

Modèle de Menaces STRIDE

Toolbox Pentest Automatisée

Version: 1.0

Date: Janvier 2026

Auteur: Ingénieur Sécurité Opérationnelle

Client: Société de Cybersécurité Offensive

1. Contexte et Objectifs

1.1 Description du Système

La toolbox est une application d'automatisation de tests de pénétration (pentest) destinée à une société de cybersécurité offensive. Elle permettra de standardiser et d'accélérer les pentests manuels en intégrant des outils open-source et propriétaires.

Objectifs principaux :

- Réduire le temps de pentest de 40 % via l'automatisation
- Standardiser les pratiques internes
- Proposer une interface simple exploitable par des analystes
- Générer des rapports standardisés et traçables
- Sécuriser l'outil lui-même et les données qu'il manipule

1.2 Architecture de Haut Niveau

L'architecture de la toolbox est une application web FastAPI standard sans conteneurisation :

- **Interface Web Frontend** : Jinja2 + HTML/CSS avec authentification JWT
- **API Backend** : FastAPI pour l'orchestration des modules de scan
- **Cache et Orchestration** : Redis + Celery pour les files d'attente de scans
- **Stockage de Données** : MariaDB pour les résultats et configurations
- **Modules de Scan** : Scripts Python appelant Nmap, OpenVAS, SQLmap via subprocess
- **Déploiement** : WSL (Windows Subsystem for Linux) en environnement de développement
- **Journalisation** : Système d'audit centralisé pour la traçabilité
- **Rapports** : Génération PDF/HTML avec signatures numériques.

2. Périmètre et Limites

2.1 Éléments Inclus

- API Backend : FastAPI avec Pydantic pour validation des entrées
- Interface Web : Jinja2, HTML/CSS avec authentification JWT
- Modules de Scan : Scripts Python appelant outils locaux (Nmap, OpenVAS, SQLmap)
- Stockage des Données : MariaDB pour résultats et configuration
- Cache et Orchestration : Redis + Celery pour files de scans
- Système de Génération de Rapports : PDF/HTML/CSV
- Journaux d'Audit : Traçabilité complète (Qui ? Quoi ? Quand ?)
- Authentification et Autorisation : JWT, RBAC (Admin / Pentesteur / Analyste)
- Chiffrement : Fernet pour secrets et données sensibles
- Déploiement : WSL avec gestion des permissions Windows/Linux

2.2 Éléments Hors Périmètre

- Infrastructure réseau du client (pare-feu, segmentation réseau)
 - Sécurité physique du serveur de déploiement
 - Gestion des vulnérabilités des dépendances tierces
 - Incidents de sécurité liés aux outils intégrés eux-mêmes
 - Infrastructure de production au-delà de WSL (déploiement cloud)
-

3. Composants Critiques

1. **API Backend FastAPI** : Point d'entrée, gestion des permissions, orchestration des scans
2. **Interface Web Jinja2** : Rendu HTML, protection CSRF, gestion JWT
3. **Scripts Python de Scan** : Exécution d'outils via subprocess (Nmap, OpenVAS, SQLmap)
4. **Système d'Authentification** : Gestion JWT, sessions, RBAC
5. **Base de Données MariaDB** : Stockage des résultats sensibles, données utilisateurs
6. **Système de Journalisation/Audit** : Immuabilité et non-répudiation
7. **Cache Redis** : Tokens, files d'attente, données sensibles temporaires

4. Modèle de Menaces STRIDE

4.1 USURPATION D'IDENTITÉ (Spoofing)

S1 : JWT Compromis

Description : Un attaquant forge un token JWT pour usurper l'identité d'un utilisateur et accéder à l'API.

Composants affectés : API Backend, Système d'Authentification

Impact : Accès non autorisé aux fonctionnalités, usurpation de scans, vol de données sensibles

Atténuations clés :

- HTTPS obligatoire (TLS 1.3+) pour toutes les communications
- JWT signés avec clé secrète forte (RS256)
- Expiration courte des tokens (15-30 minutes)
- Tokens stockés en HttpOnly cookies (pas localStorage)
- Blacklist des tokens à la déconnexion (via Redis)
- Rotation des clés de signature tous les 90 jours

S3 : Compromission Compte Administrateur

Description : Un attaquant accède au compte admin par brute force, credential stuffing ou phishing.

Composants affectés : Interface Web, Base de données utilisateurs

Impact : Compromission totale du système, accès à tous les rapports, sabotage possible

Atténuations clés :

- Limitation des tentatives de connexion (5 essais / 15 minutes)
 - CAPTCHA après N tentatives échouées
 - Politique de mots de passe stricte (12+ caractères, complexité)
 - Alertes en temps réel sur tentatives échouées
 - Journal d'audit de tout accès administrateur
-

4.2 ALTÉRATION DE DONNÉES (Tampering)

T2 : Injection dans les Paramètres de Scan

Description : Un attaquant injecte des commandes malveillantes dans les paramètres de scan (ex: 127.0.0.1; rm -rf /).

Composants affectés : API Backend, Scripts Python, Système d'exploitation

Impact : Exécution de code arbitraire, destruction de données, compromission du système

Atténuations clés :

- Validation stricte des entrées avec Pydantic (liste blanche)
 - Utiliser `subprocess.run()` sans `shell=True` pour exécution
 - Comptes système dédiés avec permissions minimales pour chaque outil
 - Ulimits stricts sur CPU, mémoire, descripteurs de fichiers (`ulimit bash`)
 - Timeouts strictes sur exécution des scans (1 heure maximum)
 - Nettoyage des caractères spéciaux avant passage aux outils
-

4.4 DIVULGATION D'INFORMATION (Information Disclosure)

I1 : Vol de Secrets (Clés API, Mots de Passe)

Description : Secrets (clés API, identifiants DB) exposés en clair dans le code, Git ou journaux.

Composants affectés : API Backend, Dépôt Git, Fichiers de configuration

Impact : Accès non autorisé aux services externes, compromission de la base de données

Atténuations clés :

- Variables d'environnement pour tous les secrets (jamais hardcodé)
- Fichier `.env` dans `.gitignore` (jamais committer)
- Scan des dépôts Git avec `git-secrets` ou `TruffleHog`
- Rotation des secrets tous les 90 jours
- Masquage des secrets dans les journaux (rédaction)
- Principe du moindre privilège : tokens avec permissions minimales
- Audit des accès aux secrets (qui a lu, quand)

I2 : Exposition des Rapports PDF

Description : Rapports sensibles contenant vulnérabilités détectées accessibles sans authentification.

Composants affectés : API de téléchargement, Stockage des fichiers

Impact : Divulgarion de vulnérabilités clients, informations exploitables pour attaques, dommages réputationnels

Atténuations clés :

- Authentification obligatoire avant accès/téléchargement
 - Contrôle d'accès : un utilisateur ne voit que ses propres rapports
 - URLs opaques pour rapports (UUID, pas IDs séquentiels)
 - HTTPS + chiffrement des fichiers au repos (Fernet)
 - Journaux d'accès à chaque téléchargement
 - Limitation de débit sur les téléchargements
 - Suppression automatique des rapports après 30 jours si non-accédés
-

4.5 DÉNI DE SERVICE (Denial of Service)

D1 : DDoS / Saturation de l'API

Description : Attaquant envoie des milliers de requêtes pour saturer l'API FastAPI.

Composants affectés : API Backend, Serveur web

Impact : Indisponibilité du service, perte de productivité, SLA non respectés

Atténuations clés :

- Limitation de débit par utilisateur/IP (ex: 100 requêtes/minute)
- Algorithme token bucket pour rate limiting
- Étranglement sur points de terminaison critiques (/api/scan)
- WAF avec détection d'attaques (si déploiement production)
- Surveillance des pics de requêtes avec alertes
- Disjoncteur pour défaillance rapide en surcharge

D2 : Épuisement des Ressources Système

Description : Scans agressifs consomment trop de CPU/mémoire, rendant le système inutilisable.

Composants affectés : Scripts Python, Système d'exploitation

Impact : Crash des processus, indisponibilité du service, perte de données temporaires

Atténuations clés :

- Ulimits stricts sur CPU, RAM, descripteurs de fichiers (ulimit bash)
 - Quotas d'utilisation par utilisateur (ex: 5 scans/jour, 1h max par scan)
 - Surveillance des ressources avec alertes (>80% CPU)
 - Délais d'attente stricts sur exécution (1 heure maximum)
 - Priorisation des scans en file d'attente (pas FIFO aveugle)
 - Isolation par comptes système (chaque scan = UID différent)
-

4.6 ESCALADE DE PRIVILÈGES (Elevation of Privilege)

E1 : Escalade via RBAC Non-Appliqué

Description : Utilisateur modifie son JWT pour changer son rôle (analyste → admin).

Composants affectés : API Backend, Système d'Authentification

Impact : Accès à toutes les fonctionnalités administrateur, manipulation de tous les rapports, compromission totale

Atténuations clés :

- JWT signé avec clé secrète forte (RS256, jamais HS256 seul)
- Signature vérifiée côté serveur (jamais faire confiance au contenu du token)
- Rôles et permissions stockés en base de données MariaDB (source de vérité)
- Vérification des permissions côté serveur avant chaque action sensible
- RBAC : 3 rôles minimum (Admin, Pentesteur, Analyste, Auditeur)
- Audit de chaque changement de rôle avec horodatage
- Logs détaillés de chaque tentative d'escalade

E3 : Injection SQL via MariaDB

Description : Entrées utilisateur non validées permettent injection SQL pour accéder à toutes les données.

Composants affectés : API Backend, MariaDB

Impact : Accès à toutes les données sensibles, modification de données, escalade de privilèges possibles

Atténuations clés :

- Requêtes paramétrées (prepared statements) obligatoires
- Validation stricte des entrées avec Pydantic (liste blanche, types spécifiques)

- Utilisation d'ORM sécurisé (SQLAlchemy avec requêtes paramétrées)
 - Moindre privilège : compte MariaDB application avec permissions minimales
 - Pas de compte root utilisé par l'application
 - WAF avec signatures de détection injection SQL (en production)
 - Tests SAST (Static Application Security Testing) dans CI/CD
-

5. Résumé des Menaces (CVSS)

ID	Menace	CVSS	Priorité
S1	JWT compromis	7.5	HAUTE
S3	Compromission Admin	9.9	CRITIQUE
T2	Injection paramètres scan	9.9	CRITIQUE
I1	Vol secrets API	8.6	HAUTE
I2	Exposition rapports PDF	7.4	HAUTE
E1	Escalade RBAC	9.1	CRITIQUE
E3	Injection SQL	9.9	CRITIQUE
D1	DDoS/Saturation API	7.5	HAUTE
D2	Épuisement ressources	7.4	HAUTE
R1	Déni d'exécution scan	6.5	MOYENNE

Table 1: Matrice de risques CVSS

6. Plan d'Atténuation

6.1 Priorités Sprint 1 (Immédiat)

Ces atténuations doivent être implémentées AVANT le MVP :

1. JWT signé (RS256) + validation côté serveur
2. HTTPS obligatoire en production (TLS 1.3)
3. Validation stricte entrées avec Pydantic (liste blanche)
4. RBAC : 3 rôles (Admin, Pentesteur, Analyste)
5. Chiffrement des secrets via variables d'environnement (.env)
6. MariaDB : TLS + moindre privilège (compte dédié)

7. Redis : authentification (AUTH) + isolation réseau
8. Subprocess sans shell=True + ulimits stricts
9. Journaux d'audit immuables (horodatage + signature)

6.2 Priorités Sprint 2-3 (Court terme)

Ces atténuations enrichissent la sécurité post-MVP :

1. AMF obligatoire pour administrateurs (TOTP)
 2. Limitation de débit API (100 req/min par IP)
 3. Chiffrement des données sensibles au repos (Fernet)
 4. Scan des secrets dans Git CI (git-secrets, TruffleHog)
 5. Alertes en temps réel sur actions sensibles
 6. Tests SAST dans pipeline CI/CD
 7. Journaux centralisés avec signatures cryptographiques
 8. Filigrane + journaux d'accès pour rapports PDF
-

7. Considérations WSL et Déploiement

7.1 Risques Spécifiques WSL

- **Isolation incomplète** : Permissions Windows/Linux peuvent créer des failles
- **Système de fichiers** : WSL2 avec Ext4 = meilleures garanties que WSL1
- **Interopérabilité** : Scripts appelant exécutables Windows doivent valider strictement
- **Persistance données** : Sauvegardes régulières en dehors de WSL (C:)

7.2 Bonnes Pratiques WSL

- Exécuter MariaDB et Redis dans WSL (pas Windows natif)
 - Comptes Unix dédiés pour chaque processus critique
 - Permissions restrictives (umask 0077 pour fichiers sensibles)
 - Sauvegarde régulière de MariaDB vers système de fichiers Windows
 - Documentation de la procédure de déploiement en WSL2
-

8. Validation et Suivi

8.1 Checklist Avant Release

Avant chaque release, valider :

1. Tous les JWT validés côté serveur (signature + expiration)
2. HTTPS activé, HTTP désactivé en production
3. Toutes les entrées utilisateur validées avec Pydantic
4. AMF activée pour les administrateurs
5. RBAC testée pour chaque rôle
6. Secrets en variables d'environnement uniquement
7. MariaDB avec authentification TLS + compte limité
8. Redis avec AUTH + ACL configuré
9. Subprocess appelé sans shell=True
10. Ulimits et timeouts configurés
11. Limitation de débit active sur tous les endpoints critiques
12. Journaux d'audit complets et immuables
13. Messages d'erreur génériques en production
14. Alertes de sécurité configurées

8.2 Processus d'Examen

- **Mensuel** : Examen des incidents de sécurité et journaux d'audit
- **Trimestriel** : Audit de sécurité applicatif (SAST + tests manuels)
- **Semestriel** : Pentest externe ou audit de conformité
- **Annuel** : Révision complète du modèle de menaces

8.3 Responsabilités

Rôle	Responsabilités
Architecte Backend	JWT, RBAC, validation, chiffrement, SubProcess sécurisé
Infrastructure	TLS, MariaDB, Redis, WSL, logs, sauvegardes
SecOps (Vous)	Audit conformité, journaux, alertes, documentation
QA	Tests de sécurité, validation atténuations

Table 2: Répartition des responsabilités

Annexe A : Glossaire

JWT (JSON Web Token) - Jetons signés pour authentification sans état

RBAC - Contrôle d'accès basé sur les rôles

Pydantic - Validation de données en Python avec types stricts

Subprocess - Exécution de processus enfants en Python

Fernet - Chiffrement symétrique authentifié

SAST - Test statique de sécurité applicatif

Version : 1.0

Date de création : 18 janvier 2026

Dernière mise à jour : 23 janvier 2026

Prochaine révision : Avril 2026 (après Sprint 3)