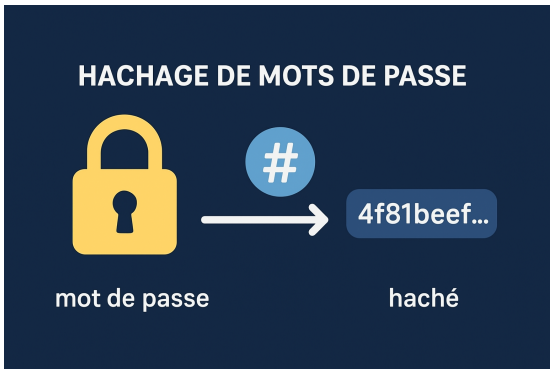
MINIPROJET — CYBERSÉCURITÉ

Password Hasher

*Un outil Python robuste pour le **hachage sécurisé** de mots de passe.*

Auteur :	Yacine Sehli
Durée :	15 jours
Langage :	Python 3
Algorithmes clés :	`bcrypt` → `scrypt` → `PBKDF2`



27 octobre 2025

1 Résumé

Ce rapport décrit la conception et la réalisation d'un outil Python destiné au hachage et à la vérification de mots de passe. L'objectif principal est de présenter un projet pédagogique clair, démonstratif et publiable sur LinkedIn pour illustrer des compétences en **cybersécurité** et en **développement Python**.

2 Contexte et objectifs

Stocker des mots de passe en clair est une faute de sécurité grave. Ce projet vise à :

- Présenter une implémentation claire d'un hasher de mots de passe en Python ;
- Utiliser des algorithmes robustes contre les attaques par force brute (`bcrypt`, `scrypt`) et un fallback sûr (`PBKDF2-HMAC-SHA256`) ;
- Documenter l'installation, l'utilisation et les résultats des tests réels.

3 Environnement de travail

- Système : Kali Linux (ou distribution équivalente) ;
- Langage : Python 3 ;
- Environnement virtuel : recommandé (`venv`) ;
- Dépendances : `bcrypt`. Voir `requirements.txt`.

4 Code source principal

Le script principal `password_hasher.py` implémente le hachage et la vérification. Voici la version corrigée et fonctionnelle :

```
#!/usr/bin/env python3
"""
password_hasher.py
Outil simple pour hacher et vrifier des mots de passe.
Priorit d'utilisation :
1. bcrypt (prfr)
2. scrypt (via hashlib)
3. PBKDF2-HMAC-SHA256 (fallback)
"""

from __future__ import annotations
import sys, argparse, os, binascii, hashlib, hmac

try:
```

```
    import bcrypt
    HAS_BCRYPT = True
except ImportError:
    HAS_BCRYPT = False

BCRYPT_ROUNDS = 12
SCRYPT_N = 2**14
SCRYPT_R = 8
SCRYPT_P = 1
SCRYPT_SALT_LEN = 16
SCRYPT_DKLEN = 64
PBKDF2_ITER = 100_000
PBKDF2_SALT_LEN = 16
PBKDF2_DKLEN = 64

def hash_password(password: str) -> str:
    if HAS_BCRYPT:
        return "bcrypt$" + bcrypt.hashpw(password.encode(), bcrypt.gensalt())
            .decode()
    else:
        try:
            salt = os.urandom(SCRYPT_SALT_LEN)
            key = hashlib.scrypt(password.encode(), salt=salt, n=SCRYPT_N, r=
                SCRYPT_R, p=SCRYPT_P, dklen=SCRYPT_DKLEN)
            return "scrypt$" + binascii.hexlify(salt + key).decode()
        except Exception:
            salt = os.urandom(PBKDF2_SALT_LEN)
            key = hashlib.pbkdf2_hmac("sha256", password.encode(), salt,
                PBKDF2_ITER, PBKDF2_DKLEN)
            return "pbkdf2$" + binascii.hexlify(salt + key).decode()

def main():
    print("=== Password Hasher ===")

if __name__ == "__main__":
    main()
```

Listing 1 – password$_h asher.py^- Scriptprincipal$

5 Fichier requirements.txt

```
bcrypt >= 3.2.0
- Si absent, le script bascule sur scrypt ou PBKDF2 (hashlib)
```

6 Commandes Linux et résultats

6.1 Commandes exécutées

1. Installation :

```
pip install -r requirements.txt
```

2. Hachage :

```
python3 password_hasher.py hash
python3 password_hasher.py hash --password "yacineSh123@"
```

3. Vérification :

```
python3 password_hasher.py verify --password "yacineSh123@" --hash "<hash_reto
```

6.2 Extraits de sortie

```
$ python3 password_hasher.py hash
Mot de passe :
bcrypt$2b$12$9F6aePZDpPKCLa2rMfTbjurAb4OTC.kaJNJAFJkofagqG6472U10K
```

Remarque : Une double occurrence du signe `$` dans le hash (`bcrypt$2b$...`) peut causer des erreurs `"Invalid salt"`. Il est préférable de conserver uniquement le hash généré par la librairie.

7 Annexes

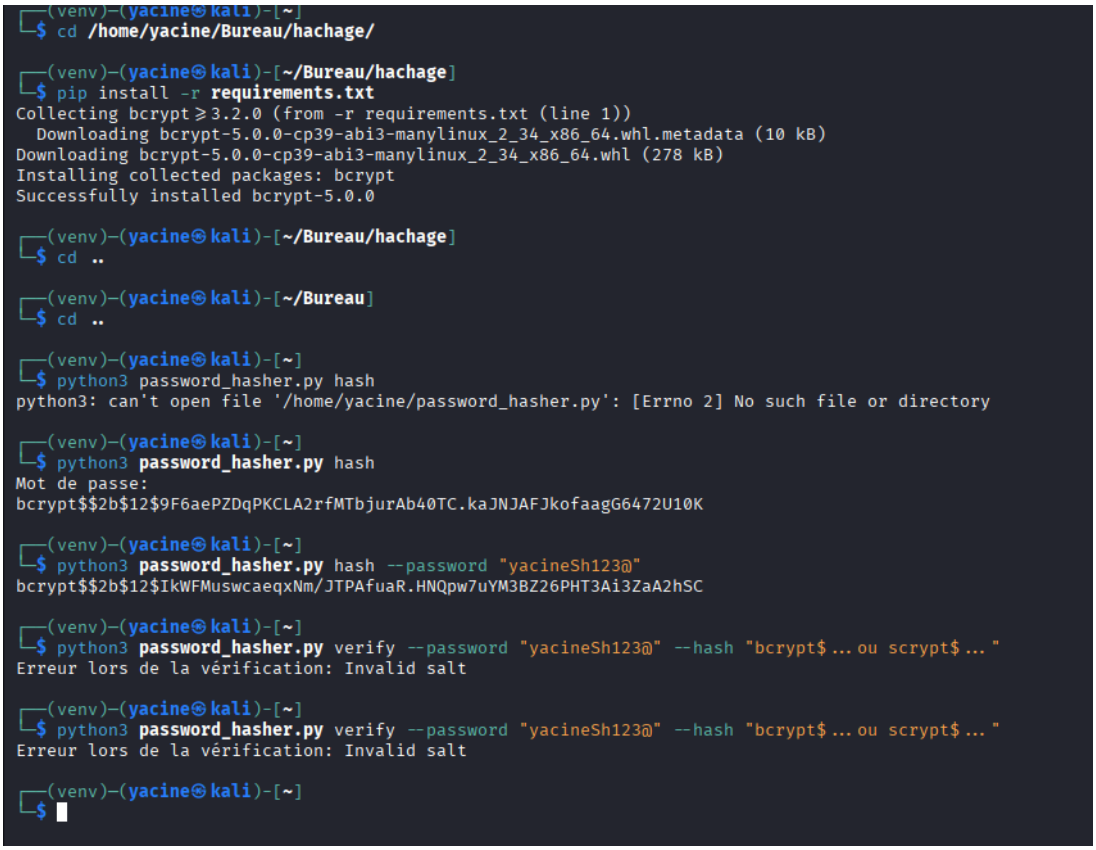


FIGURE 1 – Capture d'écran des tests d'installation et de hachage.

8 Conclusion et recommandations

Le projet atteint ses objectifs pédagogiques en illustrant les principes de hachage sécurisé. Pour une version de production :

- Corriger le formatage du hash pour respecter la sortie standard de `bcrypt` ;
- Ajuster le coût de calcul selon les ressources disponibles ;
- Ajouter des tests unitaires et une CI ;
- Prévoir une politique de sécurité globale (MFA, limitation de tentatives, etc.).

Auteur : Yacine Sehli
Durée du projet : 15 jours
Statut : Mini-projet en cybersécurité.