Actividad 1: Implementación y Evaluación de Content Security Policy (CSP)

Tema: *Protección contra XSS e inyección de scripts*

Objetivo: *Aplicar una Content Security Policy (CSP) restrictiva y evaluar su impacto*

¿Qué es CSP?

CSP (Content Security Policy) es un mecanismo de seguridad que **limita los orígenes de scripts, estilos e imágenes** en una aplicación web para evitar ataques como **XSS**.

Implementación

Crear una página web vulnerable sin CSP

Crear el archivo **vulnerable**: *index.html*

```
<!DOCTYPE html>
<html>
<head>
    <title>Prueba sin CSP</title>
</head>
<body>
    <h2>Introduce tu nombre:</h2>
    <form action="xss.php" method="GET">
        <input type="text" name="xss">
        <button type="submit">Saludar</button>
    </form>
</body>
</html>
```

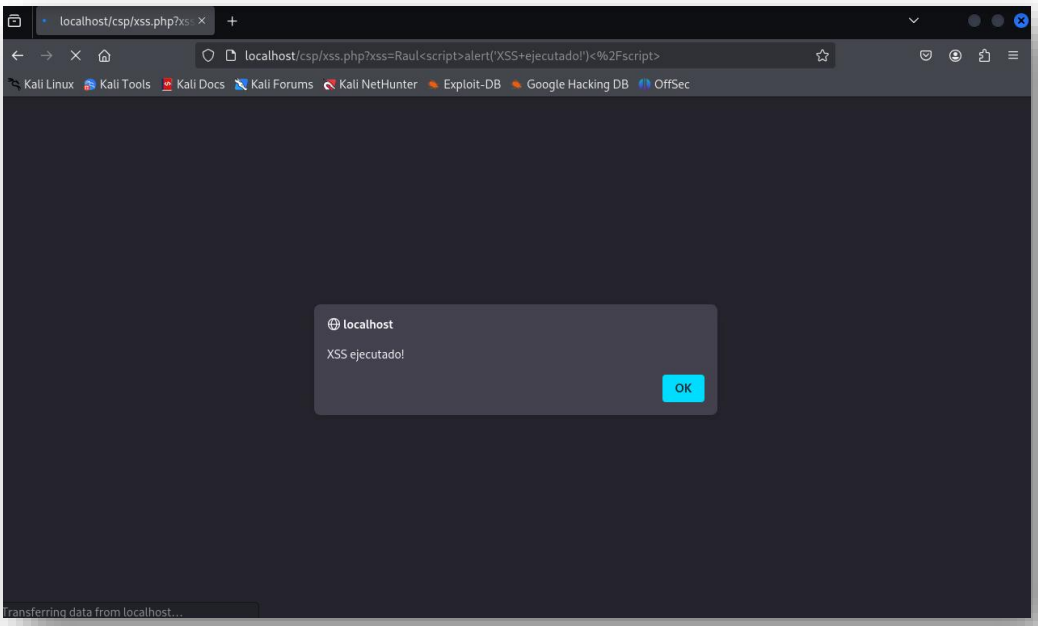
Y el fichero *xss.php*:

```
<?php
    echo "Hola, " . $_GET['xss'];
?>
```

Probar un ataque XSS básico:

```
<script>alert('XSS ejecutado!')</script>
```

Si el *alert()* se ejecuta, la página es **vulnerable**.



Implementar CSP

Para evitar la ejecución de JavaScript inyectado, usar **mod_headers** en Apache, para ello configurar CSP en Apache en el VirtualHost (*000-default.conf*)**

Editar el archivo de configuración del sitio en Apache `000-default.conf`:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Agregar la siguiente línea dentro de *<VirtualHost *:80>*:

```
<IfModule mod_headers.c>
    Header always set Content-Security-Policy "default-src 'self'; script-src
'self'"
</IfModule>
```

Header always set: "Agrega este header HTTP siempre, sin importar el tipo de respuesta." Hay otras variantes como set, append, unset, etc…
Content-Security-Policy: Es el header que ayuda a proteger el sitio de ataques XSS.

- *default-src 'self'*: Todos los recursos (imágenes, estilos, scripts, iframes, etc.) solo se pueden cargar desde el mismo origen que la página (misma IP/dominio y puerto).
- *script-src 'self'*: Específicamente los scripts solo pueden venir del mismo lugar.

¿Qué **no** permite? Cualquier script externo (como los de Google Analytics, jQuery CDN, etc.), y más importante: los scripts insertados directamente en el HTML

Habilitar *mod_headers* (si no está habilitado Apache ignora el bloque IfModule por completo) y reiniciar Apache**:**

```
sudo a2enmod headers
sudo systemctl restart apache2
```

Comprobar si CSP bloquea XSS

Abrir el navegador y acceder a**:**

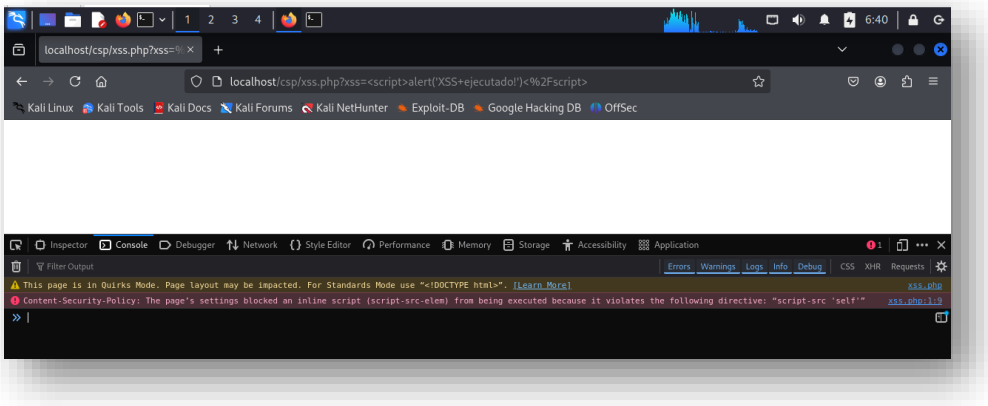
```
http://localhost/csp/index.html
```

Intentar inyectar un ataque XSS, como:

```
<script>alert('XSS ejecutado!')</script>
```

El navegador debería bloquearlo y la consola (F12 > "Consola") debería mostrar un *error* similar a:

```
Refused to execute script because 'script-src' directive disallows it.
```



Mitigación y Mejores Prácticas

*** Revisar violaciones de CSP en la consola del navegador y en logs de seguridad.**

Verificar que CSP se está aplicando después de hacer los cambios, para ello ejecutar en la terminal:

```
curl -I http://localhost/csp/index.html | grep Content-Security-Policy
```

Si muestra la política CSP en la salida, Apache la está aplicando correctamente.

¿Por qué editar 000-default.conf?

- Si CSP está en **apache2.conf** → Se aplica a **todo el servidor Apache**.
- Si CSP está en **.htaccess** → Se aplica **solo a una carpeta específica**.
- Si CSP está en **000-default.conf** → Se aplica **solo al VirtualHost principal**.

Si Apache usa múltiples sitios (*a2ensite*), también se debe editar cada archivo en */etc/apache2/sites-available/.*

*** CSP MÁS ESTRICTO PARA BLOQUEAR INLINE SCRIPTS Y eval()**

Para evitar que cualquier script inline *(<script>...</script>) o eval()* se ejecute, usar esta versión de CSP:

```
<IfModule mod_headers.c>
    Header always set Content-Security-Policy "default-src 'self'; script-src 'self'
'nonce-random123'; object-src 'none'; base-uri 'self'; frame-ancestors 'none'"
</IfModule>
```

¿Qué hace esta política?

- Solo permite scripts de la misma página (**self**).
- Bloquea **eval()**, **setTimeout('code')** y **setInterval('code')**.
- Bloquea scripts inline, a menos que usen **nonce="random123"**.
- Bloquea iframes y contenido incrustado (**frame-ancestors 'none'**).
- Evita la carga de **object** y **embed** (ataques con Flash o PDFs).
- Evita el uso de **<base>** externo (**base-uri 'self'**).

Para aplicarlo, editar **apache2.conf** o **.htaccess** o **000-default.conf**:

```
sudo nano /etc/apache2/apache2.conf
```

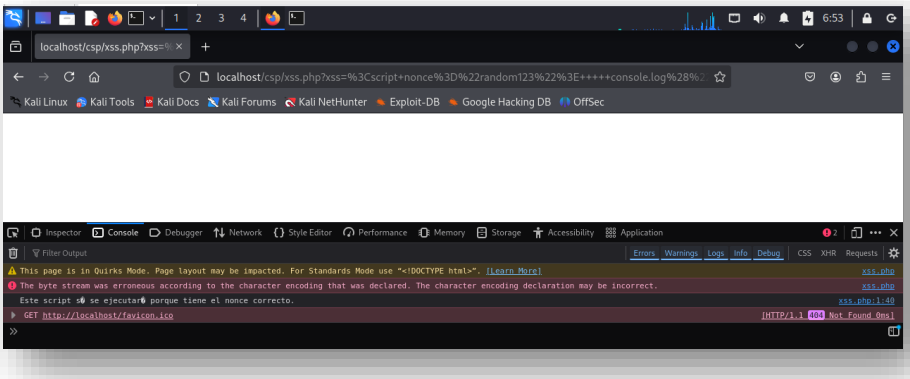
Añadir la configuración y reiniciar Apache:

```
sudo systemctl restart apache2
```

Importante: Si se tiene scripts inline **legítimos**, necesitarán el *nonce.*

¿Qué es un nonce? La palabra nonce viene de "number used once". Es un valor aleatorio, único y temporal que se le da a los scripts inline (esos escritos directamente en HTML) para que el navegador confíe en ellos y los ejecute, a pesar de que normalmente no lo haría bajo una política CSP estricta. Por ejemplo, nonce="random123":

```
<script nonce="random123">
    console.log("Este script sí se ejecutará porque tiene el nonce correcto.");
</script>
```



*** CSP MÁXIMA SEGURIDAD (Recomendada para Producción)**

Si quieres bloquear **todo lo posible**, usar esta versión aún más restrictiva:

```
<IfModule mod_headers.c>
    Header always set Content-Security-Policy "default-src 'self'; script-src 'self'
'nonce-ABC123'; style-src 'self' 'nonce-ABC123'; object-src 'none'; base-uri 'self';
frame-ancestors 'none'; upgrade-insecure-requests"
</IfModule>
```

¿Qué mejora esta versión?

- Evita **inline** scripts y estilos CSS inseguros (**style-src 'nonce-ABC123'**).
- Bloquea si la página intenta cargar contenido HTTP en un sitio HTTPS (**upgrade-insecure-requests**).
- Evita la carga de **object**, **embed** y **frames**.

Prueba BYPASS de XSS

Después de aplicar la política CSP, intenta realizar los siguientes ataques en el navegador:

Prueba inline script XSS (debe bloquearse):

```
<script>alert('XSS ejecutado!')</script>
```

Prueba eval() (debe bloquearse):

```
<script>eval("alert('XSS ejecutado!')")</script>
```

Prueba un ataque de inyección de <iframe> (debe bloquearse):

```
<iframe src="http://attacker.com"></iframe>
```

Prueba con eventos en elementos (debe bloquearse):

```
<img src="#" onerror="alert('XSS ejecutado!')">
```

Si el navegador bloquea estos ataques, entonces CSP estará funcionando correctamente.