Third Homework Cybersecurity

Andrea Morelli 1845525

1 First Ideas

The basic idea was to load a passlist from the internet and with a script try all the passwords and see if i could find a valid plaintext.
I started utilizing a python script i wrote for the second homework utilizing openssl in the cmd and modifying the system call to openssl.
The command i was using to decrypt the plaintext was openssl enc -aes-256-cbc -d -pass pass:pw -in outfile.txt.enc.
I started then loading txt dicitonaries and trying all the passwords inside the passlist.

2 Trials and Errors

The problem with this approach was that my script using system calls was slow compared to other tools that where specifically built to crack passwords, so i had to limit the maximum dimension of the dictionaries loaded otherwise it would take so much time.
I utilized a lot of dictionaries especially those of the most frequent passwords used without finding any results even if they were composed of millions of passwords.
I then started to modify the wordlist changing uppercase to lowercase and viceversa the letters of the passwords without finding results.
As suggested i started to change also o to 0 or i to 1 without finding anything.
The problem was that with my pc and with the tool i build i could try like 20 passwords in a second and i was wasting hours of pc resources without achieving any results.
I then found a tool in github specifically built to do dictionary attack on aes ciphertext called bruteforce-salted-openssl.
To utilize the software i had to install ubuntu in dual boot with windows and with a simple sudo apt install bruteforce-salted-openssl i installed the software without compiling with specific compilers.
At this point i started downlaoding big dictionaries like Weakpass3.7 or crackstation or rockyou that were all composed of billions of passwords and with the speed of this software i could easily finish them in hours.
I started the software with the command :
bruteforce-salted-openssl -v 2 -c aes256cbc -1 -d sha256 -f ”dictpath” outfile.txt.enc
In this way the tool would warn me every 2 seconds the number of password he tried, would use the cipher aes in cbc with 256 bits key,would stop once it found a plaitext of 90percent+ ascii characters,whould use the sha hash to create the first initialization vector for the cbc,would use the dictionary in the path dictpath to decript the file outfile.txt.
After tring all those dictionaries all with an unzipped dimension of 20+ gb i still couldnt find the right password even thought i at this point let my pc run for multiple days at over 4 ghz of frequencies with multiple cores.

3 Final results

After a lot of time spent trying to find the password i thought i could be doing something not in the right way and i started documenting myself.
I discovered than the default digest to initialize the vector for the cbc encryption would vary based on the openssl version.
After a certain version of openssl the default digest would be sha256 but before this specific version the

default would be md5.
At this point i retried the weak pass dictionary of like 32 gb with the same command as above but with -d md5 (digest md5).
After like 30 minutes it found the password : the0rem.
At this point with openssl i decrypted the ciphertext with openssl enc -d -aes-256-cbc -md md5 -in outfile.txt.enc and inserting the password i found the plaintext:

The House of Representatives shall be composed of Members chosen every second Year by the People of the several States, and the Electors in each State shall have the Qualifications requisite for Electors of the most numerous Branch of the State Legislature.

4 Final script source code

The first script i wrote was :

```
import os
dizionario_path="./weakpass.txt"
lettere="qwertyuiopasdfghjklzxcvbnm"
lettere={i for i in lettere}
inizio=0

#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[inizio:]
#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 -aes-256-cbc -d -pass pass:'+pw+'
    -in outfile.txt.enc > o.txt 2>e.txt')
    try:
        #apro il file con la frase decifrata come file binario
        with open("o.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=inizio
    #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")
```

The final program i used with the mds5 digest was :
https://github.com/glv2/bruteforce-salted-openssl
The final dictionary that found the password was weakpass3.7 with an unzipped dimension of 30gb.