# Laboratorio 2

Laboratorio 2 de ciberseguridad

Marcos Nicolau

## Contents

# 1 Introducción

En este laboratorio se pretende realizar un challenge a elección del sitio OWASP Juice Shop.

Juice Shop es una aplicación web que está repleta de inseguridades. Está diseñada para aprender, mejorar, reconocer y comprender los distintos tipos de vulnerabilidades existentes en la web. Cada desafío tiene su categoría y complejidad.

## 2 Resolución - Access Log

Este challenge posee 4 estrellas y pertenece a la categoría de ***Sensitive Data Exposure***, un tipo de vulnerabilidad que que hace referencia a todos los datos que pudiesen quedar expuestos, sin protección ni cifrado y que representan algún valor para la empresa o la persona que es dueña de ellos, por ejemplo: datos bancarios, información personal o familiar, información de usuarios y contraseñas, bases de datos e incluso información sobre la arquitectura de una red.

En este caso, se pretende obtener acceso a los logs del servidor.

Para ello, una buena estrategia sería hacer un análisis de todas las rutas que están abiertas en los puertos HTTP/HTTPS(80, 443), quizá en alguna de ellas se esté exponiendo información sensible.

Si uno quisiera chequear todas las rutas a mano sería una tarea interminable, por ello, junto a burp, se utiliza ffuf una herramienta que, a partir de una URL y una lista de palabras, automatiza el pedido a cada ruta.

La wordlist se obtiene del siguiente sitio SecLists en la sección de FUZZ.

Entonces, se ejecuta el siguiente comando en la terminal:

```
ffuf -w /usr/share/wordlists/ffuf.txt -u http://127.0.0.1:3000/FUZZ
```

```
dbdump.7z               [Status: 200, Size: 3748, Words: 266, Lines: 30, Duration: 35ms]
plugins/system/debug/debug.xml [Status: 200, Size: 3748, Words: 266, Lines: 30, Duration: 35ms]
.db.xml                 [Status: 200, Size: 3748, Words: 266, Lines: 30, Duration: 143ms]
db-central              [Status: 200, Size: 3748, Words: 266, Lines: 30, Duration: 150ms]
...
```

Inmediatamente se ve que todas las responses tienen el mismo *size* y son iguales. Analizando las request en el **burp** se verifica que ante toda ruta inexistente, el servidor hace una redirección al home de la aplicación por default.

Entonces al comando se le agrega una opción para filtrar los pedidos pedidos con el size 3748

```
ffuf -w /usr/share/wordlists/ffuf.txt -u http://127.0.0.1:3000/FUZZ -fs 3748
```

```
api                     [Status: 500, Size: 3017, Words: 235, Lines: 50, Duration: 57ms]
apis                    [Status: 500, Size: 3019, Words: 235, Lines: 50, Duration: 57ms]
assets                  [Status: 301, Size: 179, Words: 7, Lines: 11, Duration: 62ms]
ftp                     [Status: 200, Size: 11062, Words: 1568, Lines: 357, Duration: 339ms]
profile                 [Status: 500, Size: 1154, Words: 159, Lines: 50, Duration: 152ms]
```

```
promotion               [Status: 200, Size: 6586, Words: 560, Lines: 177, Duration: 193ms]
redirect                [Status: 500, Size: 3119, Words: 244, Lines: 50, Duration: 62ms]
rest                    [Status: 500, Size: 3019, Words: 235, Lines: 50, Duration: 62ms]
restaurants             [Status: 500, Size: 3033, Words: 235, Lines: 50, Duration: 62ms]
restore                 [Status: 500, Size: 3025, Words: 235, Lines: 50, Duration: 62ms]
restored                [Status: 500, Size: 3027, Words: 235, Lines: 50, Duration: 62ms]
restricted              [Status: 500, Size: 3031, Words: 235, Lines: 50, Duration: 62ms]
robots.txt              [Status: 200, Size: 28, Words: 3, Lines: 2, Duration: 23ms]
snippets                [Status: 200, Size: 792, Words: 1, Lines: 1, Duration: 27ms]
video                   [Status: 200, Size: 10075518, Words: 0, Lines: 0, Duration: 0ms]
Video                   [Status: 200, Size: 10075518, Words: 0, Lines: 0, Duration: 0ms]
```

De la lista vemos dos rutas que llaman la atención. **Assets** y **ftp**, ya que ambas hacen referencia a archivos del tipo que se está buscando.

Al acceder a la ruta de `/assets` uno se encuentra con una página en blanco, nada interesante. Por otro lado, al acceder `/ftp` se ve lo siguiente:

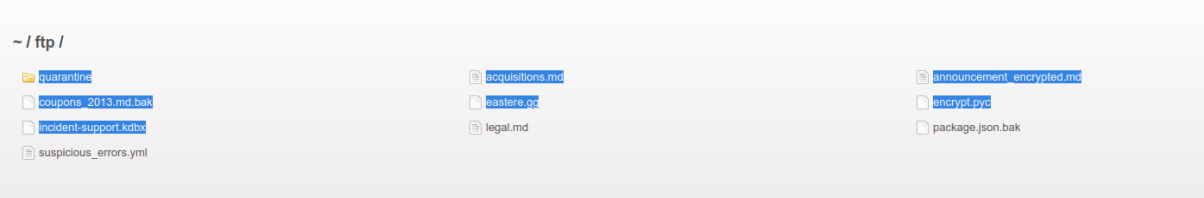


Figure 1: ftp route.png

Rápidamente destaca el archivo ***incident-support***, ya que nos da la pauta que support tuvo una incidencia de algún filtro. Nuevamente se hace una búsqueda con fuzz, pero esta vez se agrega el prefijo `/support`.

```
ffuf -w /usr/share/wordlists/ffuf.txt -u http://127.0.0.1:3000/support/FUZZ -fs 3748
```

```
logs                    [Status: 200, Size: 7778, Words: 1466, Lines: 342, Duration: 300ms]
Logs                    [Status: 200, Size: 7778, Words: 1466, Lines: 342, Duration: 300ms]
```

Obtuvimos la rutas **logs** y **Logs**, al acceder nos encontramos con el archivo de logs, se descarga y se comprueba efectivamente que son los logs de todos los requisitos que hemos hecho.


Figure 2: logs route.png

Al volver al home la aplicación nos avisa del desafío completado!

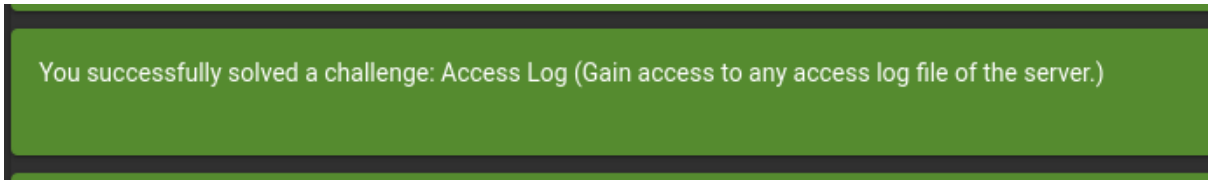


Figure 3: challenge solved.png

# 3 Conclusión

En este laboratorio, se llevó a cabo un desafío en el que se abordó una vulnerabilidad de exposición de datos sensibles.

En el ejercicio elegido quedó demostrado lo sencillo que es identificar rutas ocultas en un servidor utilizando herramientas como *ffuf*. La exposición de estas rutas subraya la importancia vital de realizar una validación exhaustiva y evitar exponer información sensible. Para ello es imprescindible implementar prácticas de desarrollo seguro para proteger la integridad y confidencialidad de los datos en servidores web.

Moraleja de la historia: siempre desarrolle con la mentalidad de que el cliente es malicioso y por tanto, uno como servidor debe validar exhaustivamente todos sus pedidos de manera tal que se pueda asegurar la fe del mismo :).