SQLi Bypasseando el firewall y pasando el examen!

Bueno amigos este tutorial lo he creado para mi amigo **Mr.Pack** para su comunidad http://r00tc0d3rs.org/ y para mi blog http://www.arthusu.blogspot.mx/ cualquier comentario es bienvenido...

Pequeño relato

Bueno empecemos con un pequeño relato xDDD :P
Estaba navegando por mi twitter y me encuentro con una publicacion de **Roberto Salgado:**

Donde viene un **reto SQLi** el cual es el siguiente:
http://ex.wargame.vn:8080/study/

Ahi elegi el reto en modo estudiante, pero lo mejor es que venia el codigo en PHP para poder entender todo el tutorial que llevare a cabo :3

Bueno el **codigo** era el siguiente:

```
<h1>List of students who passed the sql injection exam</h1>
<?php
require_once("conn.php");

$id = $_GET['student'];

if (intval($id)==0)
    $q="SELECT * FROM score";
else {
    $id = mysql_real_escape_string($id);
    if (preg_match("/ord|ascii|char|0x|like|table|column|sleep|ben
ch|[\+\-\*\/><=#]/i",$id))
        die("Attack detected");
    $q = "SELECT * FROM score WHERE id=$id";
}

$q.=" having score.point>5";
$result = mysql_query($q);

if (!$result)
    die("<h4>error in query</h4> $q");

echo "Count: ",mysql_num_rows($result);
```

```
echo "<table border=1><tr><th>student_id</th><th>Point (/10)</th><
/tr>";
while ($o=mysql_fetch_object($result)) {
    // yeah normalized to 10 point system...
    $o->point = intval($o->point)>10?10:intval($o->point);
    $o->id = intval($o->id);

    echo "<tr>";
    if ($o->id<=30)
        echo "<td><a href=?student={$o->id}>".$o->id."</a></td>";
    else
        echo "<td>error id</td>"; // I don't have that many studen
t
    echo "<td>".$o->point."</td>";
    echo "</tr>";
}
echo "</table>";
?>
```

Tal codigo se puede conseguir desde: http://ex.wargame.vn:8080/study/student.php.htm
la misma pagina del reto....

Explicando lo que hace el codigo

Y bueno el reto consiste en **pasar un examen de SQLi el cual tienes que bypassear, viendo el codigo podemos hacerlo de manera facil** :D, bueno entonces explico lo que hace el codigo es lo siguiente:

Primero incluye un archivo que tiene las credenciales de la conexión, luego tenemos **la variable que es vulnerable que se llama $id** que contiene un get hacia **student**, luego hacemos una consulta donde si **id=0 o nada nos devolveria todos los datos en una tabla.**.. pero en caso de que el id sea correcto entonces le pasa la funcion **mysql_real_escape_string()** que escapa los siguientes caracteres:

\x00 ,\n ,\r ,\ ,' ," ,\x1a

esto **no quiere decir que no sea vulnerable a una inyeccion sql ya que para ello tambien deberiamos hacer uso de la funcion stripslashes() según lo indica PHP en su documentacion**. Despues viene lo interesante que seria las e**xpresiones regulares y el WAF(Web application firewall)** en si... en este caso lo que hace es detectarnos si en la variable $id que es el Student si se encuentra alguna de estas palabras:

ord, ascii,char,0x,like,table,column,sleep,bench,+,-,*,/,>,<,=,#

si se encuentra cualquiera de estas ordenes en la url sean **minusculas o mayusculas** terminaria el script y nos mandaria un mensaje diciendo:
Attack Dettected (Ataque detectado)

Pero si no es asi entonces selecciona todos los datos de la tabla score donde id=$id luego le agrega a la consulta que nos muestre solo los valores que tengan **un valor mayor a 5 de calificacion o**

puntuacion :)

Despues hace la peticion y la guarda en la variable result, si no se completo la consulta entonces muestra un mensaje diciendo:
Error in query y aun lado la consulta....

Pero si se completa la consulta cuenta el numero de filas y lo muestra, luego en una tabla muestra el id del estudiante y la puntuacion o calificacion, aquí viene algo importante al mostrar los valores **solo acepta valores enteros, es decir, convierte cualquier cadena a 0 y cualquier decimal a entero.**... entonces solo muestra valores enteros.... y el id no puede **ser un valor mayor a 30 sino mostraria un mensaje diciendo: error id**

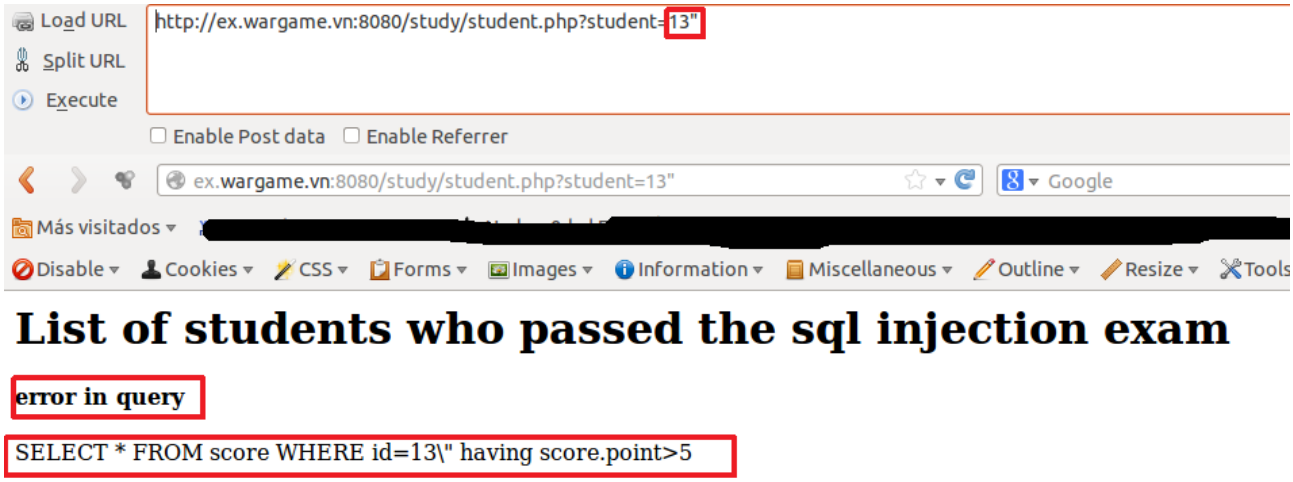
Siendo explicado el codigo **empezaremos nuestra inyeccion :3**

Inyectando!

Bueno ahora que sabemos como funciona el codigo que es lo que filtra podemos intentar nuestra SQLi, primero que nada elegi el alumno con mas baja calificacion que era un 6 con un id=13, entonces comenzare rompiendo la consulta.... con algo como:
“
con lo que se agregaria algo como esto en la consulta:

```
$q = "SELECT * FROM score WHERE id=$id”";
```

y siendo que esta utilizando **mysql_real_escape_string() no intentara escapar** \” asi.... causando un error en la base de datos:

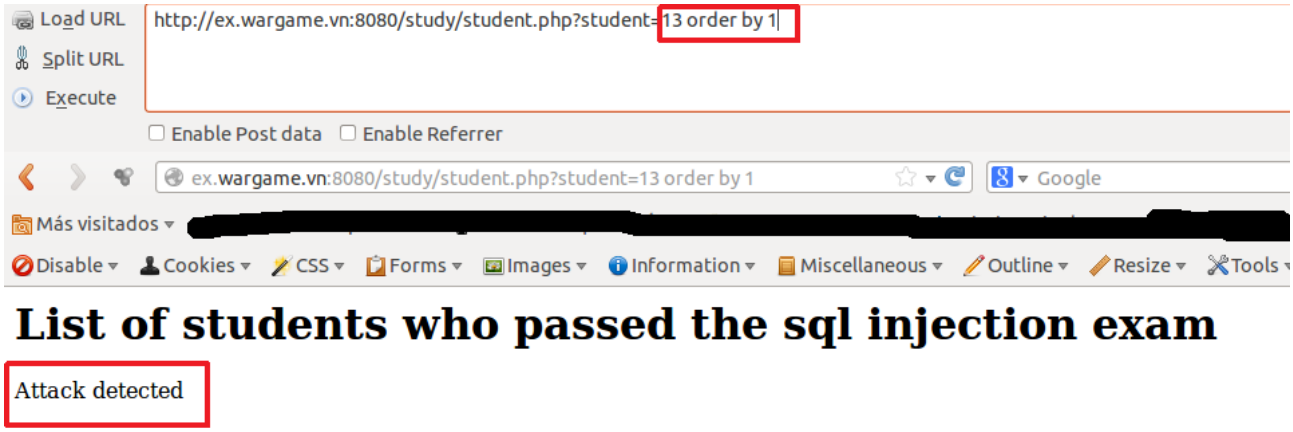


Ahi tenemos Error in query luego la consulta.... en este caso puedes ver claramente como es probado el error entonces lo siguiente seria **buscar el numero de columnas**, no sin antes decir que esta SQLi es del tipo integer....

Sacando Columnas

Aqui esta una de las cosas que mas me gustaria que entendieran, es que para poder ejecutar una

consulta en estos casos es **EVITAR EL FIREWALL** es decir me refiero si nos filtra **ord** entonces no podriamos hacer **order by**.... por que **nos detectaria ord..... y nos tiraria Attack Detected**

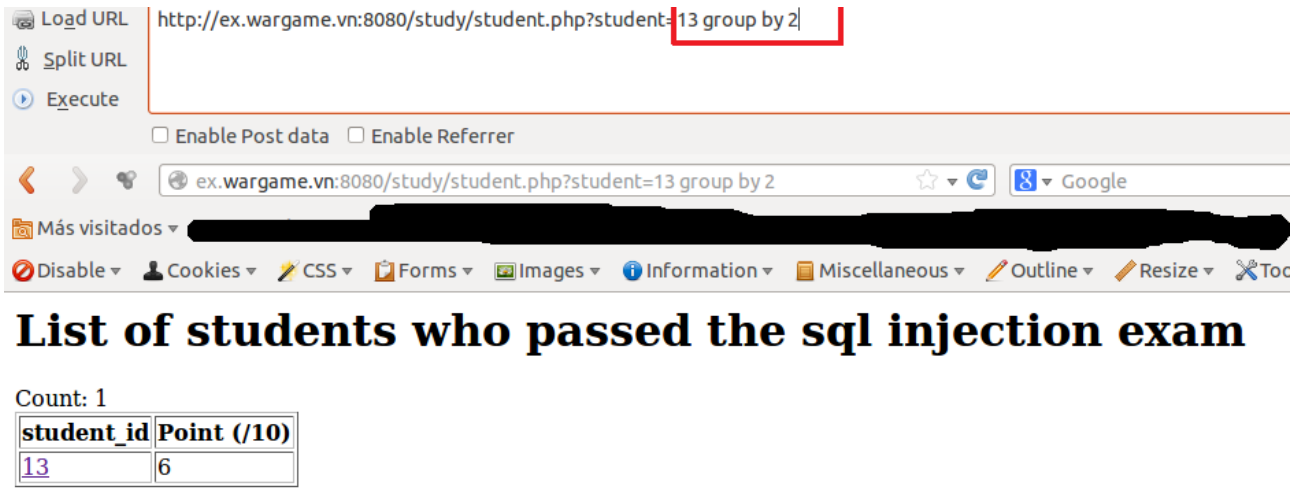


Entonces aquí evitaremos esto haciendo group by …
es decir contaremos las columnas por medio de group by … el cual no detecta el firewall por que no lo filtra... :D

entonces tenemos:

student=13 group by 1 (no error)
student=13 group by 2 (no error)
student=13 group by 3 (error)

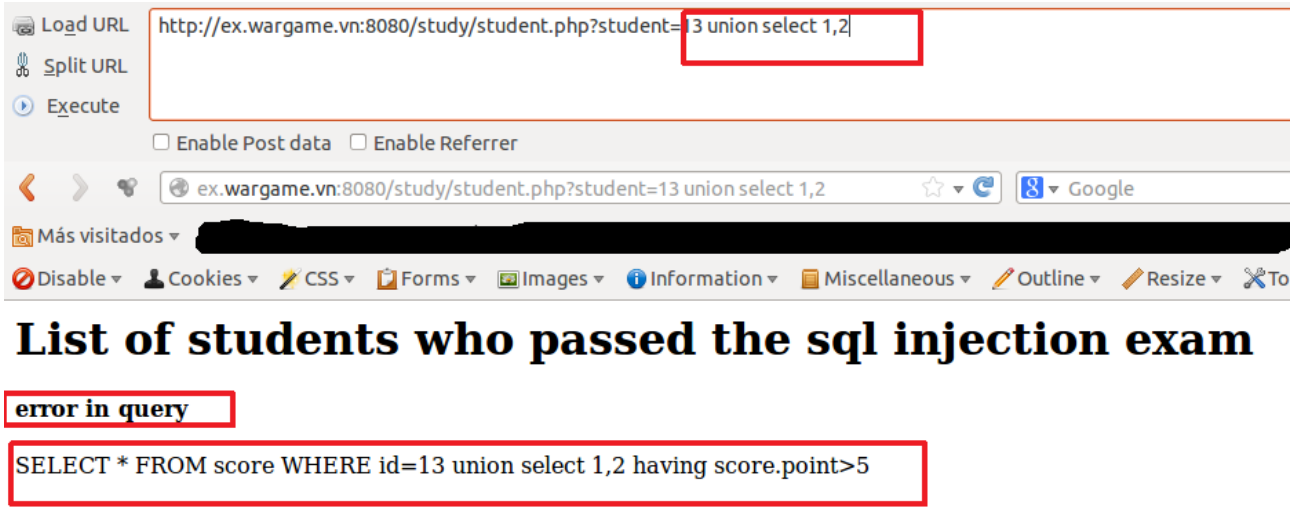
Con esto deducimos que se estan usando dos columnas que en el codigo podemos ver que son: **point e id....**



Mostrar campos vulnerables

Lo siguiente seria mostrar los campos vulnerables, para esto tenemos que hacer una consulta nueva.... para ello usamos **UNION** entonces hacemos un union select 1,2

con esto seguimos las reglas de union y seleccionamos los mismos campos que la primera consulta que seria donde se selecciona la calificacion de student....



Aqui pasa algo muy interesante, esto es que como nuestra consulta no es valida por culpa de having score.point>5 entonces nos lanza el error

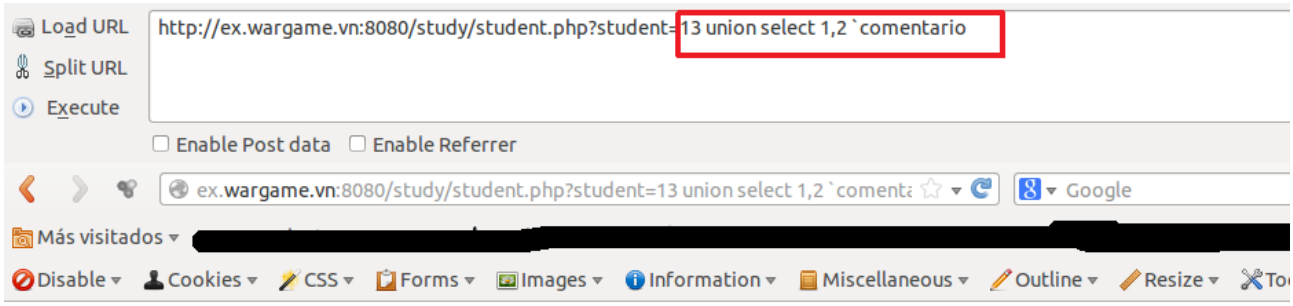
Lo que tenemos que hacer es **comentar eso.... es decir comentar having score.point>5.... para ello tenemos que usar algun comentario propio de la db que es mysql.... pero como el waf nos bloquea -- nos bloquea /* y tambien # no podemos usar ninguno de esos comentarios..... por que nos mandaria un Attack Dettected**

en ese caso usaremos el ` **(backtick)** que tambien es un comentario, con esto se mostraria la consulta correctamente....

La consulta en este caso estaria quedando de esta manera:

```
$q = "SELECT * FROM score WHERE id=$id union select 1,2
`comentario having score.point>5";
```

Lo cual es una consulta correcta.... y nos mostraria los campos vulnerables, incluyendo que ahora esta contando 2 filas y no solo 1 :)



List of students who passed the sql injection exam

Count: 2

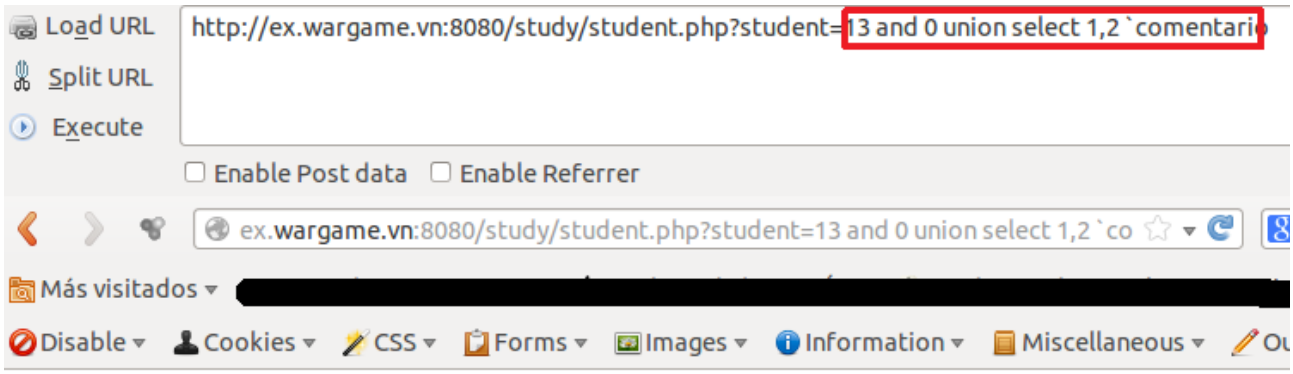
student_id	Point (/10)
13	6
1	2

Con esto ya podriamos inyectar en los campos 1 y 2, en este caso no queremos que se muestre el student_id 13 ni el point 6 que tenemos arriba con esto me refiero a que **no queremos que se muestre la primera consulta** :P
entonces en estos casos usariamos:

student=-13 union select 1,2 `comentario

pero como el **waf detecta el – entonces invalidamos la otra consulta de otra manera como por ejemplo usando and 0 cancelamos la primera consulta**:

student=13 and 0 union select 1,2 `comentario



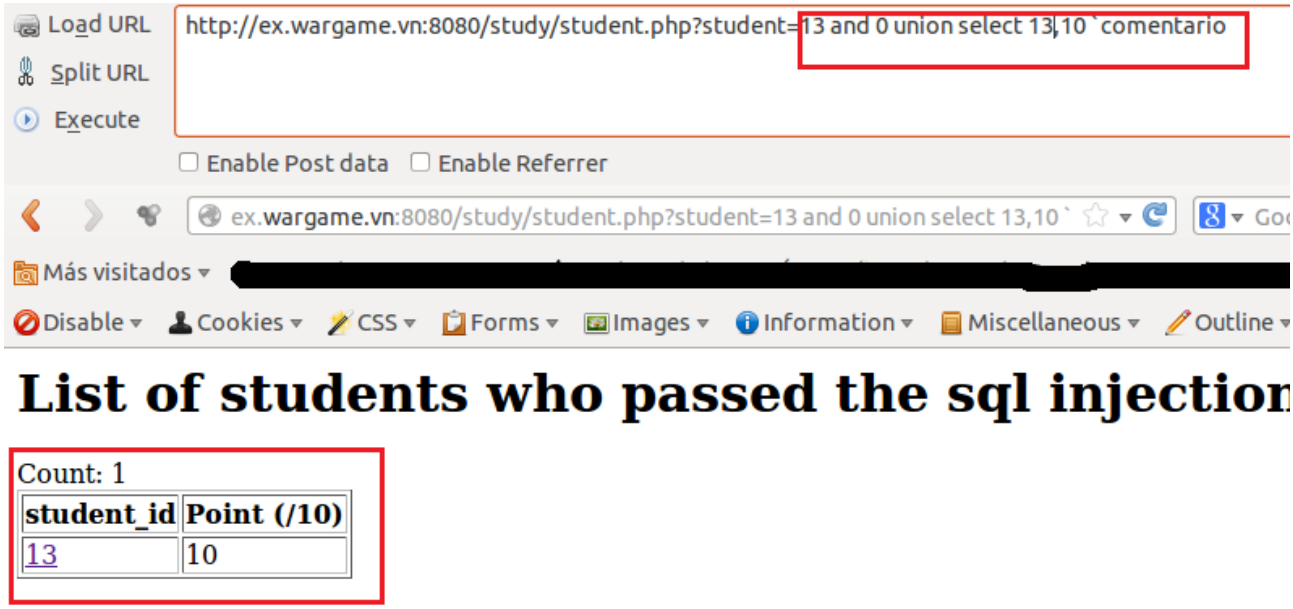
List of students who passed the sql injecti

Count: 1

student_id	Point (/10)
1	2

Entonces como lo siguiente es pasar el examen del student con id=13 solo ponemos en el campo 1 el numero 13 y en el campo 2 que seria la calificacion un 10 :)

Hemos pasado el Examen!!!



¿por que no se puede hacer mas? Simple, cualquier cadena la convierte a 0 y cualquier decimal lo convierte a entero, gracias a la funcion intval() :P

Entonces lo unico maligno que podriamos hacer en este caso es buscar la version con la funcion version() de mysql esto nos devolveria un entero con el valor 5....

Hasta aquí el tutorial
Autor: Arthusu
Dedicado a: http://r00tc0d3rs.org/ and Mr.Pack