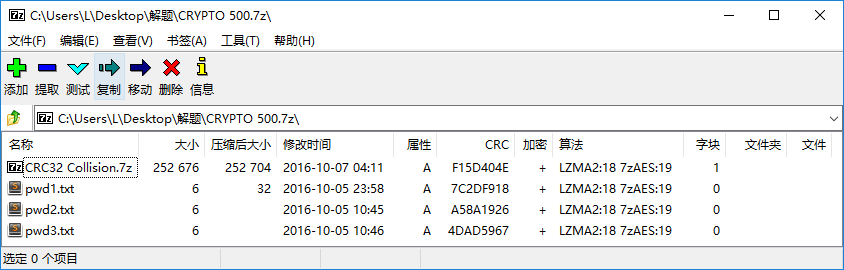
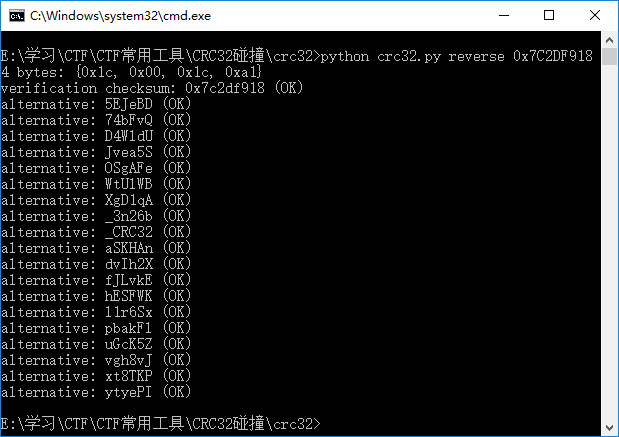
1、第一层：CRC32碰撞

得到题目压缩包



尝试解压，却发现需要密码，根据压缩包里的文件和文件名，猜测是题意是想让我们通过对pwd1，pwd2和pwd3文本文件进行CRC32碰撞获得文本内容，从而得到组合密码。

通过网上查找资料，了解CRC32碰撞原理，了解原理之后可以在[github](https://github.com/search?l=Python&q=crc32&type=Repositories&utf8=%E2%9C%93)上找到可利用的[Python脚本](https://github.com/theonlypwner/crc32)。

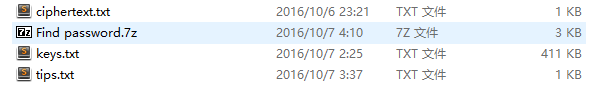


分别对3个CRC32值进行碰撞，找到最可能的组合：\_CRC32\_i5\_n0t\_s4f3，输入密码解压进入下一层。

如果是善于动手编程能力较好的小伙伴也可以自己编写一个脚本来跑，下面给出自己的demo：

1. # -\*- coding: utf-8 -\*-
2. # crc32Collision.py
3. **import** threading
4. **import** binascii
5. **import** time
7. **def** breakpassword():
8. start=time.clock()
9. crc\_num=set([0x7C2DF918,0xA58A1926,0x4DAD5967])
10. x = range(32,128)
11. **for** i **in** x:
12. **for** j **in** x:
13. **for** k **in** x:
14. **for** l **in** x:
15. **for** m **in** x:
16. **for** n **in** x:
17. mutex.acquire()
18. string=chr(i)+chr(j)+chr(k)+chr(l)+chr(m)+chr(n)
19. **if** binascii.crc32(string) **in** crc\_num:
20. **print** "crc32 of %s is-> %s" %(string,hex(binascii.crc32(string)))
21. f=open("string.txt",'a')
22. f.write(string)
23. f.close()
24. mutex.release()
25. end=time.clock()
26. **print** "Used time: %f s" % (end - start)
28. **def** main(thread\_num):
29. **print** "breaking,please wait!"
30. **global** mutex
31. mutex=threading.Lock()
32. threads=[]
33. **for** x **in** xrange(0,thread\_num):
34. threads.append(threading.Thread(target=breakpassword))
35. **for** t **in** threads:
36. t.start()
38. **for** t **in** threads:
39. t.join()
41. **if** \_\_name\_\_ == '\_\_main\_\_':
42. main(10)
43. 第二层：维吉尼亚基于字典攻击

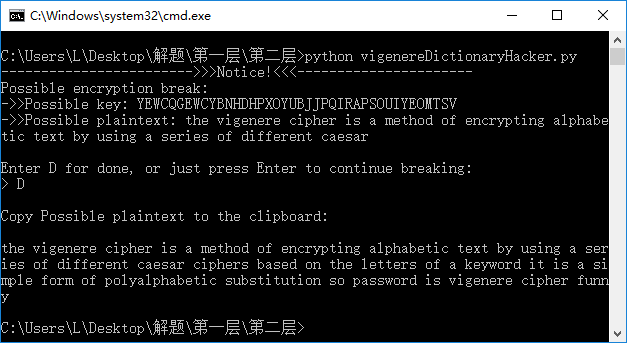
解压之后得到下图三个文件：



打开tips.txt读懂题意，要求我们在keys.txt中找到密钥解密ciphertext.txt，解密密文之后便可以得到Find password.7z的解压密码。通过网上大量的资料查阅，了解维吉尼亚密码加密解密原理和算法，搜索相应的解密工具可以在<http://inventwithpython.com/hacking/diff/>找到可以模板，需要读懂相应模块并修改使用：

1. # -\*- coding: utf-8 -\*-
2. #vigenereDictionaryHacker.py
3. **import** detectEnglish
5. LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7. **def** translateMessage(key, message, mode):
8. translated = [] # 存储加密/解密消息字符串
9. keyIndex = 0
10. key = key.upper()
11. **for** symbol **in** message: # 遍历每个消息里的字符的消息
12. num = LETTERS.find(symbol.upper())
13. **if** num != -1: # -1 意味着转换为大写在LETTERS找不到
14. **if** mode == 'encrypt':
15. num += LETTERS.find(key[keyIndex]) # 加密时相加
16. **elif** mode == 'decrypt':
17. num -= LETTERS.find(key[keyIndex]) # 解密时相减
18. num %= len(LETTERS) # 处理潜在的循环
19. # 添加转换后加密/解密字符
20. **if** symbol.isupper():
21. translated.append(LETTERS[num])
22. **elif** symbol.islower():
23. translated.append(LETTERS[num].lower())
24. keyIndex += 1 # 继续下一个用密钥字符来解密
25. **if** keyIndex == len(key):
26. keyIndex = 0
27. **else**:
28. # 字符不在LETTERS里直接添加
29. translated.append(symbol)
30. **return** ''.join(translated)
32. **def** decryptMessage(key, message):
33. **return** translateMessage(key, message, 'decrypt')
35. **def** hackVigenere(ciphertext):
36. fo = open('keys.txt')
37. words = fo.readlines()
38. fo.close()
39. **for** word **in** words:
40. word = word.strip()
41. decryptedText = decryptMessage(word, ciphertext)
42. **if** detectEnglish.isEnglish(decryptedText, wordPercentage=40):
43. **print**('------------------------>>>Notice!<<<----------------------')
44. **print**('Possible encryption break:')
45. **print**('->>Possible key: ' + str(word))
46. **print**('->>Possible plaintext: ' + decryptedText[:100])
47. **print**('Enter D for done, or just press Enter to continue breaking:')
48. response = raw\_input('> ')
49. **if** response.upper().startswith('D'):
50. **return** decryptedText
52. **def** main():
53. ciphertext = """rla xymijgpf ppsoto wq u nncwel ff tfqlgnxwzz sgnlwduzmy vcyg ib bhfbe u tnaxua ff satzmpibf vszqen eyvlatq cnzhk dk hfy mnciuzj ou s yygusfp bl dq e okcvpa hmsz vi wdimyfqqjqubzc hmpmbgxifbgi qs lciyaktb jf clntkspy drywuz wucfm"""
54. hackedMessage = hackVigenere(ciphertext)
55. **if** hackedMessage != None:
56. **print**('\nCopy Possible plaintext to the clipboard:\n')
57. **print**(hackedMessage)
58. **else**:
59. **print**('Failed to hack encryption.')
61. **if** \_\_name\_\_ == '\_\_main\_\_':
62. main()
63. # -\*- coding: utf-8 -\*-
64. # detectEnglish.py
65. # 英文单词探测模块
66. # 模块引用:
67. #   import detectEnglish
68. #   detectEnglish.isEnglish(someString) # 返回真或假
69. # 模块需要一个包含常见英文单词的"words.txt"，下载地址：http://invpy.com/dictionary.txt
71. UPPERLETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
72. LETTERS\_AND\_SPACE = UPPERLETTERS + UPPERLETTERS.lower() + ' \t\n'
74. **def** loadDictionary():
75. dictionaryFile = open('words.txt')
76. englishWords = {}
77. **for** word **in** dictionaryFile.read().split('\n'):
78. englishWords[word] = None
79. dictionaryFile.close()
80. **return** englishWords
82. ENGLISH\_WORDS = loadDictionary()
84. **def** getEnglishCount(message):
85. message = message.upper()
86. message = removeNonLetters(message)
87. possibleWords = message.split()
88. # print possibleWords
89. **if** possibleWords == []:
90. **return** 0.0 # 没有单词返回0.0
92. matches = 0
93. **for** word **in** possibleWords:
94. **if** word **in** ENGLISH\_WORDS:
95. matches += 1
96. **return** float(matches) / len(possibleWords)
98. **def** removeNonLetters(message):
99. lettersOnly = []
100. **for** symbol **in** message:
101. **if** symbol **in** LETTERS\_AND\_SPACE:
102. lettersOnly.append(symbol)
103. **return** ''.join(lettersOnly)
105. **def** isEnglish(message, wordPercentage=20, letterPercentage=85):
106. # 默认设置转换后的message中单词的20%能在words.txt中的单词列表找到
107. # 默认设置转换后的message中85%是字母或空格
108. # (not punctuation or numbers).
109. wordsMatch = getEnglishCount(message) \* 100 >= wordPercentage
110. numLetters = len(removeNonLetters(message))
111. messageLettersPercentage = float(numLetters) / len(message) \* 100
112. lettersMatch = messageLettersPercentage >= letterPercentage
113. **return** wordsMatch **and** lettersMatch

解密结果：



所以Find password.7z的解压密码就是：vigenere cipher funny

1. 第三层：SHA1

解压Find password.7z得到如下文件：

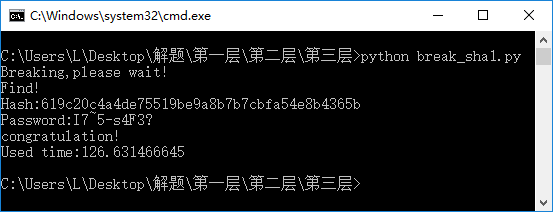
4.1

打开U need unzip password.txt读懂题意，题目要求我们通过编写脚本爆破SHA1的明文密文对，得到明文作为解压密码进入下一关。

了解了SHA1密码之后，通过查阅资料发现我们可以使用Python的hashlib模块来进行爆破，下面给出demo：

1. #break\_sha1.py
2. **import** hashlib
3. **import** time
5. **def** match(h,pwd):
6. hl=list(h)
7. **if** hl[0]=='6':
8. **if** hl[1]=='1':
9. **if** hl[2]=='9':
10. **if** hl[3]=='c':
11. **if** hl[4]=='2':
12. **if** hl[5]=='0':
13. **if** hl[6]=='c':
14. **if** hl[8]=='a':
15. **if** hl[16]=='9':
16. **if** hl[24]=='b':
17. **if** hl[32]=='e':
18. **print** "Find!"
19. **print** "Hash:%s" %h
20. **print** "Password:%s" %pwd
21. matched=1
22. **return** matched
24. **else**:
25. matched=0
26. **return** matched
28. **def** generate():
29. x=range(32,128)
30. **for** i **in** x:
31. **for** j **in** x:
32. **for** k **in** x:
33. **for** l **in** x:
34. pwd=chr(i)+'7'+chr(j)+'5-'+chr(k)+'4'+chr(l)+'3?'
35. sha1\_hash=hashlib.sha1()
36. sha1\_hash.update(pwd)
37. h=sha1\_hash.hexdigest()
38. matched=match(h,pwd)
39. **if** matched:
40. **print** "congratulation!"
41. **return** 0
42. **else**:
43. **pass**
45. **def** main():
46. start=time.clock()
47. **print** "Breaking,please wait!"
48. generate()
49. end=time.clock()
50. **print** "Used time:%s" %(end-start)
52. **if** \_\_name\_\_ == '\_\_main\_\_':
53. main()

结果：



所以得到Easy SHA1.7z的解压密码：I7~5-s4F3?

4、第四层：MD5不再安全

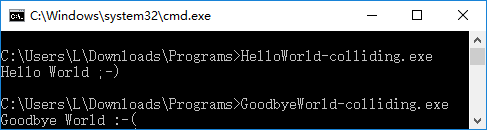
解压Easy SHA1.7z之后得到如下文件：

6.1

打开MD5\_is\_really\_safe？.txt文本，题目要让我们找到两个不同程序但是他们的MD5值却相同，这题考察我们对MD5安全性的感知度。通过搜索引擎可以找到大量关于王小云教授对MD5破解相关资料，百度关键词：MD5碰撞 MD5校验真的安全吗？ MD5真的已靠不住? 等等。



下载HelloWorld-colliding.exe GoodbyeWorld-colliding.exe运行结果：



Hello World ;-)已在提示里，那么另一个程序的输出就是：Goodbye World :-(

根据提示输入做为Vulnerable RSA.7z的解压密码，解压成功进入下一关。

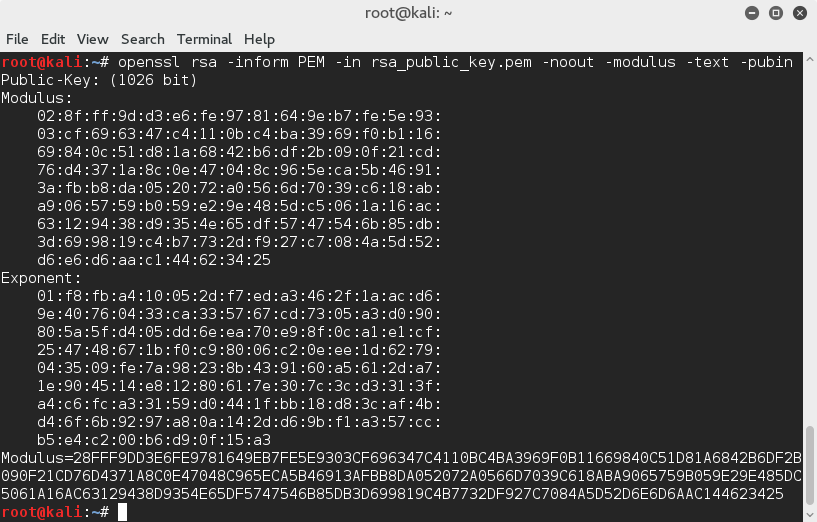
5、第五层：Vulnerable RSA

解压Vulnerable RSA.7z后得到如下文件：

7.1

在了解RSA的原理和常见的攻击方法后，我们用[OpeSSL](https://github.com/openssl/openssl)来导入公钥查看模数n，指数e。

1. openssl rsa -inform PEM -**in** rsa\_public\_key.pem -noout -modulus -text -pubin

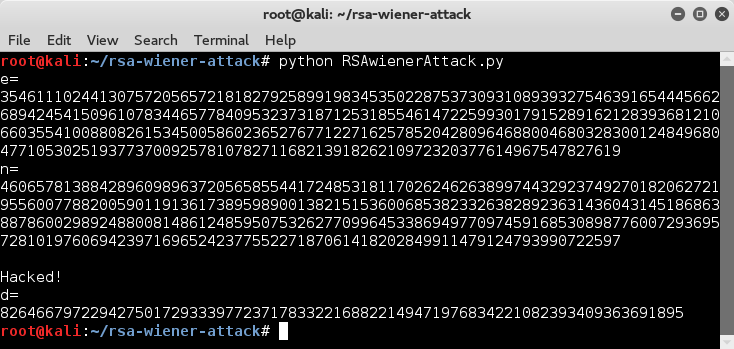


可以看到指数（Exponent）很大，在RSA中我们知道ed ≡ 1 (mod φ(n))，如果n确定，e非常大，就会导致d很小，就会出现维纳攻击（[Wiener's attack](https://en.wikipedia.org/wiki/Wiener%27s_attack)），攻击原理是使用连分式（[Continued fraction](https://en.wikipedia.org/wiki/Continued_fraction)）去求得d。

了解原理后我们可以在github找到基于维纳攻击的工具[rsa-wiener-attack](https://github.com/pablocelayes/rsa-wiener-attack)，然后将其中的RSAwienerHacker.py改写一下：

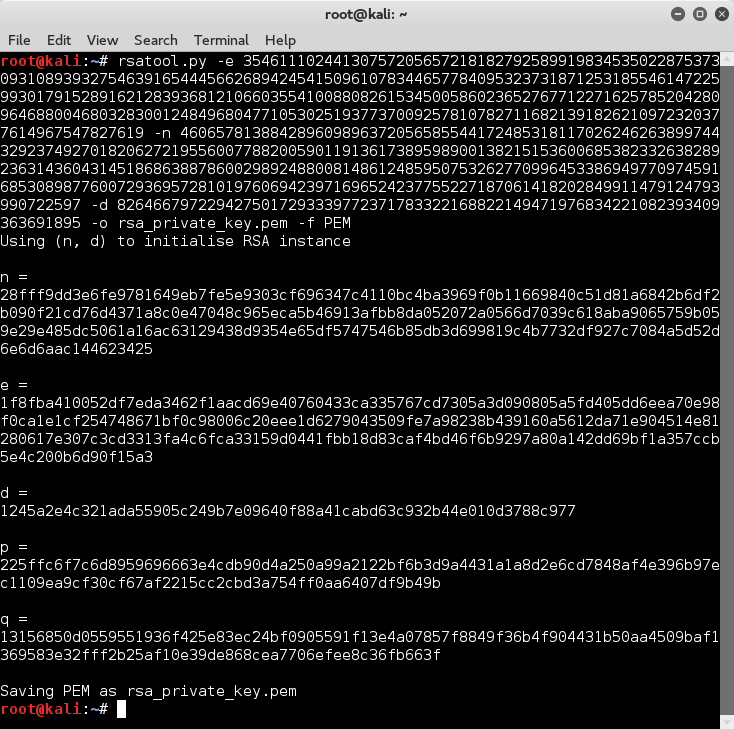
1. #RSAwienerAttack.py
2. **import** ContinuedFractions, Arithmetic
4. **def** hack\_RSA(e,n):
5. '''''
6. Finds d knowing (e,n)
7. applying the Wiener continued fraction attack
8. '''
9. frac = ContinuedFractions.rational\_to\_contfrac(e, n)
10. convergents = ContinuedFractions.convergents\_from\_contfrac(frac)
12. **for** (k,d) **in** convergents:
13. #check if d is actually the key
14. **if** k!=0 **and** (e\*d-1)%k == 0:
15. phi = (e\*d-1)//k
16. s = n - phi + 1
17. # check if the equation x^2 - s\*x + n = 0
18. # has integer roots
19. discr = s\*s - 4\*n
20. **if**(discr>=0):
21. t = Arithmetic.is\_perfect\_square(discr)
22. **if** t!=-1 **and** (s+t)%2==0:
23. **print**("\nHacked!")
24. **return** d
26. **def** main():
27. e=354611102441307572056572181827925899198345350228753730931089393275463916544456626894245415096107834465778409532373187125318554614722599301791528916212839368121066035541008808261534500586023652767712271625785204280964688004680328300124849680477105302519377370092578107827116821391826210972320377614967547827619
28. n=460657813884289609896372056585544172485318117026246263899744329237492701820627219556007788200590119136173895989001382151536006853823326382892363143604314518686388786002989248800814861248595075326277099645338694977097459168530898776007293695728101976069423971696524237755227187061418202849911479124793990722597
29. **print** "e="
30. **print** e
31. **print** "n="
32. **print** n
33. d=hack\_RSA(e,n)
34. **print** "d="
35. **print** d
37. **if** \_\_name\_\_ == '\_\_main\_\_':
38. main()

结果：



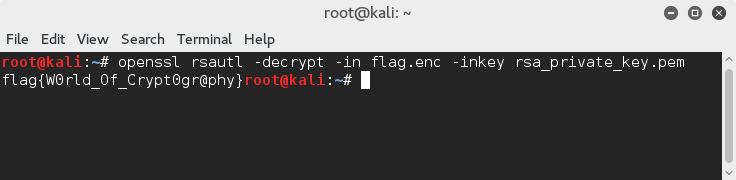
我们得到了私钥d，且知道了e，那我们就可以使用[rsatool](https://github.com/ius/rsatool)来生产私钥文件：

1. rsatool.py -e 354611102441307572056572181827925899198345350228753730931089393275463916544456626894245415096107834465778409532373187125318554614722599301791528916212839368121066035541008808261534500586023652767712271625785204280964688004680328300124849680477105302519377370092578107827116821391826210972320377614967547827619 -n 460657813884289609896372056585544172485318117026246263899744329237492701820627219556007788200590119136173895989001382151536006853823326382892363143604314518686388786002989248800814861248595075326277099645338694977097459168530898776007293695728101976069423971696524237755227187061418202849911479124793990722597 -d 8264667972294275017293339772371783322168822149471976834221082393409363691895 -o rsa\_private\_key.pem -f PEM



得到rsa\_private\_key.pem，于是我们利用OpenSSL对flag.enc解密：

1. openssl rsautl -decrypt -**in** flag.enc -inkey rsa\_private\_key.pem



最终得到flag：flag{W0rld\_Of\_Crypt0gr@phy}