Exercice Type – Partiel Python pour la Cybersécurité

Durée estimée : 45–60 min

Points :

- **Partie A (20 pts)** : Scan TCP et UDP
- **Partie B (15 pts)** : Attaque par dictionnaire sur un hash SHA256
- **Partie C (5 pts)** : Obfuscation simple d'un message dans une image

---

🔹 Partie A – Scanner TCP/UDP (20 points)

Contexte : On veut scanner les ports ouverts d'une machine cible. Aucun outil externe ne doit être utilisé.

Objectif :

1. Écrire une fonction `scan_tcp(ip, ports)` qui :
    - Prend une IP cible et une **liste de ports**
    - Pour chaque port, tente une connexion TCP
    - Affiche `[TCP] Port X ouvert` si la connexion réussit
2. Écrire une fonction `scan_udp(ip, ports)` qui :
    - Envoie un paquet UDP (ex: `b"ping"`) à chaque port
    - Affiche `[UDP] Port X silencieux` si pas de réponse
    - Affiche `[UDP] Port X a répondu` si réponse reçue

Contraintes :

- Utiliser uniquement le module `socket`

- Timeout maximum de 1 seconde par port
- Afficher un résumé à la fin (nb ports ouverts/silencieux)

🔹 Partie B – Dictionnaire SHA256 (15 points)

Contexte : On a récupéré un hash SHA256. On souhaite le "casser" par dictionnaire.

Objectif :

1. On vous donne le hash suivant (correspond à un mot dans le fichier `wordlist.txt`) :

java
CopierModifier
```
Hash cible = 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd66
d61cdd826b9b94d (c'est le hash de "password")
```

2. Écrire une fonction `crack_sha256(hash_cible, dictionnaire)` qui :
    - Prend le hash cible et un chemin de fichier
    - Teste chaque mot du fichier en le hashant (SHA256)
    - Affiche le mot trouvé si le hash correspond

Contraintes :

- Utiliser `hashlib`
- Lire les lignes du fichier proprement (sans `\n`)

🔹 Partie C – Obfuscation basique dans une image (5 points)

Contexte : On veut cacher un message texte à la fin d'un fichier image JPEG.

Objectif :

1. On vous fournit un fichier `image.jpg` et un message (ex : `"infiltration=ok"`)
2. Écrire une fonction `append_data(image_path, message)` qui :
    - Ouvre le fichier image en mode binaire `ab`
    - Ajoute le message à la fin du fichier (avec une signature ex: `##HIDDEN##`)
    - Affiche confirmation
3. Écrire une fonction `extract_data(image_path)` qui :
    - Récupère le message après `##HIDDEN##` à la lecture

Contraintes :

- Utiliser seulement `open()` en binaire
- Pas de bibliothèques d'image

✅ Bonus

- Nettoyer l'affichage avec une ligne de séparation (`print("-" * 30)`) entre les ports
- Ajouter un `--help` déclenché par `sys.argv` pour documenter rapidement votre script