soeasyheap writeup

1、分析

checksec下 保护是全开的，这倒是很正常的heap题

然后看看程序

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  const char *v3; // rdi

  v3 = "/dev/urandom";
  fd = open("/dev/urandom", 0, envp);
  if ( fd < 0 )
  {
    v3 = "open fail";
    puts("open fail");
  }
  init();
  person_ctfers(v3);
  return 0;
}
```

发现init 里面开了沙箱和获取了setcontext和free_hook的值，是后面用来判断free_hook是否被覆盖成setcontext

```
unsigned __int64 init()
{
  void *handle; // ST08_8

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  handle = dlopen("libc.so.6", 1);
  setcontext_addr = (__int64)dlsym(handle, "setcontext");
  _free_hook_addr = (__int64)dlsym(handle, "__free_hook");
  dlclose(handle);
  return sandbox();
}
```

沙箱发现 open execve 被禁止

然后继续

```
void **add()
{
  void **result; // rax
  char v1; // [rsp+Fh] [rbp-51h]
  unsigned int size; // [rsp+10h] [rbp-50h]
  unsigned int size_4; // [rsp+14h] [rbp-4Ch]
  int i; // [rsp+18h] [rbp-48h]
  unsigned int v5; // [rsp+1Ch] [rbp-44h]
```

```
  void *buf; // [rsp+20h] [rbp-40h]
  char *dest; // [rsp+28h] [rbp-38h]
  char s; // [rsp+30h] [rbp-30h]
  unsigned __int64 v9; // [rsp+48h] [rbp-18h]

  v9 = __readfsqword(0x28u);
  memset(&s, 0, 0x10uLL);
  printf("index>> ", 0LL);
  v5 = Str2Int();
  if ( v5 <= 0x1F )
  {
    printf("input your name 's size>> ");
    size = Str2Int();
    if ( size > 0x1F7 )
      size = 504;
    buf = calloc(1uLL, size);
    if ( !buf )
    {
      puts("malloc error");
      exit(-1);
    }
    printf("input your name>> ", size);
    read(0, buf, size);
    printf("input your password 's size>> ", buf);
    size_4 = Str2Int();
    if ( size_4 > 0xF8 )
      size_4 = 248;
    dest = (char *)calloc(1uLL, size_4);
    printf("input your password>> ", size_4);
    read(0, (char *)&person + 280 * v5 + 16, size_4);
    read(fd, &s, 8uLL);
    v1 = s;
    for ( i = 0; i < strlen((const char *)&person + 280 * v5 + 16); ++i )
      *((_BYTE *)&unk_203090 + 280 * v5 + i) = (v1 + *((char *)&unk_203090
+ 280 * v5 + i)) % 255;
    strcpy(dest, (const char *)&person + 280 * v5 + 16);//这里有个洞，one off
by null
    *((_QWORD *)&unk_203088 + 35 * v5) = buf;
    *((_DWORD *)&person + 70 * v5) = size;
    result = qword_203190;
    qword_203190[35 * v5] = dest;
  }
  else
  {
    puts("index error");
    result = 0LL;
  }
  return result;
}
```

存在one of by null

然后dele对 free_hook进行了判断

```
unsigned __int64 edit()
{
  size_t v0; // rax
```

```
  size_t v1; // rax
  char v3; // [rsp+3h] [rbp-4Dh]
  int i; // [rsp+4h] [rbp-4Ch]
  unsigned int v5; // [rsp+8h] [rbp-48h]
  size_t nbytes; // [rsp+Ch] [rbp-44h]
  unsigned __int64 v7; // [rsp+38h] [rbp-18h]

  v7 = __readfsqword(0x28u);
  memset((char *)&nbytes + 4, 0, 0x20uLL);
  printf("index>> ", 0LL);
  v5 = Str2Int();
  if ( *((_QWORD *)&unk_203088 + 35 * v5) && v5 <= 0x1F && qword_203190[35 * v5] )
  {
    printf("name>> ");
    read(0, *((void **)&unk_203088 + 35 * v5), *((unsigned int *)&person + 70 * v5));
    LODWORD(nbytes) = strlen((const char *)&person + 280 * v5 + 16);
    v0 = strlen((const char *)&person + 280 * v5 + 16);
    memset((char *)&person + 280 * v5 + 16, 0, v0);
    v1 = strlen((const char *)qword_203190[35 * v5]);
    memset(qword_203190[35 * v5], 0, v1);
    printf("password>> ", 0LL);
    read(0, (char *)&person + 280 * v5 + 16, (unsigned int)nbytes);
    read(fd, (char *)&nbytes + 4, 8uLL);
    v3 = BYTE4(nbytes);
    for ( i = 0; i < strlen((const char *)&person + 280 * v5 + 16); ++i )
      *((_BYTE *)&unk_203090 + 280 * v5 + i) = (v3 + *((char *)&unk_203090 + 280 * v5 + i)) % 255;
    strcpy((char *)qword_203190[35 * v5], (const char *)&person + 280 * v5 + 16);
  }
  else
  {
    puts("idx error");
  }
  return __readfsqword(0x28u) ^ v7;
```

只能show一次

```
int show()
{
  __int64 v0; // rax
  signed int i; // [rsp+Ch] [rbp-4h]

  LODWORD(v0) = SHOW_COUNT;
  if ( SHOW_COUNT == 1 )
  {
    LODWORD(v0) = puts("only one chance to show ");
    SHOW_COUNT = 0;
    for ( i = 0; i <= 31; ++i )
    {
      v0 = *((_QWORD *)&unk_203088 + 35 * i);
      if ( v0 )
        LODWORD(v0) = write(1, *((const void **)&unk_203088 + 35 * i), *((unsigned int *)&person + 70 * i));
    }
```

```
 }
  return v0;
}
```

exp编写：

首先用one off by null 来进行unlink，造成double free，然后再劫持top_chunk，然后就是控制free_hook了，但是free_hook附近并没有可以用的size，所以继续往free_hook上面的寻找，在0x1000左右有个size可以满足top_chunk，然后就是一直malloc到 free_hook,，但此时 没有栈地址，也不能利用setcontext，就只能改成printf，然后泄露stack地址和程序基地址，然后就是rop了