DVIL SECURE — Documentation Administrateur

1. Introduction & Contexte

But : Présenter le système, ses usages et la notion de “challenges” par niveaux.

Le projet DVIL Secure est une plateforme pédagogique dédiée à l’expérimentation de la sécurité IoT et web via des API RESTful. Elle permet à la fois un usage réel (contrôle d’une serrure connectée sécurisée) et la mise en évidence de failles courantes (challenges par niveaux), dans le but de former les administrateurs à la détection, l’exploitation et la remédiation de vulnérabilités typiques.

2. Authentification et JWT

But : Décrire les mécanismes de connexion, rôles, gestion du JWT.

Exemple de rédaction :

L’API repose sur une authentification forte basée sur JWT (JSON Web Token).

- Connexion : L’utilisateur soumet ses identifiants via /login. Un token JWT est généré, signé et stocké en cookie HTTPOnly.
- Rôles :
  - User : Peut consulter l’état de la serrure et la contrôler.
  - Admin : Accès complet (gestion utilisateurs, logs, panel admin).
- Déconnexion : Le token JWT est supprimé via /logout.

3. Routes Principales

- Authentification

- GET /login : Affiche la page de connexion.
- POST /login : Authentifie et délivre le JWT.
- POST /logout : Déconnecte l'utilisateur.
- GET /user/status : Retourne les infos de l'utilisateur connecté.
- ➢ Contrôle de la Serrure
- GET /status : Récupère l'état actuel de la serrure.
- POST /unlock : Déverrouille la serrure via MQTT.
- POST /lock : Verrouille la serrure via MQTT.
- c) Interface Administrateur
- GET /admin : Accès au panneau d'administration.
- GET /admin/users : Liste les utilisateurs.
- GET /admin/logs : Affichage des logs temps réel.
- POST /admin/users/add : Ajoute un utilisateur.
- PUT /admin/users/update : Modifie un utilisateur.
- DELETE /admin/users/delete : Supprime un utilisateur.
- POST /admin/reset_challenge : Réinitialise la serrure.

- ## 4. Base de Données

But : Montrer la structure, usage, sécurité.

Exemple de rédaction :

La base MySQL/MariaDB gère :

- Les utilisateurs (authentification, rôle, PIN, badge NFC).
- L'état de la serrure (locked/unlocked).

Les migrations sont automatisées (Flask-Migrate) pour garantir la cohérence et la maintenabilité du schéma.

(Insère ici le tableau complet récapitulatif des tables/colonnes fourni plus haut.)

- ## 5. Communication via MQTT

But : Expliquer l'interfaçage temps réel avec le hardware.

Exemple :

L'API communique avec le matériel (serrure connectée) via le protocole MQTT :

- Topic : DVIL/Serrure
- Utilisation : Publier les ordres de verrouillage/déverrouillage et recevoir le statut matériel.

6. Logs et Surveillance

But : Montrer le monitoring, la traçabilité.

Exemple :

Toutes les actions critiques sont loguées (serrure.log, etc).
Un thread Python diffuse ces logs en temps réel via Socket.IO, permettant à l'admin de surveiller l'état du système et de détecter tout comportement anormal.

7. Utilisation & Maintenance

But : Donner les consignes opérationnelles, bonnes pratiques de gestion.

Exemple :

- Utiliser l'interface admin pour gérer les utilisateurs et les logs.
- Effectuer des sauvegardes régulières de la base de données.
- Surveiller le système via le panel pour réagir rapidement en cas d'incident.

8. Mesures de Sécurité & Bonnes Pratiques

But : Lister ce qui protège le système, et donner les réflexes à adopter.

Exemple :

- Authentification JWT & cookies HTTPOnly.
- Validation stricte de toutes les entrées utilisateur.
- Utilisation de requêtes paramétrées (SQLAlchemy).
- Sécurisation HTTP (Flask-Talisman), limitation du débit (Flask-Limiter).
- Audits réguliers & formation continue des équipes.

Section dédiée Challenges (Niveau 1, 2, 3)

- Challenge Niveau 1 — Route non protégée
- Contexte : Présence d'un endpoint critique accessible sans authentification.

- Vulnérabilité : "Broken Authentication" — absence de contrôle d'accès sur /v1/unlock_public.
- Exploitation : Toute personne peut envoyer une requête POST et déverrouiller la serrure.
- Impact : Prise de contrôle totale, faille critique.
- Contremesures : Mettre un contrôle d'accès systématique (JWT, clé API…).
- Bonnes pratiques : Audit de toutes les routes, protection par défaut, monitoring.

- Challenge Niveau 2 — XSS & Path Traversal

- Contexte : Endpoints vulnérables à l'injection XSS (ex : /v2/xss) et à l'accès fichiers (/v2/traversal).
- Exploitation XSS : Injection de code JS dans les commentaires, exécution dans le navigateur d'un admin.
- Exploitation Path Traversal : Lecture de fichiers arbitraires via manipulation du paramètre file.
- Impact : Vol de session, accès à des fichiers confidentiels, escalade d'attaque.
- Contremesures : Validation/échappement systématique, CSP, restriction des accès fichiers.

- Challenge Niveau 3 — SSTI, Pickle, SQLi

- Contexte : Routes exposant plusieurs failles enchaînables.
- Failles :
    - SSTI (ex. /vuln/template): Injection de code Jinja2, extraction de la clé secrète.
    - Désérialisation Pickle : Modification de variables globales (ex : token d'admin).
    - SQL Injection : Altération ou extraction directe de données sensibles (/vuln/serrure).
- Impact : Exécution de code arbitraire, prise de contrôle serveur, fuite de données.
- Contremesures : Utiliser JSON au lieu de Pickle, requêtes paramétrées, validation rigoureuse.

Conclusion

Cette documentation doit te permettre d'exploiter, de maintenir et de sécuriser le système DVIL Secure en toute autonomie. Les exemples de challenges illustrent l'importance de l'audit régulier, de la défense en profondeur et de la vigilance dans la gestion des accès et des entrées utilisateur.