DVIL SECURE — Documentation Utilisateur

Introduction Générale

Bienvenue sur la plateforme DVIL Secure !
Ici, tu vas te mettre dans la peau d'un **pentester** pour apprendre à repérer, exploiter et documenter des failles classiques dans une API web connectée à une serrure intelligente.
Chaque "Niveau" correspond à une ou plusieurs vulnérabilités : à toi de les découvrir, de comprendre le fonctionnement de l'API, et d'atteindre l'objectif final (déverrouillage).

Approche Générale d'un Challenge

1. **Reconnaissance**
    I. Explore l'API (Postman, curl, Burp, navigateur…)
    II. Identifie les endpoints accessibles sans login
2. **Test des comportements**
    I. Essaie plusieurs méthodes HTTP (GET, POST, etc.)
    II. Observe les réponses (codes HTTP, messages, JSON…)
    III. Cherche des comportements "anormaux" ou des endpoints sensibles non protégés
3. **Exploitation**
    - Tente l'exploitation de la faille repérée
    - Valide l'effet obtenu (exemple : serrure déverrouillée)

Conseil : Garde trace de tes tests et captures pour le rapport !

Niveau 1 — Route Non Protégée

- ### a) Introduction

Dans ce premier challenge, ton but est d'exploiter une **route d'API critique** laissée sans authentification, permettant de déverrouiller la serrure **sans identifiants**.

- ### b) Approche

1. Recherche les endpoints (ex : /v1/unlock_public)
2. Tente une requête POST sur cette route (curl ou Postman)
3. Observe la réponse de l'API et le changement d'état de la serrure

- ### c) Vulnérabilité

  - **Type** : Absence totale de contrôle d'accès (Broken Authentication)
  - **Objectif** : Déclencher une action critique sans être authentifié

- ### d) Exemple

```
curl -X POST http://<ip>:<port>/v1/unlock_public
```

La serrure doit s'ouvrir même sans compte : c'est la faille !

Niveau 2 — XSS & Path Traversal

- ### a) Introduction

Ce challenge contient **deux failles à exploiter dans l'ordre** pour déverrouiller la serrure :

  - **XSS (Cross-Site Scripting)**
  - **Path Traversal**

- ### b) Approche

1. **XSS**
   - Cherche un endpoint où tu peux injecter du texte (ex : /v2/xss)
   - Tente d'injecter un script (<script>alert('XSS');</script>)
   - Vérifie l'exécution du code dans la page (alerte affichée)

2. **Path Traversal**

- Identifie une route vulnérable (ex : /v2/traversal?file=...)
- Essaie de lire un fichier (test.txt) dans le dossier /uploads/ ou de sortir du dossier via ../
    - L'API doit te retourner le contenu du fichier

- **c) Enchaînement et Objectif**

- Pour que /unlock fonctionne, il faut réussir XSS **et** Path Traversal (sessions "done")
- Appelle /unlock en POST pour publier le message MQTT de déverrouillage

- **d) Exemple de script d'exploitation**

```
import requests
BASE_URL = "http://<ip>:<port>"
# 1) XSS
r1 = requests.post(f"{BASE_URL}/v2/xss",
data={"comment":"<script>alert('XSS');</script>"})
# 2) Path Traversal
r2 = requests.get(f"{BASE_URL}/v2/traversal",
params={"file":"test.txt"})
# 3) Unlock
r3 = requests.post(f"{BASE_URL}/unlock")
```

Niveau 3 — SSTI, Pickle, SQL Injection

- **a) Introduction**

Dans ce niveau, il faut **enchaîner trois failles** pour déverrouiller la serrure :

1. **SSTI** (Server-Side Template Injection)
2. **Désérialisation Pickle non sécurisée**
3. **Injection SQL**

- **b) Approche par étape**

1. **Reconnaissance** : Liste tous les endpoints vulnérables (/vuln/template, /vuln/deserialize, /vuln/serrure)
2. **SSTI (Jinja2)** :
    I. Envoie une chaîne malicieuse via le paramètre q sur /vuln/template
    II. Objectif : Récupérer le SECRET_KEY ou toute autre variable sensible
    III. Ex : {{ config.__class__.__init__.__globals__['os'].popen('cat .env').read() }}

3. **Désérialisation Pickle**
    I. Crée un objet Python malicieux (Pickle) pour définir une variable globale (OVERRIDE_TOKEN)
    II. Envoie-le sur /vuln/deserialize
4. **Injection SQL**
    I. Utilise le token récupéré pour exploiter /vuln/serrure?token=...&update=...
    II. Injecte une valeur pour forcer state = 'unlock' (ex : unlock' --)

- ## c) Objectif final

- Le but est de faire passer la serrure à l'état "unlock" via ces trois failles, prouvant ainsi l'exploitation en chaîne

- # 6. Conseils & Bonnes Pratiques

✓ **Garde une trace de tes tests** : captures, scripts, retours d'API
✓ **Reste dans le cadre pédagogique** : n'attaque pas l'infra ou les comptes hors scope
✓ **Analyse les logs** pour vérifier l'état de la serrure et l'impact de tes actions
✓ **Prends le temps d'analyser** chaque étape avant d'automatiser tout le processus

- # 7. Objectif & Remarques

Ton objectif : **exploiter chaque niveau en respectant la démarche offensive et en documentant tes résultats** (payloads utilisés, étapes, captures d'écran).
Chaque challenge te prépare à des situations réelles de pentest sur API/IoT.

Bonne chance, et pense à réinitialiser l'environnement après exploitation pour les autres !