The scale, complexity, and unpredictability of large language models (LLMs) and the applications that make use of them can confound even the best traditional appsec tools. While there are measures that can and should be taken early to build models and apps more securely, many of the vulnerabilities on this list exist because they take advantage of how LLMs behave by design, meaning there's only so much that can be done with "shift-left" solutions.

Prompt Injection

01

What it is

Prompt injection (also called "jailbreaking") is when an attacker issues clever prompts that cause an LLM to behave against its underlying system prompts and guardrails, possibly leading to data leaks or remote code execution.

What can be done in design

Reduce the damage a prompt injection could cause by limiting the LLMs access and privileges to only what is necessary.

What can be done in production

To detect and prevent prompt injection vulnerabilities by monitoring and analyzing the LLM system's behavior in runtime, identifying any suspicious or malicious prompt that could lead to manipulations – whether those actions originate from user inputs or anywhere else.

Insecure Output Handling

02

What it is

The outputs of LLMs can be wildly unpredictable and may include executable code, whether by design or because bad actors are using prompt injections. Insecure output handling arises when those LLM outputs are piped into plugins or apps without sufficient validation or sanitization.

What can be done in design

In your application, an LLM should be treated with the same zero-trust approach that would be applied to any other user.

What can be done in production

Use tools (like Oligo) with runtime detection capabilities that can inspect LLM outputs for any insecure handling that leads to code execution or access that's out-of-scope, and alert you when such behavior is detected.

Training Data Poisoning

03

What it is

Large amounts of data are needed to train or tune LLMs, giving bad actors plenty of opportunities to slip poisoned data somewhere into the process. An AI model trained on poisoned data may exhibit undesirable outputs that are inaccurate, biased, or unethical.

What can be done in design

Verify that data comes from trusted sources and be careful introducing data that can change over time, with poisoned data introduced in place of previously good data (such as training data produced from crawling the internet).

What can be done in production

Use tools with features like Oligo's dynamic analysis of library behavior that can detect if an application leads to unauthorized access such as arbitrary dataset tampering in real-time, helping to identify and mitigate the risk of data poisoning, no matter how it originates.

Model Denial of Service

04

What it is

Model denial of service happens when a bad actor crafts inputs to make the LLM perform tasks that consume a large amount of resources, potentially bringing the LLM down or leaving it with few resources to respond to other requests.

What can be done in design

Where possible, limit requests per user or IP, the number of actions an LLM can take to satisfy a request, and the amount of resources that can be used per request or step.

What can be done in production

Monitoring is key. Real-time alert solutions like Oligo can detect anomalies in network access, helping to quickly mitigate the impact of model denial of service attacks.

Supply Chain Vulnerabilities

05

What it is

Like any modern application ecosystem, LLMs rely on a supply chain which can introduce risks. Vulnerabilities coming from any point in the supply chain, including dependencies and third-party components, can lead to poor or unethical LLM behavior, security breaches, and system failures.

What can be done in design

When building LLMs or applications making use of them, be sure to vet sources of data and open-source dependencies and use only those that are trusted.

What can be done in production

Tools like Oligo can detect supply chain vulnerabilities, helping to mitigate issues introduced through pre-trained models or plugin extensions.

Sensitive Information Disclosure

06

What it is

The unpredictability of LLM outputs can cause them to give up sensitive information even when instructed not to. Disclosure of sensitive information through LLMs can result in legal consequences and very unhappy customers.

What can be done in design

As with many of the top LLM vulnerabilities, input and output validation and sanitization are mandatory. When connecting LLMs to sensitive data sources, employ the rule of least-privilege (for example, read-only users with limited access to explicit resources) and ensure the data used for training and fine-tuning does not include sensitive data and Personal Identifying Information (PII).

What can be done in production

Runtime monitoring capabilities like those found with Oligo can help detect and prevent AI agents – or bad actors abusing an LLM – from accessing databases or other sources of sensitive information that might end up divulged to end users, reducing the risk of legal consequences

Insecure Plugins

07

What it is

Insecure plugins refers to those you've built to connect LLMs to external resources. If a plugin accepts LLM output without any validation regarding its contents or size, undesired behaviors including remote code execution are possible.

What can be done in design

Plugins should be designed to minimize the impact of receiving an insecure input, as well as to parameterize inputs and check them for size, range, and type.

What can be done in production

Appsec tools like Oligo can analyze plugins connecting LLMs to external resources to find vulnerabilities and prevent exploitation through malicious requests or remote code execution.

Excessive Agency

08

What it is

Due to malfunctions causing unexpected or ambiguous output, an LLM with excessive agency can cause undesired behaviors in other components it has access to.

What can be done in design

When designing systems that include LLM output, limit both permissions and functionality of the LLM to the minimum necessary to do the job.

What can be done in production

Where possible and necessary, bring in a human to approve important actions the LLM tries to take. When a human response isn't possible, available, or desirable, tools like Oligo can help enforce controls on LLMs' interactions with other systems, preventing excessive agency and reducing the risk of undesirable operations or actions.

Overreliance

09

What it is

Don't believe everything you read on the internet, even (or maybe especially) when it comes from an LLM. Overreliance on LLM outputs for decision-making without checking for accuracy can result in making poor decisions that may lead to legal issues and reputational damage.

What can be done in design

Keep in mind that LLMs hallucinate from time to time, and don't use LLMs outputs for application decision-making if they are connected to sensitive data sources. If your AI application is user-facing, remind your users to take LLM outputs with a grain of salt.

What can be done in production

Make sure that the LLMs used in your production applications cannot lead to unintended behavior, by design. Minimize the reliance on LLM outputs for controlling the flow and logic of your applications, and always sanitize the LLM prompts and responses, with explicit security guardrails.

Model Theft

10

What it is

LLMs and data crucial to their operation can be exfiltrated, in whole or in part, after access is gained through exploiting other infrastructure vulnerabilities or through prompt injection. LLM theft can result in exposure of sensitive data and the loss of reputation and competitive advantage.

What can be done in design

Regular security best practices like encryption of sensitive data, the principle of least-privilege, and strong authentication all apply here.

What can be done in production

Monitoring is vital. Runtime detection solutions like Oligo can help detect and prevent unauthorized access to proprietary LLMs, reducing the risk of theft, competitive advantage loss, and sensitive information dissemination.

Conclusion

Notice any patterns? LLMs often produce surprising (sometimes "haha" surprising, sometimes "oh no" surprising) results, and their size and capabilities naturally increase the attack surface. Luckily, the behavior of the libraries and systems composing and interacting with LLMs are very predictable, making it possible to stop threat actors from exploiting LLM vulnerabilities. Runtime observability, detection, and mitigation are crucial in securing LLM applications and the important assets they interact with.

Read the full report:
OWASP Top 10 for LLM