1. **SQL注入**

\* 注入作为owasp top 10 的首位，其常见程度非同一般，而sql注入作为其中的典型代表，作为CTFer必须要了解一下

\* sql注入的本质就是用户输入的数据被当作sql代码执行，通常在url的传参中、登录框中、head请求头中等地方可以发现注入点

先介绍下面介绍基础时的环境：

数据库名：bilala

数据表有两个：i\_am\_test 和 nisa

第一个表i\_am\_test中有三个字段：id、username、password

共有三条记录

第二个表nisa中有三个字段：id、nisa\_member、flag

共有五条记录

**先介绍一下select语句的小基础：**

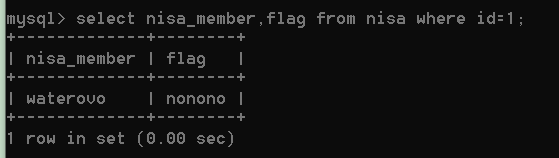
Select 字段名 from 表名 where 条件

比如：select flag from nisa where id=1



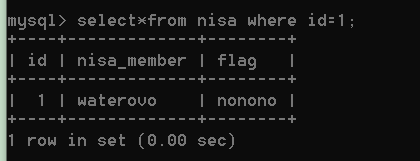
再比如：select nisa\_member,flag from nisa where id=1

*这里要说，字段名要以逗号隔开，where条件也可以没有*

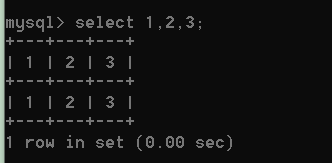


*你select了什么字段就回显什么字段内容，\*号表示所有字段*

比如：select\*from nisa where id=1 将会回显一行所有数据

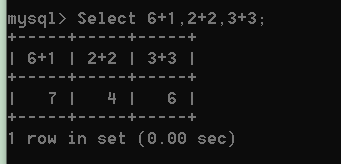


当select语句后面没有from时，比如select 1,2,3

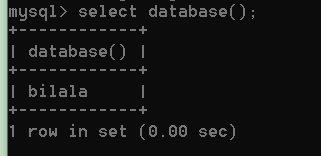


第一行的1 2 3 表示三个字段，第二行是他们的值，比如我再举个例子

Select 6+1,2+2,3+3



*最基础的函数：database() //返回当前所在的库名*



1. **SQL注入考点**

* 万能密码

万能密码很简单啊，原sql查询语句若为select\*from i\_am\_test where id= ’ $\_GET[id] ’

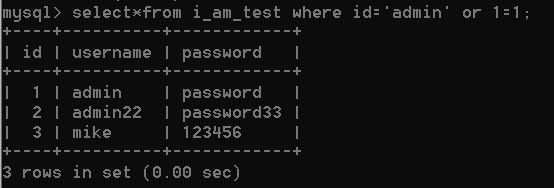
此时我们传入万能密码比如：admin’ or 1=1 --+

拼接进查询语句：select\*from i\_am\_test where id= ’admin’ or 1=1 --+’

*--+(念作杠杠加)、-- (念作杠杠空格)、#(jin号)这三个为sql语句中的注释，其中#只在mysql中有，我个人习惯-- qwe（杠杠空格qwe，因为单单杠杠空格有时候不会触发注释效果）*

拼接后第一个单引号闭合前面的单引号，or后面的1=1恒为真，后面再加上注释注释掉后面的单引号，所以执行select\*from i\_am\_test即可查出所有数据

**- - > 实操**

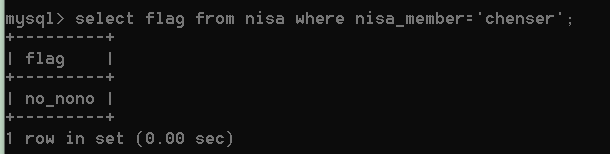


* 联合注入

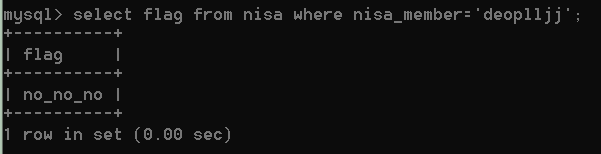
1.

利用union函数将两个sql语句一起执行并且回显

首先单独查看select flag from nisa where nisa\_member='chenser';

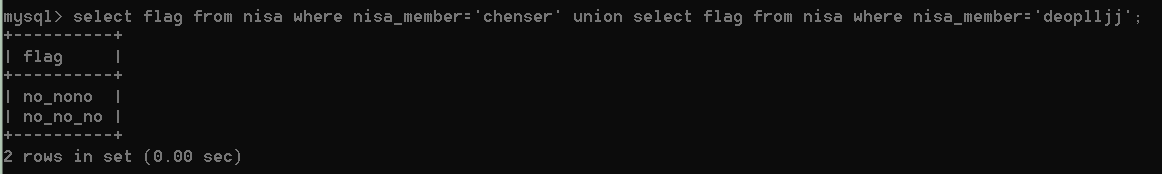


再单独查看select flag from nisa where nisa\_member='deoplljj';



再用union将两个语句拼接在一起查看

select flag from nisa where nisa\_member='chenser' union select flag from nisa where nisa\_member='deoplljj';



可以发现我们的两条语句都有回显

2.

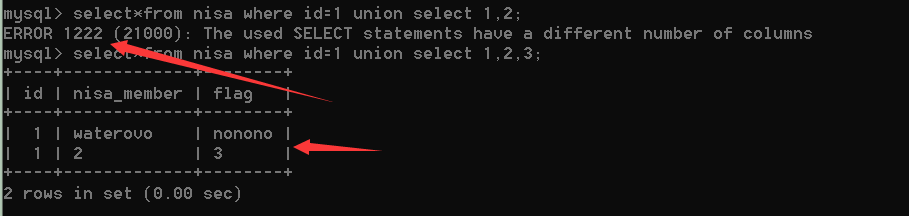
那根据万能密码的构造，我们可以联想出联合语句的构造

传：1’ union select 1,2,3 -- qwe

同样的，单引号为了闭合sql语句中的前面那个单引号

*这里要说的是union前后的select的字段数必须保持一致，否则报错*

比如：select\*from nisa where id=1 union select 1,2 将会报错，而1,2,3则不会



-- qwe 起注释作用

3.

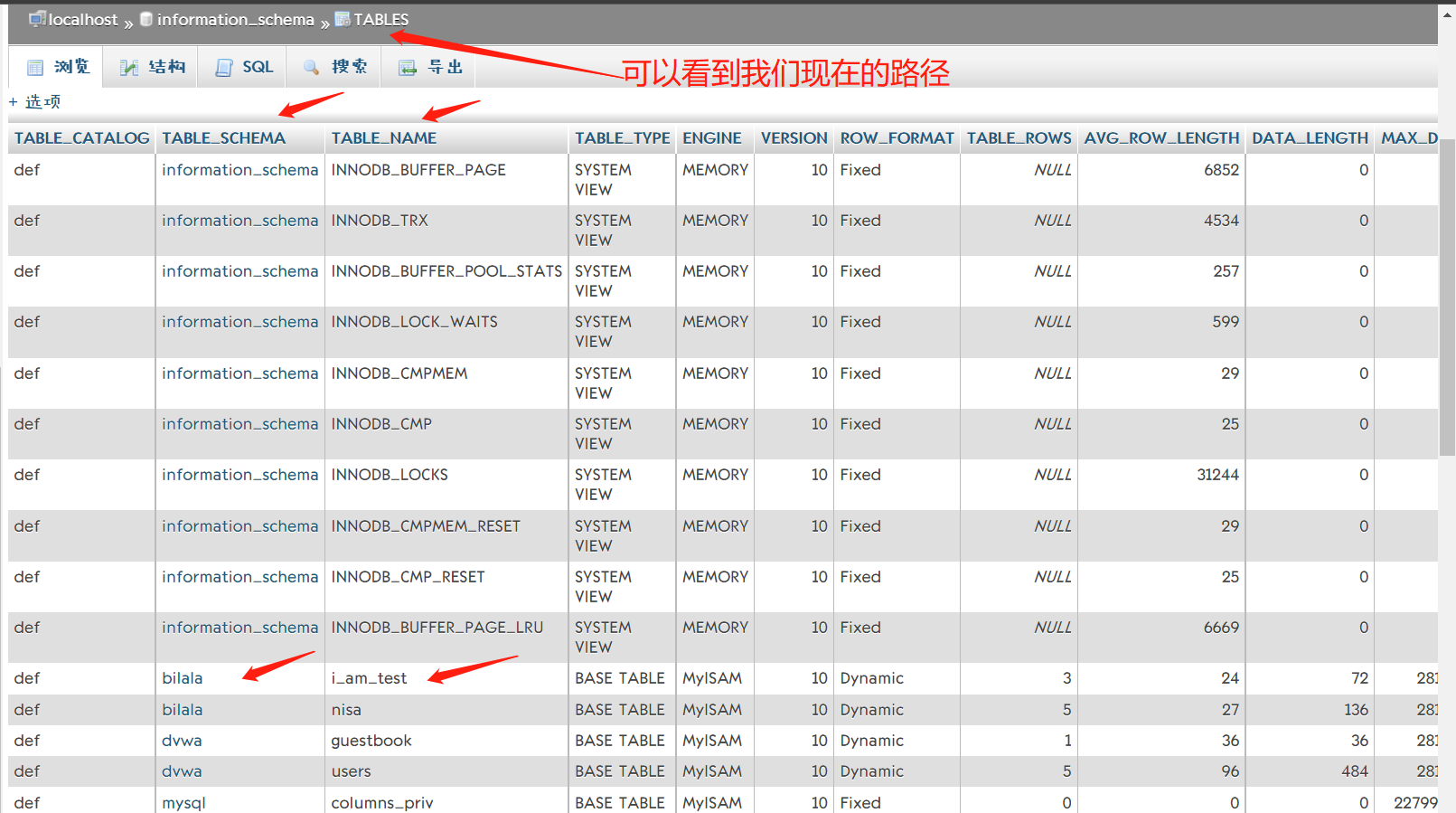
知道了基础，我们该怎么利用union来获取flag所在的库名、表名、字段名呢

这里要介绍mysql的information\_schema数据库，这个数据库存有其他包括他自己在内的所有数据库的所有信息。

在information\_schema.tables这个表中，存有所有数据库中的所有表，并且有表对应的库

在information\_schema.columns这个表中，存有所有数据库的所有表中的所有字段名，并且有其对应关系

接下来我在phpmyadmin中展示一下这个库



里面的table\_schema和table\_name都是对应的

所以对于select table\_name from information.tables where table\_schema=’nisa’ limit 0,1这个语句，大家应该能看懂了（limit 0,1就是输出从第0+1行数据开始的连续1行，也就是输出第一行数据；如果是limit 1,1就是输出第二行；limit 0,2 就是输出一二两行）

如果嫌一行一行输出麻烦，可以使用group\_concat()函数，这个就是将结果以逗号形式隔开一起输出，所以没有limit的语句为select group\_concat(table\_name) from information.tables where table\_schema=’nisa’

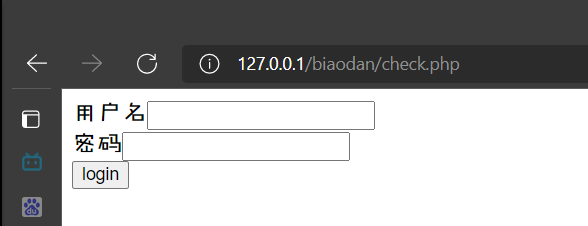
4.

Ok,到了这里，我做了一个不算题目的小题目，专门给大家实操解释

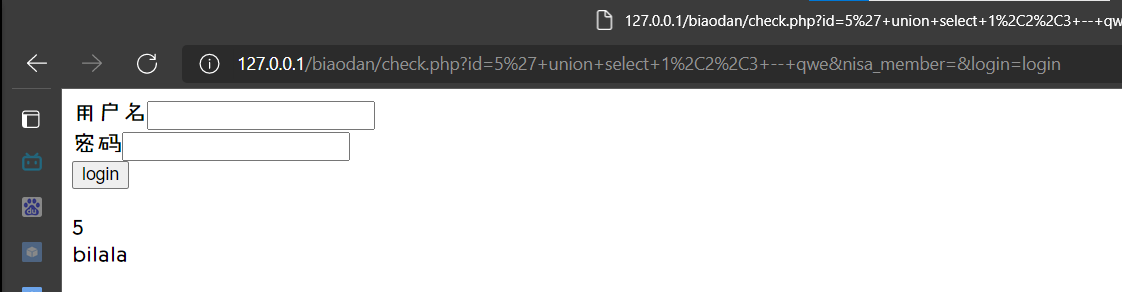
查询语句为：select\*fromselect\*from nisa where id='$id' and nisa\_member='$nisa\_member'

上面框为id的传参，下面为nisa\_member的传参

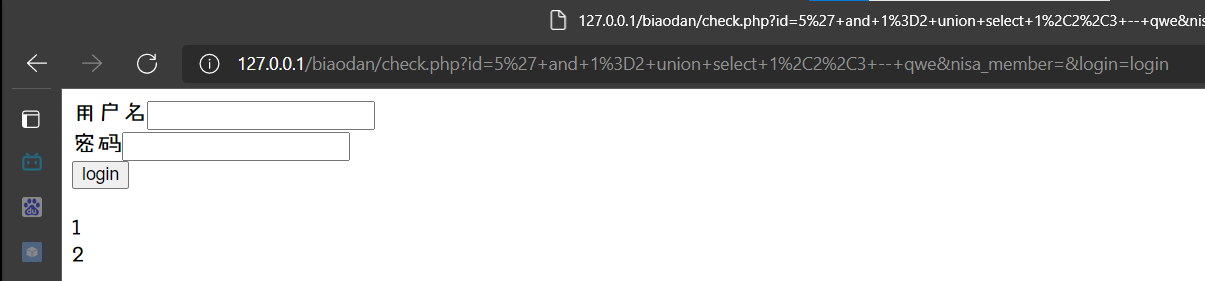
题目首页很简单，就是普通的登录框



我们传参：5’ union select 1,2,3 -- qwe

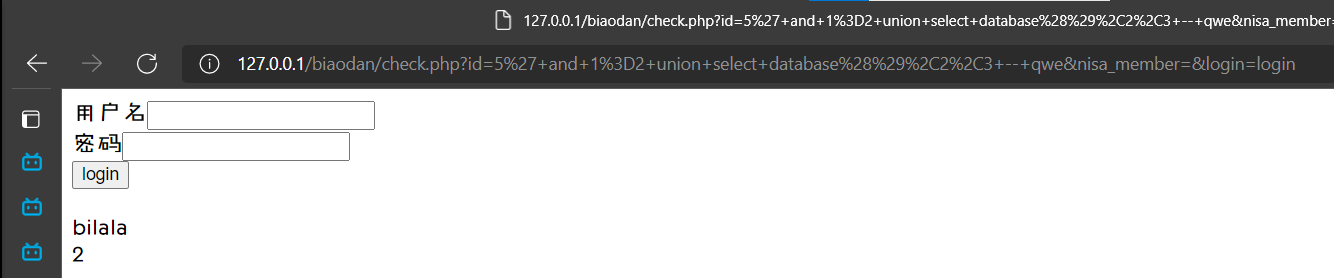


可以看到我们得到的回显是union前面那句的回显（当union前面的条件真值为假时 此时select语句的结果将完全由union select控制），所以我们需要将前面的语句弄成假，传参：5’ and 1=2 union select 1,2,3 -- qwe（或者: -1’ union select 1,2,3 -- qwe）



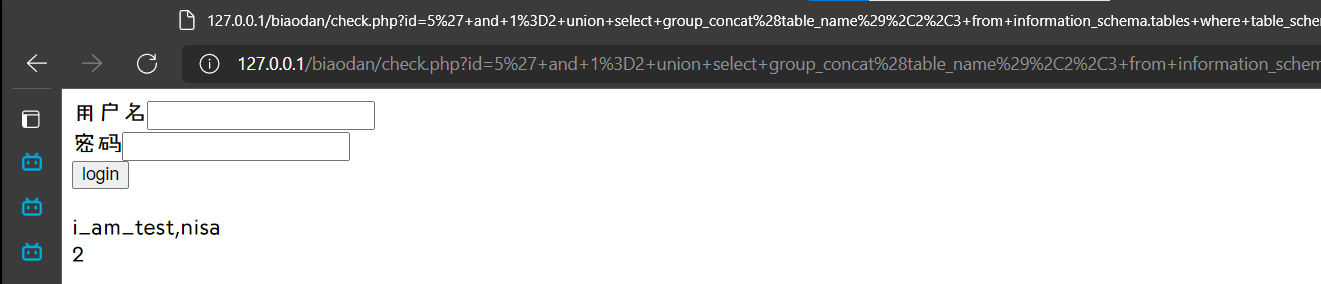
到这里，我们可以知道我们的回显点是1和2，我们随便选一个用

比如传参：5’ and 1=2 union select database(),2,3 -- qwe



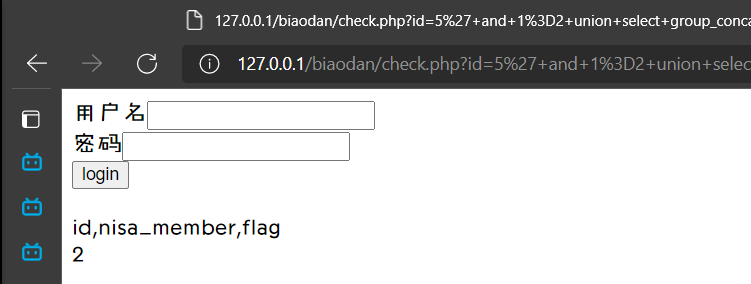
可以看到返回了当前所在的数据库库名

利用information数据库：5’ and 1=2 union select group\_concat(table\_name),2,3 from information\_schema.tables where table\_schema=database() -- qwe

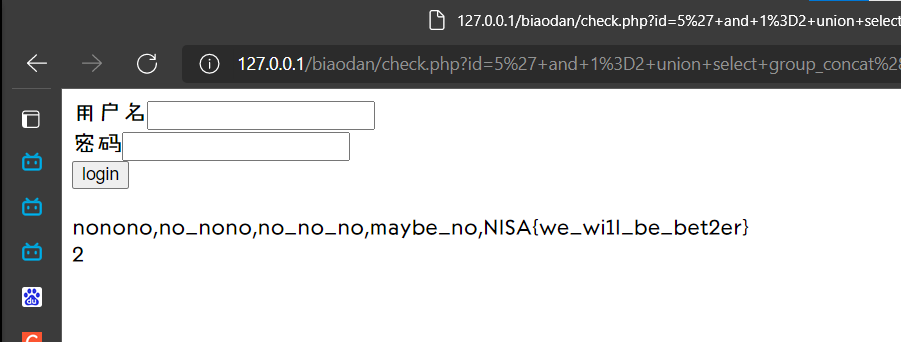


得到数据库中的所有表名

继续爆列名：5’ and 1=2 union select group\_concat(column\_name),2,3 from information\_schema.columns where table\_name=’nisa’ -- qwe



获取字段值：5’ and 1=2 union select group\_concat(flag),2,3 from nisa -- qwe



5.

由上题，可以发现我们有两个疑问点：①我怎么知道查询语句的字段数是多少（也就是union前面那句查了多少字段，因为实际做题中我们是看不到查询语句的）②我怎么知道是用单引号闭合前面还是用双引号闭合，还是不用闭合呢

对于①：很简单，select 1 select 1,2 select 1,2,3 一个一个试，或者去了解一下order by函数（也方便不了多少，也是一个一个试的）

对于②：关于用单引号还是双引号，试就好了；关于是否需要闭合，请看↓

当你传入id=1和id=2-1的页面一致时，页面大概率就是数字型，也就是不用闭合，因为此时的查询语句的条件为：where id=1 和 where id=2-1

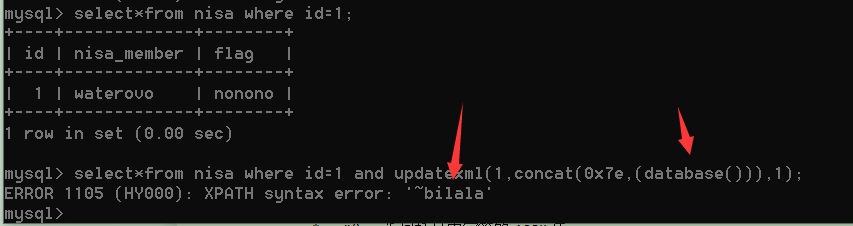
当传入id=2和id=2-1页面一致时，大概率是文字型，也就是需要闭合，因为此时查询语句的条件为：where id=’2’ 和 where id=’2-1’ ，很明显，2-1作为字符串时，是被当作2执行的

* 报错注入

（1）updatexml

从mysql5.1.5开始提供两个[XML查询和修改的函数](https://dev.mysql.com/doc/refman/5.7/en/xml-functions.html" \t "https://xz.aliyun.com/t/_blank)，extractvalue和updatexml。extractvalue负责在xml文档中按照xpath语法查询节点内容，updatexml则负责修改查询到的内容。

它们的第二个参数都要求是符合xpath语法的字符串，如果不满足要求，则会报错，并且将查询结果放在报错信息里：



我们只要把database()换成其他我们想要查询的语句，就可以进一步完成注入

格式：updatexml（1，concat（0x7e，（查询的内容）），1）

（2）extractvalue

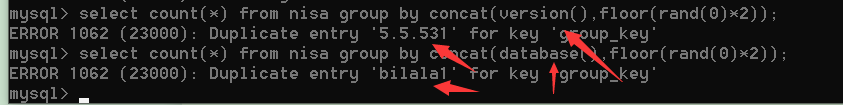
这个与updatexml一样，只不过后面的参数少了一个



格式：extractvalue（1，concat（0x7e，（查询内容）））

（3）主键重复

这里利用到了count()和group by在遇到rand()产生的重复值时报错的思路。



构造payload：select count(\*) from test group by concat(version(),floor(rand(0)\*2));

可以将version()换成其他想查询的内容

（4）join（列名重复）

这个放在无列名注入里边讲

* 布尔盲注

布尔盲注这里需要先了解几个函数

\* length() ：返回括号内字符串的长度

\* substr() ：截取字符串的一部分

\* ascii() ：返回括号内字符的ASCII值

有些题目对于报错没有回显，然后你利用union也发现不管输入啥都只有两种页面（可以执行语句返回的页面和不能执行返回的页面），这个时候就要想到布尔盲注了

布尔盲注的核心就是页面不会对你的注入有具体的内容，他只会告诉你语句有没有被正常执行。

比如利用length来判断当前所在数据库的库名长度

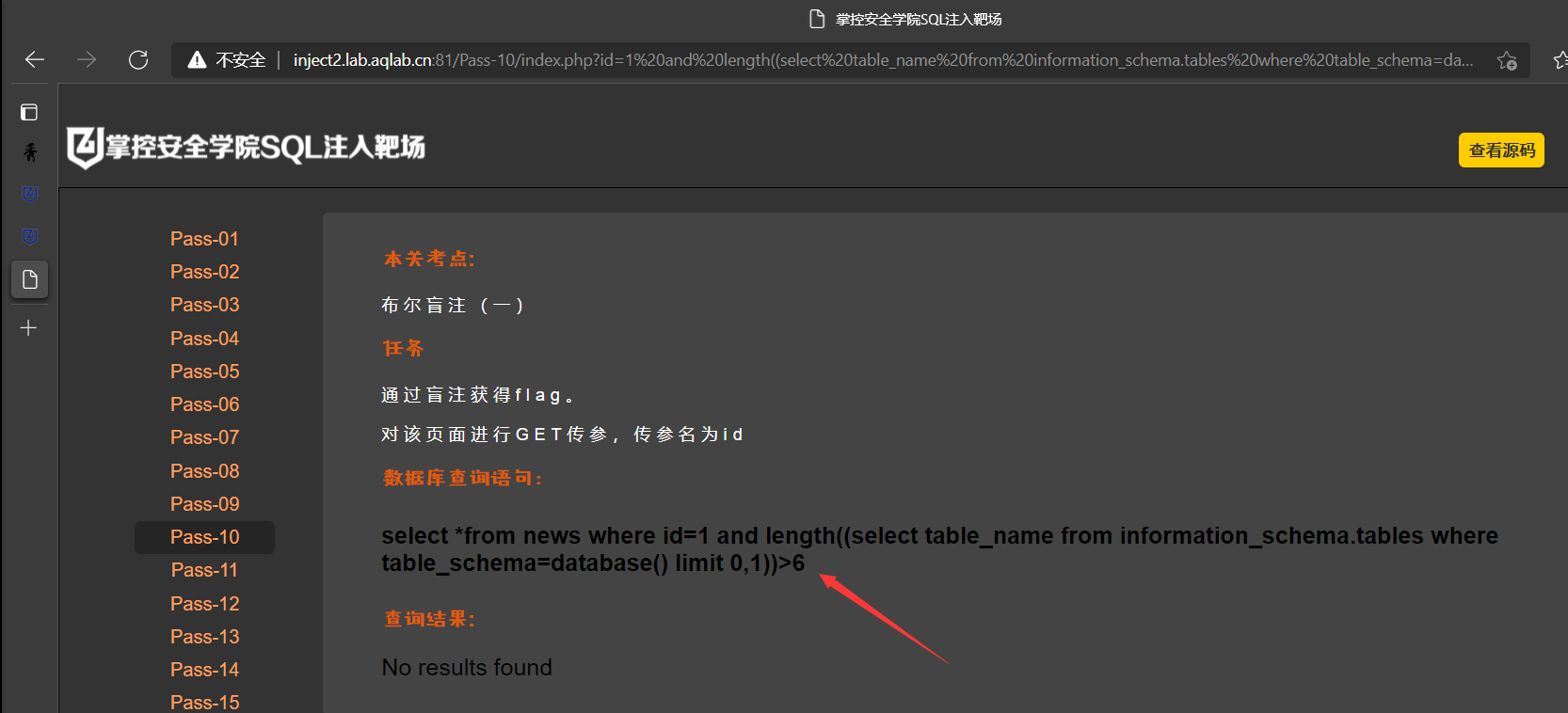




可以得知database()的长度大于11不大于12，也就是12位的长度

然后判断表名长度利用length((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1))





可以判断出第一个表名的长度为6

显然我们现在只能知道表名长度而已，要知道具体表名我们需要用到另外两个函数substr和ascii

Substr(‘abc’,1,1)：从字符串中第一个开始取，取一个也就是’a’

Substr(‘abc’,1,2)：=’ab’

Substr(‘abc’,2,1) = ’b’

Substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1,1) //这个就是得到表名的第一个字符

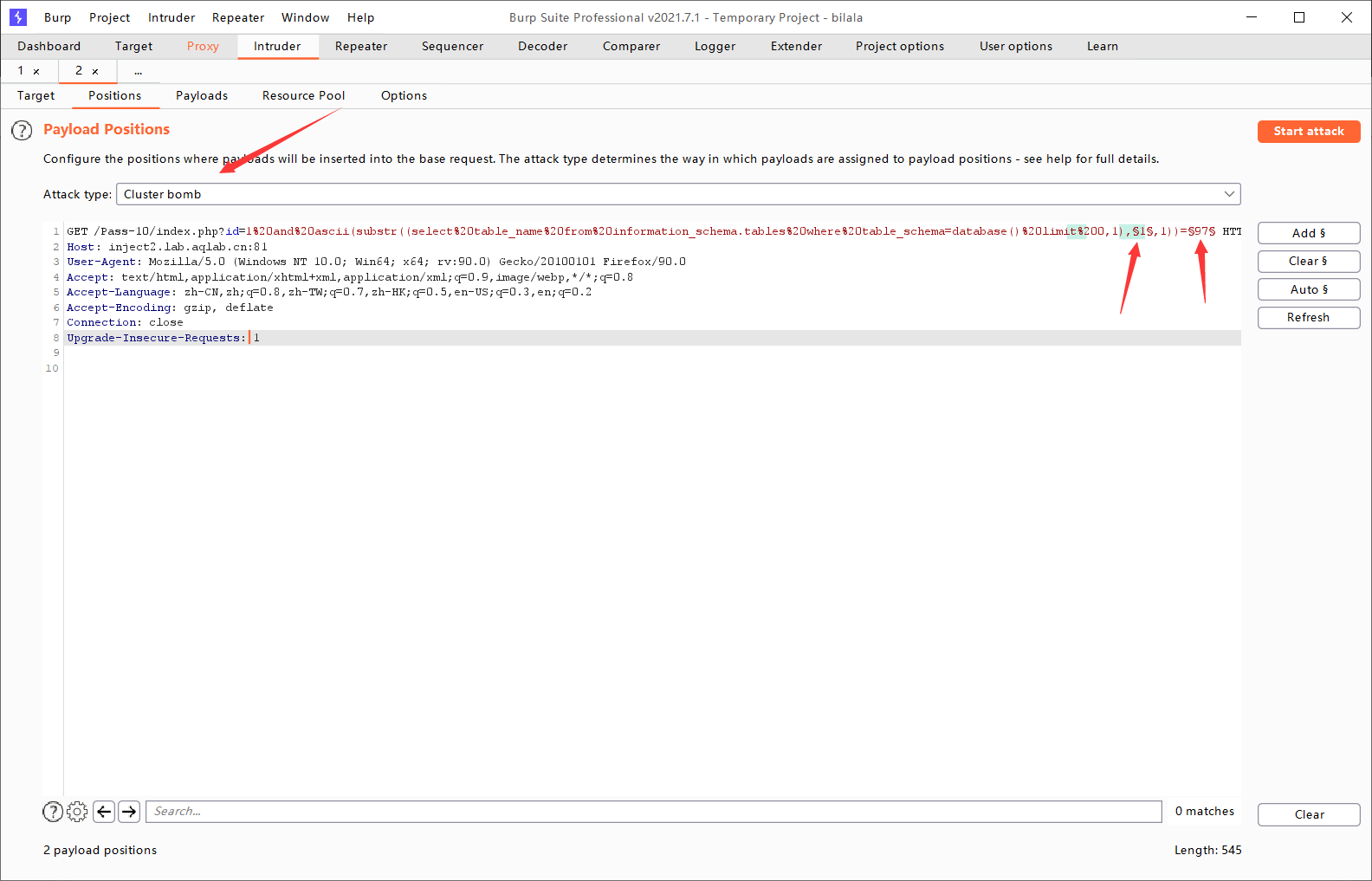
我们再将表名的第一个字符放到ascii函数中进行比较

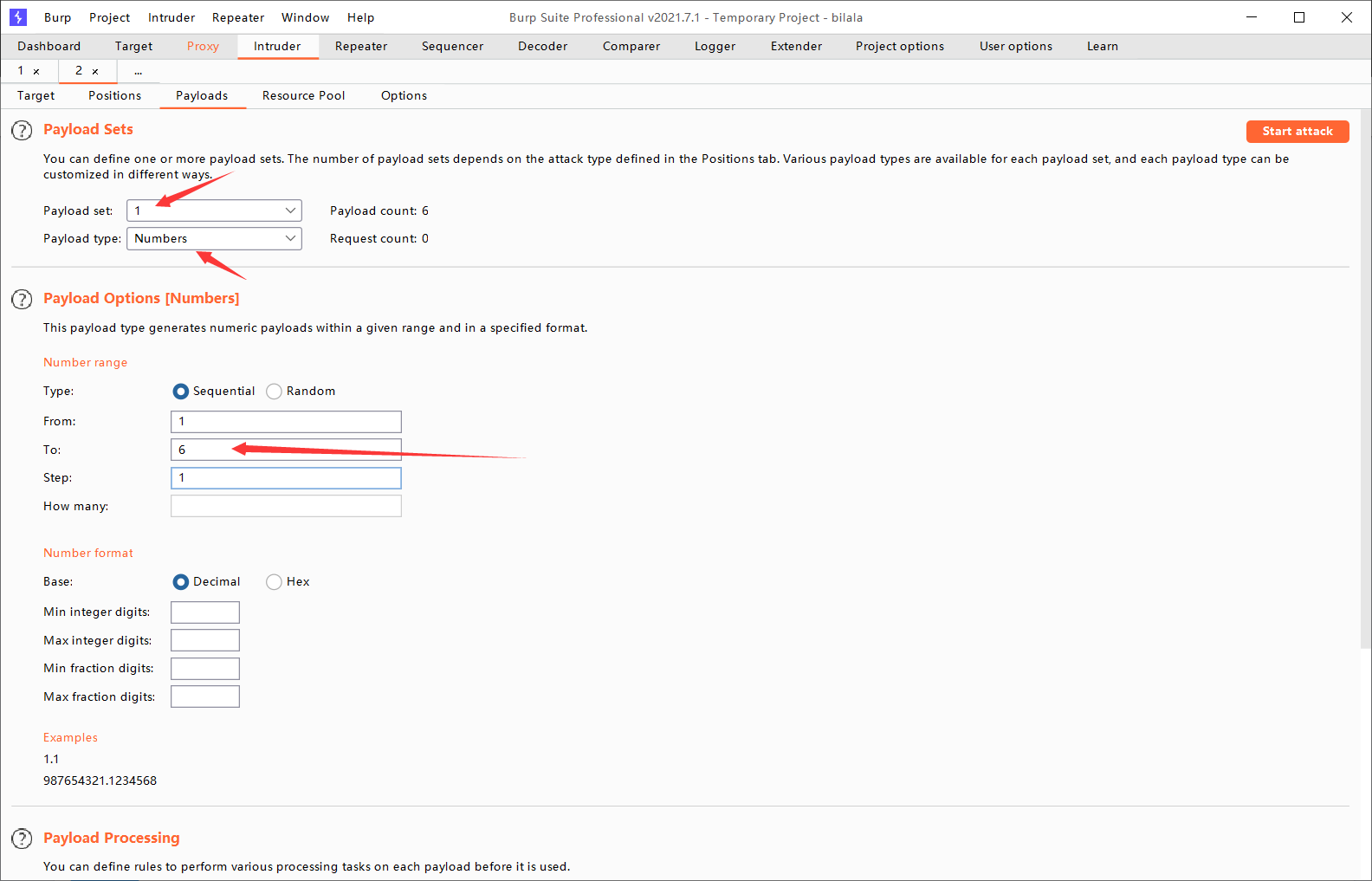
也就是ascii(Substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1,1))>97 //这个就是判断第一个字符的ASCII码是否大于97，也就是是否大于‘a’

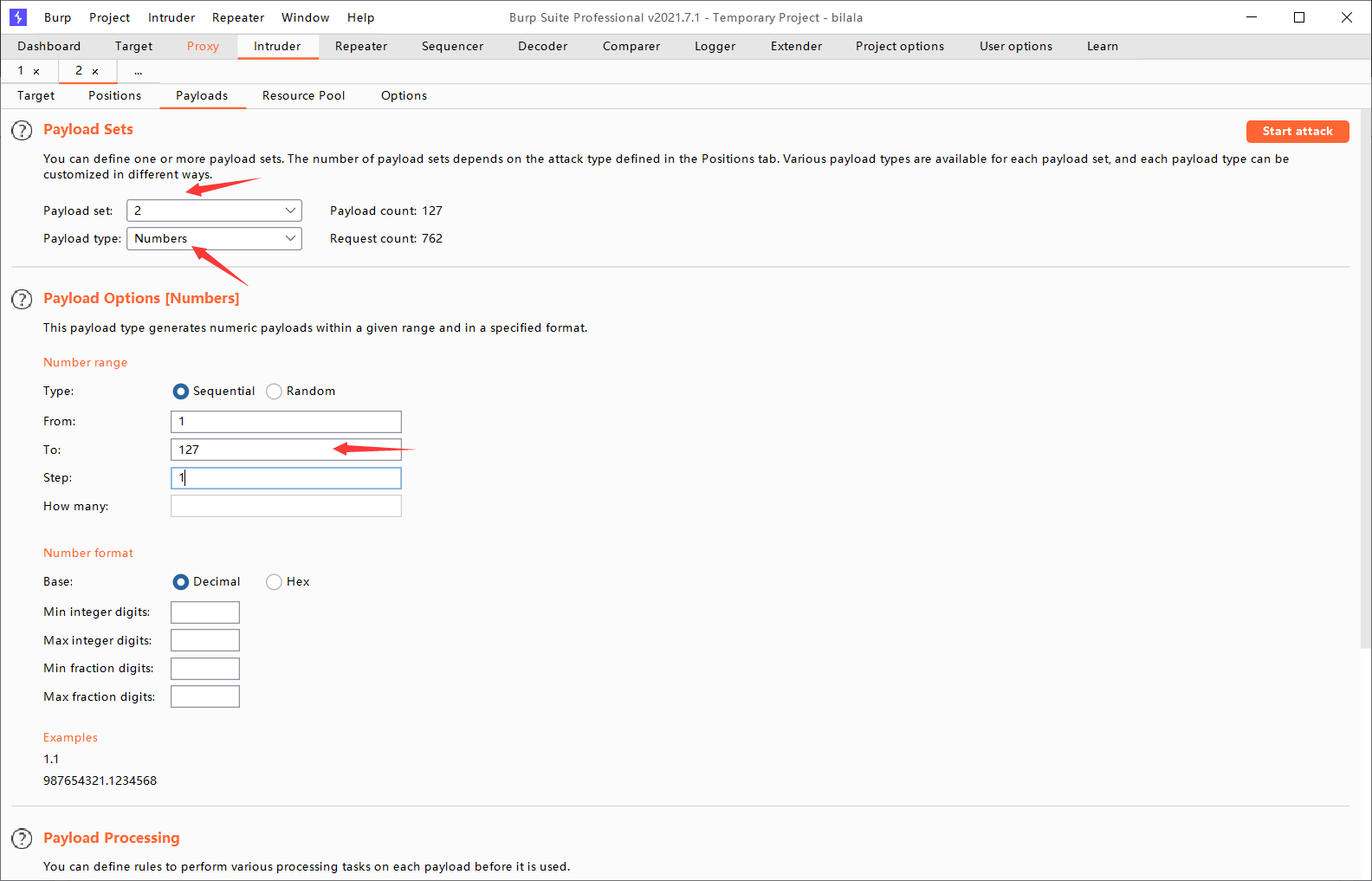
我们在手动盲注时，得到长度后，一般都要使用burp来帮助我们跑包

利用burp进行传参注入  
payload：?id=1 and ascii(substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1,1))=97

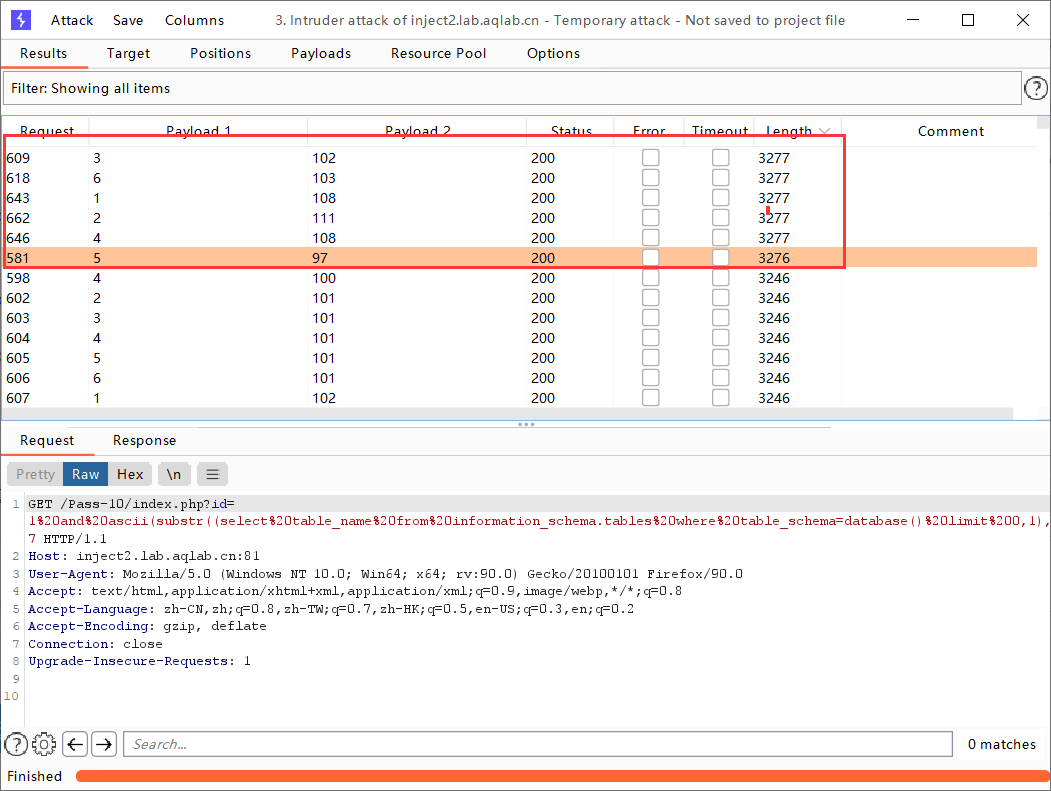
抓包放入intruder，对这两处进行跑包，并且将模式更改为cluster bomb



将第一个payload设置为↓：（因为表名长度为6）  


再将第二个payload设置为↓：（因为ASCII码有127个）  


开始跑包，找出这六个字符：



ASCII码分别为：108 111 102 108 97 103  
对应：loflag  
得到表名 loflag

接下来就是得到列名，列值，方法是一样的

当然sqlmap跑盲注也是常用的

* 时间盲注

首先了解函数：

\* sleep(number):延迟number秒

\* if(条件，成立执行，不成立执行)

比如if(length(database())>1,sleep(5),1) //如果数据库名长度大于1就延迟5s，否则1

延迟5s就是页面会先加载5s，也就是那个圈圈转5s↓



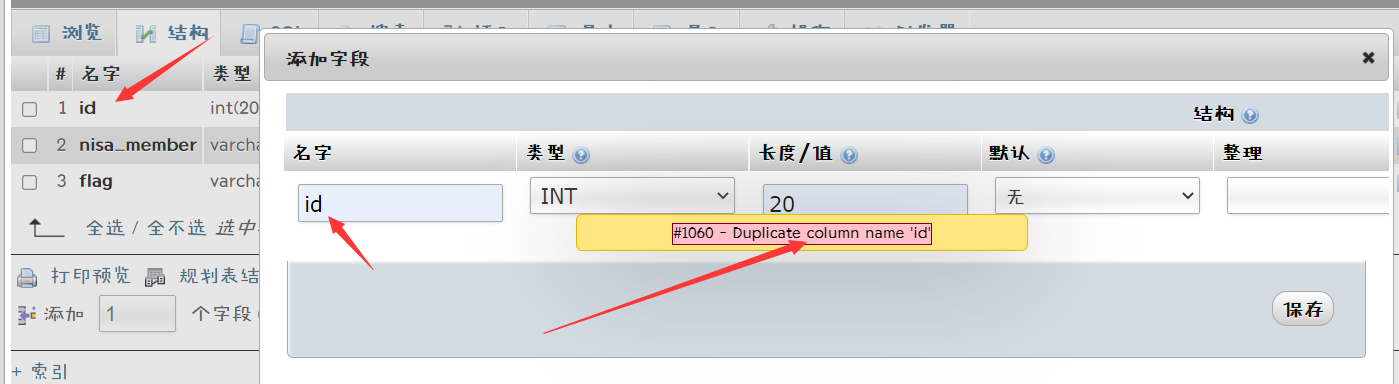
时间盲注的核心就是不管你传入什么数据，页面回显全都一样

知道了if和sleep函数后，其他操作就和布尔盲注一样了，只不过从页面回显判断语句是否正常执行变成了延迟判断

* 无列名注入

当题目中column被过滤时，我们要想到无列名注入（此时要有表名）

首先我在nisa表中已经存在了id字段的情况下再添加一个名为id的字段，出现如下错误

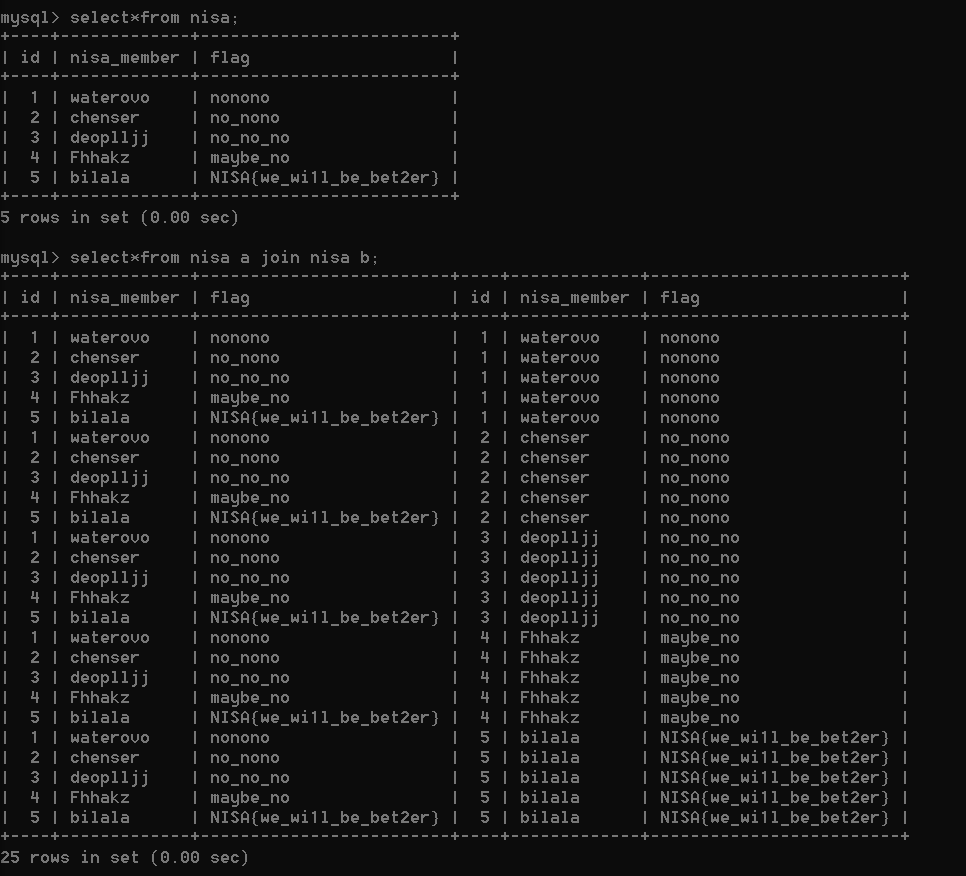


告诉我字段名重复

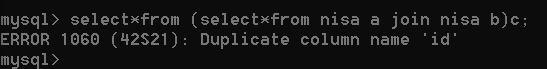
所以我们可以利用join来获得一个存在重复字段的新表，再去这个新表中查询就可以获得表中的列名，这就是无列名注入

演示：

我们先用join获得一个新表



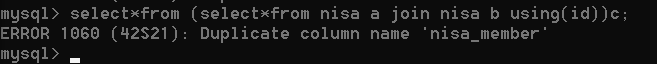
接下来如果我们在这个表中查询，就会出现列名重复的报错信息↓



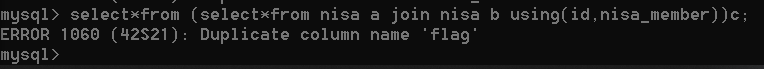
可以发现爆出了id这个字段

接下来再利用using函数

// using(id) 等价于 on a.id=b.id



爆出了第二个列名



爆出第三个

就这样我们在没有column的情况下获得了表中所有的列名

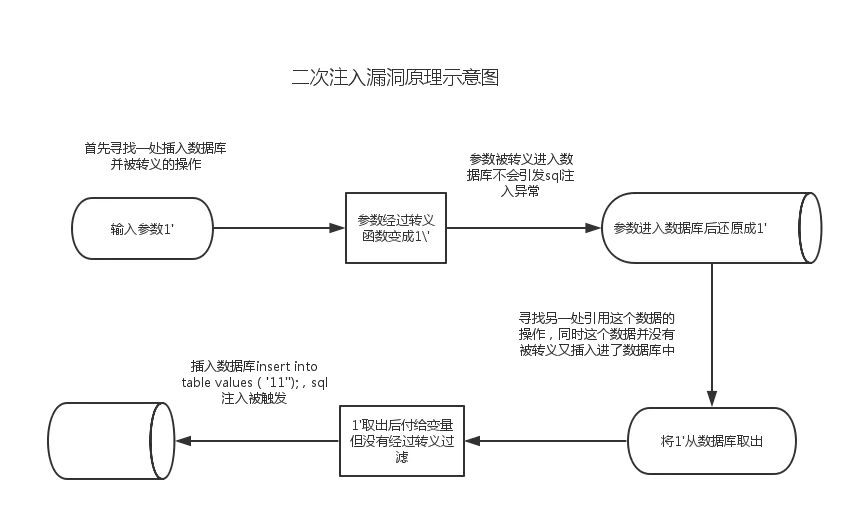
* 二次注入

二次注入可以理解为，攻击者构造的恶意数据存储在数据库后，恶意数据被读取并进入到SQL查询语句所导致的注入。防御者可能在用户输入恶意数据时对其中的特殊字符进行了转义处理，但在恶意数据插入到数据库时被处理的数据又被还原并存储在数据库中，当Web程序调用存储在数据库中的恶意数据并执行SQL查询时，就发生了SQL二次注入。

二次注入，可以概括为以下两步:

第一步：插入恶意数据  
进行数据库插入数据时，对其中的特殊字符进行了转义处理，在写入数据库的时候又保留了原来的数据。

第二步：引用恶意数据  
开发者默认存入数据库的数据都是安全的，在进行查询时，直接从数据库中取出恶意数据，没有进行进一步的检验的处理。

配合下图可以有更好的理解：  


配合sqli-labs/less24讲解

部分源码：

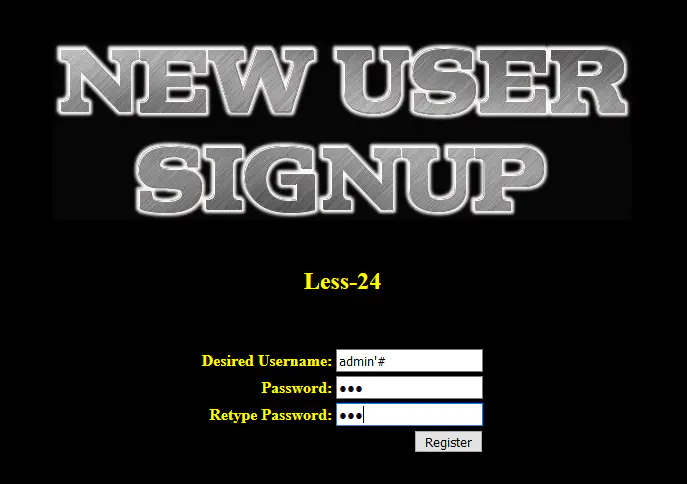
$username = mysql\_real\_escape\_string($\_POST["login\_user"]);

$password = mysql\_real\_escape\_string($\_POST["login\_password"]);

$sql = "SELECT \* FROM users WHERE username='$username' and password='$password'";

源码中调用了mysql\_real\_escape\_string，进行了转义，也就无法进行sql注入

有注册功能



这里就是二次注入中的第一次

然后我们利用这个admin’# 账户去修改密码

下面给出修改密码的源码

$sql = "UPDATE users SET PASSWORD='$pass' where username='$username' and password='$curr\_pass' ";

在用户名为 "admin'#" 时执行的实际是：

$sql = "UPDATE users SET PASSWORD='$pass' where username='admin'#' and password='$curr\_pass' ";

可以发现我们二次注入时，一次注入的admin’# 发挥了作用，也就是我们可以直接修改admin账户的密码。从而登录admin账户获取更多信息

* 堆叠注入

堆叠注入就是多个语句同时执行（分号隔开），

比如：select\*from nisa; show databases; //这两个语句会同时执行

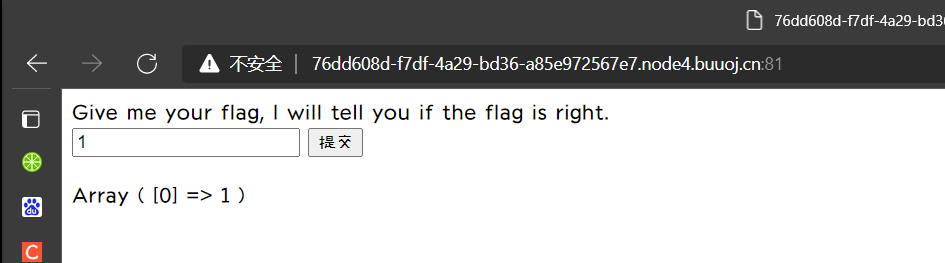
这种注入看似很厉害，只要加个分号，后面的语句我就可以对数据库进行任意的操作了

不过堆叠注入的前提是php调用了mysql\_multi\_query()函数，这个函数才是我们可以多个语句一起执行的关键，所以实际情况中，大多数都是用的mysqli\_ query()函数，这个函数只能执行一条sql语句，分号后就不再执行了。

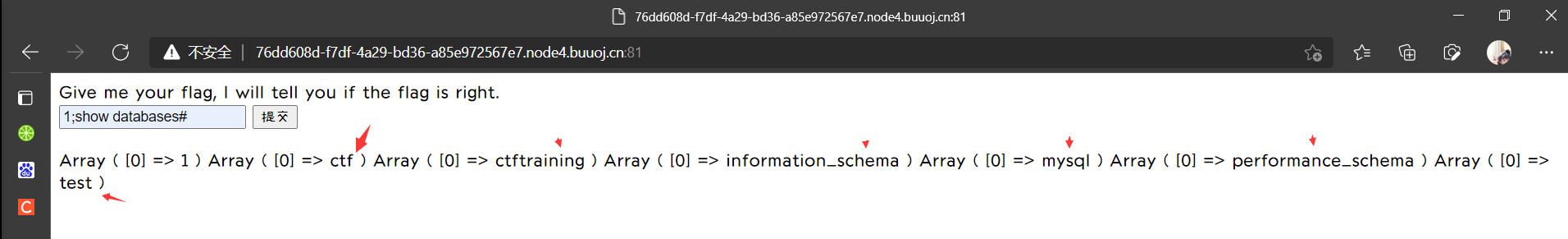
例如↓

[SUCTF 2019]EasySQL 1

传参1返回1：



这里使用堆叠注入传参：1;show databases#



可以发现执行了show databases语句，那我们就可以利用堆叠注入进行更深入的探测

* 宽字节注入

宽字节注入因为使用条件苛刻，并且很少碰到，所以放在最后讲

在php低版本中，有个配置叫做magic\_quotes\_gpc ，这个配置开启之后，你输入的参数如果带有特殊符号都将被带上转义字符“\”



也就是你无法闭合前面的单引号

在php高版本中，这个配置被移除了，改成了函数调用addslashes()



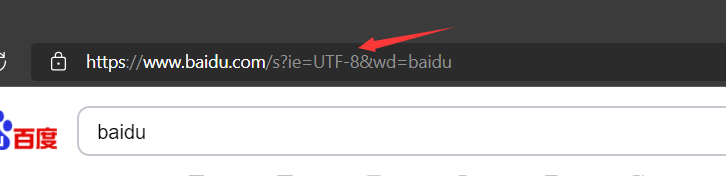
这样也达到了转义特殊字符的效果

众所周知，一个ASCII字符需要一个字节的存储，一个汉字需要两个字节

既然这个函数自带了“\”这个字符，那我们就在“\”前随便加个字符，使得加入后的字符与“\”字符构成一个汉字，从而达到单引号逃逸的目的

宽字节注入的苛刻条件就在于网站所用的数据库必须是GBK等汉字编码

那我们该如何找到能与“\”字符构成汉字的字符呢，“\”字符的url编码为%5C，在百度中这样操作，将编码改为gbk





可以发现%bb的这个url编码字符可以与“\”构成汉字（还有很多，自己试就行）

然后就可以传参：-1%bb’ union select 1,2,3 -- qwe



可以看到我们得到了想要的回显，注入成功

1. **参考文章**

报错注入： https://xz.aliyun.com/t/253

https://blog.csdn.net/silence1\_/article/details/90812612

盲注： https://www.secpulse.com/archives/153095.html

无列名注入：https://blog.csdn.net/weixin\_46330722/article/details/109605941

二次注入： https://www.jianshu.com/p/3fe7904683ac

堆叠注入： https://blog.csdn.net/qq\_45691294/article/details/107376284