INF344 2019–2020

Description | Submission view

Password Cracking

Disponible à partir de: Wednesday 24 April 2019, 12:15
Due date: Saturday 2 May 2020, 23:59
Requested files: crack.py (Download)
Nombre maximal de fichiers: 20
Type of work: Individual work

This lab

The goal of this lab is to crack the passwords of John Doe using different standard methods. This lab has to be done inside Telecom Paris network. If you are not in the school, you need to install a VPN by following the instructions given on Eole: https://eole.telecom-paris.fr/vos-services/services-numeriques/connexions-aux-reseaux#exterieur.

To do the lab, first, download the file crack.py. For each type of password, implement the corresponding function to try out all possible passwords of this type. Call the method `check_password` with each of these passwords. If a correct password is found, the method automatically registers the password on our server. Once you have completed one type of password, you can comment out the corresponding function to make your program run faster: Once you have found a password, you do not have to guess it again.

You can run the code either in the Moodle or locally on your computer. For the latter, download the file **crack.py** and the file at http://137.194.211.71:5000/passwords/[YOUR_NAME_LOWERCASE].enc (your name should be formatted as in http://137.194.211.71:5000/found). For the final submission, you have to submit your file **crack.py** in the Moodle.

In any case, you must put your name in **crack.py** line 14 in place of "**YOUR NAME HERE**". If you are not sure how to format it (e.g. composed names), it must match your name at http://137.194.211.71:5000/found.

Scores appear in real-time on http://137.194.211.71:5000/. On this website, you can also find the list of the passwords you have already found.

The deadline is today at the end of the lab!

One last advice: Some code you write can be reused in other functions. Read first all questions and write reusable code (in general, reusable code is better).

Tasks

Brute force attacks

Brute force attacks are the most simple and stupid possible attacks: one just tries all possible combinations of a set of characters and hope one finds the password. For example, the attacker will try *a*, then *b*, then *aa*, *ab*, *ba*, *bb*, *aaa*, ...

1. Via brute-force, find passwords with up to 9 digits by filling **self.bruteforce_digits()**. Write in the comments of this function why longer passwords are harder to guess using brute-force methods [4 passwords/2 points]
2. Via brute-force, find passwords with up to 5 letters, mixed case, by filling **self.bruteforce_letters()**. Write in the comments of this function why passwords with more different characters are harder to guess using brute-force methods [4 passwords/2 points]

Dictionary attacks

Dictionary attacks consist of trying all passwords in a predefined list. We are going to implement here some attacks using standard password lists.

1. Often, people use the same passwords. They consist in general of simple word or combinations of successive keys of the keyboard (like azerty). Find the list of the 10k most common passwords on the Web and try all of them by filling the function **self.dictionary_passwords()**. Note that you can upload new files in the editor of Moodle. [4 passwords/1 point]
2. To make passwords make complex to guess but easy to remember, some people use leet transformations. They consist of changing a character by another graphically similar one. For example, common transformations are: e -> 3, l -> 1, a -> @, i -> 1, o -> 0. Reuse the 10k most

common passwords list you found in the previous exercise, but with the leet transformation. Fill the function **self.dictionary_passwords_leet()** [4 passwords/2 points]

3. Another popular list of passwords is the 20k most common English words (lowercase) with randomly added hyphens (not more than three hyphens, e.g., h-e-ll-o). "Randomly" does not mean you have to generate passwords in a random way! Your password set is fixed beforehand. Guess them by filling the function **self.dictionary_words_hyphen()** [4 passwords/2 points]
4. One can also use the 20k most common English words (lowercase, considering only words of more than 6 letters) concatenated to have the minimum length of 8, concatenated with a two-digit number. Guess them by filling the function **self.dictionary_words_digits()**. [4 passwords/2 points]
5. Try the 10k most common French words with randomly removed diacritics(e.g.: é,è -> e, à -> a, ç -> c, ù -> u) by filling the function **self.dictionary_words_diacritics()**. [4 passwords/2 points]
6. Diceware passwords are passwords created by concatenating several random words. Find the diceware passwords of length up to 4 of African capital cities in English (lowercase and connected with hyphens) by filling the function **self.dictionary_city_diceware()**. If a capital name is in two words, we remove the space (Addis Ababa -> addisababa) [4 passwords/ 2 points]

Social engineering attacks

Social engineering attacks are not focused on technical implementation but human interactions. This kind of attack is undoubtedly the most common ``hacking'' technique. Hackers use people vulnerabilities to guess or extract passwords.

1. In October 2013, one of John Doe's password: "Prom3theus", was leaked in the Adobe security breach. Guess its Google account password (apply transformations that were used in previous tasks) by filling the function **self.social_google()**. [2 passwords/2 point]
2. Use the following information about John Doe to guess passwords (concatenate the words in lowercase without using any connectors): "John Doe (born April 25, 1978, in Shaker Heights, Ohio) is living in New York. He is a fan of Cleveland Indians, but his Sunday evenings, he likes to spend reading Shakespeare works.". Fill the method **self.social_jdoe()**. [4 passwords/2 points]
3. When hackers penetrate password databases or get passwords directly from people, they sometimes put them online in what is called a *paste*. Fortunately, some website reference such leaks and you can find if your account is compromised. Go to https://haveibeenpwned.com/ and check if one of your passwords leaked. We are now going to have a look at a real-world example. Check if the email *bryant.rivera@hotmail.com* is compromised. If so, find his password by doing a Google search. Note that you can force Google to search for an exact sequence of characters by surrounding them with " (e.g., "We are launching our project on natural language processing and knowledge bases with 4 industrial partners. We are hiring 4 PhD students and 3 engineers or postdocs!" appears on only one website). Fill the function **self.paste()**. [1 password/1 point]

Requested files

crack.py

```python
# Written with <3 by Julien Romero

import hashlib
from sys import argv
import sys
if (sys.version_info > (3, 0)):
    from urllib.request import urlopen
    from urllib.parse import urlencode
else:
    from urllib2 import urlopen
    from urllib import urlencode


NAME = "YOUR NAME HERE".lower()
# This is the correct location on the moodle
ENCFILE = "../passwords2020/" + NAME + ".enc"
# If you run the script on your computer: uncomment and fill the following
# line. Do not forget to comment this line again when you submit your code
# on the moodle.
# ENCFILE = "PATH TO YOUR ENC FILE"



class Crack:
    """Crack The general method used to crack the passwords"""

    def __init__(self, filename, name):
        """__init__
        Initialize the cracking session
        :param filename: The file with the encrypted passwords
        :param name: Your name
        :return: Nothing
        """
        self.name = name.lower()
        self.passwords = get_passwords(filename)

    def check_password(self, password):
        """check_password
        Checks if the password is correct
        !! This method should not be modified !!
        :param password: A string representing the password
        :return: Whether the password is correct or not
        """
        password = str(password)
        cond = False
        if (sys.version_info > (3, 0)):
            cond = hashlib.md5(bytes(password, "utf-8")).hexdigest() in \
                self.passwords
        else:
            cond = hashlib.md5(bytearray(password)).hexdigest() in \
                self.passwords
        if cond:
            args = {"name": self.name,
                    "password": password}
            args = urlencode(args, "utf-8")
            page = urlopen('http://137.194.211.71:5000/' +
                                        'submit?' + args)
            if b'True' in page.read():
                print("You found the password: " + password)
                return True
        return False

    def crack(self):
        """crack
        Cracks the passwords. YOUR CODE GOES BELOW.

        We suggest you use one function per question. Once a password is found,
        it is memorized by the server, thus you can comment the call to the
        corresponding function once you find all the corresponding passwords.
        """
        self.bruteforce_digits()
        self.bruteforce_letters()

        self.dictionary_passwords()
        self.dictionary_passwords_leet()
        self.dictionary_words_hyphen()
        self.dictionary_words_digits()
        self.dictionary_words_diacritics()
        self.dictionary_city_diceware()

        self.social_google()
        self.social_jdoe()
        self.paste()

    def bruteforce_digits(self):
        pass

    def bruteforce_letters(self):
        pass

    def dictionary_passwords(self):
        pass

    def dictionary_passwords_leet(self):
        pass

    def dictionary_words_hyphen(self):
        pass

    def dictionary_words_digits(self):
        pass

    def dictionary_words_diacritics(self):
        pass

    def dictionary_city_diceware(self):
        pass

    def social_google(self):
        pass

    def social_jdoe(self):
        pass

    def paste(self):
        pass


def get_passwords(filename):
    """get_passwords
    Get the passwords from a file
    :param filename: The name of the file which stores the passwords
    :return: The set of passwords
    """
    passwords = set()
```

```
126         with open(filename, "r") as f:
127             for line in f:
128                 passwords.add(line.strip())
129         return passwords
130
131
132
133
134
135     if __name__ == "__main__":
136         # First argument is the password file, the second your name
137         crack = Crack(ENCFILE, NAME)
```

VPL

◄ Programmation Web côté serveur

Aller à...

Sujet du TP 2 ►