Correction - Exercice Type Partiel : Python pour la Cybersécurité

Partie A - Scanner TCP/UDP

```
import socket

def scan_tcp(ip, ports):
    print("[*] Debut du scan TCP...")
    ouverts = 0
    for port in ports:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.settimeout(1)
            sock.connect((ip, port))
            print(f"[TCP] Port {port} ouvert")
            ouverts += 1
            sock.close()
        except:
            pass
    print("-" * 30)
    print(f"Resume : {ouverts} ports TCP ouverts sur {len(ports)} testes.\n")

def scan_udp(ip, ports):
    print("[*] Debut du scan UDP...")
    repondus = 0
    for port in ports:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            sock.settimeout(1)
            sock.sendto(b"ping", (ip, port))
            data, _ = sock.recvfrom(1024)
            print(f"[UDP] Port {port} a repondu")
            repondus += 1
        except socket.timeout:
            print(f"[UDP] Port {port} silencieux")
        except:
            pass
    print("-" * 30)
    print(f"Resume : {repondus} ports UDP ont repondu sur {len(ports)} testes.\n")
```

Explication : utilisation de socket pour TCP et UDP, avec timeout. Affichage clair et comptage des resultats.

Partie B - Dictionnaire SHA256

```
import hashlib

def crack_sha256(hash_cible, fichier_dico):
    print("[*] Debut du craquage SHA256...")
    with open(fichier_dico, 'r') as f:
        for ligne in f:
            mot = ligne.strip()
            hash_test = hashlib.sha256(mot.encode()).hexdigest()
```

```
            if hash_test == hash_cible:
                print(f"[+] Mot de passe trouve : {mot}")
                return mot
    print("[-] Aucun mot de passe trouve dans ce dictionnaire.")
    return None
```

Explication : lecture ligne par ligne, strip et hashage SHA256 pour comparaison avec la cible.

Partie C - Obfuscation simple dans une image

```
def append_data(image_path, message):
    with open(image_path, "ab") as f:
        f.write(b"##HIDDEN##" + message.encode())
    print(f"[+] Donnee cachee dans {image_path}")

def extract_data(image_path):
    with open(image_path, "rb") as f:
        contenu = f.read()
        idx = contenu.find(b"##HIDDEN##")
        if idx != -1:
            message = contenu[idx + 10:].decode()
            print(f"[+] Donnee extraite : {message}")
        else:
            print("[-] Aucune donnee cachee trouvee.")
```

Explication : ajout du message a la fin du fichier en binaire, recuperation avec marqueur identifiable.