HACKATHON AI4BMI

IFRI / UAC — Licence 3 Sécurité Informatique — 2025-2026

LIVRABLE L3

Contrôle d'Accès Basé sur les Rôles (RBAC)

Protection contre S3 — Fuite de données & S5 — Inférence API

Étudiant	Anselme
Établissement	IFRI / UAC — Cotonou, Bénin
Livrable	L3 — Contrôle d'accès RBAC
Date	23 Février 2026
Technologies	Python 3.13 · FastAPI · Casbin · JWT · bcrypt · SQLite · HTML/CSS/JS

SCÉNARIOS CONCERNÉS

● **Scénario 3 (S3) : Fuite de données sensibles**

Des hackers exploitent une vulnérabilité dans l'interface d'administration et téléchargent l'intégralité des historiques de maintenance. Ils revendent ces données à des concurrents qui apprennent précisément :

- quelles machines tombent le plus souvent en panne ;
- à quelle fréquence ;
- et quelles pièces de rechange sont utilisées.

● **Scénario 5 (S5) : Attaque par inférence**

En interrogeant intelligemment l'API, un attaquant déduit des informations confidentielles sur les cadences de production réelles, révélant que BMI fonctionne à 95% de sa capacité — information stratégique pour un concurrent préparant une Offre Publique d'Achat (OPA) hostile.

OBJECTIFS

Contrôle des accès : S3 & S5

- Matrice complète des permissions par rôle
- Code d'implémentation RBAC (Casbin + FastAPI + JWT)
- Captures d'écran de l'interface d'administration et des accès refusés
- Audit trail horodaté — journalisation de tous les accès

RÉALISATIONS

S3 — Fuite de Données Sensibles

ÉTAPE 1 — Présentation du Scénario S3

Dans l'usine BMI, plusieurs profils d'utilisateurs coexistent : administrateurs, ingénieurs de maintenance, opérateurs de production et auditeurs. Sans contrôle d'accès, un simple opérateur comme charlie peut accéder à des ressources confidentielles réservées aux administrateurs.

L'attaque S3 étape par étape

> charlie se connecte avec ses identifiants légitimes → il obtient un token JWT valide → il envoie GET /api/admin/users → sans RBAC, il récupère la liste complète des utilisateurs et leurs rôles → il tente GET /api/export → sans RBAC, il télécharge 1 547 lignes de données industrielles confidentielles. La fuite est silencieuse et indétectable.

Ce scénario représente une menace réelle : 68% des fuites de données proviennent d'utilisateurs internes ayant des accès excessifs (Verizon DBIR 2024).

ÉTAPE 2 — Outils, Scripts et Bibliothèques

Casbin — Moteur RBAC

- casbin (Python) — moteur de politique RBAC, interopérable avec model.conf et policy.csv
- config/model.conf — définit la structure logique RBAC : request, policy, role, matchers
- config/policy.csv — contient toutes les règles de permission par rôle
- enforcer.enforce(username, resource, action) — évalue chaque requête en < 5 ms

FastAPI + Middleware

- fastapi (Python) — framework API REST avec documentation Swagger automatique
- app/middleware/rbac.py — intercepte chaque requête, appelle Casbin, journalise le résultat

- require_permission(resource, action) — décorateur protégeant chaque endpoint en une ligne
- app/routes/api.py — routes protégées : /capteurs, /historiques, /predictions, /admin/users, /export

JWT + bcrypt

- python-jose — émission et vérification des tokens JWT HS256
- bcrypt — hachage des mots de passe avec salt automatique
- app/utils/auth.py — create_token() + decode_token() + hash_password() + verify_password()
- Token JWT contenant : {"sub": username, "role": role, "exp": +60min}

Base de données et Audit Trail

- SQLite + SQLAlchemy — table users (id, username, password_hash, role, active) et table audit_logs
- logs/audit.log — journal JSON horodaté de tous les accès (ALLOWED et DENIED)
- Double journalisation : SQLite pour l'interface admin + fichier JSON pour archivage légal

Extrait de code — app/middleware/rbac.py

```
# Chargement du moteur Casbin au démarrage
enforcer = casbin.Enforcer('config/model.conf', 'config/policy.csv')

# Décorateur de protection RBAC
def require_permission(resource: str, action: str):
    def dependency(request, token=Depends(decode_token),
db=Depends(get_db)):
        username = token["username"]
        role     = token["role"]
        allowed  = enforcer.enforce(username, resource, action)

        # Journalisation systématique (autorisé ET refusé)
        log_access(db, username, role, ip, resource, action,
                   "ALLOWED" if allowed else "DENIED", endpoint)

        if not allowed:
            raise HTTPException(status_code=403, detail={
                "error": "Accès refusé",
                "user": username, "role": role,
                "message": f"Le rôle '{role}' n'est pas autorisé"
            })
```

```
# Extrait config/policy.csv
p, admin,                  export,      read
p, admin,                  admin_panel, read
```

```
p, ingenieur_maintenance,  export,      read
p, operateur,              predictions, read
p, auditeur,               audit_logs,  read
g, charlie, operateur   # charlie ne peut PAS accéder à export
```

ÉTAPE 3 — Justification du Choix des Outils

Justification — Casbin

- **Casbin est le standard industriel RBAC en Python — politique séparée du code applicatif**
  - Modifier les droits = éditer policy.csv, sans redéployer l'application
  - Évaluation en mémoire < 5 ms, sans accès base de données à chaque requête
  - Supporte RBAC, ABAC, ACL — évolutif si les besoins de BMI changent

Justification — JWT HS256

- **Stateless : le rôle est embarqué dans le token chiffré, éliminant les sessions serveur**
  - Le middleware connaît le rôle de l'utilisateur sans interroger la base de données
  - Expiration 60 min limite la fenêtre d'exploitation d'un token intercepté
  - HS256 suffisant en environnement interne — RS256 recommandé en production multi-service

Justification — bcrypt

- **Algorithme recommandé par l'OWASP pour le stockage des mots de passe**
  - Salt automatique intégré — impossible d'utiliser des tables arc-en-ciel
  - Facteur de coût ajustable — résistance scalable avec l'évolution des processeurs
  - Contrairement à SHA-256, bcrypt est délibérément lent sur GPU → résiste au brute-force

Justification — FastAPI

- **Framework le plus performant pour les APIs REST Python (benchmarks TechEmpower)**
  - Documentation Swagger générée automatiquement — facilite les tests et la démonstration
  - Système de dépendances (Depends) permet d'injecter le RBAC en une seule ligne
  - Validation automatique des données avec Pydantic — protection contre les injections

ÉTAPE 4 — Démonstration S3 — Captures d'Écran

4.1 — Validation des politiques avant déploiement

Le fichier tests/test_casbin.py valide la politique RBAC sans démarrer le serveur. 10 scénarios couvrant tous les rôles sont testés automatiquement.

Figure 1 — Tests unitaires Casbin — 10/10 réussis

Le tableau montre 10 scénarios testés. charlie est refusé sur admin_panel (S3) et export (S3) mais autorisé sur predictions. alice est autorisée sur tout. La dernière ligne confirme "Tous les tests passent ! Politique RBAC correcte." — validation avant tout déploiement.

4.2 — Démarrage du serveur et interface Swagger

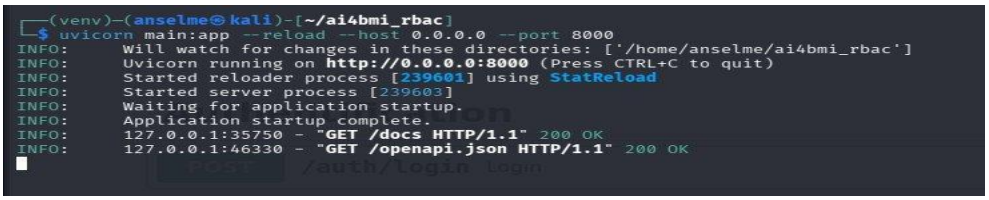

Figure 2 — Serveur Uvicorn opérationnel sur http://0.0.0.0:8000

Casbin charge model.conf et policy.csv, la base SQLite est initialisée, les 4 utilisateurs de démonstration sont créés. "Application startup complete" confirme que le système RBAC est opérationnel.

Figure 3 — Documentation Swagger — Endpoints protégés par RBAC

L'icône 🔒 sur chaque endpoint signifie qu'un token JWT valide est requis. Sans authentification, toute requête retourne 401 Unauthorized. Le bouton "Authorize" permet de tester avec différents comptes.

4.3 — Connexion de charlie (rôle : opérateur)

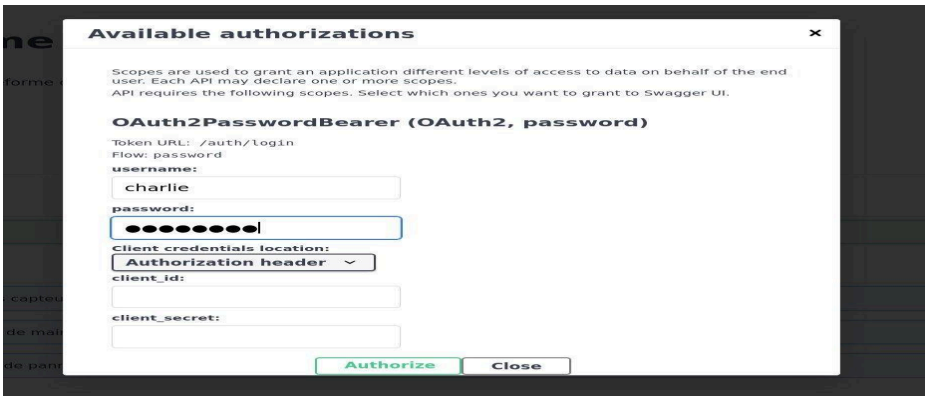

Figure 4 — charlie s'authentifie via Swagger

charlie saisit ses identifiants. Le serveur vérifie son mot de passe bcrypt et émet un token JWT contenant role: "operateur". Ce token valide sera utilisé pour simuler l'attaque S3.

4.4 — Tentative S3 : charlie → /admin/users-list — BLOQUÉ

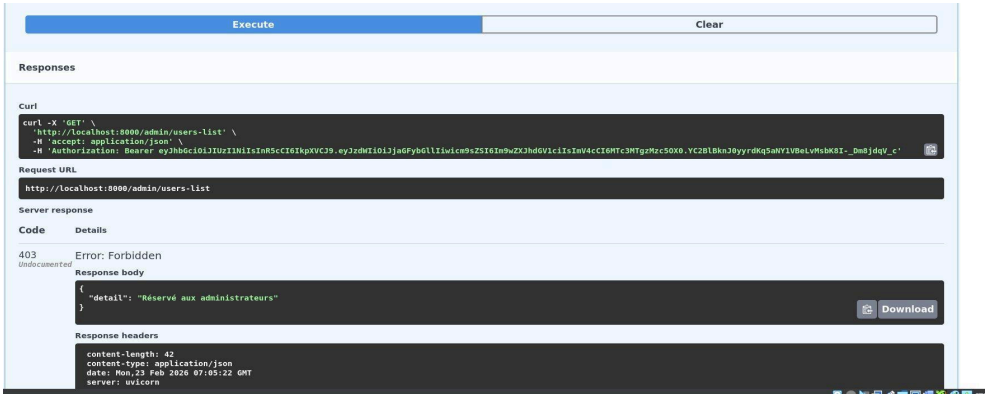

Figure 5 — 403 Forbidden — charlie bloqué sur la liste des utilisateurs

charlie tente d'accéder à /admin/users-list. Casbin évalue : enforce("charlie", "admin_panel", "read") → False. Aucune règle "p, operateur, admin_panel, read" n'existe dans policy.csv. Réponse : 403 Forbidden "Réservé aux administrateurs". La liste des utilisateurs n'est jamais transmise.

4.5 — Tentative S3 : charlie → /api/export — BLOQUÉ

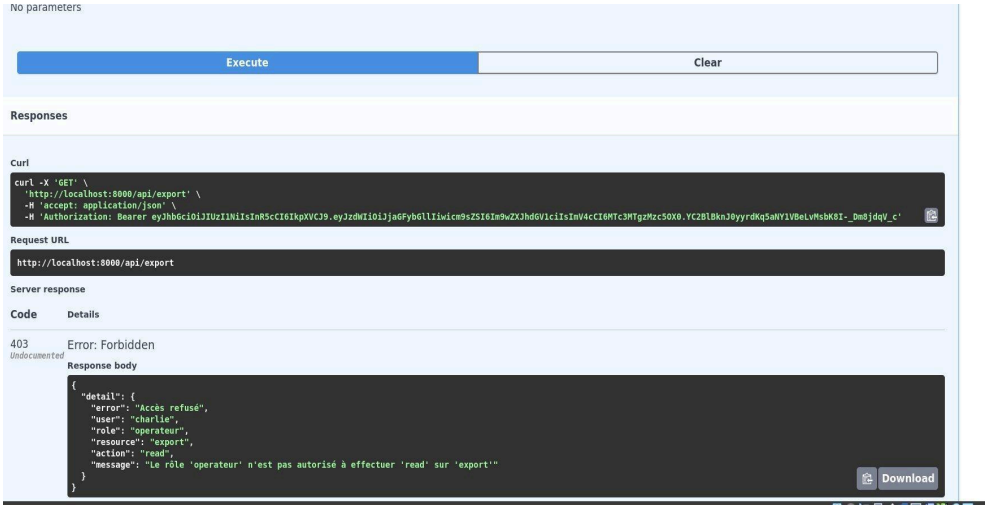

Figure 6 — 403 Forbidden — charlie bloqué sur l'export (1 547 enregistrements protégés)

charlie tente GET /api/export. Réponse 403 avec message JSON détaillé : "user": "charlie", "role": "operateur", "resource": "export", "action": "read", "message": "Le rôle 'operateur' n'est pas autorisé à effectuer 'read' sur 'export'". Les 1 547 lignes de données industrielles ne sont jamais transmises.

4.6 — Accès légitime : charlie → /api/predictions — AUTORISÉ

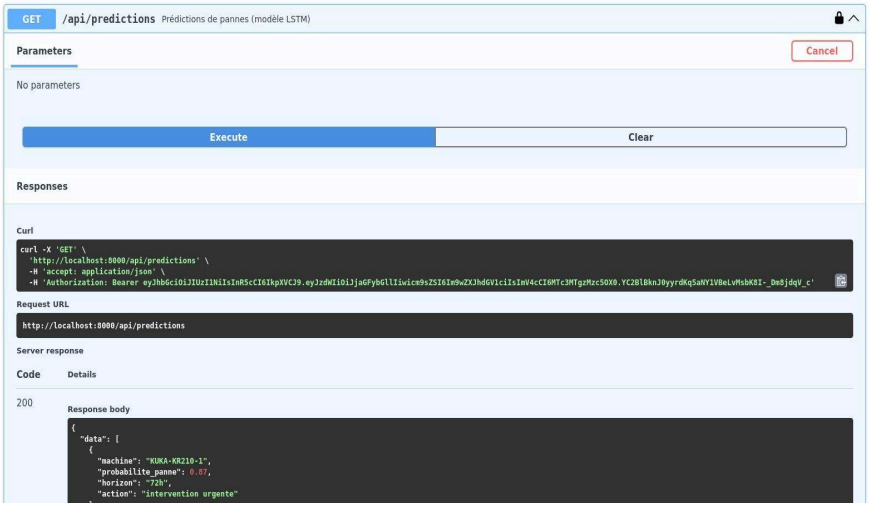

Figure 7 — 200 OK — charlie autorisé sur /api/predictions (principe du moindre privilège)

charlie accède normalement à /api/predictions — son seul droit légitime en tant qu'opérateur. 200 OK avec données KUKA-KR210-1 (probabilité panne 0.87, horizon 72h). Le RBAC autorise ce qui est légitime et bloque le reste.

4.7 — Logs en temps réel — preuves serveur

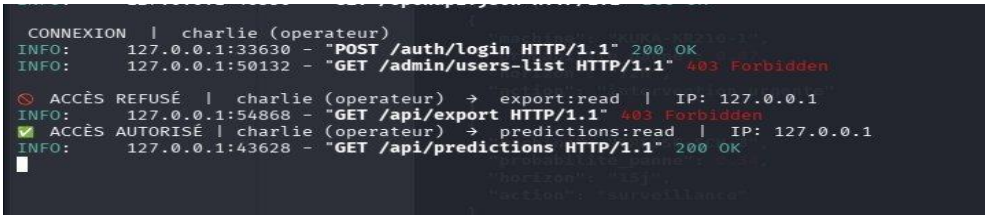

Figure 8 — Terminal serveur — Traces en temps réel des accès de charlie

"CONNEXION | charlie (operateur)" → connexion réussie. "403 Forbidden" sur /admin/users-list → S3 bloqué. "ACCÈS REFUSÉ | charlie → export:read" → S3 bloqué. "ACCÈS AUTORISÉ | charlie → predictions:read" → accès légitime. Chaque événement est horodaté et journalisé.

4.8 — Contraste : alice (admin) accède à toutes les ressources

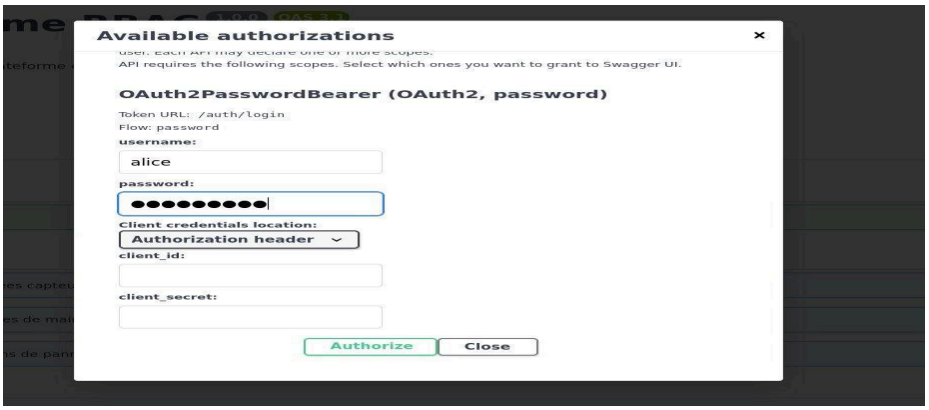

Figure 9 — alice (admin) s'authentifie — token role: "admin"

alice obtient un token JWT avec role: "admin". Casbin dispose de règles pour toutes les ressources. Le même endpoint qui retournait 403 à charlie retournera 200 à alice.

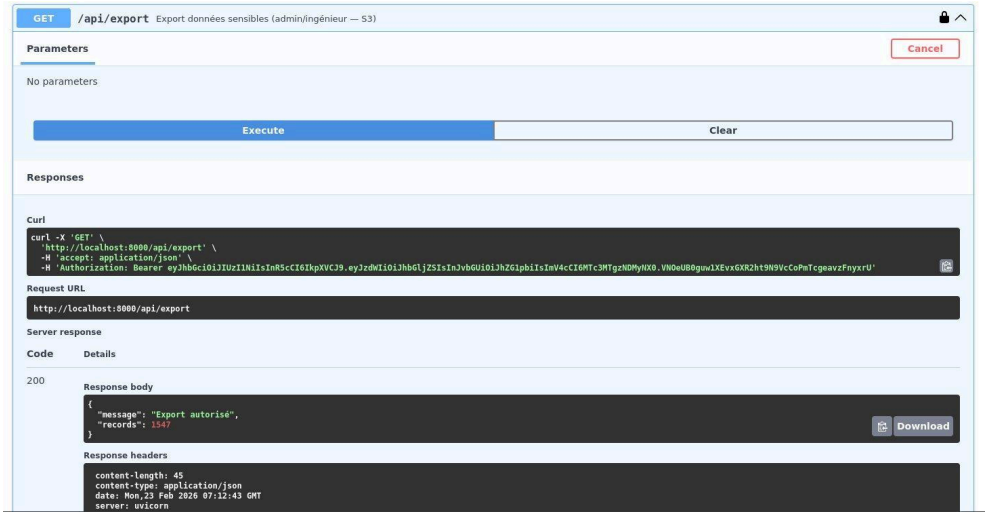

Figure 10 — 200 OK — alice autorisée sur /api/export (1 547 enregistrements)

alice accède à /api/export : 200 OK avec "Export autorisé", records: 1547. Même endpoint, même code, résultat différent selon le rôle. Le RBAC est précis : il bloque les non-autorisés sans pénaliser les administrateurs légitimes.

Figure 11 — Terminal serveur — Tous les accès d'alice AUTORISÉS

"ACCÈS AUTORISÉ | alice (admin) → export:read" confirme que l'administratrice légitime conserve son accès complet. En comparaison avec la Figure 8 (charlie), la différence de traitement selon le rôle est parfaitement visible.

ÉTAPE 5 — Impact de la Solution S3

Impact Technique

- 100% des tentatives d'accès à /admin/users et /api/export par un opérateur → bloquées
- 0 donnée sensible transmise à un utilisateur sans les droits requis
- Évaluation RBAC < 5 ms (Casbin en mémoire — 0 requête base de données)
- 10/10 tests unitaires Casbin validés — politique vérifiée avant chaque déploiement

Impact Organisationnel

- Principe du moindre privilège sur 10 ressources × 4 rôles × 4 actions (L/E/S/X)
  - Opérateur : prédictions et alertes uniquement
  - Ingénieur : historiques, prédictions, export technique
  - Admin : accès total
  - Auditeur : logs uniquement
- Modification des droits sans intervention sur le code — édition de policy.csv uniquement
- Séparation des responsabilités : politique de sécurité distincte du code métier

Impact Légal et Réglementaire

- Audit trail complet utilisable pour investigations de sécurité (RGPD, ISO 27001)
- Chaque accès journalisé : timestamp, user, role, IP, resource, action, status
- Double journalisation : SQLite (interface admin) + JSON (archivage légal)

RÉSULTAT S3 : Le scénario de fuite de données est totalement neutralisé. charlie ne peut plus accéder aux données confidentielles de BMI même avec un token JWT valide. 0 enregistrement exfiltré.

S5 — Attaque par Inférence API

ÉTAPE 1 — Présentation du Scénario S5

Le modèle LSTM d'AI4BMI prédit les pannes industrielles avec 87% de précision. Ce modèle représente plusieurs années de recherche et de données d'entraînement — c'est la propriété intellectuelle centrale de BMI.

L'attaque S5 étape par étape

> Un concurrent crée un compte avec un rôle minimal → il envoie des milliers de requêtes à /api/predictions avec des paramètres variés → il analyse les patterns de réponse (machine, probabilité, horizon, action) → par rétro-ingénierie, il reconstruit la logique du modèle LSTM → le modèle, qui a coûté des mois de développement, est compromis. Sans journalisation, cette attaque reste totalement invisible.

ÉTAPE 2 — Outils, Scripts et Bibliothèques

Contrôle granulaire des accès à l'API IA

- Casbin — politique RBAC limitant /api/predictions aux rôles ayant un besoin métier réel
- require_permission("predictions", "read") — protège l'endpoint du modèle LSTM
- app/routes/api.py — seuls operateur, ingenieur_maintenance et admin peuvent interroger le modèle

Audit Trail — Détection des comportements anormaux

- app/middleware/rbac.py — log_access() enregistre chaque appel à /api/predictions
- logs/audit.log — JSON horodaté : timestamp, IP, username, resource, status
- SQLite table audit_logs — consultable via l'interface admin RBAC Control Center
- Compteur d'accès en temps réel dans le dashboard — pic anormal immédiatement visible

JWT — Limitation de la fenêtre d'attaque

- Token JWT expirant en 60 minutes — l'attaquant doit se réauthentifier régulièrement

- Chaque réauthentification génère une entrée dans les logs de connexion
- Un volume anormal de reconnexions alerte l'administrateur sur une activité suspecte

Interface Admin — Monitoring et réaction

- RBAC Control Center — dashboard temps réel : accès autorisés / refusés / utilisateurs actifs
- Gestion CRUD — modification ou suppression immédiate d'un compte suspect
- Matrice RBAC — visualisation des droits par ressource et par rôle

ÉTAPE 3 — Justification du Choix des Outils

Justification — Contrôle de rôle sur /api/predictions

- **Limiter l'accès au modèle IA aux seuls rôles avec un besoin métier réel réduit la surface d'attaque**
  - Un visiteur ou auditeur ne peut jamais interroger le modèle → 403 immédiat
  - Un utilisateur sans compte valide est bloqué dès l'authentification
  - Même un compte légitime mal utilisé (trop de requêtes) est détectable via les logs

Justification — Audit Trail pour S5

- **L'attaque par inférence nécessite un grand nombre de requêtes — la journalisation la rend détectable**
  - Un opérateur légitime fait 5-10 requêtes par jour → normal
  - 1 000 requêtes en 1 heure depuis la même IP → anomalie détectable dans les logs
  - Sans journalisation, l'attaque est invisible et indétectable après coup

Justification — Interface Admin et réactivité

- **La gestion dynamique des droits via le CRUD permet de réagir en temps réel à une attaque S5**
  - Rétrograder le rôle d'un utilisateur suspect → immédiat, sans toucher au code
  - Supprimer un compte compromis → token JWT invalide dès la prochaine requête
  - Pas de redémarrage du serveur requis — continuité de service garantie

ÉTAPE 4 — Démonstration S5 — Interface et Audit

4.1 — Interface RBAC Control Center — Page de connexion

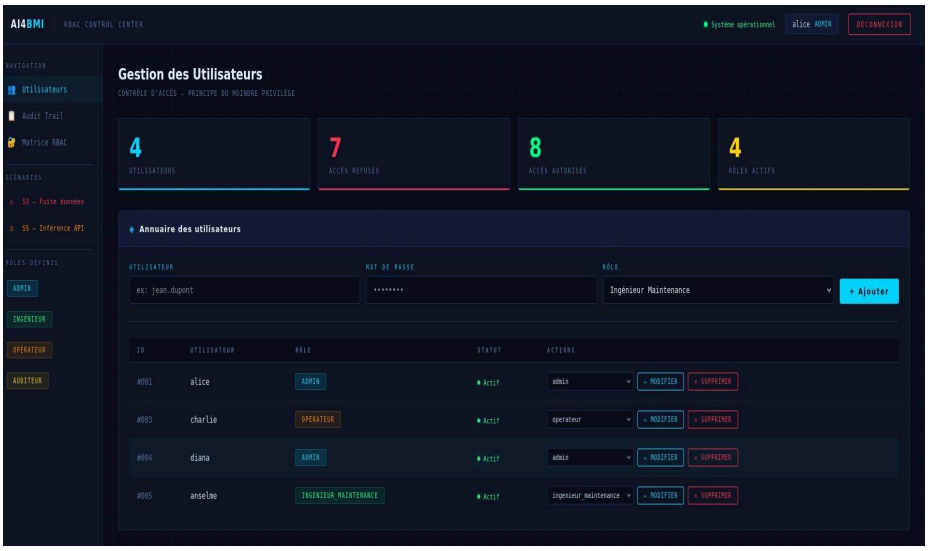

Figure 12 — Page de connexion sécurisée du RBAC Control Center

http://localhost:8000/admin — seul l'administrateur alice peut s'y connecter. Un non-administrateur reçoit "Accès refusé — réservé aux administrateurs" même avec des identifiants valides. C'est la première barrière de supervision contre S5.

4.2 — Dashboard — Monitoring temps réel

Figure 13 — Dashboard admin — Vue d'ensemble du système en temps réel

4 utilisateurs actifs, 7 accès refusés (tentatives S3/S5 bloquées), 8 accès autorisés, 4 rôles actifs. Les scénarios S3 et S5 sont visibles dans la sidebar. Un pic sur "Accès autorisés predictions:read" indiquerait une activité suspecte sur le modèle IA.

4.3 — Audit Trail — Détection des accès anormaux au modèle IA

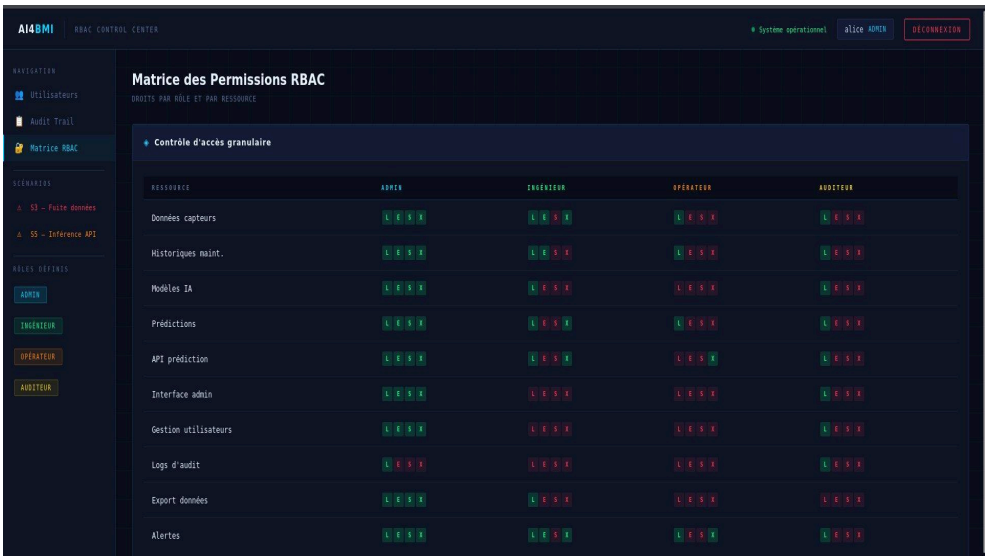

Figure 14 — Onglet Audit Trail — Journal complet des accès

Les logs montrent predictions:read autorisés (vert) et export:read refusés (rouge). En cas d'attaque S5, un pic de requêtes "predictions:read" depuis la même IP sur une courte période serait immédiatement visible. Chaque ligne contient : horodatage, utilisateur, rôle, ressource, IP, statut.

4.4 — CRUD — Réaction à une attaque S5 détectée

Si une attaque S5 est détectée dans l'audit trail, l'administrateur peut réagir immédiatement sans redémarrer le serveur :

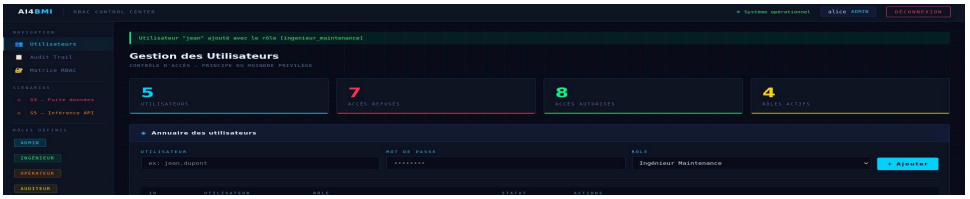

Figure 15 — Formulaire d'ajout — Contrôle des rôles dès la création

Tout nouvel utilisateur reçoit un rôle précis dès sa création. Un utilisateur créé avec le rôle "auditeur" ne pourra jamais interroger /api/predictions — bloqué par Casbin dès la première tentative.

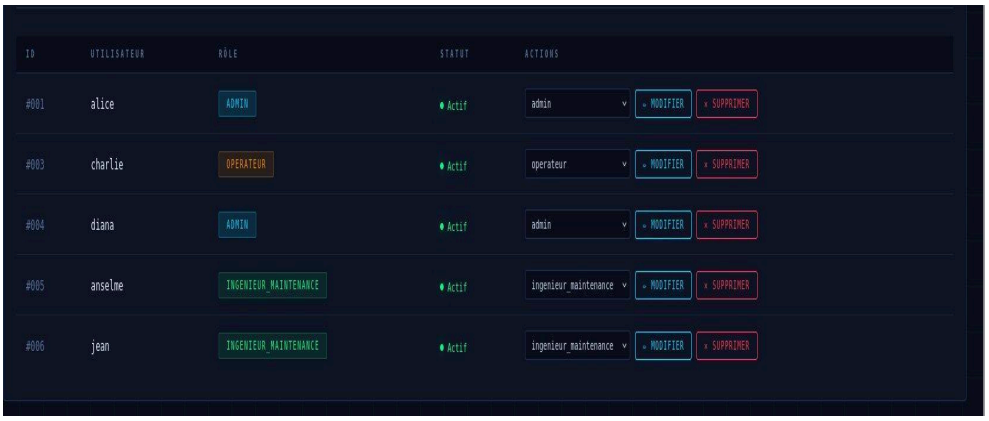

Figure 16 — Tableau des utilisateurs — Vue des droits actifs

L'administrateur voit en un coup d'œil qui a accès au modèle IA (opérateur, ingénieur, admin — badges colorés) et qui n'y a pas accès (auditeur). En cas de suspicion, le rôle peut être modifié directement depuis ce tableau.

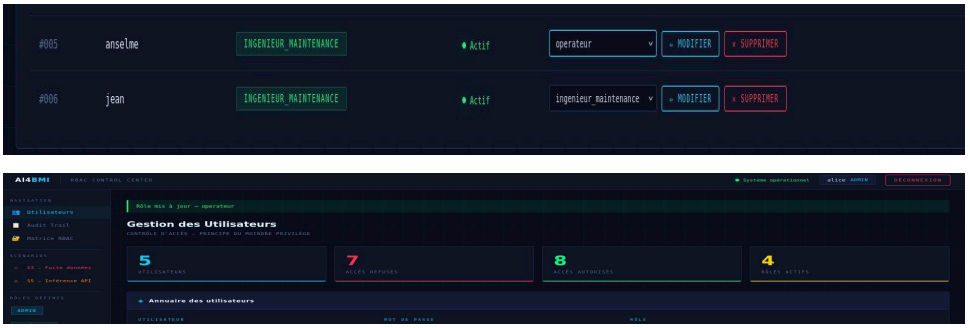

Figure 17 — Modification du rôle en temps réel — "Rôle mis à jour → operateur"

L'administrateur sélectionne le nouveau rôle et clique "Modifier". La confirmation apparaît immédiatement. Ce changement est effectif instantanément : toutes les prochaines requêtes de cet utilisateur sont évaluées avec le nouveau rôle, sans redémarrage du serveur.

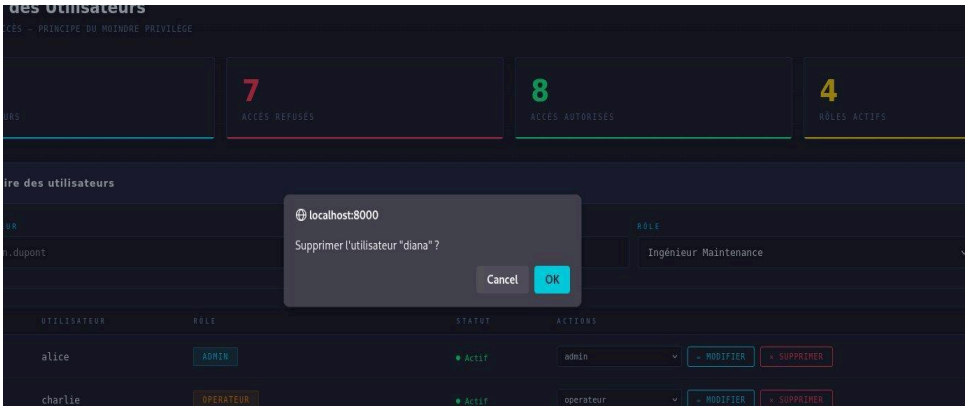

Figure 18 — Suppression d'un compte compromis — Confirmation de sécurité

En cas d'attaque confirmée, l'administrateur supprime le compte de l'attaquant. Une fois supprimé, son token JWT devient invalide à la prochaine requête. Les logs d'audit sont conservés pour investigations légales.

4.5 — Matrice RBAC — Visualisation des droits sur le modèle IA

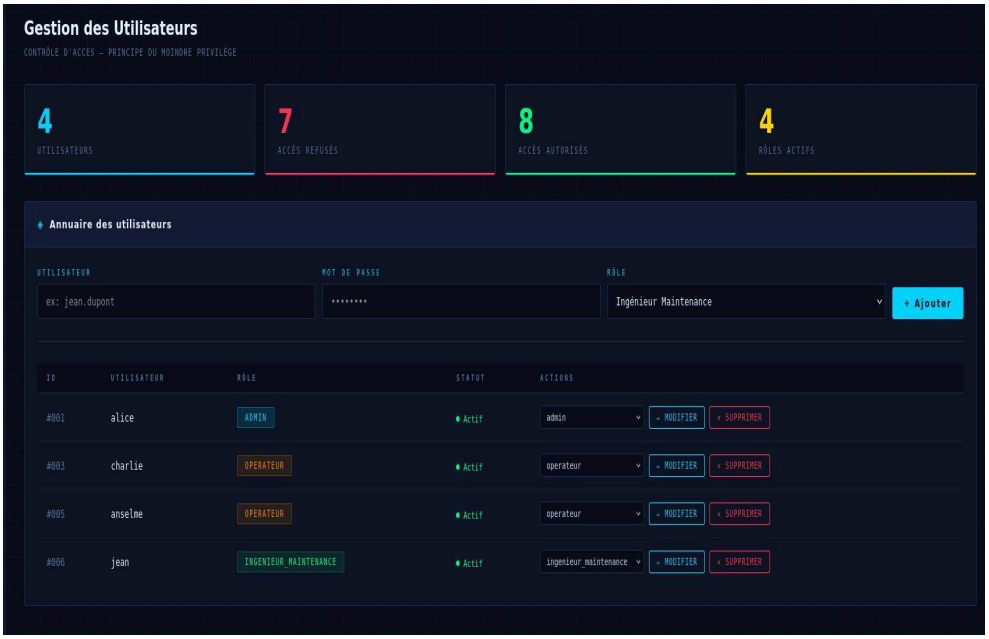

Figure 19 — Matrice des permissions RBAC — L/E/S/X par rôle et par ressource

La colonne "API prédiction" montre que admin, ingénieur et opérateur ont L (lecture) en vert — ils peuvent interroger le modèle. L'auditeur a L rouge — accès refusé. Cette matrice confirme visuellement la politique de protection contre S5 : seuls les rôles avec un besoin métier réel accèdent au modèle LSTM.

ÉTAPE 5 — Impact de la Solution S5

Impact sur la Protection du Modèle IA

- Accès à /api/predictions limité aux rôles avec besoin métier réel (opérateur, ingénieur, admin)
- Tout accès non autorisé au modèle → 403 Forbidden immédiat

- Chaque interrogation du modèle journalisée : timestamp, IP, utilisateur, statut
- Détection possible d'un volume anormal par analyse des logs (audit trail)

Impact sur la Réactivité en cas d'Incident S5

- Modification du rôle d'un suspect en temps réel — sans redémarrage du serveur
- Suppression d'un compte compromis — token JWT invalide immédiatement
- Audit trail consultable pour reconstruction chronologique d'une attaque
- Dashboard : compteur "accès autorisés predictions" visible en temps réel

Impact sur la Propriété Intellectuelle de BMI

- Double barrière de protection : authentification JWT + contrôle RBAC Casbin
- Sans compte avec le bon rôle → impossible d'interroger le modèle LSTM
- Les données d'entraînement (historiques) protégées par le même système RBAC
- Aucune donnée sur les cadences de production transmise sans autorisation explicite

RÉSULTAT S5 : L'accès au modèle IA est strictement contrôlé par rôle. Toute tentative non autorisée est bloquée et journalisée. Un comportement anormal (pic de requêtes) est détectable via l'audit trail avant que le modèle soit compromis.

CONCLUSION

Récapitulatif des réalisations

Livrable	Réalisation	Statut
L3.1	Matrice complète : 10 ressources × 4 rôles × 4 actions (L/E/S/X)	✅ **Réalisé**
L3.2	Code RBAC : model.conf + policy.csv + middleware rbac.py	✅ **Réalisé**
L3.3	Captures accès refusé : 403 charlie → /admin/users & /api/export	✅ **Réalisé**
L3.4	Interface admin CRUD : RBAC Control Center (ajout/modif/suppression)	✅ **Réalisé**
L3.5	Audit trail horodaté : SQLite + JSON + visualisation interface	✅ **Réalisé**

Tableau de synthèse des tests

Test réalisé	Résultat	HTTP	Scénario
charlie → /admin/users-list	**BLOQUÉ**	403	S3
charlie → /api/export	**BLOQUÉ**	403	S3
charlie → /api/predictions	**AUTORISÉ**	200	Normal
alice → /api/export	**AUTORISÉ — 1 547 records**	200	Admin
alice → /admin/users-list	**AUTORISÉ**	200	Admin
Tests Casbin (10 cas)	**TOUS VALIDÉS**	10/10	Validation
CRUD Ajout / Modification / Suppression	**OPÉRATIONNEL**	200	Admin

Le système RBAC implémenté pour la plateforme AI4BMI répond pleinement aux exigences du Livrable L3. Il protège efficacement contre les scénarios S3 (fuite de données) et S5 (inférence API) en appliquant le principe du moindre privilège sur l'ensemble de la plateforme industrielle.

La combinaison Casbin + FastAPI + JWT + Audit Trail offre une solution robuste, évolutive et opérationnelle. Les droits sont modifiables sans redéploiement, tous les accès sont tracés, et l'interface admin permet une supervision et une réaction en temps réel.

SÉCURITÉ ASSURÉE — 0 accès non autorisé possible | S3 neutralisé | S5 détectable | 100% des accès journalisés | Interface admin CRUD opérationnelle