Thinking about the security of AI systems

Why established cyber security principles are still important when developing or implementing machine learning models.

Martin R

If you're reading this blog, there's a good chance you've heard of large language models (LLMs) like ChatGPT, Google Bard and Meta's LLaMA. These models use algorithms trained on huge amounts of text data which can generate incredibly human-like responses to user prompts.

In our previous blog (ChatGPT and large language models: what's the risk?) we discussed some of the cyber security considerations to be aware of when using LLMs. This time we'll dig a bit deeper into a couple of the specific vulnerabilities in LLMs, namely:

- 'prompt injection' attacks
- the risk of these systems being corrupted by manipulation of their training data

However, LLMs aren't the only game in town. The cyber security fundamentals still apply when it comes to machine learning (ML), and we'll also highlight a few other things to be aware of if you're involved in scoping, developing or implementing an ML project.

Prompt injection attacks

One of the most widely reported weaknesses in the security of the current generation of LLMs is their vulnerability to 'prompt injection', which is when a user creates an input designed to make the model behave in an unintended way. This could mean causing it to generate offensive content, reveal confidential information, or trigger unintended consequences in a system that accepts unchecked input from the LLM.

Hundreds of examples of prompt injection attacks have been published. They can be mischievous (such as this example from Simon Willison, that ends up with Bing questioning its existence), but the real-world consequences could be scary too. In one example, a prompt injection attack was demonstrated against MathGPT, a model designed to convert natural language queries into code for performing mathematical operations. A security researcher identified that the model worked by evaluating user-submitted text as code, and used that knowledge to gain access to the system hosting the model. This allowed them to extract a sensitive API key before disclosing the attack.

As LLMs are increasingly used to pass data to third-party applications and services, the risks from malicious prompt injection will grow. At present, there are no failsafe security measures that will remove this risk. Consider your system architecture carefully and take care before introducing an LLM into a high-risk system.

Data poisoning attacks

An ML model is only as good as the data it is trained on, and LLMs are no exception. Their training data is typically scraped from the open internet in truly vast amounts, and will probably include content that is offensive, inaccurate or controversial. Attackers can also tamper with this information to produce undesirable outcomes, both in terms of security and bias. As noted in the NCSC's 'Principles for the security of machine learning', 'Data poisoning' of this type is a known concern for ML developers, and recent research(.pdf) by Nicholas Carlini et al has demonstrated the feasibility of poisoning extremely large models with access to only a very small part of their training data.

Designing for security

Prompt injection and data poisoning attacks can be extremely difficult to detect and mitigate. However, no model exists in isolation, so what we can do is design the whole system with security in mind. That is, by being aware of the risks associated with the ML component, we can design the system in such a way as to prevent exploitation of vulnerabilities leading to catastrophic failure. A simple example would be applying a rules-based system on top of the ML model to prevent it from taking damaging actions, even when prompted to do so.

It's also important to extend other basic cyber security principles to take account of ML-specific risks. This includes supply chain security, user education, applying appropriate access controls, and other mitigations highlighted in the NCSC's Principles for the Security of Machine Learning.

But everyone does this already, don't they?

Well...

While there's been an understandable focus on LLMs, the last few months have also seen reports of vulnerabilities or security incidents involving other ML systems, tools and workflows. Many of these could have been mitigated by applying established cyber secure principles to ML development. For example:

1. Think before you arbitrarily execute code you've downloaded from the internet (models)

Downloading pretrained models is a standard (and understandable) step in many ML workflows, but as with executing any code, it introduces risks.

One widely highlighted ML vulnerability is rooted in the widespread use of the Python Pickle library to save and load model architectures and weights. Pickle was designed for efficiency and ease of use, and is inherently insecure; deserializing files allows the running of arbitrary code.

Options to mitigate the risks include:

- using a different serialisation format (such as safetensors)
- using a Pickle scanning tool (such as Picklescan); note that these may not pick up the most sophisticated exploits, but should give some level of

/

reassurance
- most importantly, applying standard supply chain security practices like relying on trusted sources or known valid hashes and signatures

2. Keep up to date with published vulnerabilities and upgrade software regularly

Even popular and well-supported products can contain vulnerabilities. In March 2023, security researchers discovered a critical fault in the machine learning lifecycle management tool MLflow which allowed a remote attacker to read any file on the host machine running the MLflow server. Though the vulnerability was quickly fixed, the security of users is dependent on them ensuring their release version is up to date.

3. Understand software package dependencies

In December 2022, malicious files impersonating part of the PyTorch framework were uploaded to the Python Package Index code repository (PyPI). The files contained code that would extract sensitive information from the machines of users who installed them. This attack relied on a technique known as 'dependency confusion'. The malicious code had the same name as a legitimate PyTorch component which was available from the company's own repository. However, because imports from PyPI normally take precedence over other sources, many users inadvertently installed the malicious version.

4. Think before you arbitrarily execute code you've downloaded from the internet (packages)

Many ML (and other software) workflows involve downloading packages from public repositories. Attackers have sought to capitalise on the explosion in popularity of LLMs by publishing packages with malicious components. In one example, a package named **chatgpt-api** contained code to exfiltrate user credentials. Malicious actors now appear to be using LLMs to find good package names to spoof, bringing us full circle. Again, this risk can be mitigated by applying standard software supply chain principles.

Doing what we can

The AI landscape is changing quickly and will continue to do so. It's hard to know exactly what the future holds, but maintaining good cyber security hygiene remains the best way to protect **all** systems, whether or not they contain ML components. Designing for security by default, access controls, applying software updates and good supply chain security remain as important as they ever did.

Martin R
Data Science Researcher

WRITTEN BY

Martin R
Data Science Researcher, NCSC

PUBLISHED

30 August 2023

WRITTEN FOR

Cyber security professionals