NY - Avril 2023

Sécurité et vulnérabilités de l'apprentissage automatique

Avec l'expansion continue de l'usage de l'apprentissage automatique (Machine Learning - ML) dans de nombreux domaines, dont certains critiques, la sécurité des modèles de ML devient de plus en plus importante.

Ce rapport vise à présenter une vue d'ensemble des vulnérabilités courantes des modèles d'apprentissage automatique et des attaques connues qui exploitent ces vulnérabilités. Nous allons rapidement passer en revue les attaques fréquentes et connues, en détaillant leur impact sur la sécurité des modèles d'apprentissage automatique. Nous choisissons d'organiser ces attaques en trois grandes classes :

- les attaques sur l'intégrité des données
- les attaques sur la confidentialité des données
- les attaques sur la disponibilité (availability) du modèle

Nous détaillerons les types d'attaques et présenterons également des solutions de sécurité pour les prévenir ou les limiter.

Bien que le but de ce rapport soit de fournir une vue d'ensemble des attaques et des défenses en matière de sécurité des modèles d'apprentissage automatique, celui-ci a aussi pour objectif de fournir des liens vers des articles (de recherche) permettant d'approfondir les concepts présentés synthétiquement dans ce rapport.
Cependant, il est également important de noter que la recherche dans ce domaine est très récente et en constante évolution et que les techniques de défense et d'attaque mentionnées dans ce rapport ou dans les articles cités peuvent ne pas être à jour ou applicables dans tous les contextes.

Sommaire

Définitions

Connaissances de l'attaquant

Dans le contexte de la sécurité des modèles de Machine Learning, les termes "white box", "grey box" et "black box" sont couramment utilisés pour décrire le niveau de connaissance qu'un attaquant a sur le modèle (désigné ici par box).

Un modèle "white box" est un modèle dont les détails internes sont connus de l'attaquant. Cela peut inclure la connaissance des paramètres du modèle, le processus d'apprentissage, des données d'entraînement utilisées, etc. Les attaquants qui disposent de cette connaissance peuvent exploiter les faiblesses du modèle de manière plus efficace.

Un modèle "grey box" est un modèle dont certains détails internes sont connus de l'attaquant, mais pas tous. Cela peut inclure la connaissance de certaines parties du processus d'apprentissage ou d'une partie des données d'entraînement utilisées. Les attaquants qui disposent de cette connaissance peuvent avoir une compréhension partielle du modèle, mais ils ne disposent pas de toutes les informations nécessaires pour mener une attaque complète.

Enfin, un modèle "black box" est un modèle dont les détails internes sont inconnus de l'attaquant. Dans ce cas, l'attaquant doit se fier uniquement aux entrées et sorties du modèle pour mener une attaque. Les attaques sur les modèles "black box" sont en général plus difficiles à mener pour l'attaquant mais pas impossible.

Positionnement de l'attaque

Une attaque peut survenir à différentes étapes dans le processus de déploiement du modèle de ML. On peut distinguer deux moments clés :

- Lors de l'entraînement (training time) : une attaque au moment de l'entraînement signifie que l'attaquant est en mesure d'influencer directement le jeu de données d'entraînement.
- Lors de la prédiction (inference) : une attaque au moment de l'inférence signifie que l'attaquant ne peut modifier que l'entrée courante du modèle (celle dont on veut réaliser la prédiction).

En général une attaque lors de l'entraînement est plus difficile à réaliser mais les dégâts sont plus importants qu'une attaque lors de la prédiction.

Attaques sur l'intégrité des données

Ces attaques peuvent avoir des conséquences variées en fonction de l'objectif de l'attaquant mais on peut établir la classification suivante :

- Réduction de la confiance du modèle (confidence reduction) : la classe reste inchangée mais l'attaque impacte la confiance du modèle (une certaine classe sera prédite avec une probabilité moins importante).
- Mauvaise classification (misclassification) : changement de classe sans objectif spécifique (vers n'importe quelle autre classe différente de la classe originale)
- Mauvaise classification ciblée (Targeted misclassification) : changement de classe avec un objectif spécifique (changement vers une classe précise)
- Mauvaise classification source/cible (Source/Target misclassification) : modifie une entrée d'une classe spécifique (source) pour qu'elle soit classée dans une catégorie de sortie spécifique (cible)

La littérature se concentre en grande partie sur des problèmes de classification, c'est pour cela que l'on retrouve le lexique de la classification. Cependant, des papiers tentent d'élargir la recherche aux problèmes de régression https://ceur-ws.org/Vol-2916/paper_17.pdf.

Data poisoning (empoisonnement de données)

Le but de cette attaque est de modifier les données d'entraînement du modèle pour que celui se trompe lors de ses décisions.
La modification des données peut concerner la modification de données existantes ou alors l'ajout de nouvelles données.

Si l'accès aux données d'entraînement peut sembler difficile, en réalité plusieurs méthodes peuvent le permettre :

- Accès direct aux données d'entraînement
- Utilisation d'un dataset d'entraînement déjà empoisonné
- Utilisation d'un modèle par transfert learning déjà empoisonné
- Feedback malveillant
- Mise en ligne de données malveillantes avec l'espoir qu'elles soient par la suite utilisées pour entraîner des modèles

Un exemple de ce type d'attaques a été donné par Google qui a indiqué que certains groupes de spammeurs essayent d'empoisonner le modèle chargé du filtrage des spams sur Gmail. Ces derniers signalent, sur une période courte, d'énormes quantités de mails spams comme n'étant pas du spam.

Il est fréquent de voir une classification des attaques de type "data poisoning" en fonction du niveau d'accès de l'attaquant.
https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db
Les niveaux d'accès sont classés du plus faible niveau d'accès au plus important :

- Empoisonnement indirect (indirect poisoning) : l'attaquant n'a pas d'accès aux modèles et aux données mais il peut empoisonner des données utilisées par le

modèle avant le prétraitement. Cela peut intervenir dans le cadre de l'utilisation de transfer learning.
- Injection de données (data injection) : l'attaquant peut seulement ajouter de nouvelles données d'entraînement.
- Modification de données (data modification) : l'attaquant peut ajouter, modifier ou supprimer des données d'entraînement. On retrouve ici les attaques de modification d'étiquettes.
- Corruption logique (logic corruption) : l'attaquant peut directement modifier le modèle et donc la manière dont il apprend.

Une attaque fréquente de type empoisonnement est l'attaque par porte dérobée (backdoor attack). Elle permet à un attaquant d'insérer une vulnérabilité dans un modèle d'apprentissage automatique afin de pouvoir l'utiliser ultérieurement. Cette vulnérabilité est généralement introduite pendant la phase d'apprentissage en ajoutant ou modifiant des données malveillantes au jeu de données d'entraînement, ce qui amène le modèle à apprendre à reconnaître un motif spécifique. Ainsi, lorsque le modèle est déployé, l'attaquant peut exploiter cette porte dérobée et manipuler les prédictions du modèle en utilisant une entrée comportant ce motif spécifique. Cette porte dérobée compromet ainsi la sécurité et l'intégrité du modèle.

Cette attaque peut être difficile à détecter car l'insertion de la porte dérobée peut se faire sur quelques données seulement et ne pas altérer les autres données.

L'exemple classique pour illustrer les attaques de type porte dérobée est celui du panneau de signalisation stop. Ici la porte dérobée est un petit point situé sur le panneau qui modifie la prédiction du modèle en changeant la classification du panneau stop en un panneau limitation de vitesse. Ainsi lorsque le modèle rencontrera un panneau avec un point, il se trompera sur sa classification en raison de l'insertion de cette porte dérobée.
Même s'il peut s'avérer difficile de détecter cette porte dérobée au vue du nombre de données présentes dans les datasets, certains travaux proposent des techniques pour complètement cacher ce type d'attaque : https://arxiv.org/abs/1910.00033.

On peut citer d'autres types attaques ayant pour objectif l'empoisonnement de données :

On appelle attaque de type clean-label (étiquette propre) toutes les attaques dont les données empoisonnées conservent une étiquette plausible/valide.

Clean-label backdoor attack (attaque par porté dérobée à étiquette propre) : ce type d'attaque a pour objectif l'ajout d'une porte dérobée tout en conservant une étiquette plausible. Ainsi l'attaquant n'a pas à se soucier de l'étiquette de sa porte dérobée.
https://people.csail.mit.edu/madry/lab/cleanlabel.pdf

Attaque par feature collision (collision de caractéristiques) : attaque de type clean-label qui suppose que l'attaquant n'a aucune connaissance des données d'entraînement mais possède une connaissance sur le modèle et ses paramètres.
L'attaquant crée une donnée empoisonnée à partir de très légères modifications (pour ne pas se faire détecter) d'une donnée réelle d'une classe c. Le but est de rendre cette donnée

très proche de la frontière de décision afin qu'elle puisse entrer en collision avec une donnée cible dans une classe différente de la classe c.
https://arxiv.org/abs/1804.00792

Cette vidéo https://www.youtube.com/watch?v=a92SWvqgMCE couvre en 15 minutes de nombreux concepts concernant les attaques par empoisonnement.

Cet article https://arxiv.org/abs/2210.09671 étudie l'efficacité des poisons et identifie les causes de cette efficacité.

Les attaques par empoisonnement des données sont de plus en plus utilisées aujourd'hui comme obfuscation pour tromper les algorithmes de recommandation et protéger sa vie privée.

Plusieurs mécanismes de défense peuvent être mis en place pour éviter ce type d'attaque :

- S'assurer que nos données ne proviennent pas uniquement de la même source (utilisateurs, IP, …)
- Vérifier la distribution de données. En particulier si de nouvelles données sont ajoutées constamment, il peut être intéressant de vérifier que ces distributions ne varient pas brusquement
- Si on met en place une possibilité d'améliorer le modèle via le retour (feedback) utilisateur alors il faut s'assurer que ce retour soit combiné et mis en relation avec d'autres sources avant d'influencer le modèle. Cela permet d'éviter un feedback malveillant.
- Si le poison est très différent de nos données d'entraînement alors des détections de données aberrantes (outliers) et d'anomalies peuvent s'avérer efficaces
- La méthode STRIP (STRong Intentional Perturbation) qui consiste à modifier légèrement des entrées et à observer la diversité des classes prédites pour ces entrées modifiées. Si la diversité est faible alors cette entrée est probablement empoisonnée : https://arxiv.org/abs/1902.06531
- D'autres méthodes sont listées dans la partie attaques sur la disponibilité du modèle

Evasion attack/adversarial examples

Les attaques par évasion aussi appelées les exemples adversaires (evasion attack or adversarial examples) sont des techniques malveillantes visant à tromper les modèles d'apprentissage automatique. Ces attaques consistent à modifier intentionnellement des données en entrée pour que le modèle se trompe lors de la prédiction.
Le danger des ces entrées adversaires est qu'elles sont parfois indétectables par l'œil humain et indistinguables des entrées "classiques" mais pour autant elles parviennent à induire le modèle en erreur lors de la prédiction.
Une attaque par évasion trompe le modèle en perturbant la donnée d'entrée lors de la phase de prédiction (inférence) tandis qu'une attaque par empoisonnement trompe le modèle en modifiant les données d'entraînement lors de la phrase d'entraînement.

Des exemples de ce type d'attaques ont été démontrés plusieurs fois et semblent être particulièrement efficaces sur de nombreux datasets. Les perturbations de l'image en entrée peuvent être mineures mais peuvent changer complètement la prédiction d'un modèle ainsi

que sa confiance dans sa prédiction. De plus, des exemples montrent que des objets physiques peuvent jouer le rôle des perturbations.

https://towardsdatascience.com/evasion-attacks-on-machine-learning-or-adversarial-examples-12f2283e06a1
De plus, comme montré dans cet article, les attaques par évasion peuvent concerner de nombreux problèmes :

- Classification d'image
- Détection d'objets dans un flux vidéo
- Transcription audio
- Traitement du langage naturel
- Apprentissage par renforcement

Les méthodes pour réaliser une attaque par évasion sont très nombreuses, une liste non exhaustive de ces méthodes et leurs implémentation pourra être trouvées dans Adversarial Robustness Toolbox :
https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/evasion.html

Cet article
https://towardsdatascience.com/evasion-attacks-on-machine-learning-or-adversarial-examples-12f2283e06a1 propose une classification des méthodes d'attaques par évasion en 5 catégories différentes :
Les attaques par force brute (brute force) sont les attaques les plus simples et les plus naïves pour générer des exemples adverses. Ils n'utilisent pas d'optimisation pour créer des exemples adverses, mais se contentent d'appliquer des perturbations simples comme des rotations, translations ou l'ajout de bruit gaussien. Ces attaques sont souvent moins efficaces que les autres types d'attaques mais peuvent tout de même réussir à tromper certains modèles.

Les méthodes utilisant le gradient : ces méthodes ont besoin d'accéder au gradient du modèle pour pouvoir créer un exemple adversaire qui maximise la fonction de perte.
Ces méthodes sont donc des white-box attacks car elles affectent des modèles de type white-box (définis précédemment).
On retrouve dans cette classe les méthodes suivantes :

- Fast Gradient Method (FGM) : originellement développé pour la norme infinie (Fast Gradient Sign Method - FGSM), la méthode est aujourd'hui implémentée pour fonctionner avec d'autres normes (L1 et L2).
- Projected Gradient Descent (PGD) : utilise une méthode d'optimisation itérative pour calculer la perturbation.
- Carlini and Wagner (CW) : considérée comme l'une des attaques les plus performantes, elle a notamment réussi à contourner plusieurs méthodes de défenses contre les attaques par évasion. En utilisant le gradient, elle recherche par optimisation la perturbation minimale nécessaire pour modifier la sortie du modèle pour garder l'exemple adversaire aussi proche que possible de l'image originale.

Une description de ces méthodes pourra être trouvée dans cet article :
https://medium.com/swlh/gradient-based-adversarial-attacks-an-introduction-526238660dc9

Les méthodes utilisant le score de confiance (confidence score) : ces méthodes se basent sur le score de confiance de classification en sortie pour estimer les gradients du modèle. Puis une optimisation similaire aux méthodes utilisant le gradient est mise en place. Ces méthodes ne nécessitent pas de connaissance du modèle et sont donc des attaques de type black-box.
On retrouve ici des méthodes comme Zeroth-Order Optimization (ZOO) ou Simultaneous Perturbation Stochastic Approximation (SPSA).

Les attaques par étiquettes dures (Hard label attacks) se basent uniquement sur l'étiquette de sortie produite par le modèle sans avoir besoin des scores de confiance. Cela rend l'attaque plus simple, mais aussi plus réaliste, car de nombreux systèmes ne fournissent pas de scores de confiance avec leurs prédictions. L'une des méthodes les plus puissantes dans cette catégorie est Boundary/Decision-based Attack.

Les attaques par modèle de substitution (surrogate model) interviennent lorsque l'attaquant n'a pas accès aux éléments internes du modèle mais souhaite mener une attaque de type white-box. Pour cela, il peut essayer de reconstruire le modèle cible sur sa machine en utilisant une attaque d'extraction de modèle (model extraction attack), il peut essayer de deviner l'architecture et les données utilisées si le modèle est entraîné pour un problème standard ou il peut utiliser un modèle classique pour la tâche souhaitée.

Pour la génération d'exemples adversaires, on pourra consulter https://github.com/cleverhans-lab/cleverhans et https://github.com/advboxes/AdvBox.
De nouvelles approches sont envisagées pour générer des exemples adversaires en utilisant par exemple des modèles entraînés pour la génération de ce type d'entrées adversaires. Par exemple, des réseaux de neurones adversaires générateurs (GAN) ont été utilisés https://arxiv.org/abs/1801.02610.

Aujourd'hui, on utilise l'attaque AutoAttack pour évaluer la robustesse d'un modèle à des attaques adversaires. Cette attaque consiste à appliquer plusieurs attaques adversaires différentes et évaluer la capacité du modèle à résister à ces attaques.
Cela permet de fournir une mesure plus complète de la robustesse des modèles.
Les auteurs d'AutoAttack conseillent d'utiliser cette méthode comme un test minimal pour évaluer la qualité de toute nouvelle défense proposée. De plus, les résultats de l'évaluation obtenus avec AutoAttack permettent aussi d'identifier les techniques de défense les plus performantes contre les attaques adverses.
https://arxiv.org/abs/2003.01690 et https://github.com/fra31/auto-attack.

RobustBench est une plateforme qui fournit un ensemble de benchmarks standardisés pour évaluer la robustesse des modèles face aux attaques adversaires (notamment AutoAttack). Cela permet de comparer simplement et efficacement la performance de différentes défenses sur un même ensemble de données https://robustbench.github.io/index.html.

Plusieurs défenses peuvent être envisagées dans un contexte d'attaque par évasion. On peut distinguer des défenses basées sur la détection où l'objectif est de détecter les exemples adversaires pour pouvoir les éliminer et des défenses basées sur la robustesse dont l'objectif est d'améliorer la performance du modèle sur des exemples adversaires et

ainsi obtenir une performance proche de celle obtenue sur des exemples classiques https://arxiv.org/abs/1911.05268.
Concernant les défenses basées sur la détection, on peut citer :

- Adversarial classification ou Adversary Detector Networks : utiliser un réseau de détection pour classifier les images comme naturelles ou adversaires. Ce réseau est entraîné sur un ensemble de données contenant des exemples adversaires et naturels et peut ainsi parvenir à détecter certains exemples adversaires.
- Des méthodes utilisant des analyses statistiques : par exemple on peut utiliser l'analyse en composantes principales (PCA) ou de la détection d'outliers qui peuvent permettre d'identifier les exemples adversaires qui ne suivent pas la distribution naturelle des données. D'autres méthodes peuvent être utilisées comme l'estimation par noyau (Kernel Density Estimation - KDE) et l'estimation de l'incertitude bayésienne (Bayesian Uncertainty Estimation - BUE).
- Des méthodes modifiant la fonction de perte utilisée. Notamment, une fonction appelée entropie croisée inverse (Reverse-Cross Entropy - RCE) permet de produire un classifieur capable de détecter les exemples adversaires.
- Des méthodes se basant sur la compression des entrées. Par exemple, Feature Squeezing est une méthode qui consiste à comparer les prédictions d'un modèle sur des entrées non modifiées et sur des entrées modifiées en supprimant certaines caractéristiques inutiles (feature squeezing) et si la différence entre ces deux entrées est supérieure à un seuil alors l'entrée est considérée comme attaque adversaire. Cette méthode tente de réduire la surface d'attaque en réduisant la dimension des caractéristiques.

Concernant les défenses basées sur la robustesse, on peut citer :

- Les défenses utilisant l'augmentation de données (data augmentation). Notamment la génération de données adversaires à partir de données connues pour être des exemples adversaires et l'entraînement du modèle sur ces données est souvent utilisé. On appelle cette méthode l'entraînement adversaire (adversarial training). Le but est de rendre le modèle plus robuste aux attaques futures. Plusieurs techniques de génération peuvent être envisagées pour générer ces nouvelles données.
- Les méthodes se basant sur la régularisation. Par exemple, la perte contractive, aussi utilisée dans les Deep Contractive Networks (DCN), peut être pertinente dans le cas de la robustesse aux perturbations adverses. En effet, avec cette perte, le modèle apprend à ignorer les petites variations qui peuvent ne pas être significatives pour la prédiction de la sortie.
  Une autre défense pertinente utilise la distillation des connaissances (knowledge distillation) qui consiste à transférer des connaissances d'un grand modèle à un plus petit. Cette méthode appelée defensive distillation permet d'introduire de la régularisation et d'obtenir un modèle plus petit, plus rapide, moins complexe et donc plus résistant aux attaques.
- Les méthodes appliquant des transformations sur l'image avant la classification. On peut aussi ajouter de l'aléatoire, en appliquant systématiquement des transformations aléatoires sur notre image avant la classification pour rendre le calcul du gradient de la fonction de perte plus difficile.
  Des méthodes tentent aussi de "purifier" un exemple adversaire en le transformant en donnée propre (non adversaire). Ces méthodes sont parfois regroupées sous le

terme reconstruction d'entrée (input reconstruction) et on trouve la méthode PixelDefend.
- Les méthodes ensemblistes peuvent aussi être utilisées, cela consiste à combiner plusieurs modèles pour renforcer la robustesse globale du système. On peut notamment combiner des modèles avec des architectures différentes ou entraînés sur des données différentes.
- Certaines défenses se concentrent sur le problème d'optimisation à résoudre pour entraîner le modèle. En particulier, cet article https://arxiv.org/abs/1706.06083 analyse les attaques et propose une optimisation robuste comme solution de défense.

Enfin, il convient de noter que les méthodes de défense peuvent être combinées (en parallèle ou en séquence) pour renforcer la protection contre les attaques adverses.

La plateforme RobustBench, présentée précédemment, permet d'évaluer l'efficacité de nombreuses méthodes de défense.
D'après les résultats actuels, les méthodes de défenses les plus efficaces semblent être celles utilisant l'augmentation de données avec la génération de nouvelles données appelées données synthétiques.

Attaques sur la confidentialité

Attaques sur la disponibilité du modèle

Ces attaques concernent toutes les violations qui peuvent entraîner des réductions de qualité ou d'accès au modèle de ML.
Cela peut concerner par exemple un ralentissement des réponses de la part du modèle voire une indisponibilité totale mais aussi ces attaques peuvent vouloir rendre le modèle inutilisable en altérant ses performances de sorte que les prédictions du modèle seront totalement erronées.
Cette forme d'attaque peut avoir des conséquences très graves, en particulier si le modèle est utilisé pour des applications critiques et si des prédictions pertinentes et/ou rapides sont nécessaires.

Attaque par déni de service (DoS)

Ce type d'attaque est très proche de l'attaque par déni de service (Denial of Service Attack - DoS) "classique" qui tente de rendre un site Web indisponible en envoyant énormément de trafic sur ce dernier.
Aujourd'hui on entend surtout parler de DDOS pour Distributed DoS car le trafic est issu de plusieurs sources d'où le terme distribué.

Ainsi une attaque par déni de service sur un modèle de ML va envoyer de nombreuses requêtes à ce dernier pour tenter de le rendre indisponible.
Cependant, contrairement aux cas classiques, certaines requêtes spécifiques semblent plus efficaces pour saturer un modèle car elles sont plus gourmandes en calcul.
https://www.cl.cam.ac.uk/~is410/talks/shumailov_availability_april2021.pdf
https://arxiv.org/abs/2006.03463

Les résultats présentés dans l'article semblent prometteurs et ce type d'attaque utilisant ces requêtes spécifiques (nommées sponges dans l'article) fonctionnent sur de nombreux modèles.

Une défense simple peut-être mise en place pour éviter ce type d'attaque :

- limiter le nombre de requêtes d'un utilisateur pour une période donnée
- fixer un temps maximum pour le calcul d'une prédiction par le modèle ainsi nous pouvons éviter les requêtes très chronophage. Nous pouvons fixer ce temps à partir d'exemples observés sur nos données initiales.

Data poisoning

L'attaque par empoisonnement des données peut aussi être considérée comme une attaque sur la disponibilité du modèle lorsque les conséquences de l'attaque entraînent une baisse significative de la performance des prédictions du modèle.

Plusieurs méthodes, en plus de celles présentées dans la partie attaque sur l'intégrité des données, peuvent être utilisées pour mettre en place une défense face à ces attaques :

- Golden dataset : construire un (golden) dataset qui contiendra une bonne représentation de nos données et des données empoisonnées. Ainsi on peut fixer un

score qu'un modèle doit obtenir sur ce golden dataset pour pouvoir être mis en production. Cela pourra mettre en évidence la baisse de performance d'un modèle après ajout de nouvelles données par exemple, ce qui pourra nous alerter et ainsi entraîner des vérifications approfondies sur ces données.

- Optimisation robuste : modifier la procédure d'optimisation d'un modèle pour le rendre résistant à ce type d'attaque. Cependant, on peut remarquer qu'il est important de trouver un compromis entre la robustesse du modèle face à des données empoisonnées et sa performance globale.

D'autres méthodes peuvent être envisagées comme https://arxiv.org/abs/2104.06744.

Compléments

Adversarial Robustness Toolbox (ART)

Adversarial Robustness Toolbox (ART) est une librairie Python concernant la sécurité de l'apprentissage automatique. Elle prend en charge de nombreux frameworks d'apprentissage automatique comme TensorFlow, PyTorch, Keras et scikit-learn. Elle est compatible avec une majorité de type de données et de tâches d'apprentissage automatique.
ART propose de nombreuses implémentations d'attaques et de défenses présentées dans ce rapport et bien d'autres.
Cette librairie permet de mettre en place des défenses et d'évaluer des modèles d'apprentissage automatique simplement.
https://github.com/Trusted-AI/adversarial-robustness-toolbox

Scalabilité des attaques

L'utilisation croissante du "transfer learning", qui consiste à utiliser des modèles pré-entraînés pour résoudre des tâches similaires dans des domaines différents, peut également introduire des vulnérabilités de sécurité supplémentaires.
En effet, les modèles pré-entraînés peuvent contenir des biais, des failles de sécurité et des informations sensibles, qui peuvent être transférés à d'autres modèles.
La centralisation de l'apprentissage dans quelques modèles implique une grande responsabilité et il est donc important d'adopter des pratiques de sécurité appropriées lors de l'utilisation de techniques de "transfer learning", telles que l'audit des modèles pré-entraînés et la mise en place de mécanismes de détection d'attaques.

Audit

Aujourd'hui, l'audit des modèles de ML est une pratique importante pour garantir la transparence et la responsabilité des modèles de ML et permettre le développement de modèles éthiques et responsables.
Cela concerne aujourd'hui l'identification de biais, de problèmes liés à la qualité des données mais aussi l'évaluation de la performance et de la pertinence des prédictions du modèle https://www.auditingalgorithms.net/.

Cependant, il est maintenant important de considérer aussi l'aspect sécurité des modèles de ML pour pouvoir identifier les vulnérabilités potentielles des modèles de ML qui pourraient être exploitées.
Il est ainsi important de tester des attaques sur nos modèles de ML à l'image des tests d'intrusions (pentest) et d'inclure des défenses en fonction des résultats de ces attaques. Il est important que ces tests soient effectués à différentes étapes du cycle de vie du modèle.
On pourra consulter https://github.com/mitre/advmlthreatmatrix par MITRE qui regroupe les attaques sur les modèles de ML dans un cadre de référence de type ATT&CK.
Enfin, l'audit devra être inclus dans un processus continu qui vise à s'assurer que le modèle est protégé contre les dernières attaques.

Conclusion

En conclusion, la sécurité des modèles d'apprentissage automatique est devenue un enjeu crucial dans de nombreux domaines. Les différentes vulnérabilités et attaques potentielles présentées dans ce rapport soulignent la nécessité de prendre en compte la sécurité dès la conception des modèles de ML et de les auditer régulièrement. L'audit des modèles de ML permet d'identifier les éventuelles vulnérabilités et de les corriger avant qu'elles ne soient exploitées par des attaquants malveillants. Il est donc important pour les organisations et les entreprises de prendre des mesures pour renforcer la sécurité de leurs modèles d'apprentissage automatique et de s'assurer qu'ils sont résistants aux attaques potentielles. Les avancées rapides dans ce domaine nécessitent également une surveillance constante et une mise à jour régulière des mesures de sécurité pour protéger les modèles de ML contre les nouvelles menaces qui peuvent émerger.
Pour finir, l'audit des modèles de ML doit être une priorité pour assurer la fiabilité, la confidentialité et la sécurité des systèmes basés sur l'apprentissage automatique.

Sources

Classification des attaques et des défenses

- https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8269-draft.pdf
- https://linc.cnil.fr/fr/petite-taxonomie-des-attaques-des-systemes-dia
- https://www.mdpi.com/2076-3417/9/5/909
- https://elie.net/blog/ai/attacks-against-machine-learning-an-overview/
- https://arxiv.org/abs/2112.02797
- https://towardsdatascience.com/how-to-attack-machine-learning-evasion-poisoning-inference-trojans-backdoors-a7cb5832595c

Blog sur la sécurité des modèles de ML

- http://www.cleverhans.io/

Liste de ressources sur la sécurité du ML

- https://github.com/jiep/offensive-ai-compilation
- https://paperswithcode.com/area/adversarial