Second Homework Cybersecurity

Andrea Morelli 1845525

1 First Ideas

After having decrypted the zip file with the password given during the lesson i started thinking about the time that my computer could take if i bruteforced the 56 bits key used by the des cipher(56 effective key bits and 8 parity bits).I instantly knew that 2^{56} was a number of iterations too big to be analyzed.
I then was curious about the way the cipher was generated and noticed in the notes that was used a key derivation function to produce a key coming from a password (pbkdf2).
I than thought that the only thing that i could try to bruteforce was the password and not the key because only if the key was very basic i could have found the plaintext.
Bruteforcing a key derivated by a key derivation function is more expensive in terms of computer resources for the single iteration but the only way for my computer to find the plaintext was to bruteforce a password and not the key.
I therefore started to build a python script that could use openssl,utilizing the cmd without external libraries,to implement a dictionary attack to discover the plaintext.

2 Realizing the idea in a script

I built the script which would load a dictionary in the form of a txt file and,one word at a time would use it to decrypt the ciphertext using openssl via the windows command line interface.
To find if the plaintext decripted is in fact the correct one ,i calculated the percentage of letters inside the plaintext.
If the percentage is superior to the 70 percent i would stop the computation and return the decrypted plaintext and the password which generated the correct key.

3 Trials and Errors

I had to correct the script several times to obtain the right one and i had to try different wordlist to find the correct one.
I firstly tried the list of most used 100k english words without any success after like 30 minutes.
I then tried the list of the 100k most used italian words without achieving any successes even this time.
I then tried to utilize lists of most used passwords: i firstly tried the famous rockyou but in the light version without finding anything and after a lot of tries and like 6 hours of pc finding passwords at 4 ghz i found a wordlist of 280k passwords where the script wrote the correct solution with the password used to generate a key : boo
The plaintext is in the github :

`https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt`

4 Final results

The plaintext was found using the password boo that was converted with pbkdf2 and used to decrypt with des.

The command to decrypt all the words in the wordlists was:

```
openssl enc -des-cbc -d -pass pass:'pw' -pbkdf2 -in outfile.txt.enc
```

The password is : boo.

To decrypt with the found password related to the des key use :

```
openssl enc -des-cbc -d -pass pass:boo -pbkdf2 -in outfile.txt.enc
```

The plaintext deciphered is:

Ooh, your kisses, sweeter than honey
And guess what? So is my money
All I want you to do for me, is give it to me when you get home

5 Final script source code

```
import os
dizionario_path="./rn.txt"
#dizionario utilizzato che trova la password
#https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt
lettere="qwertyuiopasdfghjklzxcvbnm"
#funzione per caricare tutte le words del dizionario
def load_dizionario(path):
    d=[]
    with open(path,"r")as f:
        for i in f:
            d.append(i.strip())
    print(d)
    return d
#funzione per decriptare il ciphertext con la password pw
def attaccoAlDizionario(pw,u):
    #chiamo la funzione di openssl per decriptare con la password pw
    #il cifrario des in cbc, la key derivation function e butto gli output su out.txt
    #e gli errori in err.txt
    os.system('cmd /c openssl enc -des-cbc -d -pass pass:'+pw+' -pbkdf2 -in outfile.txt.enc
    > out.txt 2>err.txt')
    try:
        #apro il file con la frase decifrata come file binario
        with open("out.txt","rb") as f:
            s=""
            #leggo nella stringa s il file binario convertendolo in utf8
            byte=f.read(1)
            while(byte):
                s+=byte.decode("utf-8",errors="ignore")
                byte=f.read(1)
            #calcolo la percentuale di lettere presenti nella frase decifrata
            percentualelettere=0
            for i in s:
                if i.lower() in lettere:
                    percentualelettere+=1
            percentualelettere/=len(s)
            print(u,pw)
            #se la percentuale di lettere presenti è maggiore del 70%
```

```
            #allora considero la frase come decifrata
            if percentualelettere>0.7:
                print(pw+"trovato")
                print(s)
                return (True,pw,s)
    except Exception as e:
        print("Oops!", e.__class__, "occurred.")
    return (False,0,0)

if __name__ == '__main__':
    #carico il dizionario
    d=load_dizionario(dizionario_path)
    u=0
    #per tutte le parole nel dizionario provo a decriptaree se trovo un match stoppo
    for i in d:
        u+=1
        a=attaccoAlDizionario(i,u)
        if(a[0]==True):
            print(a[1],a[2])
            break
    #se trovo un match verra scritto nel file out insieme alla key
    #ed anche nella stream di output
    print("finished")
```