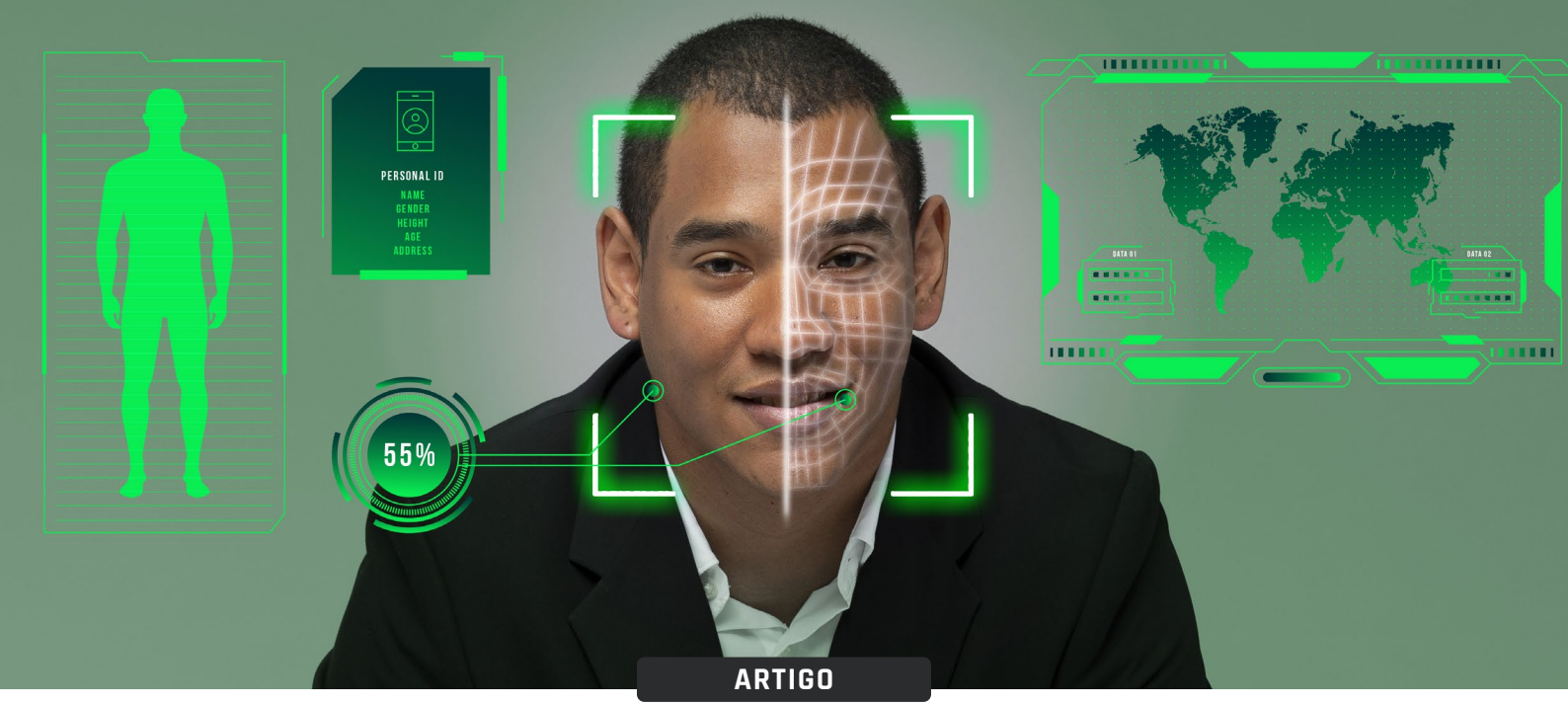

ARTIGO

A SEGURANÇA DE SOFTWARE NECESSITA DE ATUALIZAÇÃO QUANDO CONFRONTADA COM A REALIDADE DA APRENDIZAGEM AUTOMÁTICA DISTRIBUÍDA?

POR

Nuno Neves
Universidade de Lisboa
nuno@di.fc.ul.pt

O Aprendizado de Máquina (Machine Learning, ML) está a ter um impacto profundo na sociedade, transformando vários aspectos das nossas vidas. Ao analisar grandes conjuntos de dados (*datasets*), os algoritmos de ML podem fornecer recomendações personalizadas, melhorar diagnósticos médicos, otimizar processos empresariais e aprimorar experiências dos utilizadores. No entanto, a adoção generalizada do aprendizado de máquina também levanta preocupações éticas e sociais, incluindo questões relacionadas com a segurança e a privacidade.

A ML distribuída emergiu como um desenvolvimento relativamente recente, mas promissor, com o potencial de revolucionar várias áreas aplicacionais, embora acompanhada de novos desafios na cibersegurança. Torna-se assim necessário abordar esses desafios de uma forma eficaz, para garantir o desenvolvimento e implementação responsável das tecnologias de ML distribuída, maximizando os seus benefícios e mitigando os riscos potenciais para os indivíduos e a sociedade. Este aspecto é particularmente importante no ensino superior, onde os currículos devem adaptar-se às mudanças tecnológicas. É essencial incorporar as atualizações que garantam que os estudantes estão preparados para se tornarem profissionais competentes e responsáveis neste cenário em rápida evolução.

Vamos usar a Aprendizagem Federada (Federated Learning, FL) para identificar

e ilustrar benefícios e riscos específicos. A FL é um paradigma de aprendizagem distribuída que facilita o treino de modelos em vários dispositivos sem a necessidade de trocar os dados [McMahan17]. Esta abordagem inovadora trata de preocupações com a privacidade, como as delineadas no GDPR [GDPR16] e CCPA [Bukaty19], pois garante que os registros armazenados permanecem nos dispositivos respectivos, enquanto permite o treino colaborativo de um modelo global. Além disso, a FL melhora a generalização do modelo aproveitando do recolhimento descentralizado de dados, o que muitas vezes resulta em um conjunto de amostras mais diversificado. Essa diversidade contribui para uma melhor cobertura do espaço de entrada, aprimorando, em última instância, a capacidade do modelo de generalizar quando implantado em ambientes de produção.

Logo, a FL encontrou aplicação em um amplo espectro de tarefas. Exemplos como a condução autônoma e o prognóstico de doenças ilustram o seu papel crucial, ainda que numerosas outras aplicações menos críticas também explorem os seus benefícios. As plataformas como o Google GBoard para previsão e sugestão da próxima palavra e a Siri para o reconhecimento automático da fala são exemplos muito difundidos. Considerando em maior detalhe a área da saúde, as vantagens da FL tornam-se evidentes, especialmente ao enfrentar o desafio de diagnosticar doenças raras. Tipicamente, os hospitais têm poucos pacientes para cada doença rara, e os seus registros devem ser mantidos privados. Treinar um modelo com apenas esses registros levaria a que muitos diagnósticos fossem errados, uma vez que a precisão do modelo seria baixa. A FL aborda essa limitação ao permitir a colaboração de vários hospitais, cada um com alguns registros, mas que na globalidade já teriam uma dimensão apreciável, facilitando assim o desenvolvimento de modelos mais robustos.

Em mais detalhe, a FL opera da seguinte maneira: inicialmente, um modelo global é criado por um servidor central que é então distribuído por um subconjunto dos dispositivos, normalmente referidos como clientes ou participantes, onde ocorre o treino local. Durante esta fase, os parâmetros do modelo são atualizados com base no conjunto de dados disponíveis em cada dispositivo. Uma vez concluído o treino local, cada dispositivo transmite de volta para o servidor central as atualizações que ocorreram nos parâmetros (ou gradientes). Essas atualizações são agregadas no servidor para formar uma nova versão do modelo global. O processo é repetido por várias rodadas até que o modelo global atinja o desempenho desejado. Ao adotar esta abordagem descentralizada, a FL facilita o treino colaborativo de modelos enquanto protege a privacidade dos dados. Os registros armazenados localmente permanecem seguros, pois nunca são partilhados externamente aos dispositivos.

Contudo, a natureza distribuída da FL cria um ambiente ideal para entidades maliciosas (adversários) poderem manipular o comportamento do modelo global final [Fang20, Tolpegin20, Zhang22] ou tentar inferir informações sensíveis sobre

os dados de treino e/ou modelo [Yue23]. Como existe potencialmente o envolvimento de muitos dispositivos (dependendo da situação, entre algumas dezenas e as centenas de milhares [Kairouz21]), assegurar que todos exibem consistentemente um comportamento correto é uma tarefa extremamente difícil. Ademais, a detecção de uma conduta maliciosa apresenta desafios significativos, pois é inerentemente complexo distinguir entre as atualizações maliciosas e as válidas, uma vez que existe sempre alguma variabilidade decorrente da diversidade dos registros armazenados localmente. É essencial relembrar que os dados guardados nos dispositivos são não-i.i.d. (independentes e identicamente distribuídos), algo que é desejável e esperado, como mencionado anteriormente. Por estas razões, é vital compreender em maior detalhe, como é que este tipo de aplicações podem ser atacadas e como podem ser protegidas.

Vetores de Ameaça

De um ponto de vista genérico, a aprendizagem distribuída (e, em particular, a FL) está suscetível aos mesmos vetores de ameaça que nos são familiares, como os ataques à cadeia de fornecimento (*supply chain*) e os ataques on-line. No entanto, compreender efetivamente como essas ameaças se manifestam nos obriga à aquisição de conhecimentos especializados.

Por exemplo, as vulnerabilidades da cadeia de fornecimento aparecem na rede formada pelas entidades envolvidas no desenvolvimento, distribuição e manutenção do software. Essas ameaças podem surgir em várias etapas no desenvolvimento do software, incluindo a aquisição de componentes ou bibliotecas de terceiros, a integração de serviços ou dependências externas e a disseminação de atualizações de software. Exemplos comuns de ataque incluem, a introdução de código malicioso nas bibliotecas para que mais tarde este seja executado em conjunto com o resto do software, o comprometimento dos canais de distribuição das aplicações levando à disseminação de produtos adulterados ou falsificados, e a exploração de vulnerabilidades em componentes desenvolvidos por terceiros.

Todas estas ameaças são extensíveis ao software que utiliza ML distribuída. No entanto, surgem também várias ameaças especializadas que podem representar riscos consideráveis para a segurança. Alguns ataques eficazes incluem:

1. **Ataques de Envenenamento de Dados (Data Poisoning Attacks):** os adversários podem manipular os dados de treino para minar o desempenho dos modelos de ML. Uma vez que os conjuntos de dados demoram muito tempo a criar e exigem uma quantidade significativa de esforço, as organizações tendem a utilizar o que está disponível publicamente ou a adquiri-los de terceiros (pelo menos nas primeiras etapas do desenvolvimento do modelo). Além disso, esses conjuntos de dados frequentemente contêm um grande número de amostras,

tornando inviável a validação manual (por exemplo, verificar que as etiquetas corretas foram atribuídas às imagens). Logo, através da injeção de amostras cuidadosamente elaboradas nos dados de treino, os atacantes podem influenciar o comportamento final do modelo e comprometer a sua precisão ou robustez.

2. **Ataques de Envenenamento do Modelo (Model Poisoning Attacks):** na FL, enquanto o modelo é treinado, clientes controlados pelo adversário podem alterar maliciosamente as atualizações que enviam para o servidor central. Estas atualizações maliciosas podem ser criadas, por exemplo, através da manipulação do procedimento de treino local, ou modificando os hiperparâmetros, ou o critério que está a ser otimizado (loss function). Por fim, estas atualizações, quando agregadas, visam introduzir vulnerabilidades ou comportamentos maliciosos no modelo global. Logo, ao incorporar o modelo resultante numa aplicação, potencialmente pode-se causar comportamentos inesperados em condições específicas.
3. **Ataques de Inferência do Modelo (Model Inference Attacks):** se os atacantes tiverem acesso ao modelo final, podem tentar inferir informações confidenciais sobre os registros que foram utilizados durante o treino. O adversário usa as respostas produzidas pelo modelo alvo, como as suas predições sobre exemplos escolhidos, para fazer inferências probabilísticas sobre os dados de treino. Existem diversas variantes deste ataque, como aquelas que visam determinar se uma determinada amostra foi usada no treino, inferir atributos específicos como, por exemplo, informações demográficas, ou reconstruir exemplos particulares. Estes ataques comprometem a privacidade, especialmente em contextos em que são utilizados dados sensíveis, como nas atividades na área da saúde.
4. **Ataques de Reutilização do Modelo (Model Reuse Attacks):** o adversário tenta replicar um modelo alvo embora não tenha acesso aos seus parâmetros ou aos dados que foram usados no treino. Neste tipo de ataque, o adversário geralmente interage com o modelo alvo ao submeter amostras selecionadas de entrada e observar as correspondentes previsões de saída. Ao questionar estrategicamente o modelo alvo e ao analisar as suas respostas, o atacante irá construir um modelo substituto, que irá imitar o comportamento do modelo original de uma maneira muito precisa. Novamente, o objetivo é quebrar a confidencialidade, mas neste caso, do modelo propriamente dito.

Na FL, as ameaças on-line podem ser realizadas durante o treino por qualquer cliente que decida agir maliciosamente. Os ataques acima podem ser executados

com diferentes níveis de sucesso e dificuldade, uma vez que o atacante controla apenas um número limitado dos dispositivos envolvidos. Compreender as capacidades e os efeitos destas ameaças em cenários práticos é um desafio importante por si só, cuja resposta ainda não é clara, pois novas estratégias de ataque e de mitigação continuam a ser desenvolvidas a um ritmo elevado pelos pesquisadores.

Potenciais Defesas

Estas ameaças representam riscos significativos para a integridade, confidencialidade e disponibilidade dos sistemas, obrigando à implementação de mecanismos adequados de segurança, e requerendo um comportamento confiável de todas as entidades envolvidas no processo de desenvolvimento de software. Tratar destas ameaças requer a aplicação de medidas robustas, incluindo práticas seguras de manipulação de dados, revisão minuciosa do código, avaliação de vulnerabilidades, gestão de riscos na cadeia de fornecimento e monitoramento dos sistemas de ML.

No entanto, para tratar dos riscos próprios das aplicações distribuídas de ML, torna-se imperativo elaborar estratégias de defesa bem-adaptadas à ameaça específica e ao ambiente onde o software irá operar. Por exemplo, considerando a ameaça de envenenamento de dados, várias estratégias de mitigação podem ser exploradas:

1. **Filtragem e Pré-processamento de Dados:** baseia-se na utilização de técnicas de pré-processamento de dados para identificar e filtrar registros potencialmente maliciosos, antes de os incorporar no processo de treino. Contudo, é importante considerar que embora os clientes benignos possam aplicar este tipo de abordagem, ela é insuficiente num cenário de FL, pois os clientes maliciosos podem sempre optar por usar dados corrompidos.
2. **Métodos de Agregação Robustos:** o servidor utiliza algoritmos de agregação que são resilientes à influência de dados corrompidos, como a agregação de média aparada (trimmed mean aggregation). Por vezes, este tipo de abordagem pode levar a uma redução da precisão do modelo global, uma vez que os dados são diversos, introduzindo um compromisso de difícil gestão – um conflito entre a segurança e a utilidade do modelo.
3. **Privacidade Diferencial (Differential Privacy)**: a introdução de perturbações (ou ruído) nas atualizações do modelo pode garantir que as contribuições individuais não afetem significativamente o modelo resultante, reduzindo assim a eficácia dos ataques. Novamente, à medida que a quantidade de ruído aumenta, é alcançada uma melhor proteção, mas com o custo de uma redução na precisão.

4. **Deteção de Anomalias**: a utilização de métodos de detecção de anomalias no servidor serve para identificar desvios nas atualizações dos clientes, podendo reconhecer a presença de atividade maliciosa.

Normalmente, as organizações conseguem mitigar os riscos e proteger a integridade e confidencialidade dos seus ativos se utilizarem uma combinação de estratégias de defesa e adotarem as melhores práticas para o treino seguro de modelos. Todavia, esta é uma área ainda com muitas incertezas, existindo uma investigação importante no desenvolvimento de novas técnicas e metodologias para defender o software de ML contra as ameaças emergentes.

Curriculum de Segurança de Software

As competências incluídas no currículo de Segurança de Software [SBC23] mantêm-se pertinentes quando aplicadas à construção de aplicações distribuídas de ML. No entanto, estas devem ser evoluídas e adaptadas a este domínio emergente, caso contrário, a sua eficácia na proteção da informação e dos sistemas pode diminuir. Por exemplo, será que é possível delinear corretamente os requisitos de segurança de uma aplicação de ML sem o conhecimento dos ataques de envenenamento de dados? Será que os estudantes conseguem escolher mecanismos apropriados para garantir a confidencialidade se não compreenderem os ataques de reutilização de modelo? Podem os estudantes testar eficazmente um software de ML sem reconhecerem o potencial dum ataque de envenenamento do modelo? Parece-nos que a resposta "não" é a mais apropriada a todas estas questões. Por conseguinte, é imperativo integrar no currículo o conhecimento específico às ameaças na ML e as respectivas estratégias de mitigação, assegurando que os alunos estão equipados para enfrentar os desafios singulares colocados por este tipo de software.

Ademais, em cursos de engenharia e outros cursos com uma elevada formação prática, é fundamental proporcionar aos estudantes oportunidades para experimentar e avaliar os efeitos dos ataques em contextos diversos. O sucesso de um ataque muitas vezes depende das capacidades do adversário, e a restrição das mesmas leva a diferentes graus de eficácia. A implementação de medidas de segurança pode (ou não) mitigar significativamente o impacto dessas ameaças, limitando assim o seu potencial. Ao envolver os alunos em exercícios práticos que simulam cenários do mundo real e que encorajem a implementação de contramedidas, eles são levados a uma melhor compreensão dos princípios de segurança e preparados para enfrentar os respectivos desafios de forma eficaz.

Felizmente, no âmbito das aplicações de FL, surgiram algumas ferramentas para facilitar a experimentação e teste de mecanismos de segurança. Exemplos incluem o FedML [Han23] e o FADO [Rodrigues23]. O principal objetivo destas ferramentas é fornecer uma plataforma que simplifique a implementação, treino e avaliação de modelos usando FL.

Simultaneamente, elas fornecem implementações pré-construídas das classes de ataque essenciais e defesas inovadoras, permitindo que os estudantes simulem várias ameaças e avaliem os riscos mais relevantes, ganhando entendimento sobre os benefícios e limitações dos métodos de proteção existentes. Essas ferramentas também facilitam a implementação de novas estratégias de ataque e defesa, possibilitando comparações justas em condições padronizadas. No final, estas poderão contribuir para uma melhor compreensão das vulnerabilidades e medidas de segurança em FL, ajudando a orientar esforços futuros para melhorar a segurança.

Referências

1. [Bukaty19] Bukaty, P.. The California Consumer Privacy Act (CCPA): An implementation guide. IT Governance Publishing, 2019
2. [Fang20] Fang, M., Cao, X., Jia, J., Gong, N.. Local model poisoning attacks to Byzantine-robust federated learning. USENIX Conference on Security Symposium, 2020
3. [GDPR16] European Parliament and Council of the European Union, General Data Protection Regulation, 2016
4. [Han23] Han, S., et al. FedMLSecurity: A Benchmark for Attacks and Defenses in Federated Learning and Federated LLMs, arXiv:2306.04959, 2023
5. [Kairouz21] Kairouz, P., et al., Advances and open problems in federated learning. Foundations and Trends in Machine Learning, 14(1-2):1–210, 2021
6. [McMahan17] McMahan, H., Moore, E., Ramage, D., Hampson, S., Arcas, B.. Communication-efficient learning of deep networks from decentralized data. International Conference on Artificial Intelligence and Statistics, 2017
7. [Rodrigues23] Rodrigues, F., Simões, R., Neves, N., FADO: A Federated Learning Attack and Defense Orchestrator, Workshop on Dependable and Secure Machine Learning (DSML), June 2023.
8. [SBC23] Sociedade Brasileira de Computação, Referenciais de Formação para o Curso de Bacharelado em CiberSegurança, 2023
9. [Tolpegin20] Tolpegin, V., Truex, S., Gursoy, M., Liu, L.. Data poisoning attacks against federated learning systems. European Symposium on Research In Computer Security, 2020
10. [Yue23] Yue, K., et al., Gradient Obfuscation Gives a False Sense of Security in Federated Learning, USENIX Security Symposium, 2023
11. [Zhang22] Zhang, Z., et al., Neurotoxin: Durable backdoors in federated learning. International Conference on Machine Learning, 2022

NUNO NEVES é Professor Catedrático no Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (Portugal). Ele lidera a linha de investigação de Sistemas Descentralizados Seguros e Confiáveis na unidade de investigação LASIGE. Suas principais áreas de interesse são o desenho de soluções que promovam a melhoria da segurança de software e dos sistemas distribuídos. Foi coordenador do Mestrado em Segurança Informática, e ao longo dos últimos dez anos tem sido responsável pela disciplina de Segurança de Software. Tem participado em diversas atividades na comunidade dos sistemas confiáveis, tendo presidido recentemente ao IEEE Computer Society Technical Committee on Dependable Computing and Fault Tolerance (2021-2023).