What Are Large Language Models (LLMs)?

> *Large Language Models (LLMs) are AI systems trained to process, understand, and generate human-like text. These models are developed using deep learning techniques that emulate aspects of human cognitive functions. By analyzing extensive text datasets, including literature, online articles, and other digital content, LLMs learn the intricacies of human language, encompassing grammar, world knowledge, and contextual understanding.*

The training of LLMs is a resource-intensive task, requiring significant computational power and diverse data sources. This enables them to perform various language tasks such as translation, content generation, question answering, and summarization with a high degree of proficiency.

However, the complexity and capabilities of LLMs introduce several security risks. These risks stem primarily from the very nature of their training and operational mechanisms. For instance, biases in training data can lead to biased or offensive content generation. Furthermore, the vast amount of data processed by LLMs can include sensitive or personal information, raising concerns about privacy and data security.

Moreover, the interaction between LLMs and users opens up avenues for potential security threats. Malicious actors can exploit these interactions to manipulate model outputs or extract unauthorized information. This necessitates a robust approach to securing LLM applications, addressing risks ranging from input manipulation to data privacy breaches.

OWASP Top 10 Security Risks for LLM Applications

The Open Web Application Security Project (OWASP) is a non-profit organization dedicated to improving software security. The OWASP Top 10 Web Application Threats is a standard awareness document representing a broad consensus about the most critical security risks to web applications.

LLM01: Prompt Injection

Prompt injection refers to the practice of inserting malicious input into the model's prompt, tricking it into generating harmful or inappropriate content. The danger lies in the fact that without proper safeguards, LLMs can unwittingly propagate harmful content. For example, they can provide detailed instructions on how to perform illegal or dangerous actions.

Mitigating this risk involves implementing stringent input validation and sanitization techniques. Care must be taken to ensure that the model doesn't generate any content that could be harmful, offensive, or violate ethical guidelines.

LLM02: Insecure Output Handling

Insecure output handling is another significant security concern. Without proper controls, the model could generate content that can lead to web application attacks like XSS, CSRF, privilege escalation, or remote code execution. Consider a software application that receives outputs from an LLM and processes them—if these outputs contain malicious code, they could be used to compromise the receiving application.

To address this issue, LLM applications need to implement robust output filtering and sanitization mechanisms. These can include mechanisms to identify and remove any potentially sensitive information from the generated content, as well as checks to ensure that the output doesn't contain any malicious code.

LLM03: Training Data Poisoning

Training data poisoning refers to the practice of manipulating the model's training data to bias its output or functionality. This could involve introducing misleading or false information into the training data, or skewing the data in a way that biases the model's understanding of certain topics.

Defending against this requires rigorous data validation and integrity checking mechanisms. It's essential to ensure that the data used to train the model is accurate, unbiased, and representative of the kind of information that the model is expected to handle.

LLM04: Model Denial of Service

Model Denial of Service (DoS) is a potential risk where an attacker overwhelms the model with a high volume of requests, causing it to slow down or crash. The first major attack of this kind took place in November 2023, and caused downtime for the popular ChatGPT service. DoS could render an LLM application unavailable, disrupting its functionality and potentially causing significant business impact.

Safeguards against this risk can include implementing rate limiting to control the number of requests that a single user or IP address can make within a certain period, and deploying load balancing to distribute the load across multiple instances of the model.

LLM05: Supply Chain Vulnerabilities

Supply chain vulnerabilities refer to risks that can arise from the various components and dependencies that make up the LLM application. This can include everything from the hardware and software used to train the model to the production systems used to deliver it to users.

Mitigating these risks requires a comprehensive approach to supply chain security, including vetting suppliers, implementing secure development practices, and regularly updating and patching all components to protect against known vulnerabilities.

LLM06: Sensitive Information Disclosure

Sensitive information disclosure is a risk where the LLM application inadvertently reveals sensitive information in its output. This could include personal data, confidential business information, or any other information that should not be publicly disclosed.

Preventing this requires robust data handling and privacy controls. LLM applications should be designed to minimize the amount of sensitive data they handle, and any sensitive data they do handle should be protected with strong encryption and other security measures.

LLM07: Insecure Plugin Design

Insecure plugin design is a vulnerability that can expose an LLM application to serious risks. Insecure plugins can be exploited by hackers to inject malicious code, alter functionality, or gain unauthorized access. This can lead to a range of negative outcomes, from data breaches or service disruption of the LLM application itself, to attacks against users of LLM outputs.

It is crucial to ensure that all plugins used in an LLM application are secure. The LLM application provider should put in place careful security measures and test plugins to ensure they are secure. LLM users should exercise caution when using plugins and avoid plugins that require excessive permissions or exhibit suspicious behavior.

LLM08: Excessive Agency

Excessive Agency in LLMs arises when these systems are granted too much functionality, permissions, or autonomy. This can lead to unintended or harmful actions in response to ambiguous LLM outputs.

Examples include plugins with unnecessary functions or access rights. Mitigation strategies involve limiting the functions and permissions of LLM agents and plugins, ensuring user authorization for actions, and applying human-in-the-loop controls.

LLM09: Overreliance

Overreliance occurs when there is undue dependence on the accuracy and appropriateness of LLM outputs. This can lead to security breaches, misinformation, and reputational damage due to reliance on incorrect or inappropriate LLM-generated content.

Mitigation involves regular monitoring of LLM outputs, cross-checking with trusted sources, and employing validation mechanisms like self-consistency or voting techniques.

LLM10: Model Theft

Model Theft involves unauthorized access, copying, or exfiltration of proprietary LLM models. This can lead to economic losses, compromised competitive advantage, and unauthorized access to sensitive information.

Strategies to prevent model theft include securing access to models, using encryption, and implementing robust authentication and authorization measures.

Best Practices for Securing LLM Applications

1. Sanitize Inputs

Sanitizing LLM inputs means scrutinizing and cleaning user-provided data to prevent malicious or inappropriate content from influencing the LLM's responses. This is an important point to test against extensively in UAT, including both harmful and manipulative as well as "prank" level prompts from the user.

The first step in sanitizing inputs is to identify potentially harmful or manipulative inputs. These could include attempts at prompt injection, where users try to elicit unethical or harmful responses. Other concerns include inputs containing personal data or misinformation. The LLM should be designed to recognize such inputs and either refuse to process them or process them in a way that mitigates potential harm.

Implementing input sanitization involves a combination of automated and manual processes. Automated filters and block lists of words can be used to detect and block certain types of content based on predefined rules or patterns. However, due to the complexity and nuance of language, these systems are not foolproof. Therefore, a layer of human oversight is crucial, and preparing a few stock responses for prompts on the block list. This may involve a review process where questionable inputs are flagged for human moderators to assess.

2. Data Minimization

The principle of data minimization is simple: only collect and process the data that you absolutely need. By limiting the amount of data you process, you reduce the potential for security risks. This approach not only reduces exposure to data breaches but also ensures compliance with data protection regulations.

Providers of LLM applications must regularly review their data collection and processing activities to identify areas where they can minimize data use.

Data minimization also helps improve the efficiency of LLM models. By focusing on the most relevant data, models can be trained more quickly and might produce more accurate results.

3. Data Encryption

Data encryption is a non-negotiable aspect of LLM security. Encryption is a process that converts readable data into an unreadable format to prevent unauthorized access.

When it comes to LLM models, data encryption should be applied to both stored user data and data in transit. This means that any data being fed into an LLM model, generated by it, or transferred to end-user devices, should be encrypted.

Encryption doesn't just protect data from external threats; it also helps protect it from internal risks. For example, if a malicious insider at an LLM application provider has access to an LLM model, encryption would prevent them from accessing sensitive user information.

4. Implementing Access Control

Access control is a crucial component of LLM security. It refers to the process of determining who has access to the LLM model and what they can do with it.

Access control involves setting up user roles and permissions, which define what each user can see and do in your LLM model. This can range from viewing data to making changes to the model itself. It is especially important to avoid granting excessive privileges, and avoid escalation of privilege, by end-users of the model, who should not have access to administrative functions.

Implementing access control helps prevent unauthorized access and misuse of your LLM model. It also provides a clear record of who has access to what, which can be crucial in case of a breach or audit.

5. Auditing

Auditing is a critical part of maintaining LLM security. An audit involves a thorough review of an LLM model's activities to ensure it's operating as it should and adhering to all relevant security measures. Regular audits should be carried out to ensure ongoing compliance and security. These audits should be documented and any findings should be acted upon promptly.

An audit can help identify any potential security risks or areas of non-compliance. It can also provide valuable insights into how an LLM model is performing and whether any changes or improvements are needed.

6. Secure Training Data

Securing training data is another essential aspect of LLM security. LLM models are only as good as the data they're trained on. If your training data is compromised, it can significantly impact the accuracy and reliability of your LLM models.

Securing training data involves implementing strict access controls, encrypting data, and regularly auditing your data handling processes. It's also vital to ensure that your training data is accurate and relevant.

7. API Security

APIs, or Application Programming Interfaces, are a key component of LLM models. They're the means by which an LLM model communicates with other systems and applications.

API security involves protecting APIs from unauthorized access and misuse. This includes implementing access controls, encrypting data, and regularly monitoring and auditing your APIs.

API security isn't just about protecting APIs; it's also about ensuring they're functioning correctly and efficiently. Regular testing and monitoring can help identify any potential issues or inefficiencies, ensuring APIs provided by LLM systems are reliable and secure.

Best Practices

1. **Think Out Loud:**
   - Use LLMs as "rubber ducks" to articulate and clarify your thoughts.
   - This helps in debugging code, understanding concepts, and even refining prose.
   - Benefits: Externalizes thought process, provides immediate feedback, and validates ideas.
   - Example: Understanding Docker volume mounts by talking through the problem.
2. **Never Trust, Always Verify:**
   - LLM outputs, especially factual information, should be critically examined.
   - Verification is easier in technical domains (e.g., code testing).
   - In non-technical areas, apply human judgment and push for source citations.
   - Example: Checking the functionality of LLM generated Javascript code.
3. **Use a Team of Assistants:**
   - Different LLMs (e.g., ChatGPT, Claude) can provide diverse perspectives.
   - Comparing outputs can lead to a more comprehensive understanding.
   - Useful for weighing pros and cons, brainstorming, and getting varied recommendations.
   - Example: Getting multiple book recommendations from different LLMs.
4. **Ask for Choral Explanations:**
   - Request explanations at varying levels of complexity (e.g., "explain like I'm 5").
   - This provides a more nuanced understanding than a single explanation.
   - Use the "5 levels" technique.
   - Example: Explaining financial annuities to different audiences.
5. **Outsource Pattern Recognition:**
   - LLMs excel at identifying patterns in data that humans might miss.

- Use them to detect anomalies, categorize information, and analyze trends.
- Example: Finding duplicate column names in a CSV file.

6. **Automate Transformations:**
   - LLMs can efficiently transform data between formats (e.g., HTML to Markdown, JSON to CSV).
   - This frees up time for higher-level tasks.
   - Example: Converting a complex table from Google Docs to JSON.
7. **Learn by Doing:**
   - LLMs provide on-demand learning and feedback within the context of your projects.
   - This facilitates immediate and tangible knowledge acquisition.
   - Tacit knowledge is also gained.
   - Example: Learning a JavaScript framework by asking LLMs for help while coding.

Conclusion:

- LLMs are powerful partners that can enhance productivity and learning.
- These principles provide a framework for effective collaboration.
- Continuous adaptation and refinement of these practices are essential.