPDATE users SET username='{username}' WHERE id={id};"  #
    cur.execute(query)
    conn.commit()
    return {"status": "updated"}

```

**Figura 10**

*Método DELETE para eliminar usuarios*

```python
# ----------------------------------------
# Endpoint vulnerable: eliminar SQL Injection
# ----------------------------------------
@app.delete("/users/{id}")
def delete_user(id: int):
    conn = connect()
    cur = conn.cursor()
    query = f"DELETE FROM users WHERE id={id};"  #
    cur.execute(query)
    conn.commit()
    return {"status": "deleted"}

```

**Figura 11**

*Modulo vulnerabilidad XSS y Login Seguro en PostgreSQL*

```
105  # Endpoint vulnerable a XSS (solo pruebas)
106  # ----------------------------------------
107  @app.get("/xss")
108  def xss_test(text: str):
109      return {"echo": text}   #
110
111
112  # ----------------------------------------
113  # NUEVO: Simulated Login con PostgreSQL
114  # ----------------------------------------
115  @app.post("/auth/simulated-login")
116  def simulated_login(username: str = Form(...), password: str = Form(...)):
117      conn = connect()
118      cur = conn.cursor()
119
120      # Consulta segura
121      cur.execute("SELECT password FROM users WHERE username=%s;", (username,))
122      result = cur.fetchone()
123
124      # Usuario no encontrado → HTTP 401
125      if not result:
126          raise HTTPException(
127              status_code=401,
128              detail="Invalid username or password"
129          )
130
```

***Tabla 3***

*Endpoints API vulnerable*

| **Endpoints API vulnerable** | | | | |
|---|---|---|---|---|
| *Endpoint* | *Método* | *Función* | *Estado de seguridad* | *Tipo de vulnerabilidad* |
| /users | GET | Lista todos los usuarios | Seguro | Ninguna |
| /users/search | GET | Busca usuarios | Inseguro | SQL Injection |
| /users/{id} | PUT | Modifica un usuario | Inseguro | SQL Injection |
| /users/{id} | DELETE | Elimina un usuario | Inseguro | SQL Injection |
| /xss | GET | Devuelve lo enviado | Inseguro | XSS |
| /auth/simulated-login | POST | Login seguro | Seguro | Ninguna |

Esta API se encuentra dentro de la instancia de AWS a la cual se realizará ataques para constatar las vulnerabilidades y posteriormente realizar las pruebas, pero implementando los WAFs comercial y código abierto para verificar el rendimiento de cada uno.

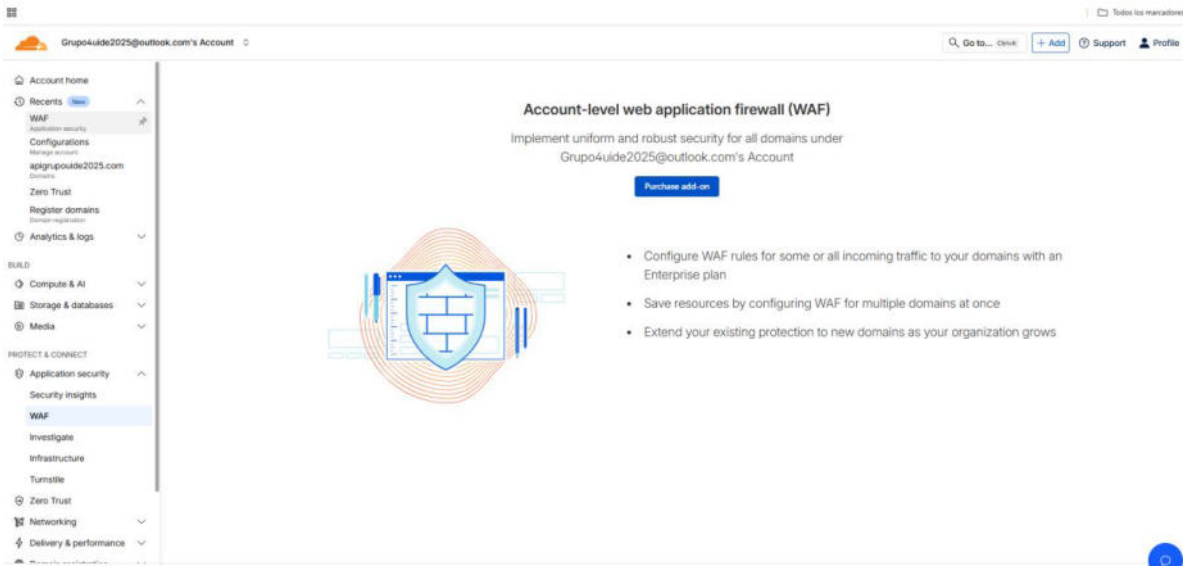
### *3.4. Implementación WAF comercial*

Para la elección del WAF utilizado en este proyecto, se evaluaron varias alternativas disponibles en el mercado. Después de comparar características como facilidad de despliegue, capacidad de detección, rendimiento, soporte y reputación, se determinó que Cloudflare era la opción más adecuada. Su trayectoria como una de las soluciones más robustas y confiables, junto con su constante actualización frente a nuevas amenazas, lo convirtió en el candidato ideal para la implementación del WAF de pago dentro de este trabajo.

Según se observa en la Figura 1 de la arquitectura de aplicación Cloudflare actúa inspeccionando todo el tráfico entrante antes de que llegue a la infraestructura interna de la API, aplicando reglas de seguridad, filtrando solicitudes maliciosas y mitigando ataques comunes como SQL Injection, Cross-Site Scripting (XSS), fuerza bruta o escaneo automatizado.

**Figura 12**

*Cloudflare web application firewall (WAF)*




Para poder comenzar con la configuración uno de los requisitos es tener registrado un dominio, esto implica modificar los registros DNS para que el tráfico sea enrutado a través de los servidores de Cloudflare. Una vez que el dominio está activo, se habilitan las funciones de seguridad disponibles en el panel, entre ellas las Reglas Administradas del WAF, controles de tasa de solicitudes, validación de métodos HTTP, inspección de cabeceras y filtros personalizados. Estas reglas permiten detectar patrones de ataque sin necesidad de instalar agentes adicionales en el servidor.

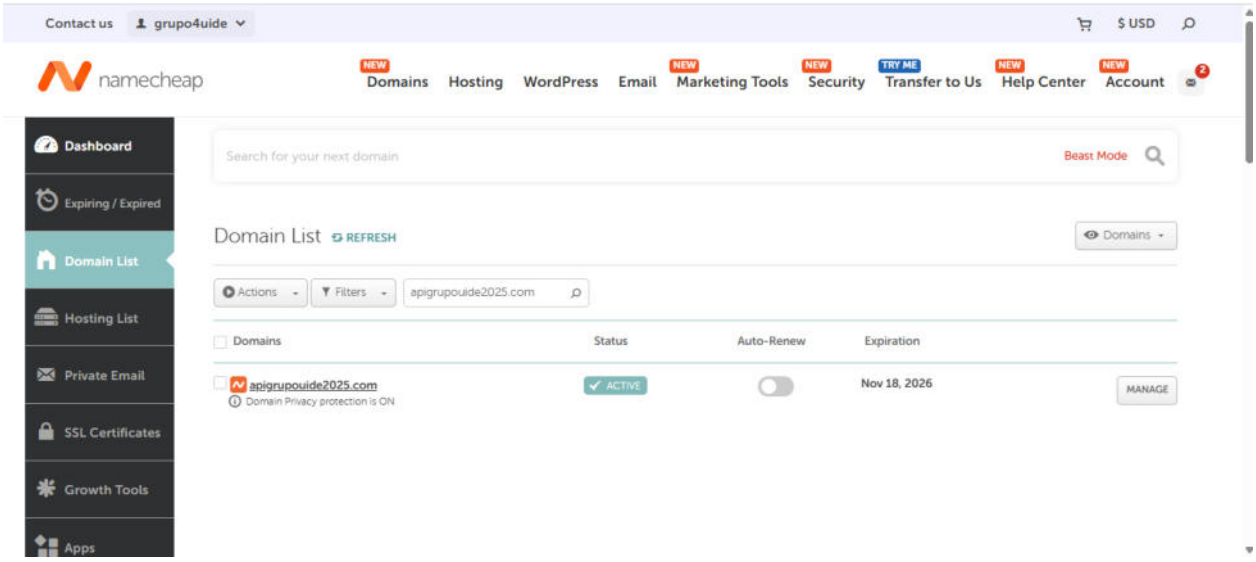
### Dominio para WAF comercial

Para su funcionamiento, Cloudflare requiere únicamente que el dominio esté administrado desde su panel y que el servidor de destino (en este caso la API REST en AWS) permita recibir el tráfico proveniente de los rangos IP autorizados. Para esto, se procedió a comprar un dominio, en este caso utilizamos namecheap para realizar la compra del dominio y asociar a nuestra IP pública de la instancia de AWS.

Para la compra del dominio se debe registrar una cuenta y buscar la extensión que más convenga para el laboratorio, en este caso se seleccionó el dominio **apigrupouide2025.com** con al cual se va a configurar en AWS y en CloudFlare para realizar el despliegue del WAF.

**Figura 13**

*Dominio para API integrada con WAF comercial*




### Configuración de Cloudflare

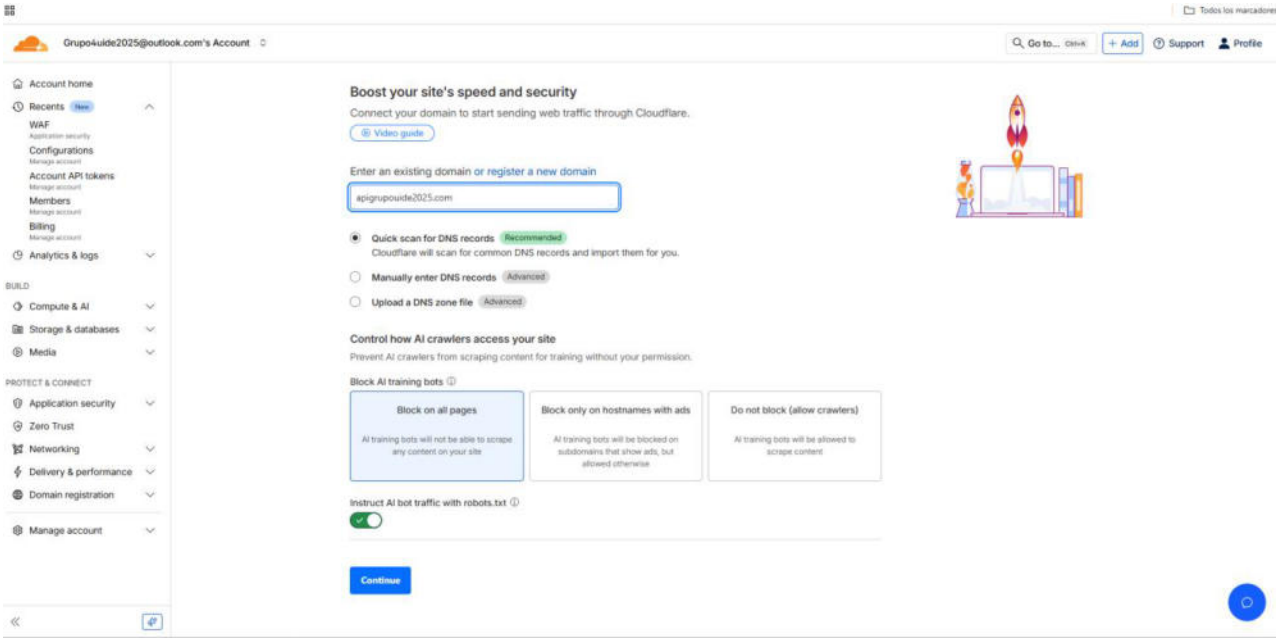
La implementación de Cloudflare como WAF está basada en la capacidad de actuar como una capa intermedia entre el usuario y la API que se encuentra alojada en el servidor de AWS.

Para poder utilizar la plataforma de Cloudflare se debe dirigir el tráfico del dominio hacia la plataforma, para que el servicio del WAF inspeccione y filtre las solicitudes antes que llegue al servidor.

El primer paso es registrar el dominio que se tiene asociado en la API en la plataforma de Cloudflare como se muestra en la Figura 14, posteriormente colocar el tipo de plan que se requiere según las necesidades.

**Figura 14**

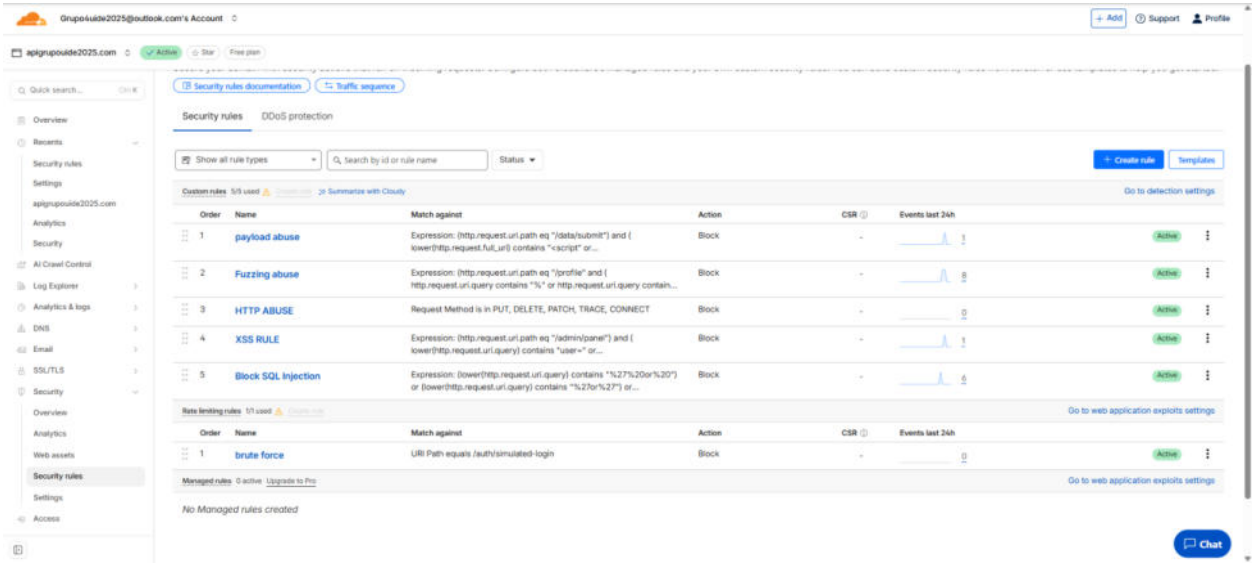
*Registro de dominio a filtrar en Cloudflare*




En la Figura 15 se muestra el panel donde se realiza la configuración de las reglas del WAF, en esta sección se visualizan las reglas de seguridad creadas específicamente para la protección de la API. Para este proyecto se configuró las siguientes reglas que nos van a proteger para ataques de SQL Injection, XSS, fuzzing, abuso de payloads y fuerza bruta. Se procede a detallar cada una de las configuraciones.

**Figura 15**

*Panel de configuración de reglas para la API*




### Configuración reglas inyección SQL

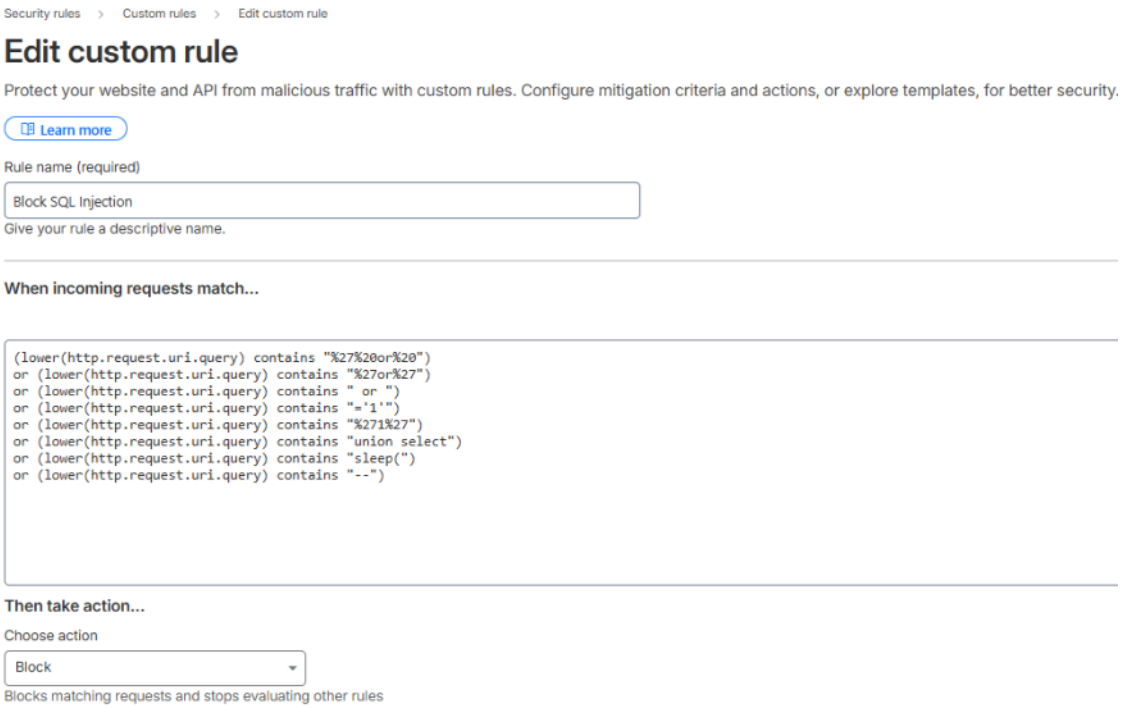
Para proteger los ataques de inyección SQL se configuró una serie de reglas personalizadas en Cloudflare que nos permite detectar patrones utilizados durante intentos de explotación. La expresión aplicada analiza el contenido del parámetro query dentro de la URL, convirtiéndolo previamente a minúsculas para evitar técnicas de evasión basadas en cambios. Las reglas están configuradas para identificar los siguientes indicadores de ataque asociados a SQL Injection:

- Comillas simples codificadas en URL
- Uso del operador lógico OR en la consulta
- Comparaciones maliciosas orientadas a bypass
- Intentos de concatenación de consultas SQL

- Uso de funciones de tiempo para SQL Injection ciega
- Inserción de comentarios SQL

**Figura 16**

*Configuración reglas prevención ataques inyección SQL*



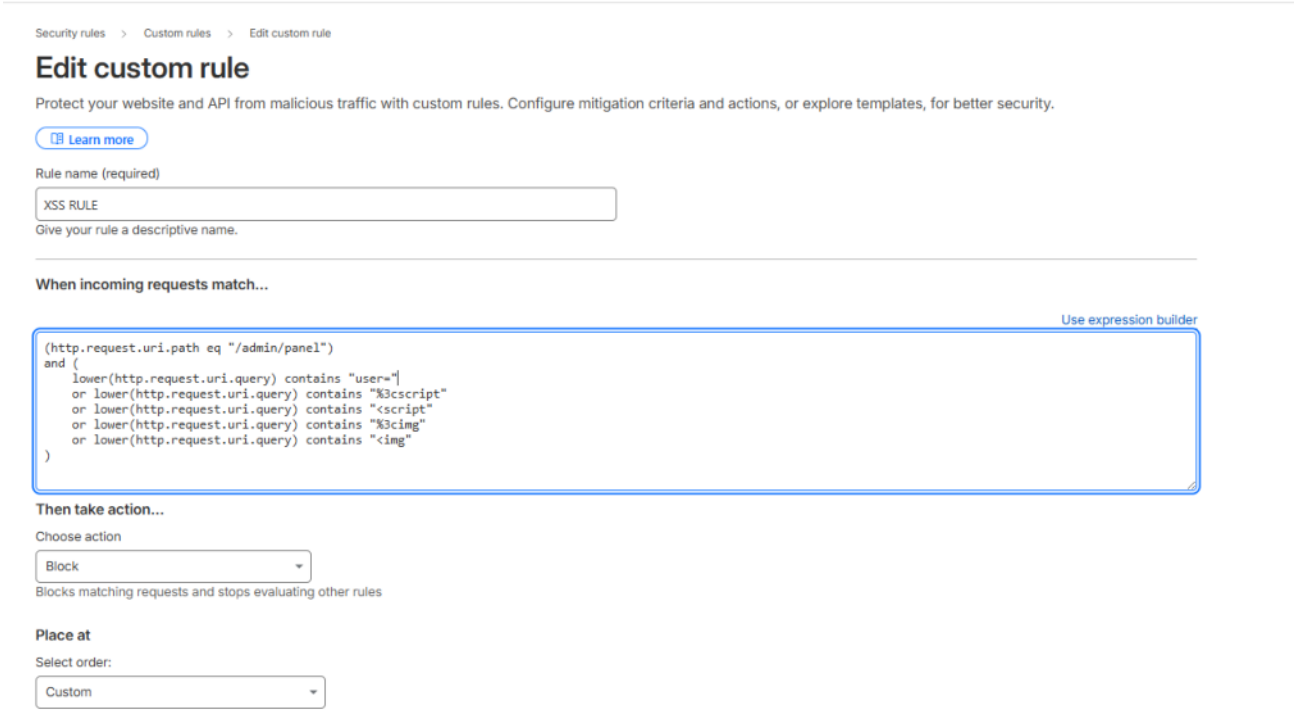

### Configuración reglas Cross-Site Scripting XSS

La configuración de las reglas tiene como objetivo proteger un ataque de XSS identificando alguno de estos patrones y bloqueando las solicitudes automáticamente. Las reglas que están configuradas para identificar los ataques de XSS son los siguientes:

- Acceso sospechoso al endpoint administrativo.
- Intentos de Cross-Site Scripting (XSS)
- Payloads maliciosos codificados y no codificados

**Figura 17**

*Configuración reglas prevención ataques XSS*




### Configuración reglas Fuzzing Abuse

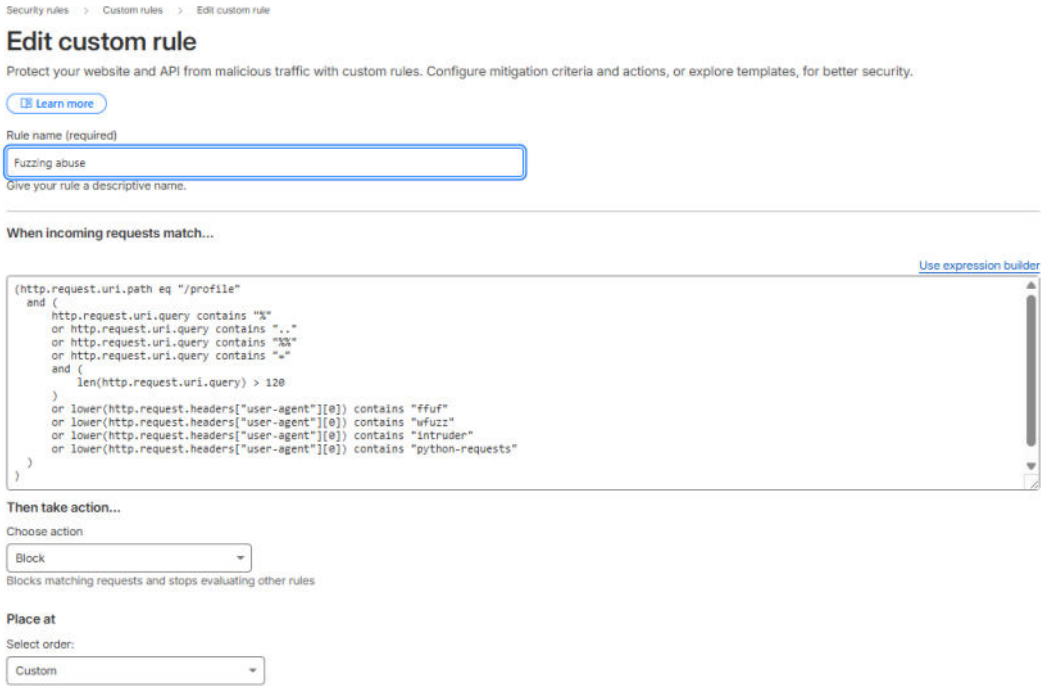
La configuración de las reglas tiene como objetivo mitigar ataques clasificados como Fuzzing Abuse. Estos ataques consisten en el envío masivo y automatizado de parámetros manipulados con el fin de descubrir rutas ocultas, vulnerabilidades lógicas, errores del servidor o comportamientos inesperados en la aplicación. El fuzzing es comúnmente usado para mapear la superficie de ataque o provocar fallos que revelen información sensible. Las reglas que están configuradas para identificar los ataques de Fuzzing son los siguientes:

- Caracteres sospechosos en parámetros (indicadores de fuzzing)
- Parámetros demasiado largos

- Herramientas automatizadas de fuzzing (User-Agent)

**Figura 18**

*Configuración reglas prevención ataques Fuzzing*




### Configuración reglas HTTP Abuse

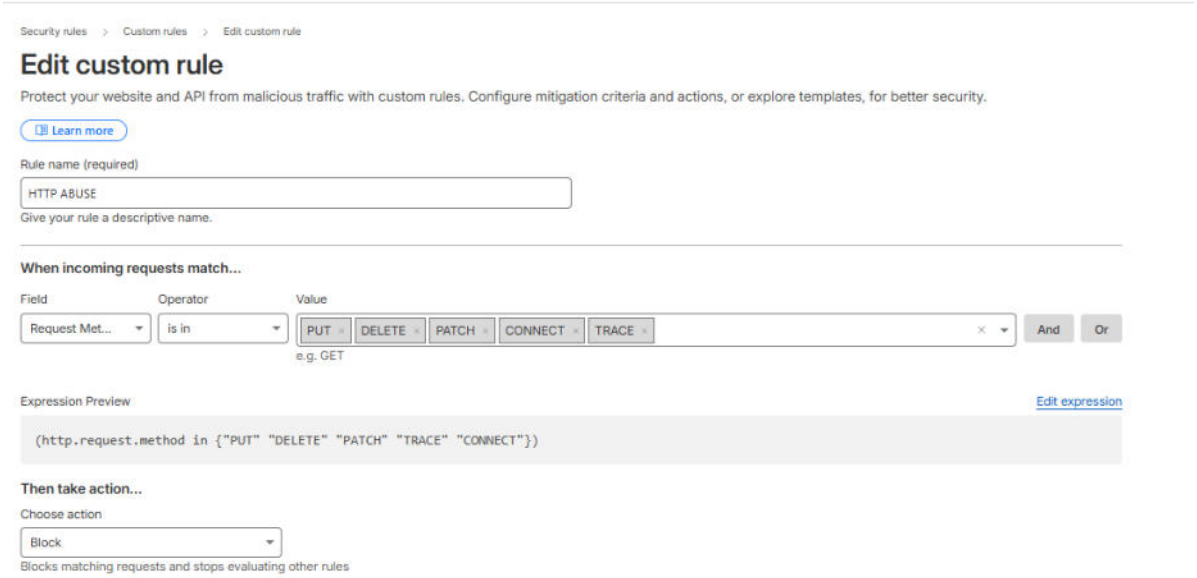
La configuración de las reglas tiene como objetivo mitigar solicitudes que utilicen métodos HTTP potencialmente peligrosos y que no forman parte del funcionamiento legítimo del servicio de la API. La exposición de métodos como *PUT*, *DELETE*, *PATCH*, *TRACE* y *CONNECT* puede abrir la puerta a ataques de manipulación de recursos, modificación de datos, trazado de solicitudes o túneles no autorizados. Las reglas que están configuradas para identificar los ataques de HTP son las siguientes:

- Modificación de recursos sin autorización (PUT / PATCH)
- Eliminación de datos (DELETE)

- Cross-Site Tracing (TRACE)
- Tunneling de tráfico o evasión de firewall (CONNECT)

**Figura 19**

*Configuración reglas prevención ataques HTTP*




**Configuración reglas ataque de Fuerza bruta.**

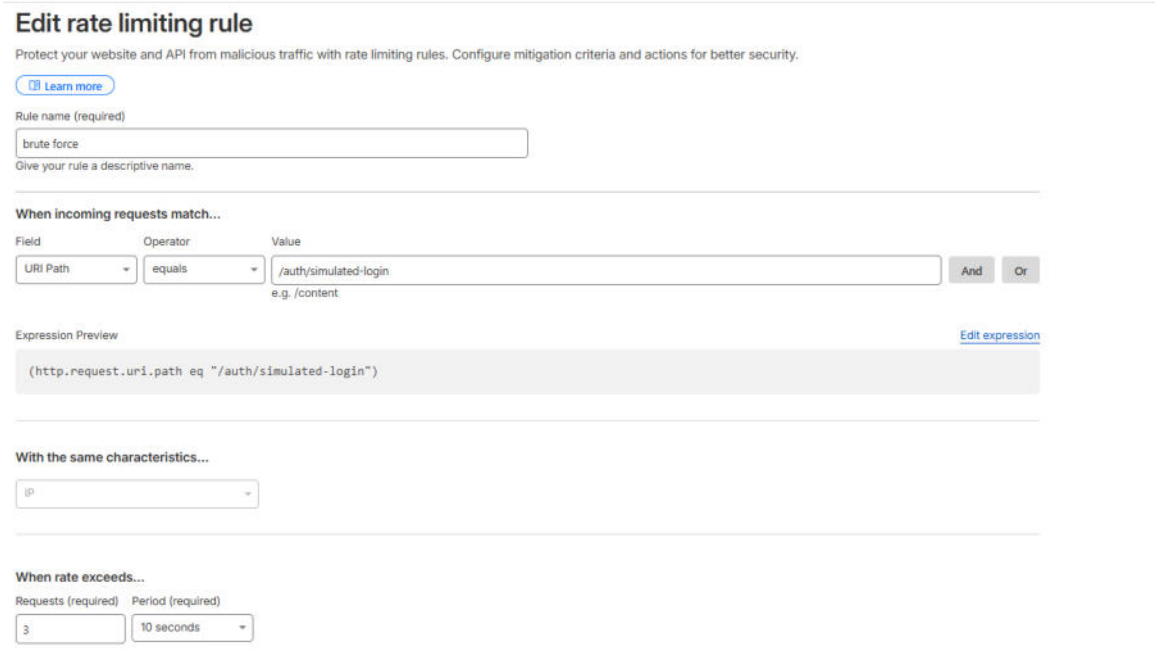
La configuración de las reglas tiene como objetivo identificar patrones de comportamiento anómalos basados en un número elevado de solicitudes generadas desde una misma dirección IP en un intervalo reducido de tiempo, lo cual es característico de ataques como fuerza bruta, bots maliciosos o intentos de denegación de servicio a pequeña escala.

- Agrupación de tráfico por IP.
- Umbral de activación 3 solicitudes en un tiempo de 10 segundos.
- Bloqueo de IP si supera el umbral.

- Tiempo de bloqueo de 10 segundos.

**Figura 20**

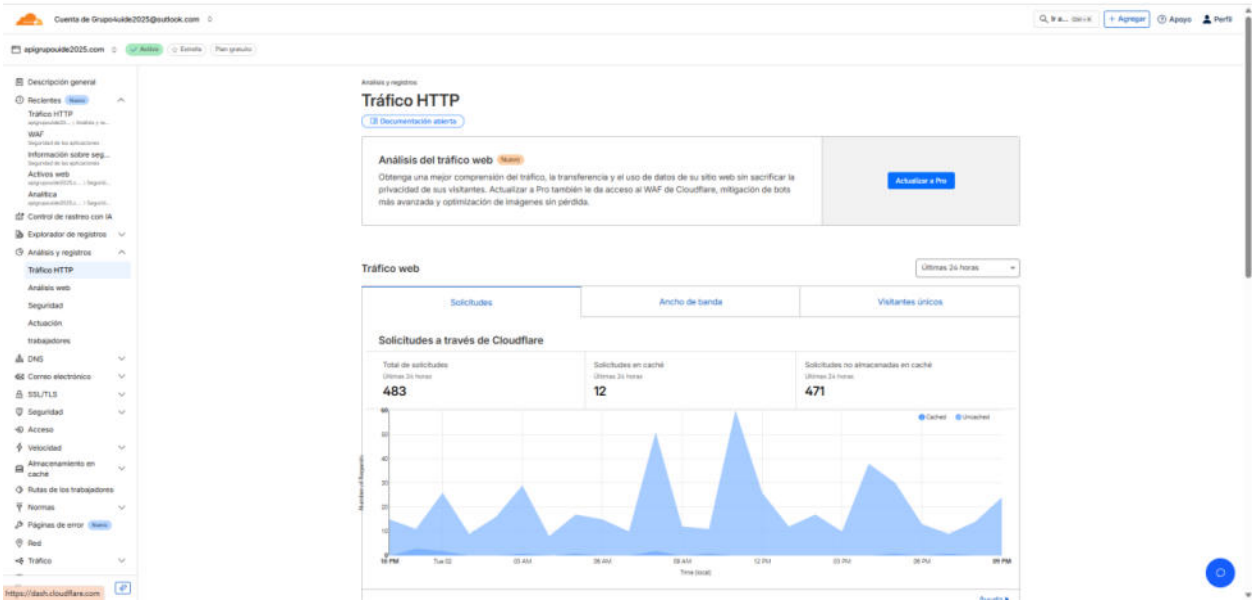
*Configuración reglas prevención ataques Fuerza bruta*




Adicionalmente, la plataforma de Cloudflare cuenta con un panel de control (Dashboard) que permite visualizar diferentes métricas relacionadas con el tráfico y los eventos de seguridad. Esta información resulta clave para evaluar la efectividad del WAF frente a los ataques dirigidos a nuestra API. En la Figura 21 se observa, por ejemplo, el comportamiento del tráfico del dominio durante las últimas 24 horas, lo que facilita identificar patrones inusuales o intentos de ataque.

**Figura 21**

*Tráfico HTTP analizado a través de Cloudflare*




### *3.5. Implementación WAF código abierto*

Para la elección del WAF de código abierto, se evaluaron varias alternativas disponibles y se decidió incluir a SafeLine como WAF de nueva generación debido a su arquitectura liviana, despliegue basado en contenedores Docker y capacidad de detección inteligente. SafeLine protege aplicaciones web y APIs mediante inspección profunda del tráfico, reglas personalizadas y módulos capaces de identificar patrones anómalos. La característica principal de este WAF es un servicio onpremise que requiere un sistema operativo Linux con ciertas características mínimas como se muestra en la Tabla 4.

***Tabla 4***

*Requisitos mínimos para despliegue de SafeLine*

| **Dependencias mínimas del sistema** | |
|---|---|
| *Dependencia* | *Valor* |
| Sistema operativo | Linux |
| Arquitectura | x86_64, arm64 |
| Versión de docker | Docker ≥ 20.10.14 / Docker Compose ≥ 2.0.0 |
| CPU | 1 CPU |
| RAM | 1 GB |
| Almacenamiento | 5 GB |

Adicionalmente, SafeLine destaca por su arquitectura basada en contenedores, lo que facilita un despliegue rápido, reproducible y estandarizado. Esto resulta esencial en un entorno de investigación, ya que permite realizar pruebas controladas bajo las mismas condiciones y asegurar que los resultados obtenidos sean consistentes. Su instalación mediante Docker simplifica la integración con la infraestructura existente sin necesidad de configuraciones complejas o dependencias invasivas.

**Despliegue de SafeLine WAF**

Para el despliegue y considerando que la infraestructura del proyecto utiliza los servidores de AWS, se crea una nueva instancia con el sistema operativo de Ubuntu. Esta instancia fue configurada de acuerdo con los requisitos mínimos establecidos por SafeLine para

su instalación, garantizando así un entorno compatible y estable para la implementación del WAF.

**Figura 22**

*Creación de instancia AWS para instalación WAF SafeLine*

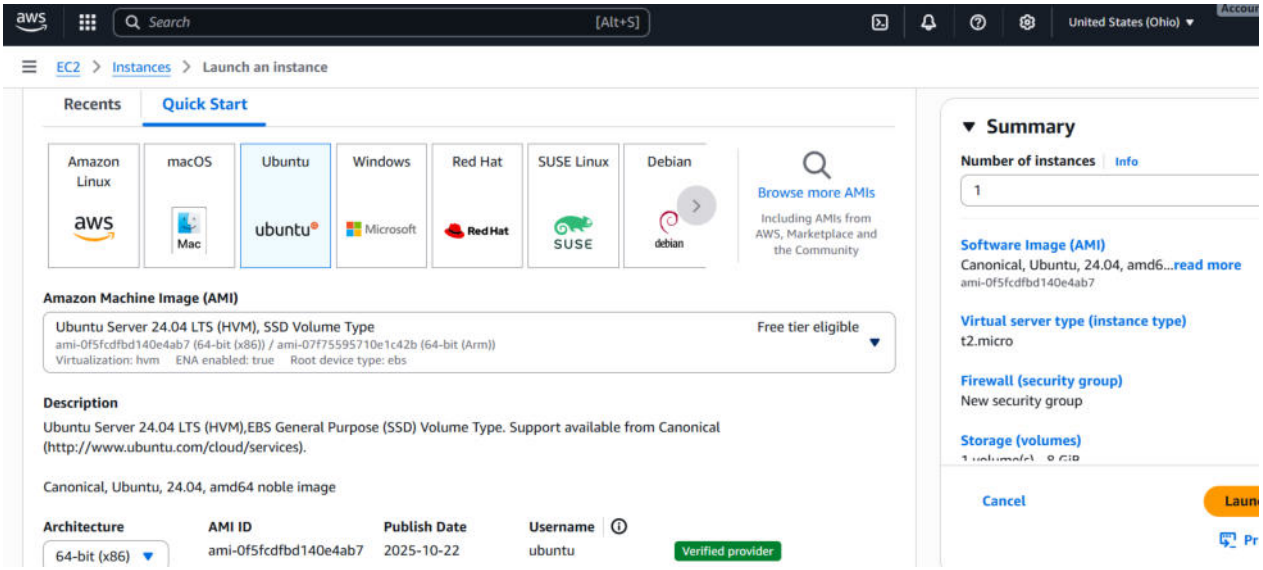
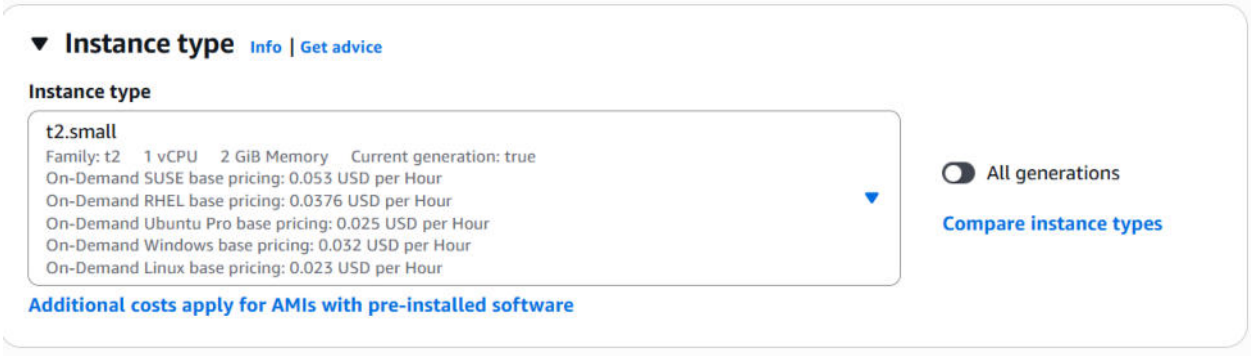



**Figura 23**

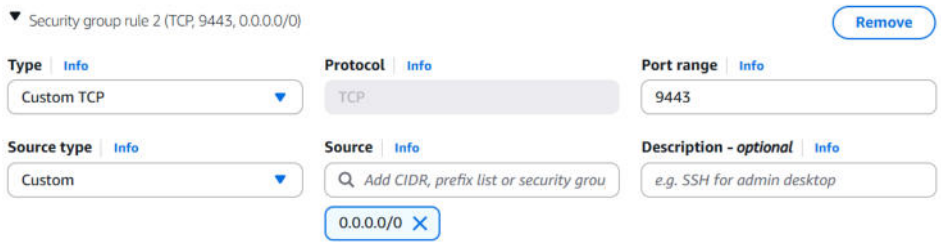
*Modificación de características de hardware para instalación de SafeLine*




Para habilitar el servicio es necesario habilitar el puerto 9443 por el cual se levantará el aplicativo.

**Figura 24**

*Configuración del Security Group TCP para de administración del WAF*




Para la instalación del WAF en el servidor de Ubuntu seguimos el manual indicado por el fabricante (chaitin, 2025). Se inicia instalando Docker con la versión indicada, y una carpeta de SafeLine donde se almacenará la última versión de Docker compose.

**Figura 25**

*Instalación de Docker y Docker compose*



Una vez creada la carpeta de se procede a definir los valores .env recomendados por el proveedor y finalmente instalamos todo el servicio con el comando de Docker compose como se muestra en la Figura 26.

**Figura 26**

*Configuración de archivo .env SafeLine*

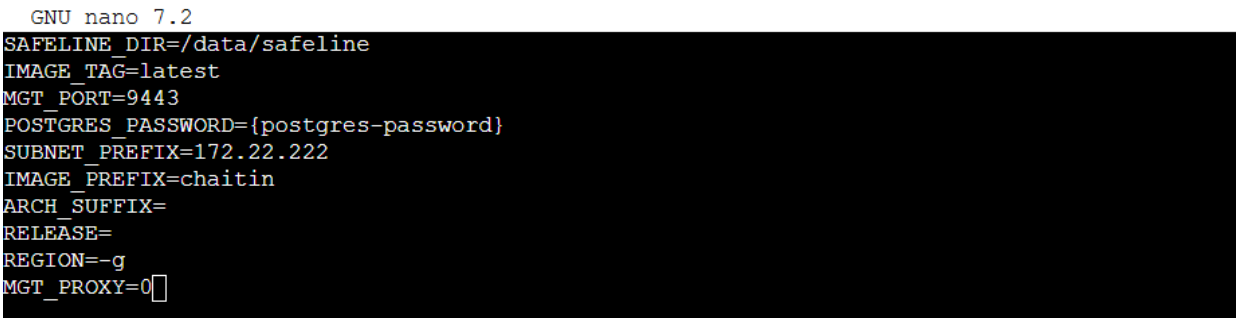
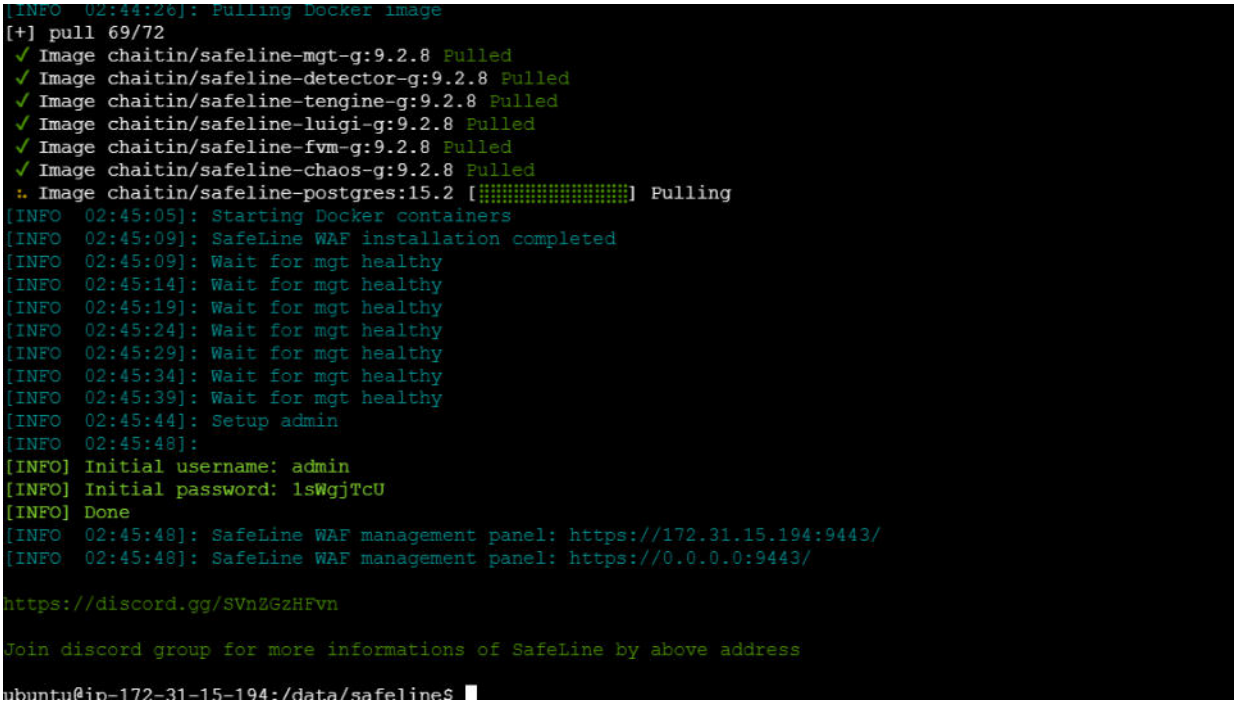
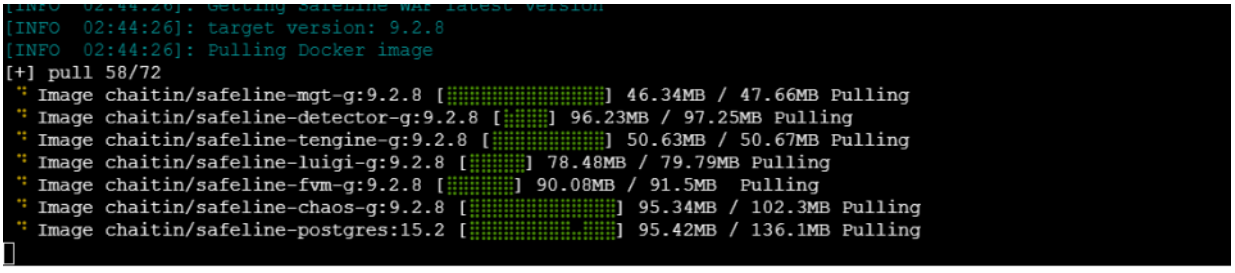



**Figura 27**

*Instalación de Safeline con el comando Docker compose up -d*



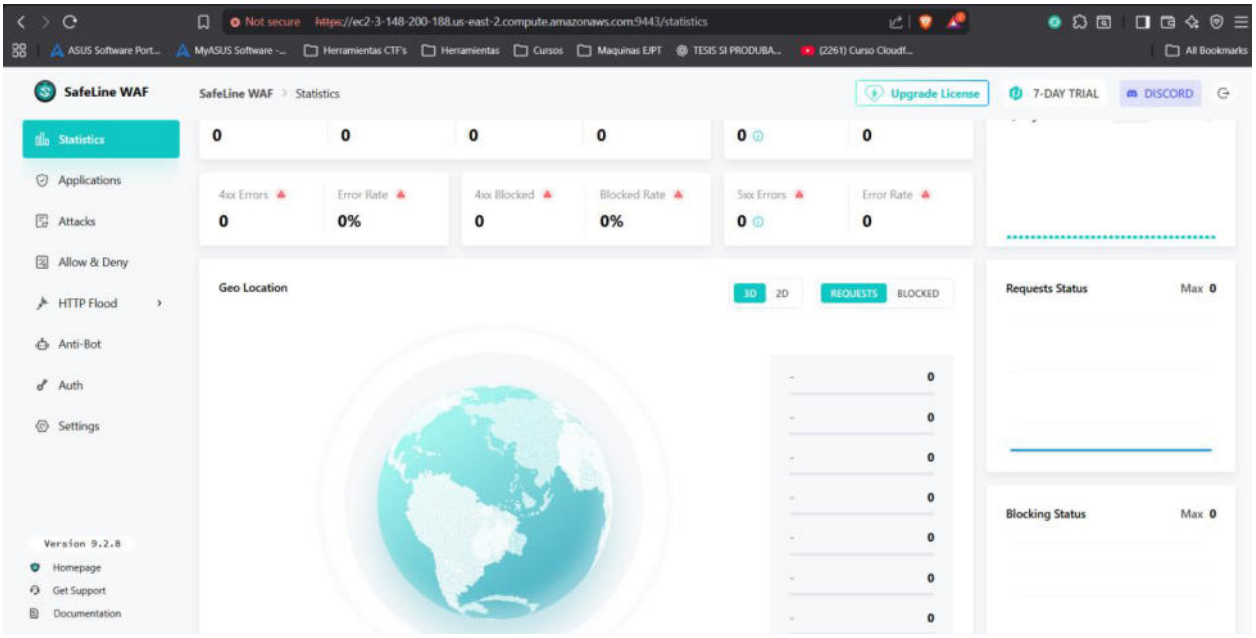



Para comprobar el funcionamiento de SafeLine se debe ingresar a la IP publica que de la instancia de AWS con el puerto 9443 para acceder al Dashboard de administración de Safeline.

https://ec2-3-151-89-137.us-east-2.compute.amazonaws.com:9443/

**Figura 28**

*Dashboard de administración Safeline WAF*

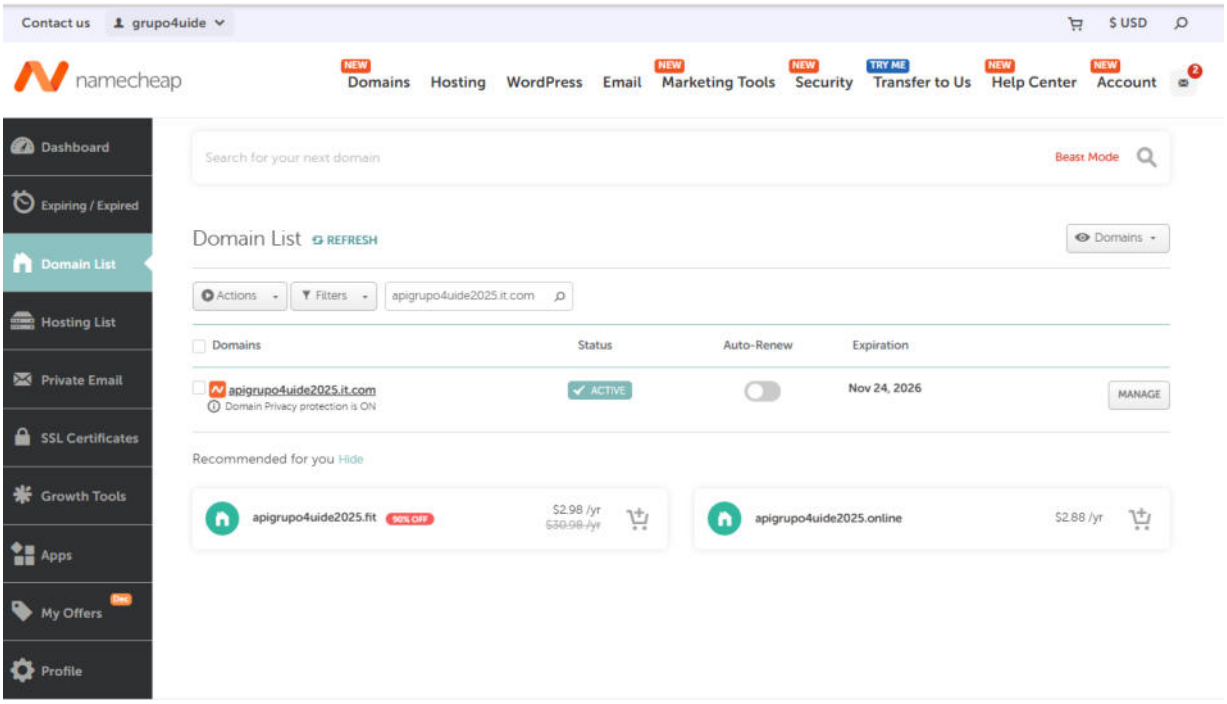



**Dominio para WAF código abierto**

El dominio adquirido para que se integre con el WAF fue **apigrupo4uide2025.it.com** en namecheamp**.** Al igual que Cloudflare, Safeline requiere únicamente que el dominio esté administrado desde su panel y que el servidor de destino permita recibir el tráfico proveniente de los rangos IP autorizados.

**Figura 29**

*Dominio para API integrada con WAF código abierto*




**Configuración SafeLine y reglas del WAF**

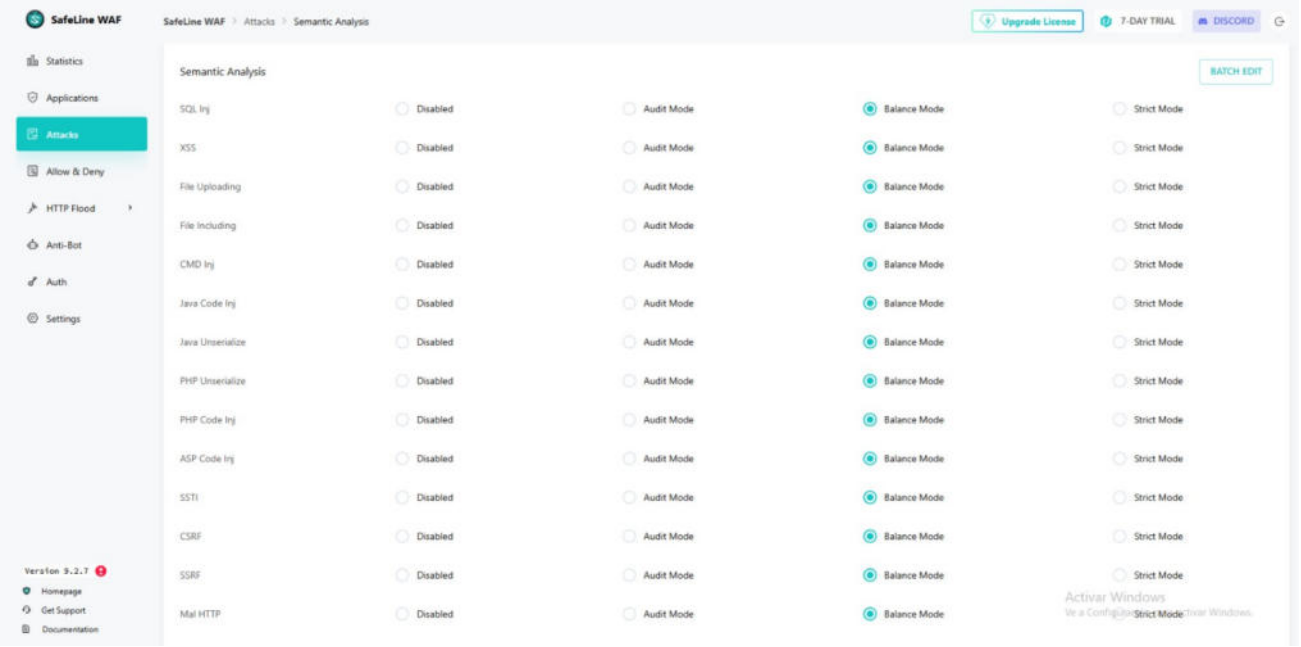
En particular en este aplicativo no se puede configurar las reglas, por defecto ya vienen habilitadas las para los siguientes tipos de ataques; Estas reglas se pueden ser personalizadas, pero para poder comprar con el WAF comercial vamos a utilizar las reglas predeterminadas.

- SQL Injection
- Cross-Site Scripting (XSS)
- Remote Command Execution
- Directory Traversal
- File Inclusion

- Brute Force
- Fuzzing / Reconocimiento automatizado
- Ataques Zero-Day detectados mediante patrones heurístico

**Figura 30**

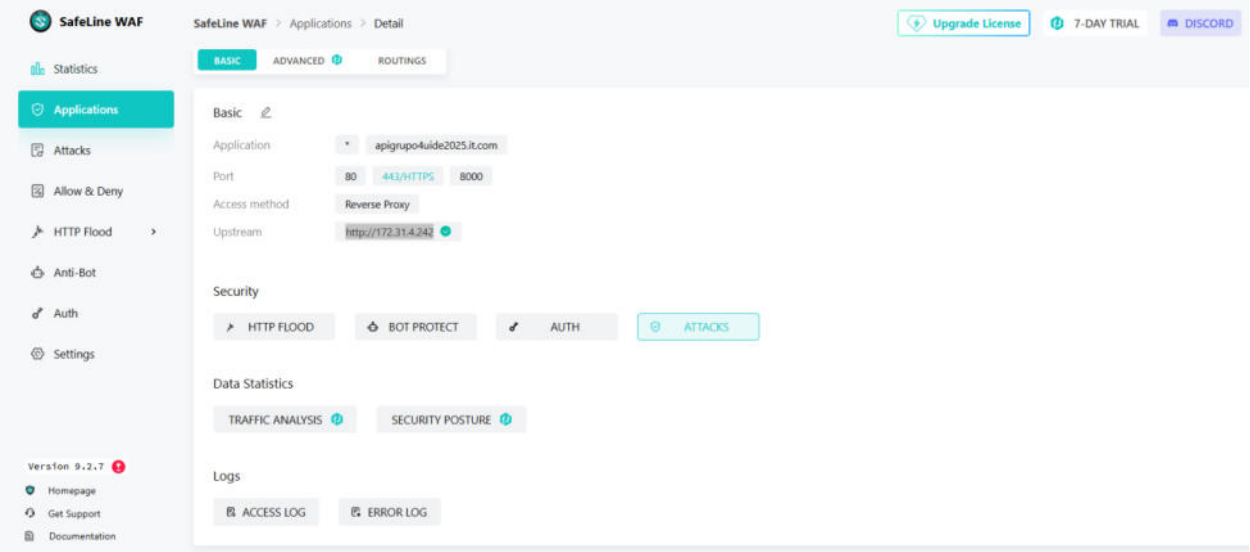
*Activación de modos de protección SafeLine*




Finalmente se procede al registro oficial en la aplicación, incluyendo el dominio registrado, la dirección IP en la que se encuentra desplegada la API, y el certificado digital generado por la plataforma Safeline.

**Figura 31**

*Registro de dominio en SafeLine WAF*



### *3.6. Máquina atacante*

Para la implementación de la máquina atacante se decidió utilizar Kali Linux. distribución de Linux basada en Debian, específicamente diseñada para temas de seguridad muy variados, como análisis de redes, ataques inalámbricos, análisis forenses, etc. Los recursos utilizados en este equipo son:

- CPU: 2 núcleos
- RAM: 4 GB
- Disco duro: 50 GB

Creamos un entorno Python, instalando las librerías necesarias para el correcto funcionamiento.

sudo apt install python3-venv python3-pip

La máquina Kali Linux está preparada con un script automatizado para realizar las pruebas. A continuación, se describe el listado de pruebas para evaluar los WAFs Cloudflare **y** Safeline**,** enfocado únicamente en pruebas autorizadas, análisis, validación de reglas y observabilidad en entornos controlados.

El script permite seleccionar los dominios y pruebas a realizar.

Los dominios configurados en cada WAF o directamente la IP pública de la API:

"WAF SafeLine", "http://apigrupo4uide2025.it.com"

"WAF Cloudflare", "http://www.apigrupouide2025.com"

"IP Directo", "http://3.137.13.237"

Opciones para realizar las pruebas:

1. SQL Injection

2. XSS

3. Fuzzing

4.Payload Tampering

5.Solicitudes legítimas tipo curl

6. Mostrar resumen y exportar métricas (CSV)

# Capítulo 4

## 4. Análisis de Resultados

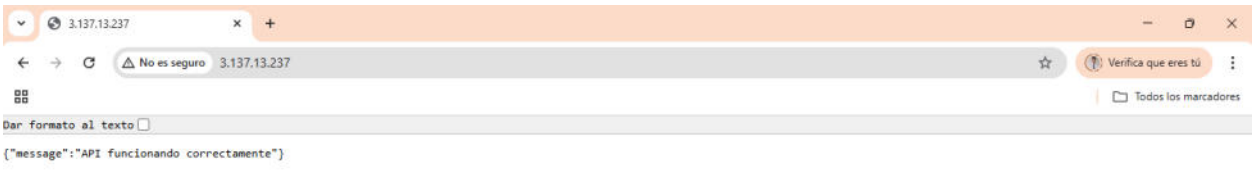
En esta sección se detallan las medidas de rendimiento obtenidas en cada proceso planteado en el desarrollo. Estas métricas corresponden a la fase de ejecución de ataques sobre la API vulnerable, tanto en escenarios sin protección como con los WAFs configurados.

### *4.1 API ante ataques sin protección.*

Primero validamos que la API se encuentra habilitada en la IP pública de AWS donde se configuró la instancia. En este caso la IP http://3.137.13.237 es donde se encuentra habilitado la API y se puede ver en la Figura 32.

**Figura 32**

*IP pública del API expuesta ante ataques sin protección*




Se ejecutó una serie de acciones sobre la API con el fin de verificar su correcto funcionamiento y determinar si alguna de ellas era identificada erróneamente como un ataque. Al no contar con protección desde su código todos los comandos fueron ejecutados con éxito. Por lo cual no se puede evaluar la eficiencia del API ante ataques.

### *4.2 Evaluación de la efectividad del WAF de Cloudflare ante ataques sobre un API Rest*

Para el segundo escenario, utilizando la máquina atacante, se generó un total de 31 solicitudes, tanto interacciones legítimas como intentos de ataque con el objetivo de evaluar la efectividad del WAF en la protección de la API Rest. Tal como se observa en la Figura 33, estas solicitudes permitieron analizar el comportamiento del sistema ante tráfico mixto y medir su capacidad de detección y bloqueo.

**Figura 33**

*Prueba de solicitudes legitimas y ataques a API Rest con WAF Cloudflare*

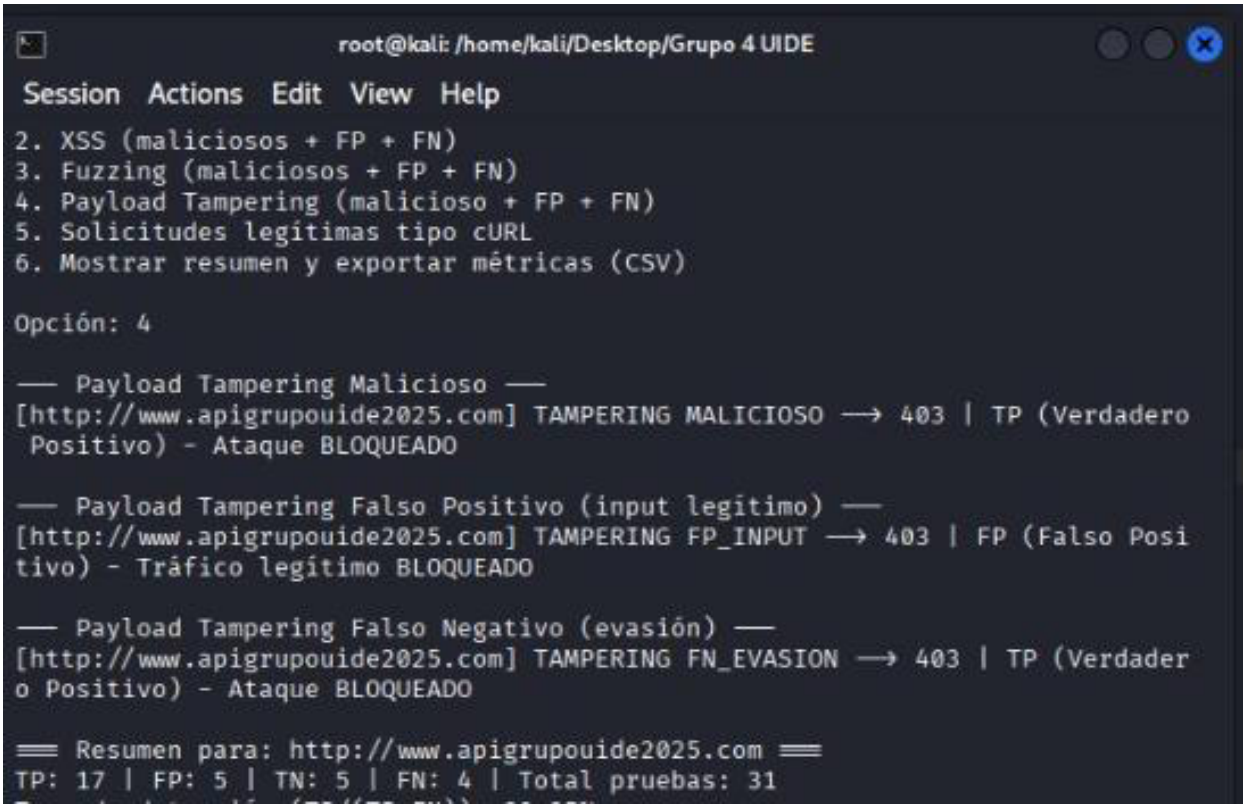


**Tabla 5**

*Detalle de solicitudes realizadas*

| **Tipos de Ejecuciones** | |
|---|---|
| *Ataques* | |
| *Tipo* | *Intentos* |
| SQL Injection | 6 |
| XSS | 6 |
| Fuzzing | 7 |
| Payload Tampering | 2 |
| *Acción* | |
| Solicitudes legitimas | 10 |

Al realizar las pruebas se obtuvieron los resultados presentados en la Figura 34. A partir de estos datos, fue posible calcular las métricas correspondientes, las cuales se detallan en la Tabla 6.

**Figura 34**

*Matriz de confusión del WAF Cloudflare*

| **WAF** | **TP** | **TN** | **FP** | **FN** |
|---|---|---|---|---|
| Cloudflare | 17 | 5 | 5 | 4 |

**Tabla 6**

*Métricas para evaluar la efectividad del WAF Cloudflare*

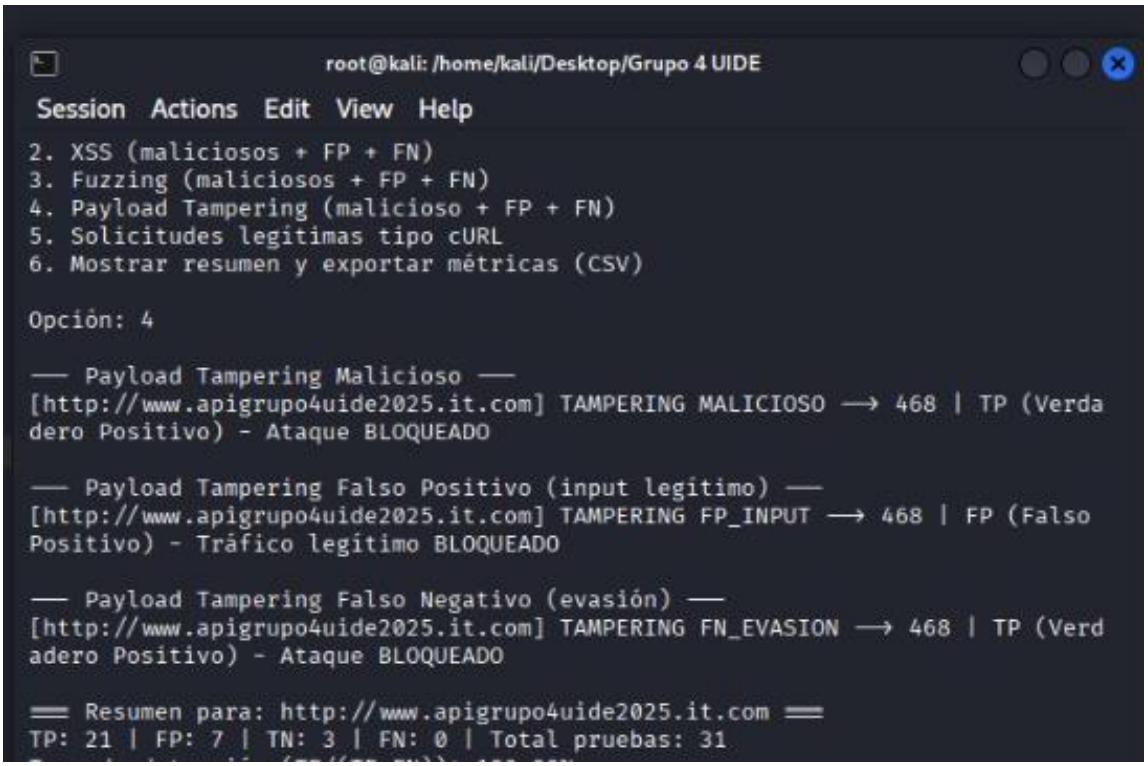
| WAF | TPR(%) | FPR(%) | TNR(%) | Precisión equilibrada(%) |
|---|---|---|---|---|
| Cloudflare | 81,00 | 50,00 | 50,00 | 65,48 |

### *4.3 Evaluación de la efectividad del WAF de Safeline ante ataques sobre un API Rest*

Aplicamos el mismo escenario utilizando la máquina atacante, se generó un total de 31 solicitudes tanto interacciones legítimas como intentos de ataque con el objetivo de evaluar la efectividad del WAF en la protección de la API Rest. Tal como se observa en la Figura 35, estas solicitudes permitieron analizar el comportamiento del sistema ante tráfico mixto y medir su capacidad de detección y bloqueo.

**Figura 35**

*Prueba de solicitudes legitimas y ataques a API Rest con WAF Safeline*




Al realizar las pruebas se obtuvieron los resultados presentados en la Figura 36. A partir de estos datos, fue posible calcular las métricas correspondientes, las cuales se detallan en la Tabla 6.

**Figura 36**

*Matriz de confusión del WAF Safeline*

| WAF | TP | TN | FP | FN |
|---|---|---|---|---|
| **Safeline** | **21** | **3** | **7** | **0** |

**Tabla 7**

*Métricas para evaluar la efectividad del WAF SafeLine*

| WAF | TPR(%) | FPR(%) | TNR(%) | Precisión equilibrada(%) |
|---|---|---|---|---|
| Safeline | 100,00 | 70,00 | 30,00 | 65,00 |

### *4.4 Comparación del Desempeño de los WAF Evaluados*

Al analizar el comportamiento de ambos WAF frente a las pruebas realizadas, se evidencian diferencias claras en su capacidad para procesar adecuadamente tanto el tráfico malicioso como el legítimo. En esta evaluación, Safeline muestra un enfoque más eficiente hacia la detección, alcanzando un TPR del 100%, lo que significa que detectó la totalidad de los ataques enviados. No obstante, este nivel de sensibilidad también incrementó su tasa de falsos positivos, reflejada en un FPR del 70% y un TNR del 30%, lo que indica dificultades para reconocer correctamente el tráfico legítimo.

Cloudflare, en cambio, presenta una detección de ataques del 81%, aceptable pero inferior a la alcanzada por Safeline. Sin embargo, gestiona mejor el tráfico válido, con un FPR del 50% y un TNR del 50%, lo que reduce la probabilidad de bloquear solicitudes legítimas y ofrece un comportamiento más equilibrado en entornos donde la continuidad del servicio es prioritaria.

A pesar de estas diferencias, la métrica de precisión equilibrada muestra valores muy similares en ambos casos, ubicándose cerca del 65%. Esto refleja que cada solución compensa sus fortalezas y debilidades de forma distinta: Safeline se inclina hacia una detección total de

ataques, mientras que Cloudflare prioriza mantener un mejor balance entre seguridad y accesibilidad del tráfico permitido.

**Tabla 8**

*Métricas de comparación efectividad de los WAF*

| *Métrica* | *Safeline* | *Cloudflare* | *Mejor* |
|---|---|---|---|
| **TPR (detección de ataques)** | 100% | 81% | **Safeline** |
| **FPR (errores contra tráfico legítimo)** | 70% | 50% | **Cloudflare** |
| **TNR (identifica tráfico legítimo)** | 30% | 50% | **Cloudflare** |
| **Precisión Equilibrada** | 65.00% | 65.48% | **Cloudflare** |

# Capítulo 5

## 5. Conclusiones y Recomendaciones

### *5.1. Conclusiones*

De acuerdo con el proceso de configuración de reglas en los diferentes WAF evaluados, se identifica que el WAF Cloudflare requiere mínimo mantenimiento y su gestión es más sencilla a diferencia del WAF SafeLine que requiere personal técnico dedicado para su gestión y mantenimiento.

Luego de ejecutar las pruebas comprendidas por ataques y solicitudes reales, a través de los resultados, se puede concluir que existen diferencias en el rendimiento y enfoque operativo entre un WAF comercial de pago, como Cloudflare, y un WAF de código abierto, como SafeLine. Entre las principales diferencias se identifica:

- Los resultados evidencian que el WAF SafeLine alcanza una tasa de detección de ataques (TPR) del 100 %, lo que demuestra que, a pesar de ser una solución de código abierto, proporciona una alta capacidad de identificación y bloqueo de amenazas, evidenciando eficacia en escenarios donde la prioridad de la seguridad es alta y donde se busca reducir al mínimo la probabilidad de que un ataque pueda evadir el sistema de protección. Este alto nivel de detección en SafeLine se ve acompañado por una tasa elevada de falsos positivos (70 %) y una tasa baja de reconocimiento de tráfico legítimo (TNR del 30 %), lo cual puede afectar la experiencia del usuario y la continuidad de los servicios. Esto indica que, aunque

tiene buena efectivad en seguridad, el WAF de código abierto presenta limitaciones en el afinamiento de sus reglas y en la optimización del procesamiento del tráfico legítimo.

- De acuerdo con las pruebas realizadas, el WAF de Cloudflare, al ser una solución comercial de pago, presenta un desempeño más equilibrado. Aunque su tasa de detección de ataques (81 %) es inferior a la de SafeLine, presenta menores errores sobre tráfico legítimo (FPR del 50 %) y mejor reconocimiento de solicitudes válidas (TNR del 50 %). Este resultado representa mayor precisión equilibrada (65.48 %), lo que sugiere un mejor balance entre seguridad, rendimiento y usabilidad, características esperadas en una solución comercial diseñada para entornos productivos y de alta disponibilidad.

- Los resultados indican que el WAF como SafeLine de código abierto ofrece un nivel de protección elevado frente a ataques, siendo adecuado para entornos de pruebas, laboratorios o sistemas con alta tolerancia a bloqueos de tráfico legítimo. Al comparar un WAF de pago como Cloudflare justifica su costo al ofrecer mayor estabilidad, menor impacto en el tráfico legítimo y un rendimiento global más balanceado, lo que lo convierte en la opción más recomendada para aplicaciones en producción y servicios críticos.

Luego de haber finalizado esta evaluación, se evidencia la importancia de utilizar métricas estandarizadas para la comparación de soluciones de seguridad, lo que permite una toma de decisiones informada y basada en evidencia, lo cual resulta fundamental para la implementación de mecanismos efectivos de protección en aplicaciones web.

## *5.2. Recomendaciones*

Diseñar y optimizar las arquitecturas, adoptando estándares o regulaciones internacionales y mejores prácticas de seguridad, así como la integración de tecnologías de inteligencia artificial y aprendizaje automático, permitirá reducir los riesgos y facilitará el cumplimiento de normativas.

Priorizar la seguridad e integrarla desde el inicio del ciclo de vida del desarrollo y en todas las fases es fundamental, a través de colaboración, automatización y procesos claros, para evitar costos de posibles incidentes y asegurar la continuidad de las operaciones y servicio al usuario final.

La seguridad de una API REST requiere un enfoque integral, incluyendo: control de acceso y autenticación, validación de datos, protección contra amenazas, gestión de errores y vulnerabilidades, monitoreo constante, para crear una API robusta, confiable y capaz de resistir intentos de explotación y garantizar un ecosistema de API seguro y fiable.

En entornos de producción se debe implementar herramientas de vanguardia que permitan proteger las aplicaciones contra amenazas y vulnerabilidades en constante evolución,

por lo que es crucial asociarse con proveedores de seguridad adecuados para mantener defensas sólidas de todos los servicios.

Se recomienda realizar pruebas periódicas que permitan evaluar los WAFs, utilizando métricas estandarizadas como TPR, FPR, TNR y precisión equilibrada, para a medir la evolución del rendimiento del sistema de seguridad ante nuevos tipos de ataques y cambios en los componentes o comportamiento del tráfico.

Para implementaciones en entornos productivos, se recomienda utilizar capas de seguridad, como sistemas de detección de intrusos (IDS/IPS), firewall, autenticación robusta y monitoreos continuos, ya que el WAF por sí s olo no garantiza una protección completa frente a todas las amenazas web.

## 6. Referencias Bibliográficas

Akamai Technologies. (s. f.). *¿Qué es WAAP? Las soluciones WAAP ofrecen protección contra amenazas en constante evolución*. https://www.akamai.com/es/glossary/what-is-waap

Amazon Web Services (AWS). (s. f.). *AWS Web Application Firewall.* https://aws.amazon.com/es/waf/

Amazon Web Services. (2024). *Amazon EC2: Virtual server hosting in the cloud.* Amazon EC2: Virtual server hosting in the cloud.: https://aws.amazon.com/ec2/

Babu, S. & Geekflare. (2025, febrero 4). *17 Mejor software de cortafuegos de aplicaciones web [Selección de los mejores]*. https://geekflare.com/es/cybersecurity/best-web-application-firewall/

Chaitin. (2025, diciembre). *Documentación SafeLine*. Documentación SafeLine: https://docs.waf.chaitin.com/en/GetStarted/Deploy

Check Point Software Technologies Ltd. (2024–2025). *CloudGuard WAF Comparison Project: Report*. https://www.checkpoint.com

Check Point Software Technologies Ltd (s. f.). Key parameters when evaluating a Web Application Firewall (WAF). https://www.checkpoint.com/cyber-hub/cloud-security/what-is-web-application-firewall/key-parameters-when-evaluating-a-web-application-firewall-waf/

Cibersafety. (2025, abril 22). *¿Qué es Burp Suite y para qué sirve? Cibersafety.* https://cibersafety.com/que-es-burp-suite-herramienta-ciberseguridad/

Ciberseguridad. (s. f.). *Nikto, un práctico escáner de vulnerabilidades de sitios web*. Ciberseguridad. https://ciberseguridad.com/herramientas/software/nikto/#%C2%BFQue_es_Nikto

Cloudflare. (s. f.). *WAF de Cloudflare*. Recuperado de https://www.cloudflare.com/es-la/application-services/products/waf/

Código6. (s. f.). *Introducción y ejemplos prácticos de OWASP ZAP para seguridad web*. Código6. https://www.codigo6.com/introduccion-y-ejemplos-practicos-de-owasp-zap-para-seguridad-web/

Coherent Market Insights. (2025, junio 3). *Web application firewall market size and share analysis – growth, trends & forecasts (2025–2032).* https://www.coherentmarketinsights.com/industry-reports/web-application-firewall-market

Cyber Writes Team. (2025, febrero 3). *Top 10 Best Web Application Firewall (WAF) in 2025*. Cyber Security News. https://cybersecuritynews.com/best-web-application-firewall-waf

DataScientest. (2025, marzo 21). *Fuzzing: ¿Qué es? ¿Cómo se utiliza?* DataScientest. https://datascientest.com/es/fuzzing-que-es

Datos.gob.es. (2025, febrero 5). *Guía para la publicación de APIs: diseño REST.* https://datos.gob.es/sites/default/files/doc/file/guia_publicacion_apis.pdf?utm_source=chatgpt.com

Imagina Formación. (30 de octubre de 2024). *¿Qué es Postman y para qué sirve?* https://imaginaformacion.com/tutoriales/que-es-postman-y-para-que-sirve Fielding, R. T.

(2000). *Architectural Styles and the Design of Network-based Software Architectures.* California: University of California, Irvine.

Fortinet. (s. f.). *Diseño de un WAF resistente: De la arquitectura a la implementación*. https://www.fortinet.com/lat/resources/cyberglossary/waf-architecture

F5. (s. f.). *OWASP API Security Top 10 Overview and Best Practices*. F5. https://www.f5.com/es_es/glossary/owasp-api-security-top-10

Gartner Research. (2019, febrero 28). *Defining Cloud Web Application and API Protection Services*. https://www.gartner.com/en/documents/3903064#:~:text=Summary,will%20improve%20their%20security%20posture

IBM. (2025, abril 24). *¿Qué es la API REST? IBM Think*. https://www.ibm.com/mx-es/think/topics/rest-apis

Marquez, D. (2025). *Todo lo que necesitas saber sobre API REST: Glosario de términos esenciales y más. DEV Community*. https://dev.to/dennysjmarquez/todo-lo-que-necesitas-saber-sobre-api-rest-glosario-de-terminos-esenciales-y-mas-29pc

Namecheap. (2025). *Domains list*. https://ap.www.namecheap.com/domains/list/

OWASP. (2019). *API8:2019 Injection*. https://owasp.org/API-Security/editions/2019/en/0xa8-injection/

OWASP Foundation. (2023). *OWASP API Security Top 10*. https://owasp.org/API-Security/?utm_source

OWASP Foundation. (2023). *OWASP API Security Project*. https://owasp.org/www-project-api-security/

OWASP. (2021). *Injection* . OWASP Top 10: https://owasp.org/Top10/A03_2021-Injection/

OWASP Foundation. (2023). *OWASP Top 10 API Security Risks – 2023.* https://owasp.org/API-Security/editions/2023/en/0x11-t10/

OWASP Foundation, (2025, noviembre). *OWASP Top Ten*. https://owasp.org/www-project-top-ten/

Palo Alto Networks. (s. f.). *What Is Web Application and API Protection? Palo Alto Networks Cyberpedia*. https://www.paloaltonetworks.lat/cyberpedia/what-is-web-application-and-api-protection

PortSwigger. (s.f.). *Authentication*. PortSwigger Web Security: https://portswigger.net/web-security/authentication

Radware. (2025). *Radware: Security, Application Delivery & Cloud Solutions*. https://www.radware.com/

Ravoof, S. (2025, enero 17). *API Rest vs API Web: Todo lo que necesitas saber. Kinsta*. https://kinsta.com/es/blog/api-rest-vs-api-web/

SciSimple. (2025, Octubre 23). *Avances en técnicas de fuzzing para la seguridad del software*. https://scisimple.com/es/articles/2025-10-23-avances-en-tecnicas-de-fuzzing-para-la-seguridad-del-software--akyojox

SysAdminOK. (23 de Junio de 2023). *Qué es y para qué sirve el Fuzzing: Un enfoque integral a la seguridad de software*.https://www.sysadminok.es/blog/ciberseguridad/que-es-y-para-que-sirve-el-fuzzing-un-enfoque-integral-a-la-seguridad-de-software/

The Hacker News (2025, mayo 23). *SafeLine WAF: Open Source Web Application Firewall with Zero-Day Detection and Bot Protection*. https://thehackernews.com/2025/05/safeline-waf-open-source-web.html

Wallarm. (2025, marzo 30). *¿Qué es WAF (Firewall de aplicaciones web)?*. https://lab.wallarm.com/what/waf-firewall-de-aplicaciones-web/?lang=es#:~:text=Un%20WAF%20puede%20afectar,See%20Wallarm%20In%20Action

## 7. Anexo

**Link con máquinas y accesos:**

https://mailinternacionaledu-my.sharepoint.com/:f:/g/personal/cacadenapa_uide_edu_ec/EgzT774UsWtHnztEXCM_ywQBzX-5qqZZsKv2kljmK1NFBA?e=cua3wX