Google

Guidelines for Responsible Machine Learning

Guidelines

Exercise caution when control flows are modified based on the output of ML models.

Workflows that consume the output of a machine learning primitive can be modified to produce attack surfaces that are difficult to secure. The current advice is that it is not advisable to ingest data into an execution workflow that is not explicitly expected.

Avoid direct access to models and model metadata.

Direct interaction with machine learning primitives can lead to a number of different attack surfaces; exfiltration, inversion and others. Keep these objects away from users and allow controlled, approved-only interactions, and ensure output is attached to an established feature - like a database. Do not provide output directly to the user.

Ensure that models use integrity checking on both training and post-implementation.

Modifications to software can occur along multiple stages within the build and deployment pipeline; the same is true for machine learning. Check, and then check again. Only run what you know is expected: Authenticate before you operate.

Complexity of features may produce complexity of controls.

Keep interactions brief, to the point and succinct. Large, complex workflows represent much greater risks of exposure in terms of successful attacks.

Start with extensional logic for training, and move to intensional logic for features and inference, where needed.

Using a vetted, well-known good set for training is a good way to avoid data poisoning and other attacks. However, to create features that have greater inference capabilities, new data must be ingested; testing the results of this training carefully based on expected outcomes from known good input results in more resilient systems.

Ensure integration testing is created along the intended use-case of the primitive.

Machine learning workflows require integration and unit testing like any other software. The heuristic nature of ML software can make it difficult to build reliable assertions, so it's recommended to focus on the desired, long-term outcomes of the workflow.

Look for attacks that use machine synthesis, particularly around controls that focus on 'what-you-are'.

Always ask 'Can a machine do this, now?' before making assumptions on authenticity of voice, video and text -- particularly when used to provide access rights.

Use mutual policies to keep ahead of the economic model and ensure a strong, low-cost security model.

If the cost of an attack is lower than the cost of its subsequent remediation, the control is likely to be ineffective. Operationalise security costs to adjust automatically based on the impact of successful attacks to gradually increasing costs on attacker.

Google

Avoid placing models trained on proprietary data in control of the user, such as on a device.

Consider the use-case involved in creating the machine learning primitive and assume, in theory, that the training data that created it can be recovered. Use this for business continuity planning and risk assessments when developing products.

Use rate limiting to avoid extrapolation attacks and prevent unnecessary resource consumption.

Traditional controls are still largely effective; layer controls such as access control lists, rate limiting and authentication when designing and implementing machine learning technologies.