AYI NEDJIMI Consultants

Sécuriser un Pipeline MLOps : Bonnes Pratiques et 2026

Catégorie : Intelligence Artificielle Lecture : 24 min Publié le : 13/02/2026 Auteur : Ayi NEDJIMI

Guide complet sur la sécurisation des pipelines MLOps : menaces sur les données d'entraînement, empoisonnement de modèles, sécurité de l'inférence,.

Table des Matières

Avez-vous évalué les risques d'injection de prompt sur vos systèmes d'IA en production ?

1 Les Menaces Spécifiques aux Pipelines ML

Les **pipelines MLOps** représentent en 2026 une surface d'attaque considérablement plus étendue que les pipelines DevOps classiques. Là où un pipeline CI/CD traditionnel manipule du code source et des artefacts binaires, un pipeline ML ajoute des dimensions critiques : **datasets massifs**, **modèles entraînés** pesant plusieurs gigaoctets, **feature stores** contenant des données sensibles, et des **notebooks d'expérimentation** souvent partagés sans contrôle d'accès. Selon le rapport Gartner 2025 sur la sécurité de l'IA, plus de **65% des organisations** déployant des modèles ML en production n'ont pas de stratégie de sécurité spécifique pour leurs pipelines d'apprentissage automatique.

Surface d'attaque élargie vs DevOps classique

Un pipeline MLOps typique comprend des composants absents du DevOps traditionnel qui constituent autant de vecteurs d'attaque potentiels. Les **données d'entraînement** peuvent être empoisonnées pour introduire des biais ou des backdoors dans le modèle final. Les **feature stores** centralisent des transformations de données qui, si elles sont compromises, affectent tous les modèles en aval. Les **hyperparamètres** et configurations d'entraînement peuvent être manipulés pour dégrader subtilement les performances. Les **registres de modèles** stockent des artefacts binaires opaques dont l'intégrité est difficile à vérifier. Enfin, les **endpoints d'inférence** exposent les modèles à des attaques adversariales en temps réel.

Taxonomie des attaques sur les pipelines ML

Les menaces spécifiques aux pipelines ML se répartissent en plusieurs catégories fondamentales. Le **data poisoning** consiste à injecter des données malveillantes dans le dataset d'entraînement pour modifier le comportement du modèle — par exemple, faire qu'un modèle de détection de spam laisse passer certains patterns. Les **model backdoors** sont des comportements cachés implantés dans un modèle qui s'activent uniquement en présence d'un trigger spécifique, permettant à un attaquant de contrôler les sorties du modèle à volonté. Le **model theft** vise à extraire un modèle propriétaire via des requêtes systématiques à l'API d'inférence, reconstruisant progressivement une approximation fonctionnelle. Les **adversarial inputs** sont des entrées soigneusement perturbées qui trompent le modèle en production tout en paraissant normales à un observateur humain.

MITRE ATLAS : framework de référence

Le framework **MITRE ATLAS** (Adversarial Threat Landscape for Artificial-Intelligence Systems) est devenu en 2026 la référence incontournable pour modéliser les menaces sur les systèmes ML. Inspiré du célèbre **MITRE ATT&CK**, ATLAS catégorise les techniques d'attaque spécifiques à l'IA en une matrice structurée couvrant la reconnaissance, le développement de ressources, l'accès initial, l'exécution, la persistance, l'évasion et l'impact. Le framework documente plus de **90 techniques** spécifiques à l'IA, incluant le ML Supply Chain Compromise, le Backdoor ML Model, l'Exfiltration via ML Inference API, et le Data Poisoning. Pour un RSSI, ATLAS permet de mapper les contrôles de sécurité existants aux menaces ML et d'identifier les lacunes dans la posture de sécurité du pipeline MLOps.

Cas réels d'attaques 2024-2026

Les attaques sur les pipelines ML ne relèvent plus du théorique. En **mars 2024**, des chercheurs ont démontré comment des modèles populaires sur **Hugging Face** contenaient des payloads malveillants cachés dans des fichiers Pickle sérialisés, exécutant du code arbitraire au chargement. En **juin 2024**, une attaque de supply chain sur **PyTorch** a

compromis le package torchtriton via le registre PyPI, affectant des milliers de pipelines d'entraînement. En **janvier 2025**, une campagne complexe de data poisoning ciblant des datasets publics de **Common Crawl** a été découverte, injectant des biais subtils dans les modèles de langage entraînés sur ces données. Plus récemment, en **novembre 2025**, une vulnérabilité critique dans **MLflow** permettait l'exécution de code à distance via les artefacts du registre de modèles, soulignant que même les outils d'orchestration MLOps eux-mêmes constituent des cibles de valeur pour les attaquants.

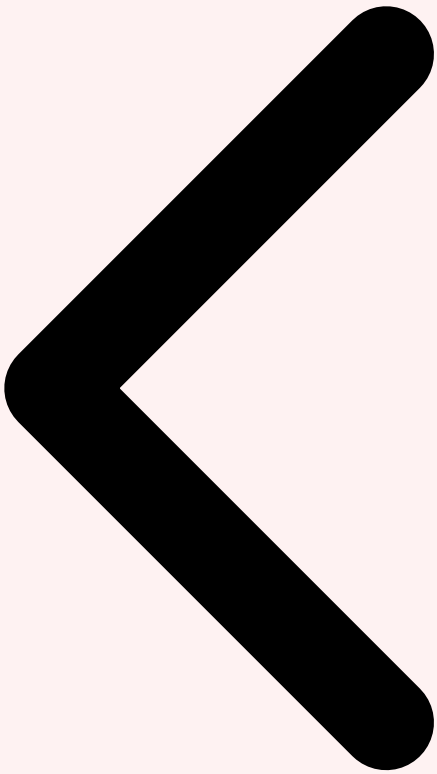
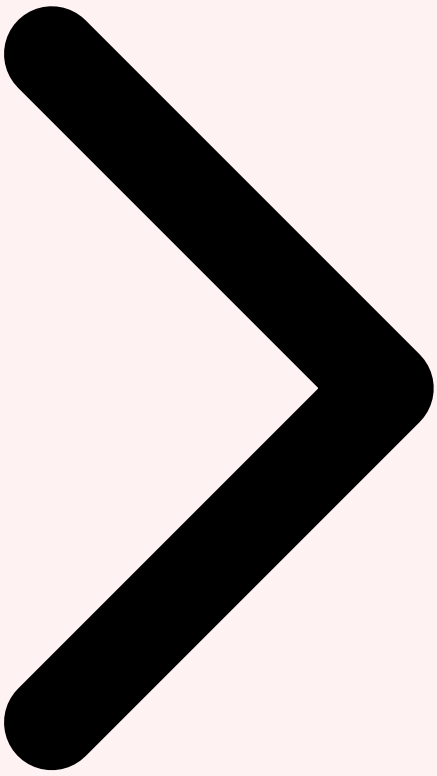


Table des Matières Menaces Pipeline ML Sécurité Données



Critere	Description	Niveau de risque
Confidentialite	Protection des donnees d'entrainement et des prompts	Eleve
Integrite	Fiabilite des sorties et detection des hallucinations	Critique
Disponibilite	Resilience du service et gestion de la charge	Moyen
Conformite	Respect du RGPD, AI Act et politiques internes	Eleve

Notre avis d'expert

La gouvernance de l'IA est le prochain grand chantier de la cybersécurité. Les attaques par prompt injection, l'empoisonnement de données d'entraînement et l'extraction de modèles sont des menaces concrètes que nous observons de plus en plus lors de nos missions. Ne pas s'y préparer, c'est accepter un risque majeur.

2 Sécurité des Données d'Entraînement

Les **données d'entraînement** constituent le fondement de tout modèle ML et représentent simultanément le vecteur d'attaque le plus critique et le moins surveillé des pipelines MLOps. Un modèle n'est aussi fiable que les données qui l'ont formé : si un attaquant parvient à compromettre le dataset, il contrôle indirectement le comportement du modèle en production. La sécurisation des données d'entraînement exige une approche multicouche couvrant la **provenance**, l'**intégrité**, la **confidentialité** et le **versionnement**.

Provenance et intégrité des datasets

Le concept de **data lineage** (traçabilité des données) est essentiel pour garantir l'intégrité d'un pipeline ML. Chaque dataset utilisé pour l'entraînement doit disposer d'un **pedigree vérifiable** : d'où viennent les données, quelles transformations elles ont subies, qui les a modifiées, et quand. En pratique, cela implique de calculer des **checksums SHA-256** à chaque étape du pipeline, de maintenir un registre immuable des sources de données, et d'utiliser des outils comme **Apache Atlas** ou **OpenLineage** pour tracer automatiquement le flux de données. Chaque modification du dataset doit être journalisée avec l'identité de l'auteur, le timestamp, et la justification métier. Cette traçabilité permet non seulement de détecter des modifications non autorisées, mais aussi de répondre aux exigences de l'**AI Act européen** en matière d'auditabilité des systèmes IA à haut risque.

Détection de data poisoning

Le **data poisoning** est une attaque où un adversaire injecte des échantillons malveillants dans le dataset d'entraînement pour altérer le comportement du modèle. La détection repose sur plusieurs techniques complémentaires. L'**analyse statistique d'outliers** identifie les échantillons qui dévient significativement de la distribution attendue — des outils comme **PyOD** (Python Outlier Detection) et **Alibi Detect** implémentent plus de 30 algorithmes de détection d'anomalies. La bibliothèque **Cleanlab** détecte automatiquement les étiquettes erronées ou suspectes dans les datasets en utilisant la théorie du **confident learning**. Pour les attaques de type backdoor, les techniques de **spectral signature analysis** analysent la représentation latente des données pour identifier des clusters anormaux qui pourraient correspondre à des triggers implantés. L'approche recommandée est de combiner ces méthodes dans un pipeline de validation automatique exécuté avant chaque entraînement.

Protection des données sensibles

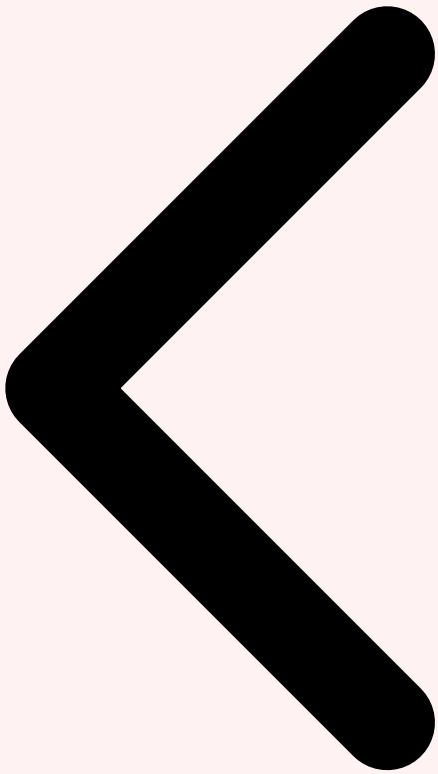
Les datasets d'entraînement contiennent fréquemment des **données personnelles identifiables (PII)** qui doivent être protégées conformément au RGPD et aux réglementations sectorielles. La première ligne de défense est la **détection automatique de PII** via des modèles NER (Named Entity Recognition) spécialisés comme **Microsoft Presidio** ou **AWS Comprehend**, capables d'identifier noms, adresses, numéros de téléphone, emails et identifiants dans les données textuelles. L'**anonymisation** remplace les PII par des substituts réalistes mais fictifs, tandis que la **pseudonymisation** utilise des

tokens réversibles pour les cas où la réidentification reste nécessaire. Pour une protection mathématiquement prouvable, la **differential privacy** ajoute un bruit calibré aux données ou aux gradients d'entraînement, garantissant qu'aucun échantillon individuel ne peut être reconstruit depuis le modèle final — des frameworks comme **Opacus** (PyTorch) et **TensorFlow Privacy** implémentent cette approche nativement. Pour approfondir, consultez Coût d'Inférence des LLM : Optimiser sa Facture Cloud.

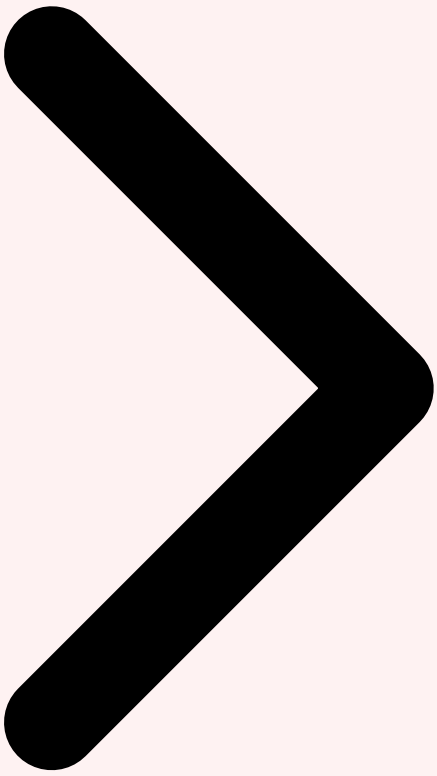
Vos pipelines de données d'entraînement sont-ils protégés contre l'empoisonnement ?

Data versioning sécurisé

Le versionnement des données est aussi critique que le versionnement du code, mais les outils traditionnels comme Git ne sont pas conçus pour des fichiers de plusieurs gigaoctets. **DVC (Data Version Control)** étend Git avec un suivi des fichiers volumineux stockés sur des backends distants (S3, GCS, Azure Blob), en ne versionnant dans Git que les métadonnées et les checksums. **LakeFS** fournit un système de branches et de commits pour les data lakes, permettant de créer des snapshots atomiques de datasets complets avec des contrôles d'accès RBAC granulaires. **Delta Lake** ajoute des garanties ACID aux data lakes Spark avec un journal de transactions immuable. Pour chaque outil, la sécurisation implique le chiffrement at-rest et in-transit des données, des politiques d'accès Zero Trust, la signature cryptographique des commits de données, et des mécanismes d'audit trail permettant de retracer toute modification jusqu'à son auteur. L'objectif est de garantir que chaque version d'un dataset utilisée pour l'entraînement est reproductible, vérifiable et non falsifiable.



Menaces Pipeline ML Sécurité Données Sécurité Entraînement



Cas concret
L'attaque par prompt injection sur les systèmes GPT documentée par OWASP en 2023 a révélé que des instructions malveillantes dissimulées dans des documents pouvaient détourner le comportement de chatbots d'entreprise, accédant à des données internes sensibles sans aucune authentification supplémentaire.

3 Sécurité de la Phase d'Entraînement

La phase d'entraînement est le moment le plus vulnérable du pipeline MLOps : des ressources computationnelles coûteuses traitent des données sensibles pour produire un artefact — le modèle — qui encodera dans ses poids tout ce qu'il a appris, y compris d'éventuels biais ou backdoors implantés par un attaquant. La sécurisation de cette phase exige un contrôle rigoureux de l'**environnement d'exécution**, des **bibliothèques utilisées**, et des **artefacts produits**, avec une traçabilité complète de chaque étape de l'entraînement.

Environnements d'entraînement isolés

L'isolation des environnements d'entraînement est la première ligne de défense contre la compromission du processus ML. Le **sandboxing** des notebooks et des jobs d'entraînement empêche un code malveillant d'accéder au réseau, au système de fichiers hôte, ou à d'autres workloads. En pratique, cela se traduit par l'utilisation de **conteneurs renforcés** avec des profils AppArmor ou SELinux restrictifs, l'isolation réseau via des **network policies Kubernetes** empêchant toute communication non autorisée, et le chiffrement des volumes de données montés dans les pods d'entraînement. Pour les entraînements sur **GPU**, l'isolation est plus complexe : les technologies **NVIDIA MIG** (Multi-Instance GPU) et **NVIDIA Confidential Computing** permettent de partitionner physiquement les GPU et de chiffrer les données en mémoire GPU, empêchant un workload malveillant de lire les données d'un autre entraînement sur le même cluster.

Protection contre le model backdooring

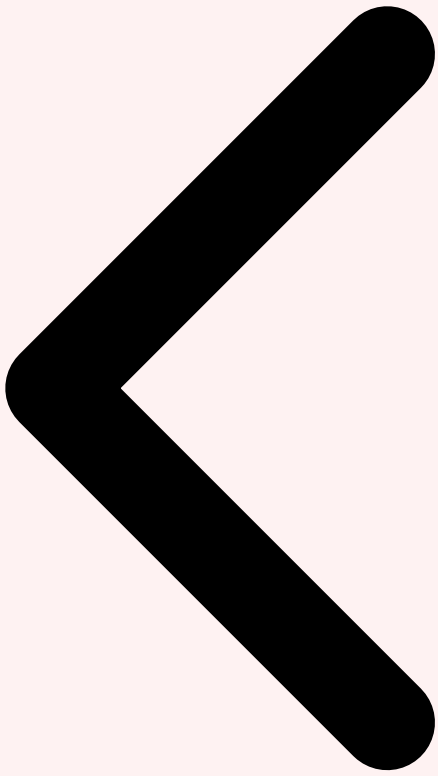
Un **model backdoor** est un comportement caché implanté dans les poids d'un modèle qui s'active uniquement en présence d'un pattern déclencheur spécifique — par exemple, un pixel dans un coin d'image fait qu'un classifieur identifie systématiquement un objet comme inoffensif. La détection de backdoors repose sur plusieurs techniques avancées. L'**analyse spectrale** examine les représentations latentes du modèle pour détecter des séparations anormales dans l'espace des features, caractéristiques de la présence d'un trigger. Le **fine-pruning** consiste à supprimer progressivement les neurones dormants (rarement activés sur des données propres mais actifs sur les données de trigger), puis à réentraîner le modèle sur un sous-ensemble de données vérifiées. Les techniques de **Neural Cleanse** inversent le processus d'attaque en cherchant le plus petit pattern qui fait basculer les prédictions, identifiant ainsi les triggers potentiels. L'**Activation Clustering** groupe les activations des neurones pour identifier les clusters correspondant aux données empoisonnées. En 2026, des outils comme **IBM ART** (Adversarial Robustness Toolbox) intègrent ces techniques dans une suite unifiée de détection de backdoors.

Audit trail de l'entraînement

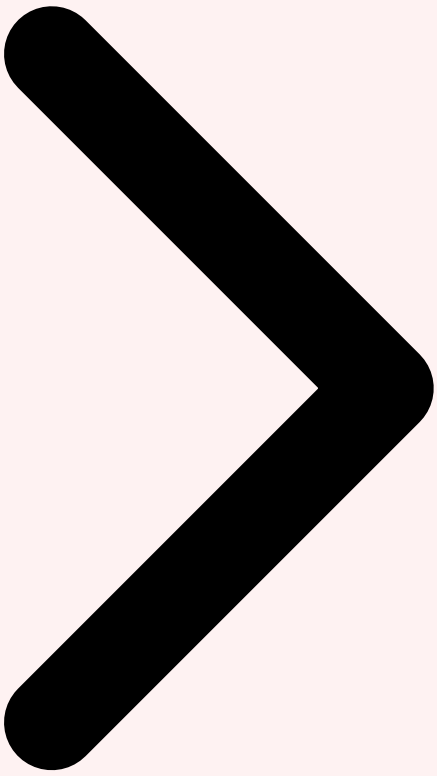
Chaque entraînement doit produire un **audit trail complet** permettant de reproduire exactement le modèle et d'investiguer tout comportement suspect. Cet audit trail doit capturer les **hyperparamètres** exacts (learning rate, batch size, epochs, architecture), les **seeds aléatoires** pour la reproductibilité, un **snapshot référencé du dataset** (hash DVC ou commit LakeFS), la liste complète des **dépendances** avec leurs versions exactes (pip freeze), les **métriques** d'entraînement et de validation à chaque epoch, et l'**identité** de l'opérateur ou du pipeline CI/CD ayant lancé l'entraînement. Des outils comme **MLflow Tracking**, **Weights & Biases**, ou **Neptune.ai** capturent automatiquement ces informations. L'audit trail doit être stocké de manière immuable — idéalement dans un système WORM (Write Once Read Many) — pour garantir qu'il ne peut pas être altéré après coup.

Supply chain ML et code signing

La **supply chain ML** est devenue un vecteur d'attaque majeur, les bibliothèques Python de machine learning comptant parmi les plus téléchargées sur PyPI. Un **pip audit** régulier vérifie les dépendances contre les bases de vulnérabilités connues (CVE), tandis que **Safety** (de PyUp) effectue des vérifications similaires avec une base de données propriétaire plus étendue. Pour aller plus loin, **Bandit** analyse le code Python à la recherche de patterns de sécurité dangereux, et **Semgrep** permet de définir des règles personnalisées pour détecter des usages risqués spécifiques au ML (par exemple, l'utilisation de `pickle.load()` sans validation). Le **code signing** des artefacts de training — scripts d'entraînement, configurations, Dockerfiles — garantit que seul du code approuvé et vérifié est exécuté dans le pipeline. En combinant **Sigstore/cosign** pour la signature et **SLSA** (Supply-chain Levels for Software Artifacts) pour l'attestation de provenance, on crée une chaîne de confiance vérifiable depuis le commit Git jusqu'au modèle déployé en production.



Sécurité Données Sécurité Entraînement Sécurité Modèles



4 Sécurité des Modèles et Artefacts

Les **modèles ML** sont des artefacts binaires complexes qui encapsulent des millions, voire des milliards de paramètres. Contrairement au code source qui peut être lu et audité, un modèle est essentiellement une **boîte noire** dont le comportement ne peut être vérifié que par des tests empiriques. Cette opacité intrinsèque rend la sécurisation des artefacts de modèles particulièrement critique : un modèle compromis peut passer toutes les vérifications fonctionnelles standard tout en contenant un backdoor activable à la demande de l'attaquant.

Formats sûrs : SafeTensors vs Pickle

Le choix du **format de sérialisation** des modèles est une décision de sécurité fondamentale. Le format **Pickle** de Python, historiquement utilisé par PyTorch et scikit-learn, est intrinsèquement dangereux : le mécanisme de désérialisation exécute du code Python arbitraire, permettant à un fichier `.pkl` malveillant d'exécuter des commandes système au moment du chargement. En 2024, des chercheurs de Trail of Bits ont démontré

qu'un modèle Pickle pouvait contenir un reverse shell invisible tout en fonctionnant normalement comme classifieur d'images. Le format **SafeTensors**, développé par Hugging Face, résout ce problème en stockant uniquement les tenseurs numériques dans un format binaire simple sans mécanisme d'exécution de code. SafeTensors est désormais le format recommandé par Hugging Face, PyTorch et la majorité de l'écosystème ML. La migration vers SafeTensors doit être accompagnée d'une **politique stricte** interdisant le chargement de fichiers Pickle non vérifiés, avec des scans automatiques via **Fickling** (outil d'analyse statique de fichiers Pickle) pour détecter les payloads malveillants dans les modèles hérités. Pour approfondir, consultez Agentic AI 2026 : Autonomie en Entreprise.

Model signing et vérification d'intégrité

La **signature cryptographique des modèles** est le mécanisme clé pour garantir qu'un modèle déployé en production est exactement celui qui a été validé et approuvé. L'écosystème **Sigstore**, avec son outil **cosign**, permet de signer des artefacts OCI (conteneurs, modèles) avec des certificats éphémères liés à une identité OIDC, éliminant le besoin de gérer des clés privées de longue durée. Le workflow recommandé est le suivant : à la fin de l'entraînement, le pipeline CI/CD calcule le hash SHA-256 du modèle, le signe avec cosign en utilisant l'identité du pipeline (par exemple, un service account GitHub Actions), et publie la signature dans un registre de transparence **Rekor**. Avant chaque déploiement, le système vérifie que la signature est valide et correspond à un pipeline autorisé. Cette approche crée une **chaîne de confiance** vérifiable de bout en bout, depuis l'entraînement jusqu'à l'inférence.

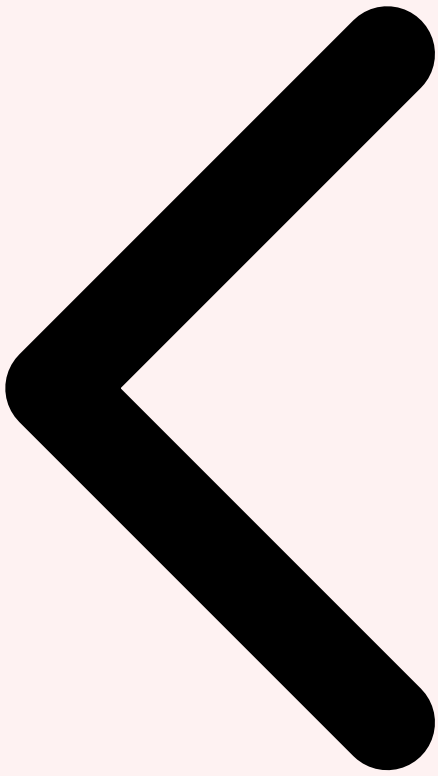
Model registry sécurisé

Un **model registry** sécurisé est l'équivalent d'un registre de conteneurs pour les modèles ML, fournissant un référentiel central avec contrôle d'accès, versionnement et audit. **MLflow Model Registry** offre un système de stages (Staging, Production, Archived) avec des transitions contrôlées par des politiques d'approbation, des webhooks pour l'intégration avec les systèmes de sécurité, et une authentification LDAP/OIDC. **Weights & Biases** ajoute une traçabilité complète des expériences avec des contrôles d'accès par projet et une piste d'audit immutable. Pour les organisations utilisant Kubernetes, **Seldon Core** et **KServe** intègrent le registre de modèles directement dans le déploiement, vérifiant l'intégrité des modèles au moment du chargement. La clé est de traiter le model registry comme une **infrastructure critique** : accès Zero Trust, chiffrement systématique, sauvegardes régulières, et monitoring des accès suspects comme le téléchargement massif de modèles qui pourrait indiquer une tentative de vol de propriété intellectuelle.

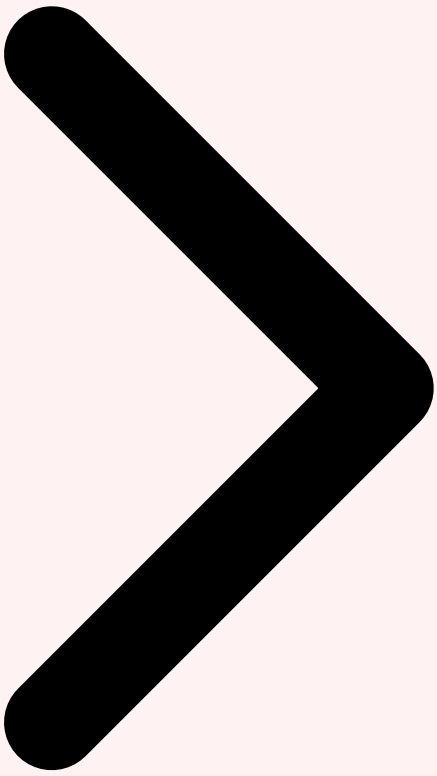
Protection contre le model theft

Le **vol de modèle** représente un risque majeur pour les organisations dont les modèles ML constituent un avantage compétitif. Les attaques par **extraction de modèle** utilisent des requêtes systématiques à l'API d'inférence pour reconstruire une approximation fonctionnelle du modèle cible. La défense repose sur plusieurs couches. Le **watermarking** implante un filigrane invisible dans les prédictions du modèle — des patterns

caractéristiques qui permettent de prouver la paternité en cas de copie. Le **fingerprinting** utilise des ensembles de requêtes spécifiques (adversarial examples) qui produisent des réponses uniques à chaque modèle, permettant d'identifier un modèle volé même s'il a été affiné. La détection d'extraction repose sur le monitoring des patterns d'utilisation de l'API : un volume anormalement élevé de requêtes systématiques, couvrant uniformément l'espace d'entrée, est caractéristique d'une attaque d'extraction. Des outils comme **Protect AI ModelScan** et **HiddenLayer** offrent des solutions commerciales de protection contre le vol et la manipulation de modèles, combinant watermarking, monitoring et détection d'anomalies dans une plateforme unifiée.



Sécurité Entraînement Sécurité Modèles Sécurité Inférence



5 Sécurité de l'Inférence en Production

L'**inférence en production** est la phase où le modèle ML est exposé directement aux utilisateurs et, par conséquent, aux attaquants. Contrairement aux phases d'entraînement qui se déroulent dans des environnements contrôlés, l'inférence opère en temps réel sur des entrées non fiables, dans un contexte où la latence et la disponibilité sont critiques. La sécurisation de l'inférence exige un équilibre délicat entre **protection robuste** et **performance opérationnelle**, en intégrant des mécanismes de défense qui n'impactent pas de manière prohibitive les temps de réponse.

Input validation et sanitization

La **validation des entrées** est la première ligne de défense contre les attaques adversariales en inférence. Au-delà de la validation de schéma classique (types, formats, bornes), les endpoints ML nécessitent des contrôles spécifiques. La **détection d'inputs adversariaux** utilise des détecteurs statistiques qui comparent la distribution de l'entrée à celle des données d'entraînement — une entrée significativement hors distribution (OOD)

est suspecte. Des outils comme **Alibi Detect** implémentent des détecteurs OOD basés sur des autoencodeurs, des tests de Kolmogorov-Smirnov, et des méthodes de score d'isolement. Pour les modèles de vision par ordinateur, les détecteurs de **perturbations adversariales** analysent les gradients de l'entrée pour identifier les patterns artificiels caractéristiques d'une attaque FGSM, PGD ou C&W. Pour les modèles de langage, la validation inclut la détection de **prompt injection** — des instructions malveillantes cachées dans l'entrée utilisateur visant à détourner le comportement du modèle. L'architecture recommandée place un **proxy de validation** devant chaque endpoint d'inférence, appliquant ces contrôles avant que la requête n'atteigne le modèle.

Output filtering et protection des données

Le filtrage des sorties est tout aussi critique que la validation des entrées. Un modèle peut involontairement exposer des **données personnelles** mémorisées pendant l'entraînement — un phénomène documenté dans les LLM où des requêtes spécifiques peuvent extraire des données d'entraînement verbatim. Le **PII scanning** des sorties détecte et masque automatiquement les informations sensibles (noms, adresses, numéros de carte) avant qu'elles n'atteignent l'utilisateur. Les filtres de **toxicité** évaluent les sorties textuelles pour bloquer les contenus inappropriés, offensants ou dangereux. La **détection d'hallucinations** compare les sorties du modèle à des sources factuelles pour identifier les informations inventées. Pour les modèles génératifs, des guardrails comme **NVIDIA NeMo Guardrails** ou **Guardrails AI** permettent de définir des politiques de filtrage déclaratives, combinant des règles statiques et des modèles de classification pour garantir que les sorties respectent les politiques de l'organisation.

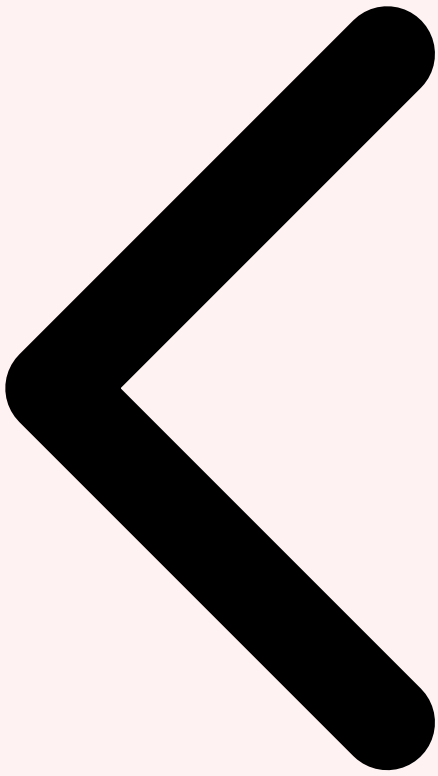
Rate limiting et abuse prevention

Le **rate limiting** des endpoints d'inférence ML va au-delà du simple contrôle de débit HTTP. Les attaques d'extraction de modèle nécessitent un grand nombre de requêtes systématiques, et un rate limiting intelligent peut les détecter et les bloquer. L'approche recommandée combine plusieurs niveaux : un **rate limit global** par API key limitant le volume total de requêtes, un **rate limit par utilisateur** empêchant un compte compromis de monopoliser les ressources, et un **rate limit adaptatif** qui réduit les quotas en cas de détection de patterns suspects. La détection de patterns inclut l'analyse de la **couverture de l'espace d'entrée** — un utilisateur légitime envoie des requêtes concentrées autour de cas d'usage spécifiques, tandis qu'une attaque d'extraction balaye systématiquement l'espace d'entrée. Les **API gateways** comme Kong, Tyk ou AWS API Gateway fournissent les mécanismes de base, mais doivent être complétés par des règles spécifiques au ML implémentées dans le proxy de validation.

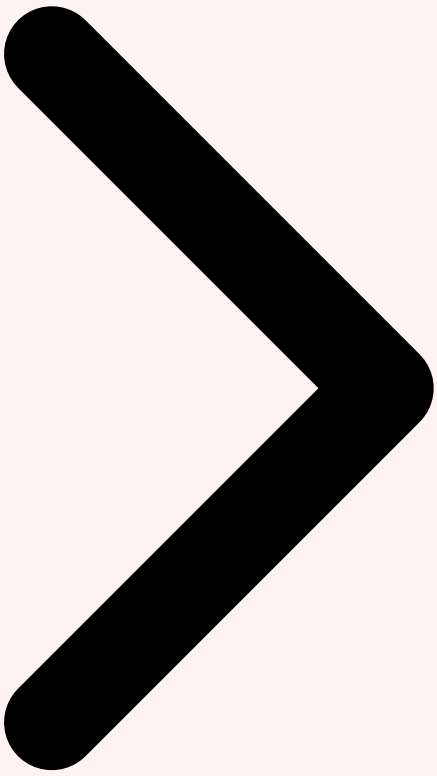
Monitoring de drift et détection d'attaques

Le **monitoring continu** en production est le dernier filet de sécurité du pipeline MLOps. Le **data drift** détecte quand la distribution des données d'entrée en production s'éloigne significativement de celle des données d'entraînement — ce qui peut indiquer soit une évolution naturelle de l'usage, soit une attaque coordonnée envoyant des données

atypiques. Le **concept drift** surveille la dégradation des performances du modèle dans le temps, qui peut résulter d'un empoisonnement progressif. Des plateformes comme **Evidently AI**, **Fiddler**, et **Arize AI** offrent des dashboards de monitoring ML spécialisés avec des alertes configurables sur les métriques de drift. Pour la **détection d'attaques adversariales** en temps réel, l'approche la plus efficace combine un modèle sentinelle dédié qui analyse les patterns de requêtes et un système de scoring de risque qui agrège les signaux (requêtes OOD, couverture spatiale, timing) pour identifier les sessions suspectes et les remonter au SOC via l'intégration SIEM.



Sécurité Modèles Sécurité Inférence Supply Chain ML



6 Supply Chain ML : Dépendances et Modèles Tiers

La **supply chain ML** constitue l'un des vecteurs d'attaque les plus insidieux et les moins maîtrisés dans les pipelines MLOps modernes. Un pipeline ML typique repose sur des dizaines de bibliothèques Python, des images Docker contenant des frameworks d'entraînement, et souvent des **modèles pré-entraînés tiers** téléchargés depuis des registres publics. Chacun de ces composants représente un point de confiance implicite que les attaquants exploitent activement. L'attaque SolarWinds a démontré l'impact critique des compromissions de supply chain dans le software classique — les pipelines ML ajoutent la dimension des **modèles comme vecteurs d'attaque**, un risque que peu d'organisations évaluent correctement.

Risques des modèles Hugging Face non vérifiés

Hugging Face Hub héberge en 2026 plus de 800 000 modèles publics, dont la vaste majorité n'a subi aucun audit de sécurité. Les risques sont multiples et documentés. Les modèles au format **Pickle** peuvent contenir du code d'exécution arbitraire déguisé en poids

de modèle. Des chercheurs de JFrog ont identifié en 2024 plus de **100 modèles malveillants** sur Hugging Face, incluant des reverse shells, des cryptominers, et des exfiltrateurs de credentials. Même les modèles SafeTensors ne sont pas à l'abri de risques fonctionnels : un modèle peut avoir été entraîné sur des données empoisonnées pour contenir un backdoor sémantique qui s'active sur des triggers textuels spécifiques, sans aucune trace de code malveillant dans l'artefact lui-même. La politique recommandée est de maintenir un **registre interne de modèles approuvés**, de n'autoriser que les modèles ayant passé un scan de sécurité complet (ModelScan, Fickling, analyse des métadonnées), et de bloquer le téléchargement direct depuis les registres publics dans les environnements de production.

SBOM pour ML : Software et Model Bill of Materials

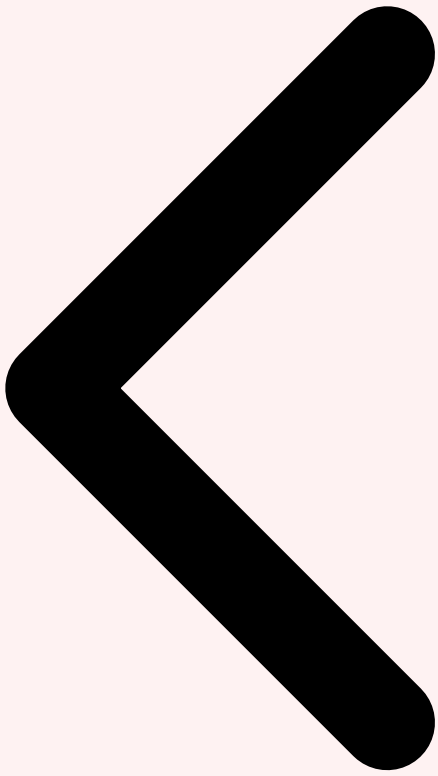
Le concept de **Software Bill of Materials (SBOM)**, rendu obligatoire pour les fournisseurs du gouvernement américain par l'Executive Order 14028, doit être étendu aux pipelines ML avec un **ML-BOM** (Machine Learning Bill of Materials) qui documente non seulement les dépendances logicielles mais aussi les **données d'entraînement**, les **modèles pré-entraînés** utilisés comme base, les **hyperparamètres**, et les **métriques de validation**. En pratique, un ML-BOM comprend le SBOM logiciel classique au format **SPDX** ou **CycloneDX** pour les bibliothèques Python et les images Docker, une **data card** décrivant les sources de données, leur taille, leur distribution et leurs biais connus, une **model card** documentant l'architecture, les performances, les limitations et les risques éthiques, et les **attestations de provenance** SLSA pour chaque artefact. L'outil **CycloneDX-ML** étend le standard CycloneDX avec des composants spécifiques au ML, tandis que **Protect AI** propose un format ML-BOM propriétaire intégré à leur plateforme de sécurité IA.

Vérification des dépendances Python

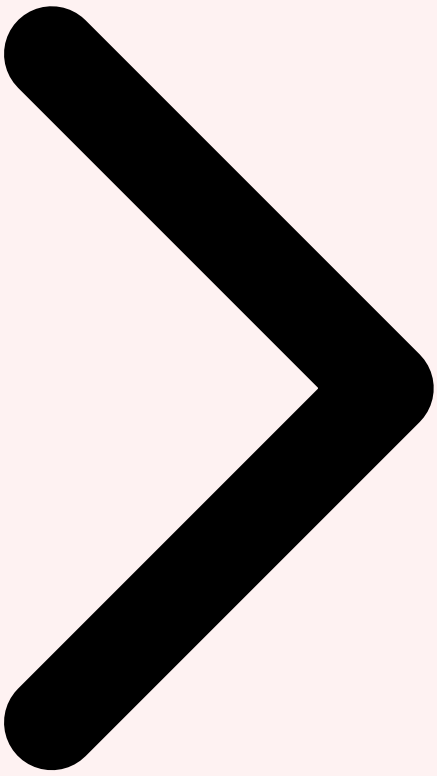
Les bibliothèques Python de ML sont des cibles privilégiées pour les attaques de supply chain en raison de leur large adoption et de la complexité de leurs dépendances transitives. **pip-audit**, développé par Google, vérifie toutes les dépendances installées contre la base de vulnérabilités OSV (Open Source Vulnerabilities) et signale les packages avec des CVE connues. **Snyk** offre une analyse plus approfondie avec une base propriétaire couvrant des vulnérabilités non encore publiées en CVE, plus un monitoring continu des nouvelles vulnérabilités sur les dépendances déclarées. **Dependabot** de GitHub automatise la création de pull requests pour les mises à jour de sécurité, mais doit être configuré avec des politiques strictes pour les bibliothèques ML qui ont souvent des contraintes de compatibilité complexes (par exemple, les versions spécifiques de CUDA, cuDNN et PyTorch doivent être alignées). L'approche défensive recommandée combine un scan automatique dans le CI/CD avec un **lock file** strict (`pip freeze` ou `poetry.lock`), la vérification des hashes de packages avec `pip install --require-hashes`, et un miroir PyPI interne (comme **Artifactory** ou **devpi**) filtrant les packages non approuvés.

Container security et policies d'approvisionnement

Les **images Docker ML** sont particulièrement volumineuses et complexes, intégrant des drivers GPU, des frameworks de calcul scientifique, et des bibliothèques de machine learning avec des centaines de dépendances transitives. **Trivy** et **Docker Scout** analysent ces images pour identifier les vulnérabilités dans les packages système et les bibliothèques, mais les images ML présentent des défis spécifiques : les images NVIDIA CUDA officielles contiennent régulièrement des vulnérabilités connues que NVIDIA met du temps à corriger, et les frameworks ML comme TensorFlow et PyTorch embarquent des dépendances C++ dont les vulnérabilités ne sont pas toujours traçables par les scanners standards. La politique d'approvisionnement recommandée inclut la construction d'**images de base curatées** à partir de distroless ou d'images minimales, la mise en place d'un **registre de conteneurs interne** avec des scans automatiques de chaque image poussée, des **admission controllers Kubernetes** (OPA/Gatekeeper ou Kyverno) bloquant le déploiement d'images non signées ou contenant des vulnérabilités critiques, et un processus de **revue périodique** des modèles tiers utilisés en production pour détecter les modèles abandonnés ou compromis après leur déploiement initial.



Sécurité Inférence Supply Chain ML MLSecOps Pratique



7 MLSecOps en Pratique : Pipeline de Référence

Après avoir analysé les menaces et les contrôles de sécurité spécifiques à chaque phase du pipeline MLOps, cette section synthétise les bonnes pratiques dans une **architecture de référence MLSecOps** déployable en production. L'objectif est de fournir aux RSSI et aux équipes ML un cadre concret et actionnable pour intégrer la sécurité dans chaque étape du cycle de vie des modèles, sans créer de friction excessive qui ralentirait l'innovation.

Architecture de référence MLSecOps

L'architecture MLSecOps de référence s'articule autour de quatre couches intégrées. La couche **Source Control** utilise Git avec des branches protégées, des revues de code obligatoires, et le scanning pré-commit (secrets, PII, code dangereux) via des hooks **pre-commit** configurés avec Gitleaks, Bandit et des règles Semgrep personnalisées pour le ML. La couche **CI/CD ML** orchestre l'entraînement et la validation avec des gates de sécurité automatisées : scan des dépendances (pip-audit), validation des datasets (integrity checks, outlier detection), test de robustesse adversariale (ART), et scan des modèles produits

(ModelScan). La couche **Model Registry** centralise les modèles validés avec des contrôles d'accès RBAC, des signatures cryptographiques (cosign), et un workflow d'approbation multietapes (Staging, Canary, Production). La couche **Serving Infrastructure** déploie les modèles derrière un proxy de sécurité intégrant la validation d'entrées, le filtrage de sorties, le rate limiting, et le monitoring de drift, le tout connecté au SIEM de l'organisation pour une visibilité unifiée.

Intégration des contrôles dans CI/CD

L'intégration de la sécurité dans le pipeline CI/CD ML repose sur trois types de gates. Les **pre-commit gates** bloquent les commits contenant des secrets, des dépendances vulnérables, ou des patterns de code dangereux — ces vérifications s'exécutent localement en quelques secondes et constituent la première ligne de défense. Les **training gates** vérifient l'intégrité des données d'entraînement, exécutent un scan de poisoning sur les nouveaux échantillons, valident que les hyperparamètres sont dans les plages approuvées, et testent la robustesse du modèle entraîné contre une suite d'attaques adversariales standardisées. Les **deployment gates** vérifient la signature du modèle, scannent l'image de conteneur de serving, valident que les métriques de performance sont dans les seuils acceptables, et exécutent un canary test sur un sous-ensemble de trafic réel avant le déploiement complet. Chaque gate produit une **attestation SLSA** signée qui est stockée dans le registre de transparence, créant une chaîne de confiance auditable de bout en bout. En cas d'échec d'un gate, le pipeline s'arrête, notifie l'équipe ML et l'équipe sécurité, et crée un ticket d'investigation automatique.

Monitoring continu et compliance

Le **monitoring continu** en production surveille trois dimensions critiques. Le **model drift** — déviation entre la distribution des données en production et celle des données d'entraînement — est surveillé via des tests statistiques automatisés (KL divergence, Population Stability Index) avec des seuils d'alerte configurables. La **performance degradation** est mesurée par des métriques métier (précision, rappel, latence, taux d'erreur) comparées à des baselines établies lors de la validation. La **détection d'attaques** analyse les patterns de requêtes en temps réel pour identifier les tentatives d'extraction de modèle, d'empoisonnement en ligne, ou d'exploitation adversariale. Pour la **compliance**, le pipeline doit répondre aux exigences de l'**AI Act européen** (classification des risques, documentation technique, monitoring post-déploiement), du **NIST AI Risk Management Framework** (governer, mapper, mesurer, gérer les risques IA), et des standards sectoriels applicables. Un dashboard unifié agrège ces métriques de sécurité, de performance et de compliance pour fournir au RSSI une vue en temps réel de la posture de sécurité de tous les modèles en production. Pour approfondir, consultez Pydantic AI et les Frameworks d'Agents Type-Safe en 2026.

Checklist RSSI : 15 contrôles essentiels

Pour conclure ce guide, voici les **15 contrôles essentiels** que tout RSSI doit vérifier pour un pipeline ML sécurisé :

- ▹**1. Data provenance** — Traçabilité complète de chaque dataset avec checksums SHA-256 et registre de sources immuable.
- ▹**2. PII scanning** — Détection et anonymisation automatique des données personnelles dans les datasets d'entraînement.
- ▹**3. Poisoning detection** — Analyse d'outliers et validation statistique des données avant chaque entraînement.
- ▹**4. Environment isolation** — Conteneurs renforcés avec AppArmor/SELinux et isolation GPU pour les workloads d'entraînement.
- ▹**5. Dependency scanning** — pip-audit et Snyk dans le CI/CD avec lock files et vérification des hashes.
- ▹**6. SafeTensors enforcement** — Interdiction du format Pickle, migration vers SafeTensors obligatoire.
- ▹**7. Model signing** — Signature cosign/Sigstore de chaque modèle avec vérification au déploiement.
- ▹**8. Backdoor scanning** — Analyse spectrale et ModelScan sur chaque modèle avant la mise en production.
- ▹**9. Adversarial testing** — Suite de tests de robustesse adversariale (IBM ART) dans le pipeline de validation.
- ▹**10. Input validation** — Proxy de validation avec détection OOD et anti-prompt-injection devant chaque endpoint.
- ▹**11. Output filtering** — PII scanning, filtrage de toxicité et guardrails sur les sorties du modèle.
- ▹**12. Rate limiting ML** — Limitation adaptative avec détection de patterns d'extraction de modèle.
- ▹**13. Drift monitoring** — Surveillance continue du data drift et du concept drift avec alertes automatiques.
- ▹**14. ML-BOM** — Bill of Materials ML complet (SBOM + data card + model card) pour chaque modèle en production.
- ▹**15. SIEM integration** — Remontée des alertes ML au SOC avec playbooks de réponse spécifiques à l'IA.

La sécurisation d'un pipeline MLOps est un processus itératif qui doit évoluer avec le paysage des menaces. Les organisations les plus matures adoptent une approche **risk-based**, priorisant les contrôles en fonction de la criticité des modèles et de leur exposition aux menaces. Un modèle de classification d'images interne n'exige pas le même niveau de protection qu'un LLM exposé sur Internet traitant des données clients. L'essentiel est de **commencer maintenant** avec les contrôles fondamentaux — SafeTensors, model signing, dependency scanning — et d'enrichir progressivement la posture de sécurité à mesure que l'organisation gagne en maturité MLSecOps. Le coût de la sécurisation d'un pipeline ML est

toujours inférieur au coût d'un modèle compromis en production — que ce soit en termes de réputation, de conformité réglementaire, ou de confiance des utilisateurs dans les systèmes IA de l'organisation.

Ressources open source associées

HF Dataset mlops-infrastructure-fr HF Space mlops-infrastructure-explorer (démo)

Besoin d'un accompagnement expert ?

Nos consultants en cybersécurité et IA vous accompagnent dans vos projets. Devis personnalisé sous 24h.

Références et ressources externes

- OWASP LLM Top 10 — Les 10 risques majeurs pour les applications LLM
- MITRE ATLAS — Framework de menaces pour les systèmes d'intelligence artificielle
- NIST AI RMF — AI Risk Management Framework du NIST

- arXiv — Archive ouverte de publications scientifiques en IA
- HuggingFace Docs — Documentation de référence pour les modèles de ML

Pour approfondir ce sujet, consultez notre outil open-source ml-model-security-audit qui facilite l'évaluation de la sécurité des modèles ML.

Sources et références : ArXiv IA · Hugging Face Papers

FAQ

Qu'est-ce que Sécuriser un Pipeline MLOps ?

Le concept de Sécuriser un Pipeline MLOps est détaillé dans les premières sections de cet article, qui couvrent les fondamentaux, les enjeux et le contexte opérationnel. Pour un accompagnement sur ce sujet, contactez nos experts.

Pourquoi Sécuriser un Pipeline MLOps est-il important en cybersécurité ?

La compréhension de Sécuriser un Pipeline MLOps permet aux équipes de sécurité d'améliorer leur posture défensive. Les sections « Table des Matières » et « 1 Les Menaces Spécifiques aux Pipelines ML » détaillent les raisons de cette importance. Pour un accompagnement sur ce sujet, contactez nos experts.

Comment mettre en œuvre les recommandations de cet article ?

Les recommandations pratiques sont détaillées tout au long de l'article, avec des commandes, des outils et des méthodologies éprouvées. La section « Conclusion » fournit une synthèse actionnable. Pour un accompagnement sur ce sujet, contactez nos experts.

Conclusion

Cet article a couvert les aspects essentiels de Table des Matières, 1 Les Menaces Spécifiques aux Pipelines ML, 2 Sécurité des Données d'Entraînement. La mise en pratique de ces recommandations permet de renforcer significativement la posture de securite de votre organisation.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr