LAB 1

潘亦晟

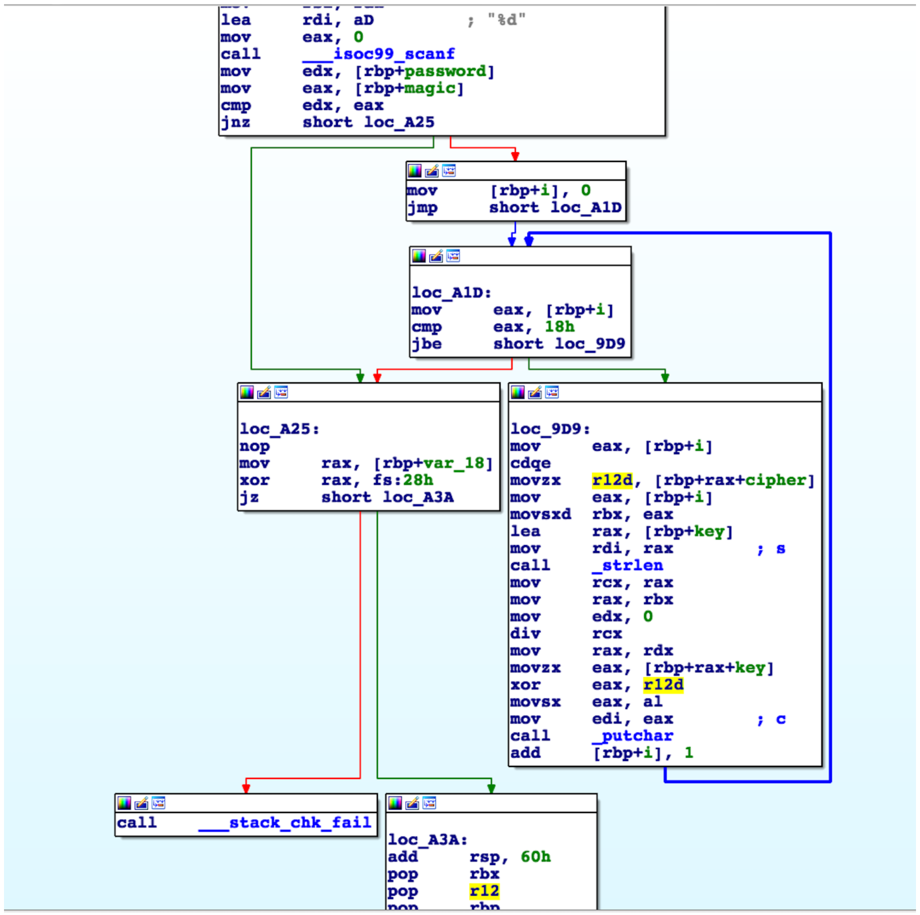
515021910384

1 ELF文件逆向分析

1.1 编译并去掉符号表,试比较有无符号表的区别

- 去掉符号表后，IDA中无法显示真实函数名
- 去掉符号表后，文件相对有符号表变小

1.2 使用IDA分析程序算法



```
lea     rdi, aD         ; "%d"
mov     eax, 0
call    ___isoc99_scanf
mov     edx, [rbp+password]
mov     eax, [rbp+magic]
cmp     edx, eax
jnz     short loc_A25
```

```
mov     [rbp+i], 0
jmp     short loc_A1D
```

```
loc_A1D:
mov     eax, [rbp+i]
cmp     eax, 18h
jbe     short loc_9D9
```

```
loc_A25:
nop
mov     rax, [rbp+var_18]
xor     rax, fs:28h
jz      short loc_A3A
```

```
loc_9D9:
mov     eax, [rbp+i]
cdqe
movzx   r12d, [rbp+rax+cipher]
mov     eax, [rbp+i]
movsxd  rbx, eax
lea     rax, [rbp+key]
mov     rdi, rax        ; s
call    _strlen
mov     rcx, rax
mov     rax, rbx
mov     edx, 0
div     rcx
mov     rax, rdx
movzx   eax, [rbp+rax+key]
xor     eax, r12d
movsx   eax, al
mov     edi, eax        ; c
call    _putchar
add     [rbp+i], 1
```

```
call    ___stack_chk_fail
```

```
loc_A3A:
add     rsp, 60h
pop     rbx
pop     r12
pop     rbp
```

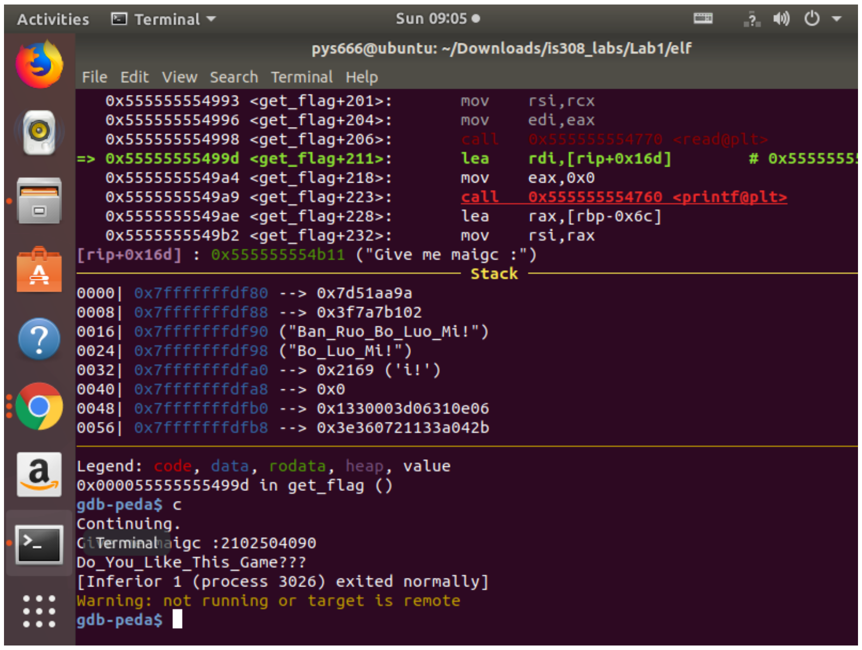
Figure 1: IDA 分析

- 从随机数表中获取password;

- 输入magic，并与password进行比较；
- 若相等，输出secret。secret由cipher[i] 异或上 key[i% strlen(key)]得到。

1.3 使用gdb调试，观察password的值，输出secret

1.3.1 步骤：


Activities
Terminal
Sun 09:05
pys666@ubuntu: ~/Downloads/is308_labs/Lab1/elf
File Edit View Search Terminal Help

```
   0x555555554993 <get_flag+201>:       mov    rsi,rcx
   0x555555554996 <get_flag+204>:       mov    edi,eax
   0x555555554998 <get_flag+206>:       call   0x555555554770 <read@plt>
=> 0x55555555499d <get_flag+211>:       lea    rdi,[rip+0x16d]        # 0x5555555
   0x5555555549a4 <get_flag+218>:       mov    eax,0x0
   0x5555555549a9 <get_flag+223>:       call   0x555555554760 <printf@plt>
   0x5555555549ae <get_flag+228>:       lea    rax,[rbp-0x6c]
   0x5555555549b2 <get_flag+232>:       mov    rsi,rax
[rip+0x16d] : 0x555555554b11 ("Give me maigc :")
----------------------------------------- Stack -----------------------------------------
0000| 0x7fffffffdf80 --> 0x7d51aa9a
0008| 0x7fffffffdf88 --> 0x3f7a7b102
0016| 0x7fffffffdf90 ("Ban_Ruo_Bo_Luo_Mi!")
0024| 0x7fffffffdf98 ("Bo_Luo_Mi!")
0032| 0x7fffffffdfa0 --> 0x2169 ('i!')
0040| 0x7fffffffdfa8 --> 0x0
0048| 0x7fffffffdfb0 --> 0x1330003d06310e06
0056| 0x7fffffffdfb8 --> 0x3e360721133a042b
-----------------------------------------------------------------------------------------
Legend: code, data, rodata, heap, value
0x000055555555499d in get_flag ()
gdb-peda$ c
Continuing.
GiTerminalaigc :2102504090
Do_You_Like_This_Game???
[Inferior 1 (process 3026) exited normally]
Warning: not running or target is remote
gdb-peda$
```

Figure 2: gdb 调试

- 我们使用gab-peda调试si命令，进入get flag()函数中
- 使用ni命令，观察到执行call 0x555555554770 read@plt后，栈中写入password值
- 在栈中读出0x7fffffffdf80 地址中的数据，将其转换为十进制数即得到password值。
- 使用c命令继续执行剩余代码，屏幕中显示“Give me maigc :”
- 输入之前得到的password值,即可得到secret值

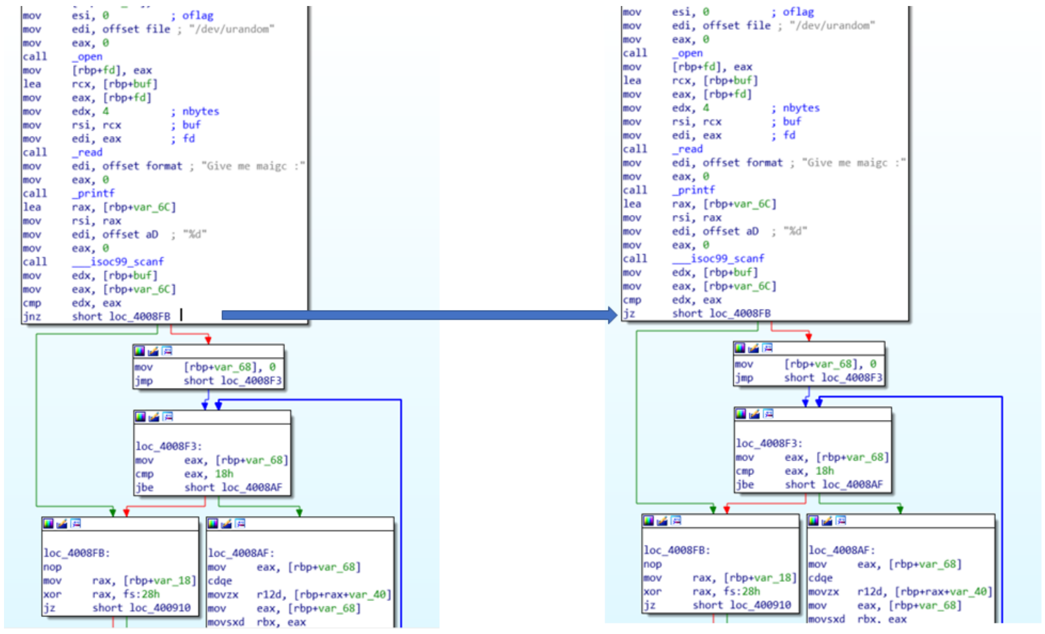
1.3.2 结论：

Secret: $Do\_You\_Like\_This\_Game$???

1.4 其他得到secret值的方法

1.4.1 原理：

使用IDA中的修改汇编代码的功能，可将get flag函数中用来判断magic和password是否相等的跳转语句jnz short loc_A25 改为 jz short loc_A25，即可得到输入任何不等于password的magic得到secret值的程序。



```
mov     esi, 0          ; oflag
mov     edi, offset file ; "/dev/urandom"
mov     eax, 0
call    _open
mov     [rbp+fd], eax
lea     rcx, [rbp+buf]
mov     eax, [rbp+fd]
mov     edx, 4          ; nbytes
mov     rsi, rcx        ; buf
mov     edi, eax        ; fd
call    _read
mov     edi, offset format ; "Give me maigc :"
mov     eax, 0
call    _printf
lea     rax, [rbp+var_6C]
mov     rsi, rax
mov     edi, offset aD  ; "%d"
mov     eax, 0
call    ___isoc99_scanf
mov     edx, [rbp+buf]
mov     eax, [rbp+var_6C]
cmp     edx, eax
jnz     short loc_4008FB

mov     [rbp+var_68], 0
jmp     short loc_4008F3

loc_4008F3:
mov     eax, [rbp+var_68]
cmp     eax, 18h
jbe     short loc_4008AF

loc_4008FB:
nop
mov     rax, [rbp+var_18]
xor     rax, fs:28h
jz      short loc_400910

loc_4008AF:
mov     eax, [rbp+var_68]
cdqe
movzx   r12d, [rbp+rax+var_40]
mov     eax, [rbp+var_68]
movsxd  rbx, eax
```

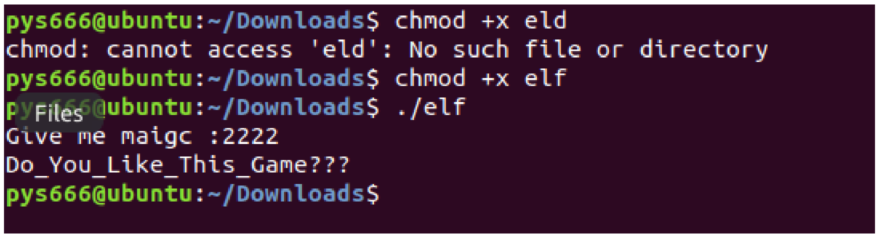
```
mov     esi, 0          ; oflag
mov     edi, offset file ; "/dev/urandom"
mov     eax, 0
call    _open
mov     [rbp+fd], eax
lea     rcx, [rbp+buf]
mov     eax, [rbp+fd]
mov     edx, 4          ; nbytes
mov     rsi, rcx        ; buf
mov     edi, eax        ; fd
call    _read
mov     edi, offset format ; "Give me maigc :"
mov     eax, 0
call    _printf
lea     rax, [rbp+var_6C]
mov     rsi, rax
mov     edi, offset aD  ; "%d"
mov     eax, 0
call    ___isoc99_scanf
mov     edx, [rbp+buf]
mov     eax, [rbp+var_6C]
cmp     edx, eax
jz      short loc_4008FB

mov     [rbp+var_68], 0
jmp     short loc_4008F3

loc_4008F3:
mov     eax, [rbp+var_68]
cmp     eax, 18h
jbe     short loc_4008AF

loc_4008FB:
nop
mov     rax, [rbp+var_18]
xor     rax, fs:28h
jz      short loc_400910

loc_4008AF:
mov     eax, [rbp+var_68]
cdqe
movzx   r12d, [rbp+rax+var_40]
mov     eax, [rbp+var_68]
movsxd  rbx, eax
```

Figure 3: 更改汇编代码

1.4.2 步骤：

- 使用IDA中的Assemble功能对原elf文件进行更改
- 使用apply patches to input file保存更改后的elf文件
- 在Ubuntu中运行elf文件，输入任何magic值验证猜想

1.4.3 结果：



```
pys666@ubuntu:~/Downloads$ chmod +x eld
chmod: cannot access 'eld': No such file or directory
pys666@ubuntu:~/Downloads$ chmod +x elf
pys666@ubuntu:~/Downloads$ ./elf
Give me maigc :2222
Do_You_Like_This_Game???
pys666@ubuntu:~/Downloads$
```

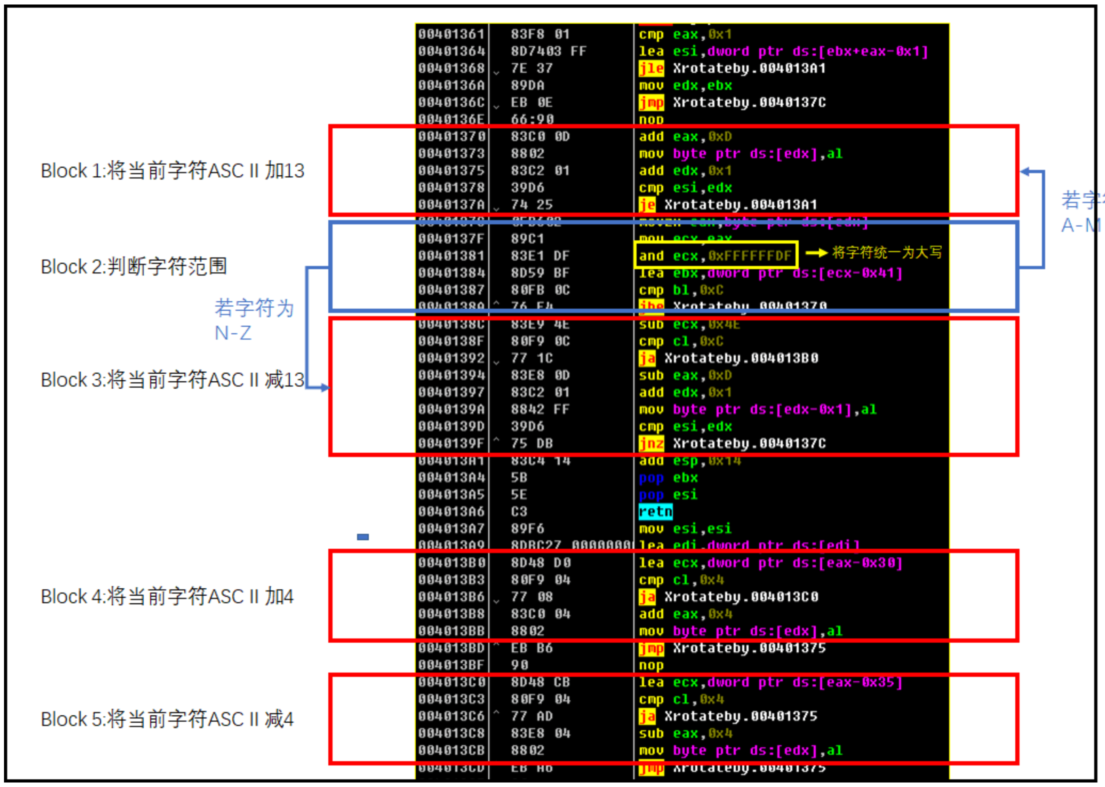
Figure 4: other method

2 EXE文件分析

2.1 分析程序算法

2.1.1 还原算法

- 输入字符串；
- 将字符串进过预处理。若字符为A-M或者a-m，则将其ACSII码加13；若字符为N-Z或者n-z，则将其ACSII码减13；若为0-4的数字，则加4；若为5-9的数字，则减4；


Block 1:将当前字符ASC II 加13
Block 2:判断字符范围
若字符为 N-Z
Block 3:将当前字符ASC II 减13
Block 4:将当前字符ASC II 加4
Block 5:将当前字符ASC II 减4
若字符为 A-M
将字符统一为大写

```
00401361    83F8 01         cmp eax,0x1
00401364    8D7403 FF       lea esi,dword ptr ds:[ebx+eax-0x1]
00401368  ⌄ 7E 37           jle Xrotateby.004013A1
0040136A    89DA            mov edx,ebx
0040136C  ⌄ EB 0E           jmp Xrotateby.0040137C
0040136E    66:90           nop
00401370    83C0 0D         add eax,0xD
00401373    8802            mov byte ptr ds:[edx],al
00401375    83C2 01         add edx,0x1
00401378    39D6            cmp esi,edx
0040137A  ⌄ 74 25           je Xrotateby.004013A1
0040137C    0FB602          movzx eax,byte ptr ds:[edx]
0040137F    89C1            mov ecx,eax
00401381    83E1 DF         and ecx,0xFFFFFFDF
00401384    8D59 BF         lea ebx,dword ptr ds:[ecx-0x41]
00401387    80FB 0C         cmp bl,0xC
0040138A  ^ 76 E4           jbe Xrotateby.00401370
0040138C    83E9 4E         sub ecx,0x4E
0040138F    80F9 0C         cmp cl,0xC
00401392  ⌄ 77 1C           ja Xrotateby.004013B0
00401394    83E8 0D         sub eax,0xD
00401397    83C2 01         add edx,0x1
0040139A    8842 FF         mov byte ptr ds:[edx-0x1],al
0040139D    39D6            cmp esi,edx
0040139F  ^ 75 DB           jnz Xrotateby.0040137C
004013A1    83C4 14         add esp,0x14
004013A4    5B              pop ebx
004013A5    5E              pop esi
004013A6    C3              retn
004013A7    89F6            mov esi,esi
004013A9    8DBC27 00000000 lea edi,dword ptr ds:[edi]
004013B0    8D48 D0         lea ecx,dword ptr ds:[eax-0x30]
004013B3    80F9 04         cmp cl,0x4
004013B6  ⌄ 77 08           ja Xrotateby.004013C0
004013B8    83C0 04         add eax,0x4
004013BB    8802            mov byte ptr ds:[edx],al
004013BD  ^ EB B6           jmp Xrotateby.00401375
004013BF    90              nop
004013C0    8D48 CB         lea ecx,dword ptr ds:[eax-0x35]
004013C3    80F9 04         cmp cl,0x4
004013C6  ^ 77 AD           ja Xrotateby.00401375
004013C8    83E8 04         sub eax,0x4
004013CB    8802            mov byte ptr ds:[edx],al
004013CD    EB A6           jmp Xrotateby.00401375
```

Figure 5

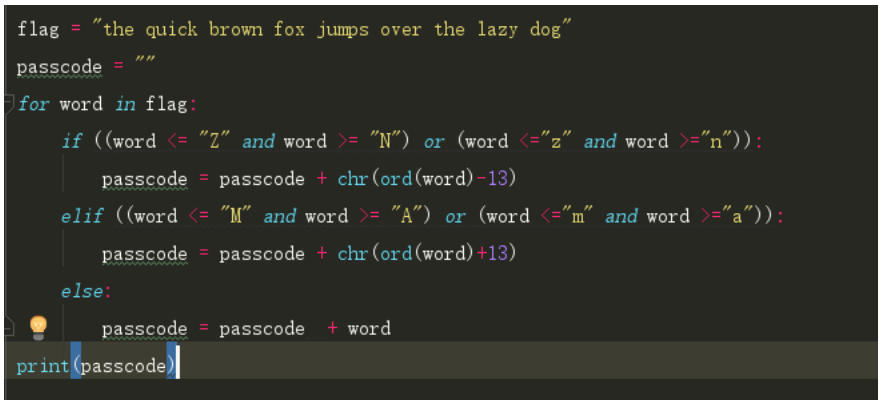
- 将预处理后的字符串与预设flag（“the quick brown fox jumps over the lazy dog”）进行比较;
- 若相等，则输出”Big Cong !!! ”；若不相等，则输出”Try again ... ”、

2.1.2 还原passcode

具体代码见 6
Passcode: gur dhvpx oebja sbk whzcf bire gur ynml qbt

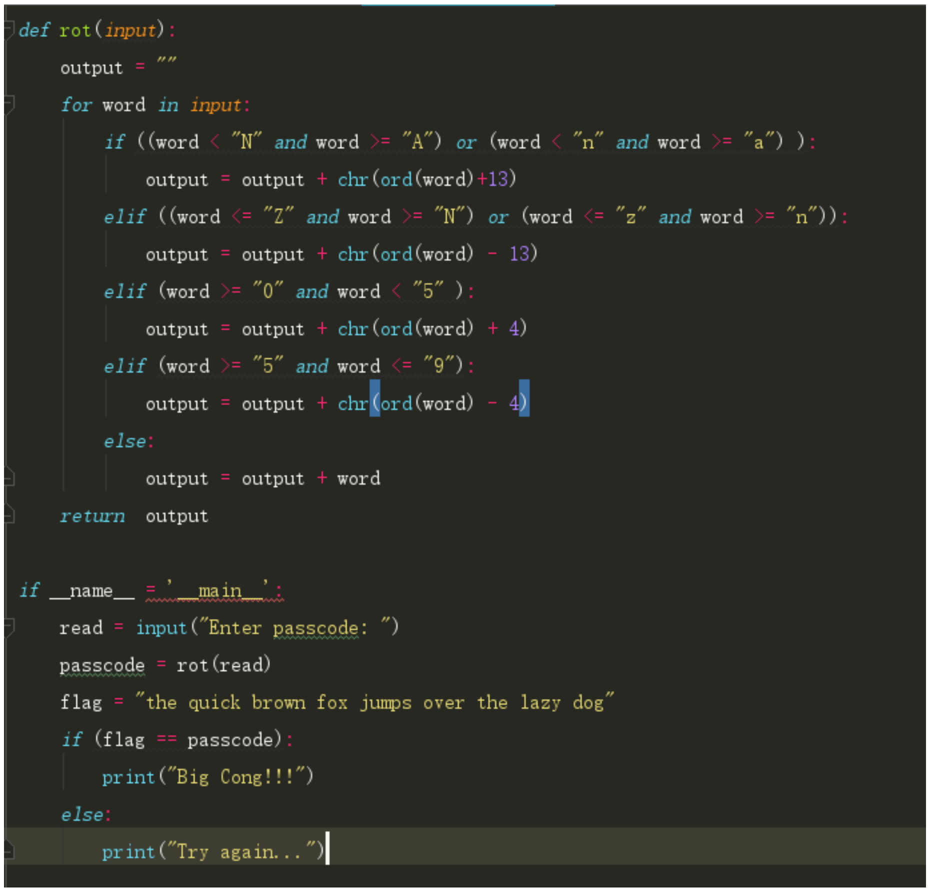
2.2 使用Python还原程序主要功能

具体代码见 7



```python
flag = "the quick brown fox jumps over the lazy dog"
passcode = ""
for word in flag:
    if ((word <= "Z" and word >= "N") or (word <="z" and word >="n")):
        passcode = passcode + chr(ord(word)-13)
    elif ((word <= "M" and word >= "A") or (word <="m" and word >="a")):
        passcode = passcode + chr(ord(word)+13)
    else:
        passcode = passcode  + word
print(passcode)
```

Figure 6: decode



```python
def rot(input):
    output = ""
    for word in input:
        if ((word < "N" and word >= "A") or (word < "n" and word >= "a") ):
            output = output + chr(ord(word)+13)
        elif ((word <= "Z" and word >= "N") or (word <= "z" and word >= "n")):
            output = output + chr(ord(word) - 13)
        elif (word >= "0" and word < "5" ):
            output = output + chr(ord(word) + 4)
        elif (word >= "5" and word <= "9"):
            output = output + chr(ord(word) - 4)
        else:
            output = output + word
    return  output

if __name__ = '__main__':
    read = input("Enter passcode: ")
    passcode = rot(read)
    flag = "the quick brown fox jumps over the lazy dog"
    if (flag == passcode):
        print("Big Cong!!!")
    else:
        print("Try again...")
```

Figure 7: python code