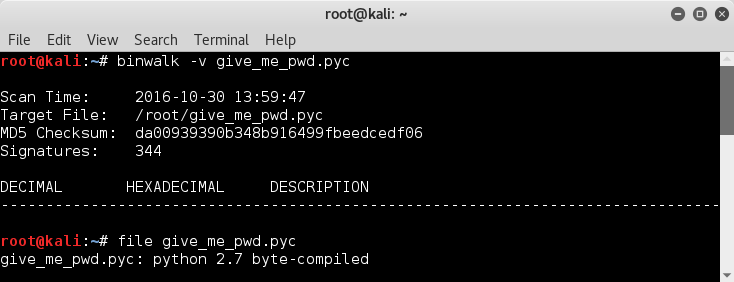
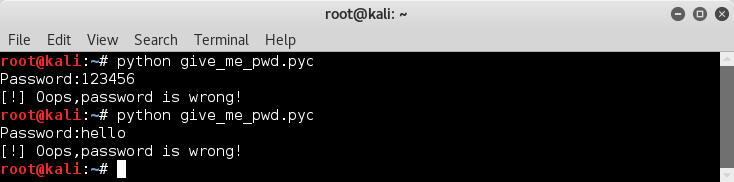
1、反编译pyc

得到pyc后缀的文件，先使用binwalk和file识别文件



结果显示确实是pyc字节码文件，试着运行：



提示输入密码Password，试探性输入几个密码，提示密码错误，通过搜索引擎搜索查找资料，可以通过[uncompyle6](https://github.com/rocky/python-uncompyle6)反编译pyc文件得到源码，根据说明安装使用uncompyle6。

1. root@kali:~# git clone https://github.com/rocky/python-uncompyle6.git
2. root@kali:~# cd python-uncompyle6/
3. root@kali:~/python-uncompyle6# pip install -r requirements.txt
4. root@kali:~/python-uncompyle6# python setup.py install
5. root@kali:~# uncompyle6 give\_me\_pwd.pyc > original.py

2、分析orignal.py源代码中的算法

1. # uncompyle6 version 2.9.3
2. # Python bytecode 2.7 (62211)
3. # Decompiled from: Python 2.7.12+ (default, Sep  1 2016, 20:27:38)
4. # [GCC 6.2.0 20160927]
5. # Embedded file name: original.py
6. # Compiled at: 2016-10-30 15:41:32
7. pwd = raw\_input('Password:')
8. letter = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'z', 'x', 'c', 'v', 'b', 'n', 'm']
9. flag\_list = [20, 13, 22, 13, 11, 1, 3, 6, 55, 39, 7, 24, 85, 30, 76, 1, 55, 64, 26, 51, 21, 91, 93, 15, 28, 12]
10. randomKey = ''
11. **for** i **in** range(1, len(pwd) + 1):
12. i = i \*\* (i \* i / i // i + i - i) << i | i ^ i & i \*\* (i \* i / i // i + i - i) \* i \*\* (i \* i / i // i + i - i) << i | i ^ i & i \*\* (i \* i / i // i + i - i) << i | i ^ i & i \*\* (i \* i / i // i + i - i) \* i \*\* (i \* i / i // i + i - i) << i | i ^ i
13. randomKey += letter[i % len(letter)]
15. wrongPassword = False
16. **for** i **in** range(len(flag\_list)):
17. **if** flag\_list[i] != ord(pwd[i]) ^ ord(randomKey[i % len(randomKey)]):
18. wrongPassword = True
19. **break**
21. **if** wrongPassword:
22. **print** '[!] Oops,password is wrong!'
23. **else**:
24. **print** '[\*] Good,password is the flag!'
25. # okay decompiling original.pyc

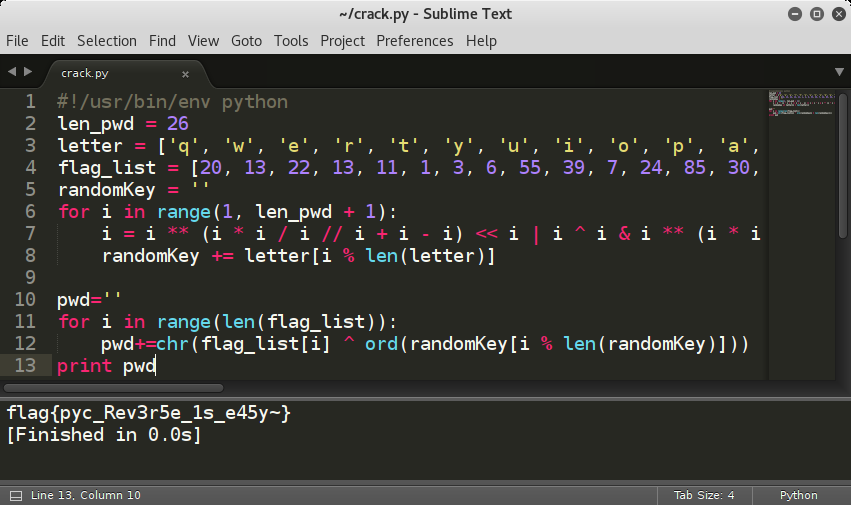
通过分析我们知道核心语句为

1. **if** flag\_list[i]!=ord(pwd[i])^ord(randomKey[i%len(randomKey)]):

其中flag\_list我们已经知道，而pwd是我们输入的password，而randomkey是根据输入pwd长度通过复杂的算法生成的，仔细分析这个生成算法我们知道生成的randomkey只与输入的长度有关，但是我们不知道password的长度，这里需要我们打开脑洞，既然是异或运算，且flag\_list的长度为26，所以大胆猜测pwd的长度为26，这样一来randomkey就已知了。接下来分析核心语句if，如果if里判断为真，意味着输入pwd不对，wrongPassword为true，就会提示输入password错误，如果if判断都通过，那么wrongPassword为false，那么我们输入的pwd就是password。但是我们怎么知道正确的pwd，似乎没什么办法，但是咨询分析if语句的运算，我们发现只是简单的pwd和randomkey的异或，而现在flag\_list，randomkey已知，根据异或的特点，我们可以反向计算出pwd，所以编写逆向求解代码。

3、编写逆向求解代码

1. #!/usr/bin/env python
2. len\_pwd = 26
3. letter = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'z', 'x', 'c', 'v', 'b', 'n', 'm']
4. flag\_list = [20, 13, 22, 13, 11, 1, 3, 6, 55, 39, 7, 24, 85, 30, 76, 1, 55, 64, 26, 51, 21, 91, 93, 15, 28, 12]
5. randomKey = ''
6. **for** i **in** range(1, len\_pwd + 1):
7. i = i \*\* (i \* i / i // i + i - i) << i | i ^ i & i \*\* (i \* i / i // i + i - i) \* i \*\* (i \* i / i // i + i - i) << i | i ^ i & i \*\* (i \* i / i // i + i - i) << i | i ^ i & i \*\* (i \* i / i // i + i - i) \* i \*\* (i \* i / i // i + i - i) << i | i ^ i
8. randomKey += letter[i % len(letter)]
10. pwd=''
11. **for** i **in** range(len(flag\_list)):
12. pwd+=chr(flag\_list[i] ^ ord(randomKey[i % len(randomKey)]))
13. **print** pwd



最终得到flag。