low_re

这个题其实算是逆向题, 考虑到这个题不是常规逆向思路, 并且有点谜语, 所以我把这个题放misc里面了.

出题人比较菜, re和misc师傅们轻点骂, orz.

这个题目的预期解是通过pintool之类的插桩工具进行指令计数来进行爆破, 但是我也看到有一些师傅使用钩子来获得信息的一些思路, 甚至于做了一些vmp的逆向, 师傅们tql.

师傅们基本上都是用的官方给的代码来构建, 但是pin官方给的指令计数是算了库的, 这样的话计算出来的指令正常波动就会非常大, 所以我使用了2560次的hash加密, 希望能够使得正确答案的指令大小更为明显, 但是我自己爆破的时候, 如果从头开始爆破, 在得到最后几个字符的时候可能会发生错误.如果不计算库的话, 指令波动要小很多, 也就不会出现爆破不出来的情况了.

至于插桩工具可以使用pin或者dynamorio, 不过得去阅读文档才能改改, 达到不计算库的目的

爆破脚本:

```
from winpwn import *
import threading
from concurrent.futures import ThreadPoolExecutor, Future
import re

def test(data):
    tmp = data
    cmd = '' #your pin.exe path
    module = '' #your pintool.dll path
    p = process([cmd, '-t', module, '--', '.\\low_re.exe'])
    #p = process([cmd, '-c', module, '-only_from_app', '--', '.\\low_re.exe'])
//dynamorio的命令行
    p.recvline()
    p.recvline()
    p.sendline(data)
    data = p.recvuntil("\n")
    if (data[0:5] != "Count"):
        data = p.recvuntil("\n")
    data = int(data.split("Count ")[1].split(" ")[0])
    #if (data[0:5] != "Instr"): //dynamorio的输出
        #data = p.recvuntil("\n")
    #data = int(re.findall('\d+', data)[0])
    p.close()
    return data

pool = ThreadPoolExecutor(8)


testlen = 17

if (testlen):
    flag = ""
    while len(flag) != testlen:
        tasks = []
        results = []
        for i in range(32, 0x7f):
            data = flag + chr(i) + 'i' * (testlen - 1 - len(flag))
```

```
            tasks.append(pool.submit(test, data))
        for t in tasks:
            results.append(t.result())
        #print(chr(a.index(max(a)) + 32))
        flag += chr(results.index(max(results)) + 32)
        print(flag)
else:
    i = 1
    a = []
    while i < 64:
        a.append(test(i * '1'))
        i += 1
    print(a.index(max(a)) + 1)
```