SQL Injection (SQLi)

Nicolás Montejano

Javier Calvo

¿Qué es un SQL Injection y qué tipos hay? ¿Qué es una inyección ‘ciega’?

- El SQL Injection es una vulnerabilidad que puede permitir a un atacante acceder y manipular información de una base de datos a través de una aplicación web, utilizando técnicas de inyección de código malicioso. Los atacantes explotan esta vulnerabilidad mediante la inserción de comandos SQL en los campos de entrada de datos de la aplicación web, lo que permite la ejecución de operaciones no autorizadas.
- Existen varios tipos de SQL Injection, como el basado en errores, tiempo y unión, cada uno con diferentes técnicas de ataque y niveles de complejidad. Por ejemplo, el SQL Injection basado en errores utiliza valores malintencionados que causan errores en la consulta SQL, mientras que el basado en tiempo utiliza consultas que se ejecutan lentamente para obtener información a través del tiempo de respuesta.
- Por otro lado, la inyección ciega se refiere a una variante de SQL Injection en la que la aplicación web no proporciona ningún mensaje de error en caso de que la consulta SQL sea incorrecta. En este caso, los atacantes utilizan técnicas de prueba y error para obtener información de la base de datos, como la enumeración de usuarios y contraseñas. Es importante tener en cuenta que, aunque este tipo de ataque es más difícil de detectar y prevenir, existen soluciones y prácticas de seguridad que pueden ayudar a mitigar el riesgo de esta vulnerabilidad.

¿Cuáles son los pasos para explotar una inyección basada en errores?

- La inyección basada en errores, también conocida como "Blind SQL injection" o "SQL injection a ciegas", es una técnica de explotación de vulnerabilidades en aplicaciones web que puede permitir a un atacante comprometer un sistema y acceder a datos confidenciales. A continuación, se presentan algunos pasos generales que pueden seguirse para explotar una inyección basada en errores:
- Identificar la existencia de la vulnerabilidad: El primer paso para explotar una inyección basada en errores es identificar si existe la vulnerabilidad en la aplicación web. Se puede utilizar herramientas de escaneo de vulnerabilidades o realizar pruebas manuales para detectar la presencia de la vulnerabilidad
- .Identificar la entrada vulnerable: Una vez identificada la vulnerabilidad, se debe identificar la entrada que es vulnerable a la inyección de código. La entrada puede ser un campo de formulario, una URL, una cookie o cualquier otra entrada que acepte datos de entrada del usuario.
- Identificar la base de datos y la estructura de la tabla: Una vez identificada la entrada vulnerable, el siguiente paso es identificar la base de datos subyacente y la estructura de la tabla. Esto se puede hacer mediante la prueba de diferentes valores y la observación de las respuestas de la aplicación.

¿Qué tipo de filtros se suelen poner en las aplicaciones para evitar los SQLi?

- Existen diferentes tipos de filtros que se pueden implementar en las aplicaciones web para prevenir o mitigar los ataques de inyección SQL (SQLi). Algunos de los filtros más comunes son:
- Validación de entrada: La validación de entrada es un proceso que se utiliza para asegurarse de que los datos ingresados por el usuario sean válidos y cumplan con ciertas reglas y restricciones. Esto puede incluir comprobar que los caracteres especiales sean reemplazados por caracteres escapados, restringir la longitud de la entrada, validar los tipos de datos de entrada, entre otros.
- Sanitización de entrada: La sanitización de entrada es un proceso que se utiliza para limpiar y normalizar los datos de entrada antes de ser procesados. Esto puede incluir la eliminación de caracteres especiales, conversión de caracteres a mayúsculas o minúsculas, eliminación de comentarios, entre otros.
- Parámetros de consulta preparados: Los parámetros de consulta preparados son una técnica que se utiliza para separar la consulta SQL de los datos de entrada del usuario. Los parámetros de consulta preparados permiten que los datos de entrada se envíen por separado de la consulta SQL, lo que hace que sea mucho más difícil para un atacante inyectar código malicioso.

¿Cómo sería posible evadir filtros? Indicar un par de ejemplos?

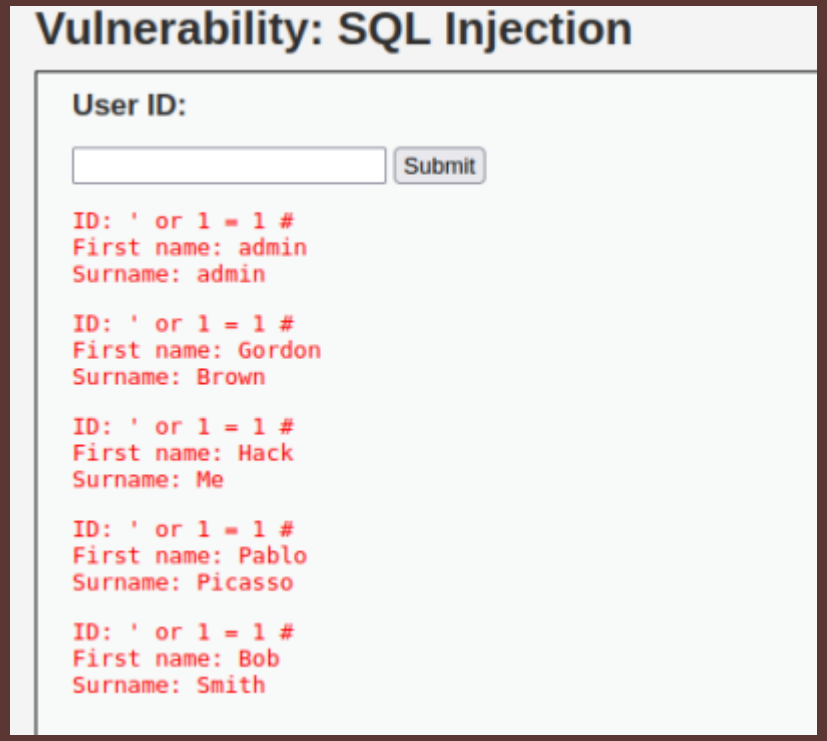
- Evadir filtros en una aplicación web puede ser posible mediante varias técnicas que pueden ser utilizadas por un atacante con conocimientos de seguridad informática. A continuación, se presentan algunos ejemplos de cómo se podría evadir filtros en una aplicación web:
- Utilización de técnicas de codificación y obfuscación: Una técnica común para evadir los filtros es utilizar técnicas de codificación y obfuscación para ocultar el código malicioso. Por ejemplo, se puede utilizar la técnica de codificación base64 para convertir el código malicioso en una cadena de caracteres que no sea detectada por el filtro.
- Utilización de caracteres especiales alternativos: Otra técnica común para evadir los filtros es utilizar caracteres especiales alternativos para los que el filtro no está diseñado para detectar. Por ejemplo, en lugar de utilizar el caracter ' (comilla simple) que puede ser bloqueado por el filtro, se puede utilizar el caracter ` (acento grave) que no es bloqueado.
- Uso de técnicas de inyección condicional: Los filtros se basan en patrones específicos para detectar código malicioso, por lo que una técnica de evasión común es utilizar técnicas de inyección condicional para engañar al filtro. Por ejemplo, en lugar de inyectar todo el código malicioso de una vez, se puede dividir en varias partes y utilizar técnicas de inyección condicional para evadir el filtro.

¿Es posible ejecutar comandos sobre una máquina o leer ficheros con un SQLi?

- Sí, es posible ejecutar comandos en una máquina o leer ficheros utilizando una inyección SQL (SQLi) en una aplicación web que no esté protegida adecuadamente.
- Por ejemplo, si una aplicación web no realiza una validación adecuada en los campos de entrada y es vulnerable a una inyección SQL, un atacante podría inyectar código malicioso en una consulta SQL y ejecutar comandos en la base de datos subyacente. A partir de ahí, si el usuario que se está utilizando para la conexión a la base de datos tiene suficientes permisos, el atacante podría leer o escribir en el sistema de archivos del servidor o incluso ejecutar comandos en la línea de comandos.
- Por otro lado, si la aplicación web permite la carga de archivos y no valida correctamente la extensión del archivo, un atacante podría subir un archivo malicioso como un archivo de imagen y luego utilizar una inyección SQL para ejecutar el archivo en el servidor. Si el archivo es un script PHP, por ejemplo, el atacante podría ejecutar cualquier comando que tenga permiso para ejecutar.

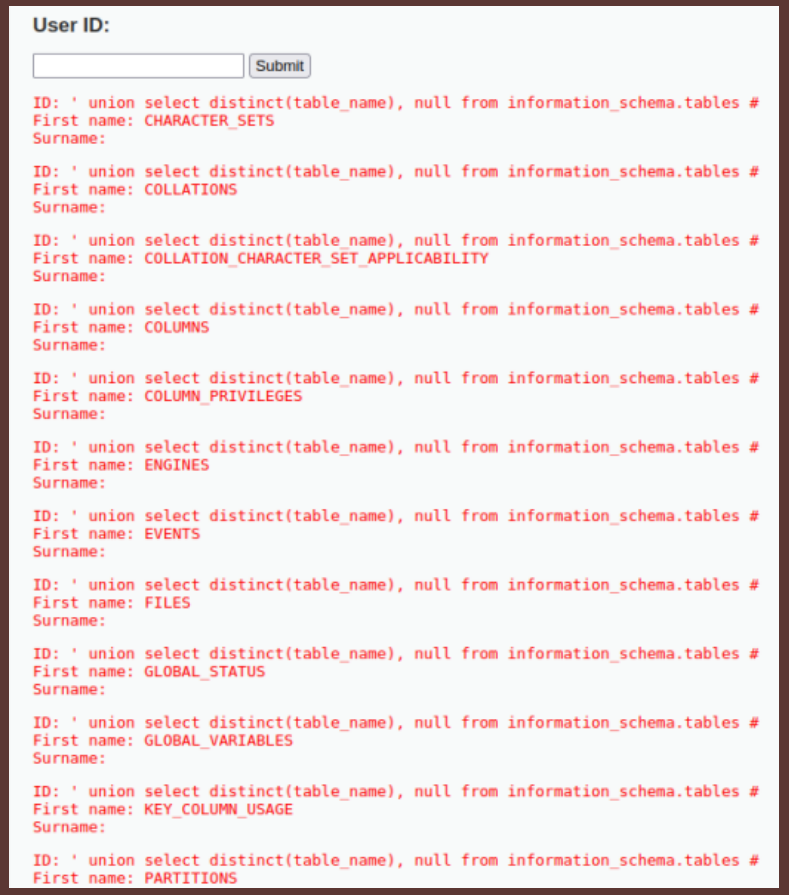
Explotar una inyección SQL en DVWA y extraer información de la base de datos

' or 1 = 1 #:



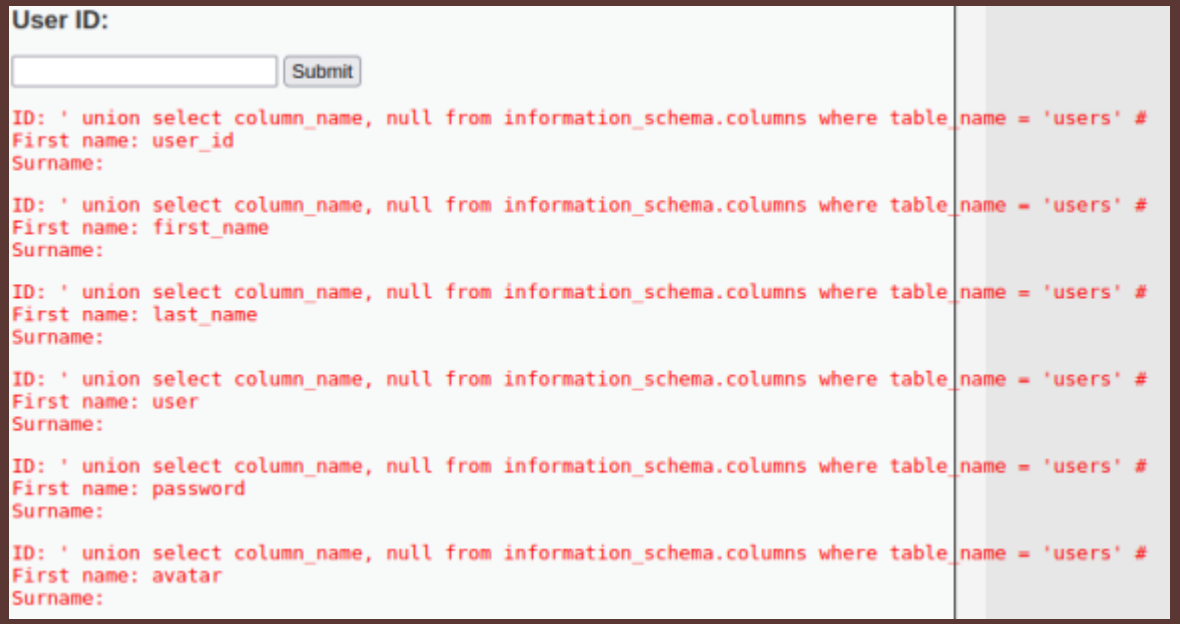
Explotar una inyección SQL y evadir los filtros de Web for Pentester

' union select distinct(table_name), null from information_schema.tables #



Explotar una inyección SQL mediante la herramienta SQLmap

: ' union select column_name, null from information_schema.columns where table_name = 'users' #



Identificar una máquina vulnerable en Vulnhub e intentar resolverla

'union select concat(user, 0x3a, password), null from users #