# Top considerations for addressing risks in the OWASP Top 10 for LLMs

## 01 Prompt injection

Prompt injection (similar to SQL injection) is the manipulation of an LLM by providing inputs designed to trick the LLM into performing tasks it shouldn't be allowed to do.

- Educate your teams on prompt injection, and consider gamifying the learning with a tool like Gandalf from Lakera.
- Use the principles of least priviledge between your LLM and your data/functionality.
- Limit the sensitive data your LLM has access to.
- Treat your LLM like a user data and sanitize actions and responses.
- Use function calling where possible to avoid unstructured data which might change the context for the LLM/desired behavior

## 02 Insecure output handling

Avoid your LLM directly accesses sensitive data, or to executing functions without validation.

- Treat your LLM like your user data and be careful of direct access between your LLM and data.
- Adhere to the rules of least privilege.
- Don't allow your LLM to be able to execute dangerous functions when it shouldn't need to.

## 03 Training data poisoning

This is the manipulation of training data or fine tuning processes to make the LLM output less secure, more biased, less effective, or less performant.

- Validate/verify data sources.
- Use sandboxing of data sources, so external sources can't be added easily.
- Perform attestation/signing of data.

**Read the full OWASP Top 10 for LLMs**

## 04 Model Denial of Service

MDoS (similar to ReDoS) occurs when malicious users provide input/prompts to an LLM that consume significant system resources to create a denial of service for the request or wider system.

- Consider LLM frameworks, such as LangChain that restrict number of steps in input evaluation.
- Consider following LLM architecture references from vendors and software architecture best practices such as circuit breakers.
- Typical DoS advice still apply: sanitization and validation of input, rate limiting calls, etc.

## 05 Supply chain vulnerabilities

Vulnerabilities can exist in the training or tuning data (similar to vulnerable components in apps) that was taken from third-party sources — making them part of the software supply chain.

- List the data source dependencies you're using for the training/tuning of your LLM.
- Use attestation/signing techniques to validate data you use.
- Responsibly evaluate traditional security risks of malicious and typosquatting libs related to AI and LLMs devtools SDKs.

## 06 Sensitive information disclosure

This occurs when an LLM discloses sensitive and confidential data back to the user in an unauthorized manner.

- Don't give your LLM more data than it needs to do it's job.
- Add input and output guard checks around interactions to sanitize input from users and output from LLM.

## 07 Insecure plugin design

Homegrown LLM plugins that are triggered by the LLM can be open to the risk of malicious input from the LLM.

- Treat LLM output similar to how you treat user input.
- Use various param input into plugins vs. single text queries.
- Think about plugin interactions similar to an API contract and follow OWASP Top 10 API Security Risks best practices.

## 08 Excessive agency

An LLM with excessive agency may choose to invoke particular functionality in a malicious or unexpected way causing unexpected results.

Three types of excessive agency include:

- Excessive Functionality: Insufficient functionality granularity in plugins can lead to LLM applications taking actions that are too broad.
- Excessive Permissions: An LLM plugin provides more access/permissions than required.
- Excessive Autonomy: An LLM application automatically taking dangerous actions without user interaction.

## 09 Overreliance

Overreliance occurs when a user incorrectly assumes that generated code (for example) are correct, secure, legal etc.

- Treat generated code, like junior developer code: validate, test, correct.
- Use tools like Snyk Code to automate AI generated code security testing and fix first party code in the IDE where code is generated, as well as Snyk Open Source to test AI suggested open source library usage.
- Educate teams about hallucinations and disinformation by from generative AI.

## 10 Model theft

Protect your digital IP from an an attacker trying to gain unauthorized access to your LLM by using existing best practices like RBAC and more.