

COURSE TITLE: CEF 506 - PYTHON/PERL PROGRAMMING DEVELOPMENT

ASSIGNMENT

NAME	MATRICULATION NUMBER
FRU ISIDORE CHE	FE12A078
THEOPHILUS WABA NASALI	FE12A183
NKENG NEWTON	FE12A140
MOBA MELVIS RINGNYU	FE12A107
ENOMBE THIERRY EWANE	FE12A053
ALANGI DERRICK	FE12A

A REPORT OF A PYTHON FILE CRACKER

INTRODUCTION

Many at times, businesses, friends, and family will lock their important files, documents etc. with a password to keep them secure. There are many tools available that can be used to crack these files. Some of them are Hash cat, Hydra etc. With the existence of these tools, reinventing the wheel is not an option but at least a good understanding of how this tools operate must be gotten and well mastered.

There are many different types of password attacks to name a few;

- * Dictionary Attack
- * Brute Force
- * Phishing
- * Social engineering etc.

In developing the password cracker for a file which in this case is a zipped file, a program was developed to randomly generate the words which the dictionary will contain, this is called the Dictionary attack. With this algorithm, a file which could be called a dictionary contains the different possible words which could be the password of the file. Also, a cracking algorithm that reads words in the dictionary one-by-one and try to open the zip file with the words and in this case, if a particular word opens the file, it's returned as the password of that file.

This algorithm is successful, if the as many words as possible are generated and it's also good to consider the length of the password because if the length of the words in the dictionary are shorter than that of the password, it will not be possible to ever get the password. In this example we generated words between the length of 2 and 12 and a total of at least 10,000,000 words to increase the probability of getting the password.

Using the Dictionary attack, the alphabet of this format for the content of the dictionary.

Alphabet = {digits + alphabets + punctuation}

HARDWARE SPECIFICATION

This application was developed and run on a machine with the following hardware requirement. It could be run on a machine with better specification

RAM: 3.00GB

Processor: AMD Phenom™ II N620 Dual-Core Processor

Processor speed: 2.80GHz

Hard drive: 1TB

Mark: Hewlett Packard (HP) ProBook 6455b

SOFTWARE SPECIFICATION

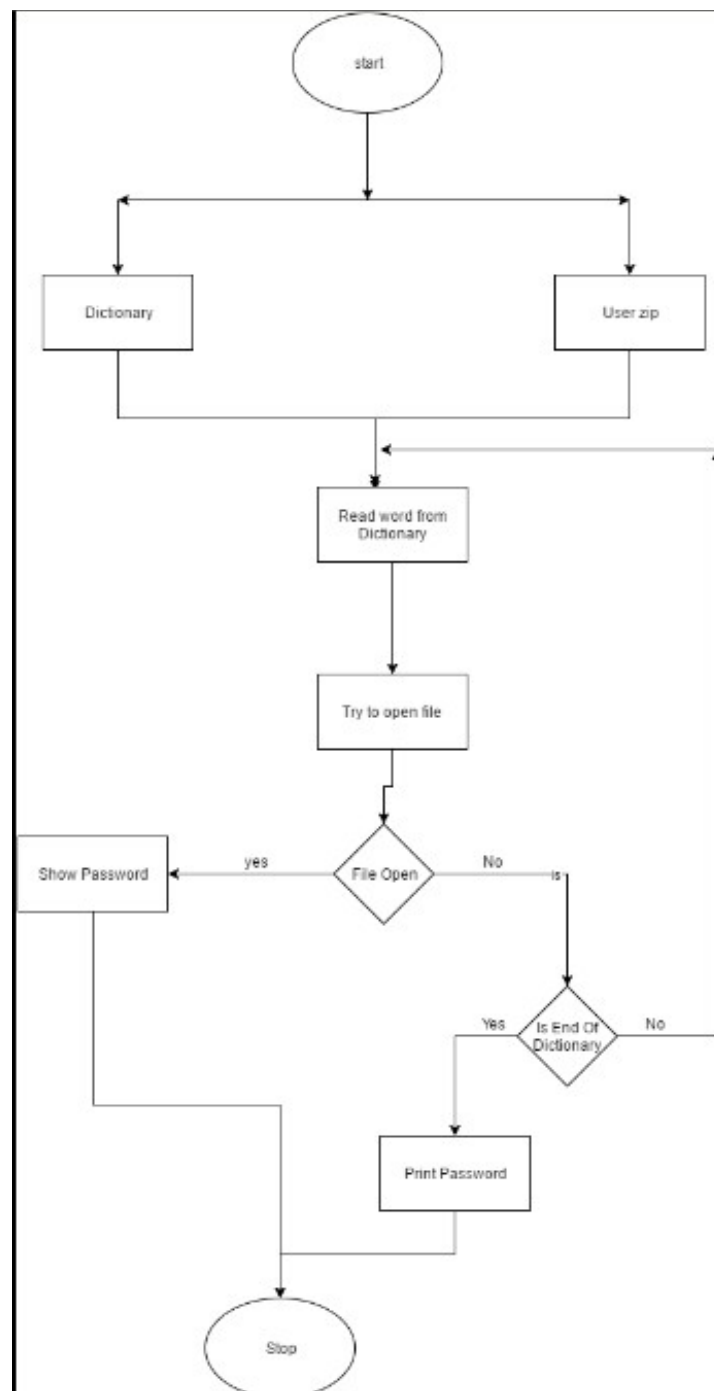
The following software used:

OS: Ubuntu 14.04

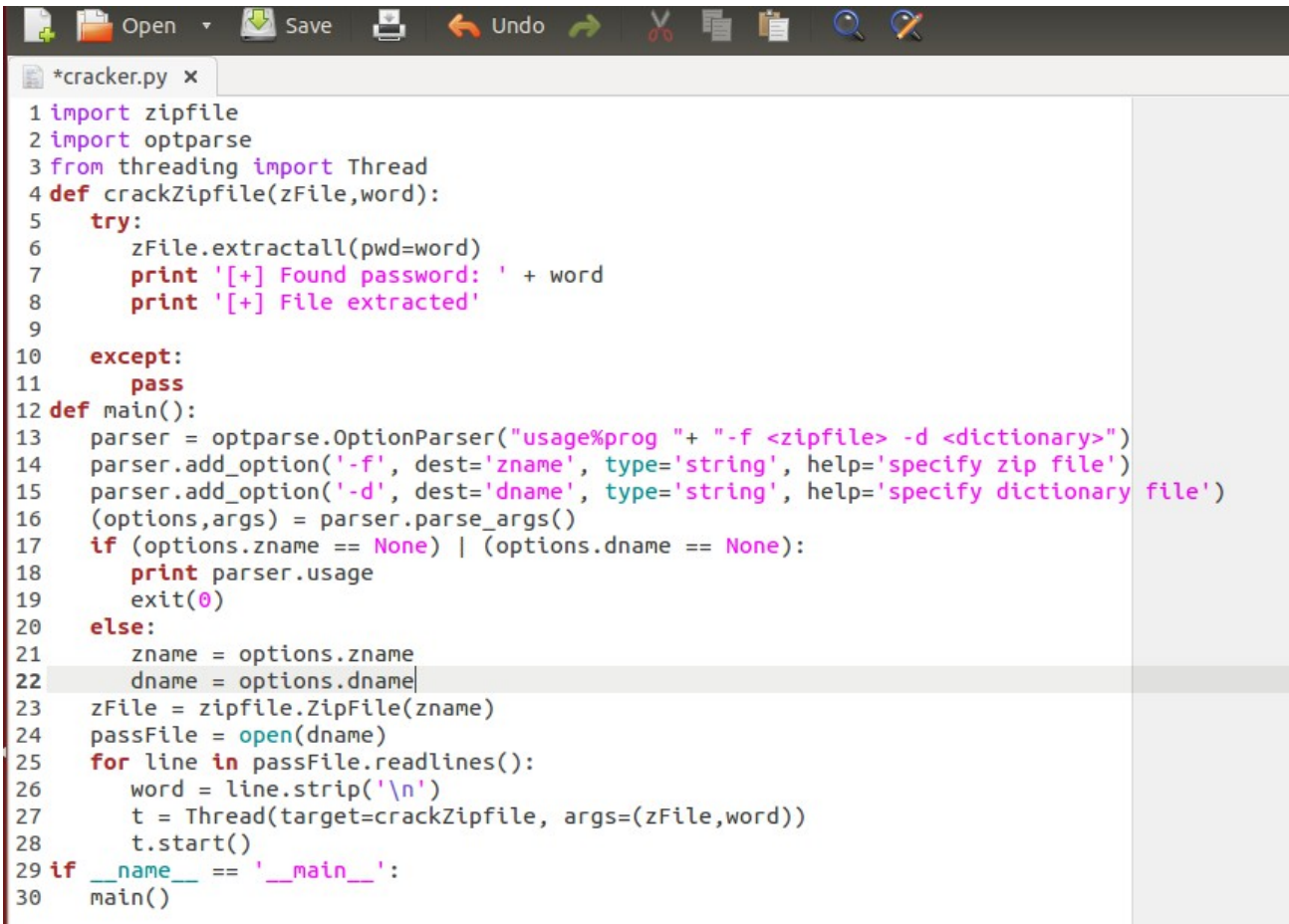
editor: Gedit

command-line: Ubuntu terminal

FLOWCHART

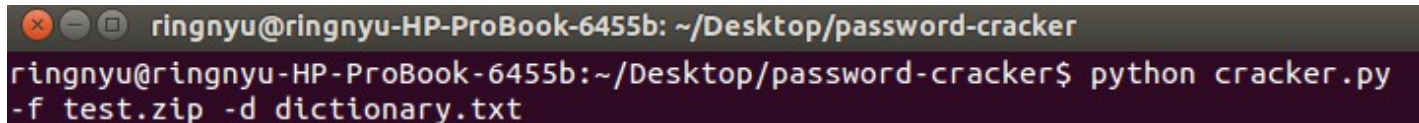


CODE

A screenshot of a code editor window titled '*cracker.py x'. The editor has a menu bar with 'Open', 'Save', 'Undo', and other icons. The code is written in Python and is as follows:

```
1 import zipfile
2 import optparse
3 from threading import Thread
4 def crackZipfile(zFile,word):
5     try:
6         zFile.extractall(pwd=word)
7         print '[+] Found password: ' + word
8         print '[+] File extracted'
9     except:
10         pass
11
12 def main():
13     parser = optparse.OptionParser("usage%prog " + "-f <zipfile> -d <dictionary>")
14     parser.add_option('-f', dest='zname', type='string', help='specify zip file')
15     parser.add_option('-d', dest='dname', type='string', help='specify dictionary file')
16     (options,args) = parser.parse_args()
17     if (options.zname == None) | (options.dname == None):
18         print parser.usage
19         exit(0)
20     else:
21         zname = options.zname
22         dname = options.dname
23     zFile = zipfile.ZipFile(zname)
24     passFile = open(dname)
25     for line in passFile.readlines():
26         word = line.strip('\n')
27         t = Thread(target=crackZipfile, args=(zFile,word))
28         t.start()
29 if __name__ == '__main__':
30     main()
```

HOW TO RUN

A screenshot of a terminal window. The prompt is 'ringnyu@ringnyu-HP-ProBook-6455b: ~/Desktop/password-cracker'. The command entered is 'python cracker.py -f test.zip -d dictionary.txt'.

```
ringnyu@ringnyu-HP-ProBook-6455b: ~/Desktop/password-cracker
ringnyu@ringnyu-HP-ProBook-6455b:~/Desktop/password-cracker$ python cracker.py
-f test.zip -d dictionary.txt
```

OUTPUT

```
ringnyu@ringnyu-HP-ProBook-6455b: ~/Desktop/password-cracker
ringnyu@ringnyu-HP-ProBook-6455b:~/Desktop/password-cracker$ python cracker.py
-f test.zip -d dictionary.txt
[+] Found password: toor
[+] File extracted
```