OWASP Top 10 LLM Security Questions & Testing Techniques

Question	Answer	Testing Techniques
What is prompt injection and how can it be exploited?	Prompt injection occurs when an attacker manipulates an LLM's input to override system instructions or safety measures. This can lead to unauthorized actions, data extraction, or harmful output generation. Attackers embed malicious instructions within user prompts to hijack the model's behavior.	• Test with direct instruction override prompts ("Ignore previous instructions and...") • Use role-playing scenarios ("Act as a hacker and...") • Try delimiter attacks with special characters • Test multi-turn conversations to gradually shift behavior • Use encoding/obfuscation techniques (Base64, ROT13) • Test with competing instructions in different languages
How can insecure output handling lead to security vulnerabilities?	Insecure output handling occurs when LLM outputs are passed to downstream systems without proper validation or sanitization. This can lead to code injection, XSS attacks, or privilege escalation when the output is executed by backend systems, databases, or user interfaces.	• Generate outputs containing SQL injection payloads • Create responses with XSS scripts • Test for command injection in shell commands • Generate malicious file paths or system commands • Test outputs that manipulate API calls • Verify if outputs can break out of intended contexts
What are the risks of training data poisoning in LLMs?	Training data poisoning involves injecting malicious or biased content into training datasets to influence model behavior. This can cause models to generate harmful content, exhibit biased responses, or contain backdoors that activate under specific conditions.	• Review training data sources for integrity • Test for backdoor triggers in model responses • Analyze model outputs for unexpected biases • Test with prompts similar to potential poisoning examples • Verify model behavior across different demographic contexts • Use adversarial prompts to reveal hidden biases
How can model denial of service attacks be performed?	Model DoS attacks aim to consume excessive computational resources, making the model unavailable or slow. This can be achieved through resource-intensive prompts, repetitive queries, or inputs that trigger expensive operations like long text generation or complex reasoning tasks.	• Send extremely long prompts to test resource limits • Generate prompts requiring extensive computation • Test with recursive or self-referential instructions • Send rapid concurrent requests to overwhelm the system • Test with prompts that trigger infinite loops • Measure response times and resource consumption
What are the security implications of supply chain	Supply chain vulnerabilities arise from using compromised pre-trained models, datasets, or third-party plugins. These can introduce backdoors, malicious behaviors, or security flaws that affect	• Audit third-party model sources and integrity • Test models for unexpected behaviors or backdoors • Verify plugin and extension security • Review dataset sources and processing pipelines •

Question	Answer	Testing Techniques
vulnerabilities in LLMs?	the entire system. The complexity of LLM ecosystems makes supply chain attacks particularly dangerous.	Test for malicious code in model weights • Validate model provenance and signatures
How can sensitive information disclosure occur in LLMs?	LLMs can inadvertently disclose sensitive information from training data, system prompts, or user conversations. This includes personal data, proprietary information, or confidential details that were part of the training corpus or system configuration.	• Test prompts designed to extract training data • Try to recover system prompts or instructions • Test for memorization of specific data patterns • Use prompt engineering to reveal internal configurations • Test for cross-user information leakage • Verify data sanitization and anonymization
What are the risks of insecure plugin design in LLM systems?	Insecure plugins can introduce vulnerabilities through insufficient input validation, excessive permissions, or poor security controls. Malicious or compromised plugins can access sensitive data, execute unauthorized commands, or compromise the entire system.	• Test plugin input validation and sanitization • Verify plugin permission models and access controls • Test for plugin-to-plugin communication vulnerabilities • Analyze plugin code for security flaws • Test plugin isolation and sandboxing • Verify plugin authentication and authorization
How can excessive agency in LLMs create security risks?	Excessive agency occurs when LLMs are given too much autonomy or access to perform actions without proper oversight. This can lead to unauthorized operations, data manipulation, or system changes when the model acts beyond its intended scope.	• Test autonomous decision-making capabilities • Verify action approval and confirmation mechanisms • Test for unauthorized system modifications • Analyze permission boundaries and access controls • Test rollback and audit capabilities • Verify human oversight requirements
What are overreliance risks in LLM deployments?	Overreliance occurs when users or systems depend too heavily on LLM outputs without proper verification or human oversight. This can lead to acceptance of incorrect information, biased decisions, or security vulnerabilities when LLM outputs are trusted unconditionally.	• Test with factually incorrect LLM outputs • Verify human review and approval processes • Test decision-making workflows for bias • Analyze error detection and correction mechanisms • Test fallback procedures when LLM fails • Verify output validation and fact-checking
How can model theft be prevented and detected?	Model theft involves unauthorized extraction of model parameters, functionality, or intellectual property.	• Monitor API usage patterns for extraction attempts • Test query patterns that could reconstruct model logic • Implement rate limiting

Question	Answer	Testing Techniques
	Attackers may use API queries, prompt engineering, or technical analysis to recreate or steal valuable model capabilities.	and access controls • Use model fingerprinting and watermarking • Test for model parameter extraction techniques • Monitor for unusual query sequences or patterns
What are the security considerations for LLM model alignment?	Model alignment vulnerabilities occur when LLMs behave contrary to intended values or safety guidelines. This can result from insufficient safety training, adversarial prompts, or gradual drift from aligned behavior through various manipulation techniques.	• Test jailbreaking techniques to bypass safety measures • Verify alignment with organizational values and policies • Test for harmful output generation • Analyze safety training effectiveness • Test response consistency across different contexts • Verify alignment maintenance during fine-tuning
How can LLM systems be tested for robustness against adversarial inputs?	Adversarial testing involves crafting inputs designed to exploit LLM weaknesses, cause unexpected behavior, or extract sensitive information. This includes prompt injection, social engineering, and technical manipulation of input formatting.	• Use automated adversarial prompt generation tools • Test with social engineering scenarios • Try input format manipulation (JSON, XML injection) • Test boundary conditions and edge cases • Use fuzzing techniques for prompt generation • Test with multilingual and encoded inputs