A2
SEGURETAT A LES APLICACIONS

Aitor Sánchez Salgado

DAM2 2024/25

Contingut:

Xifrat d'usuaris

- Implementa la funció encrypt_data per xifrar la informació dels usuaris.(2p)

```
def encrypt_data(data, output_file):
    ret = gpg.encrypt(data, symmetric = True, recipients=None, passphrase=os.environ.get('PassWord'))
    if(os.path.exists("users.txt")):
        with open(output_file,'a') as file:
            file.write(str(ret)+"////")
    else:
        with open(output_file,'w') as file:
            file.write(str(ret)+"////")
```

- Implementa la funció decrypt_data per desencriptar la informació.(2p)

```
def decrypt_data(input_file):
    lst = []
    if(os.path.exists("users.txt")):
        with open(input_file,'r') as file:
            splited = file.read().split("////")
            for s in splited:
                lst.append(str(gpg.decrypt(s, passphrase=os.environ.get('PassWord'))).strip())
    else:
        print("l'arxiu no existeix")

    return lst
```

- Podràs fer servir qualsevol algorisme de xifrat (simètric o asimètric) amb gnupg, però hauràs de justificar per què has escollit aquest algorisme i per què és adequat per a l'escenari de la pràctica. (1p)

He utilitzat un algorisme de xifrat simètric en el que es necessita una passPhrase, ja que, al no saber qui utilitzarà l'aplicació, no hi hauria cap manera d'utilitzar la clau pública/privada i el recipient correcte cada vegada, ja que no tothom tindrà el necessari per a utilitzar-les.

Una manera per a tenir guardada la passPhrase i que no sigui visible, seria tenir-la com a variable d'entorn i que python la demanés a través de la base de dades al servidor on es guarden totes les dades.

Per a provar el funcionament, he fet que al iniciar el programa, es creï o subscribeixi una variable d'entorn de manera local la qual agafo quan vaig a encriptar o desencriptar.

```
os.environ['PassWord'] = "totallySavePassPhrase"
```

Gestió d'usuaris

- Implementa la funció save_user per guardar les credencials dels usuaris en un arxiu, utilitzant xifrat.

```
def save_user(username, password):
    encrypt_data(f"{username};{password}","users.txt")
```

- Implementa la funció verify_user per verificar l'usuari i contrasenya proporcionats durant el procés de login.

```
def verify_user(username, password):
    list = []
    list = decrypt_data("users.txt")
    if list.__contains__(f"{username};{password}"):
        return True
    return False
```

Arxiu d'usuaris

- Implementa un mecanisme per generar un arxiu que emmagatzemi els usuaris registrats de forma segura (xifrat).

L'arxiu generat és un arxiu on es guarden la clau i l'usuari junts estant encriptades. Després, es separen entre elles amb "////" per a poder-les separar i tractar a l'hora de desencriptar-ho després.