Protect Against OWASP Top 10 Large Language Model Security Risks

SHAKUDO

SHAKUDO

Table of Contents

SHAKUDO

This overview document from Shakudo outlines the OWASP Top 10 security vulnerabilities for Large Language Models (LLMs) in 2023, highlighting critical risks including prompt injection, insecure output handling, and model theft. We provide actionable mitigation strategies and emphasize the importance of implementing robust security protocols. Our insights aim to equip developers and security professionals with the necessary tools and knowledge to effectively safeguard LLM deployments.

Protect Against OWASP Top 10 LLM Security Risks

SHAKUDO

LLM01
Prompt Injection

Definition

This vulnerability occurs when malicious inputs are injected into the model's prompts, leading to unintended behavior or output manipulation.

OWASP Suggestion

Implement prompt validation and sanitization of inputs

Shakudo Comments

1. Shakudo provides several tools such as LLM Guard that apply guardrails to LLM applications. These tools help ensure appropriate output and reduce risk of misinformation.
2. The Shakudo Retrieval Augmented Generation (RAG) stack provides built-in controls to guardrail against inappropriate responses, and ensures that outputs are generated only from your data.
3. The mitigation approach for preventing prompt injection attacks is similar to the mitigation of Overreliance (Section 9.c).

LLM02
Insecure Output Handling

Definition

This vulnerability occurs when an LLM output is accepted without scrutiny, exposing backend systems. Misuse may lead to severe consequences like XSS, CSRF, SSRF, privilege escalation, or remote code execution.

OWASP Suggestion

1. Implement thorough input validation for the responses the model sends to backend functions.
2. Encode the output from the model to the users to prevent any unintended code execution or interpretation.

Shakudo Comments

The mitigation approach for insecure output handling is similar to the mitigation of Overreliance (Section 9.c).

SHAKUDO

LLM03
Training Data Poisoning

Definition

injecting malicious or biased data into the LLM's training dataset, resulting in skewed model outputs or compromised performance.

OWASP Suggestion

1. Ensure the authenticity of data sources during both the model training and fine-tuning phases.
2. Develop separate models for different use cases, each with its own specific training data.
3. Implement rigorous screening or input filters for training data and for different types of data sources.

Shakudo Comments

1. This is not specific to LLMs — training data poisoning is also applicable to regular models.
2. Shakudo ensures that all data connections are authenticated and encrypted with TLS and additionally authenticated via SSO.
3. Data residing within the Shakudo platform is only accessible by authenticated and authorized users, with full logging of activities.

LLM04
Model Denial of Service

Definition

this vulnerability occurs when an attacker interacts with an LLM in the way that consumes a large amount of resources

OWASP Suggestion

1. Implement input validation and sanitization to ensure inputs adhere to defined limits
2. Enforce API rate limits to restrict the number of requests an individual user or IP address can make, and cap resource use per request or step
3. Limit the number of queued actions and the number of total actions in a system reacting to LLM responses.

Shakudo Comments

1. Shakudo offers fine-grained access controls and request throttling on per-microservice and per-endpoint granularities.
2. Access to all components on Shakudo are gated through istio, and access to model endpoints will only be through internal IPs. All network access requests are logged through zipkin. In addition, compute (CPU, RAM, GPU) usage metrics are tracked, with configurable alerting thresholds and targets.
3. Shakudo microservices use Kubernetes-native capabilities for self recovery, autoscaling, and isolation. Even in cases of non-malicious usage where resource limits are reached, other cluster resources will not be impacted, and the services are designed to self-recover and scale up according to needs.

SHAKUDO

LLM05

Supply Chain Vulnerabilities

Definition

LM application lifecycle can be compromised by vulnerable components or services, leading to security attacks. Using third-party datasets, pre-trained models, and plugins add vulnerabilities.

OWASP Suggestion

1. Thoroughly evaluate data sources and utilize security systems that have passed independent audits.
2. Only use plugins that are reliable and have been tested to meet specific needs.
3. Adhere to MLOps (Machine Learning Operations) best practices for managing your models.
4. Sign both the model and code when using external models to ensure authenticity.
5. Set up systems to monitor for vulnerabilities and enforce a consistent update and patching routine.
6. Conduct regular security and access reviews of your suppliers.

Shakudo Comments

1. Shakudo performs additional scanning using the GCP artifact registry security scanner of every image prior to deploying it to customers. This is in addition to signature verification and any scanning done by the original model developers.
2. Every stack component on Shakudo gets thoroughly vetted by our solution engineering team for breadth of adoption, quality of product and active development.
3. Shakudo provides multiple options for MLOps management including MLFlow, Weights & Biases to ensure best practices can be adhered to by the users of the platform.
4. Shakudo continuously monitors security advisories and notices that may affect stack components deployable through Shakudo, and roll out hot fixes in cases where zero day vulnerabilities are announced.
5. Shakudo performs routine security and access reviews as part of SOC-2 compliance.

SHAKUDO

LLM06

Sensitive Information Disclosure

Definition

LLM applications may inadvertently disclose sensitive information, proprietary algorithms, or confidential data

OWASP Suggestion

1. Sanitize, scrub, anonymize data before training
2. Limit access to external sources and maintain strict access control on internal sources
3. Implement robust input validation and sanitization
4. Restrict the types of data returned by the LLM

Shakudo Comments

1. The mitigation approaches of sensitive information disclosure are related to those of Prompt Injection and Overreliance.
2. Data within Shakudo is protected by the overall Shakudo access control and security controls, similarly to protection for code and model parameters. In addition, Shakudo offers workflows for data anonymization and sanitation before usage for model training or data analytics.

SHAKUDO

LLM07

Insecure Plugin Design

Definition

LLM plugins can have insecure inputs and insufficient access control due to lack of application control. Attackers can exploit these vulnerabilities, resulting in severe consequences like remote code execution.

OWASP Suggestion

1. Strict input validation with parameterization, type, and range checks to prevent invalid or harmful data entry.
2. Comprehensive inspections and tests using Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Interactive Application Security Testing (IAST) to identify vulnerabilities.
3. Implementation of proper authentication mechanisms using identities and API Keys to ensure that only authorized entities can access the system.
4. The requirement for manual user authorization for any actions executed by sensitive plugins to ensure deliberate and reviewed actions.

Shakudo Comments

1. Any plugins deployed on Shakudo are protected by the overall Shakudo access control and security controls, like any other component or microservice
2. Specific controls include:
    a. SSO authentication for every action and API call
    b. Namespace isolation in the underlying Kubernetes environment prevents components from impacting each other or accessing each other's data.

SHAKUDO

LLM08

Excessive Agency

Definition

LLM-based systems may undertake actions leading to unintended consequences. The issue arises from excessive functionality, permissions, or autonomy granted to the LLM-based systems.

OWASP Suggestion

1. Restrict the plugins/tools an LLM can use to only those that are absolutely necessary, and also limit the functions within those plugins/ tools.
2. Opt for plugins that offer specific functionality rather than those with broad capabilities.
3. Ensure that all actions require human approval and keep track of who has authorized what.
4. Keep records of how LLM plugins/tools are used, monitor them for any issues, and set up controls to prevent too many unwanted actions from happening.

Shakudo Comments

1. Our recommendation would be to avoid autonomous agents that take actions during the initial phases of AI adoption
2. By default, Shakudo does not come deployed with any action taking agents out of the box. Those are available, but require careful planning around the available APIs that the agents can call. For the initial rollout we recommend using AI models only for processing of data and text and not function calling.

SHAKUDO

LLM09
Overreliance

Definition

Systems or people overly depending on LLMs without oversight may face misinformation, miscommunication, legal issues, and security vulnerabilities due to incorrect or inappropriate content generated by LLMs.

OWASP Suggestion

1. Frequently monitor and review what LLMs produce.
2. Verify the LLM's output against reliable sources.
3. Improve the LLM by fine-tuning it or adding embeddings.
4. Set up systems that automatically check the validity of outputs.
5. Divide complex tasks into smaller, more manageable ones.
6. Make the potential risks and limitations of LLMs clear to all involved.
7. Adopt secure coding practices within development environments.

Shakudo Comments

1. Shakudo provides several tools such as LLM Guard that apply guardrails to LLM applications. These tools help ensure appropriate output and reduce risk of misinformation.
2. The Shakudo Retrieval Augmented Generation (RAG) stack provides built controls to prevent hallucination and ensures that model responses are generated from previously ingested documents.
3. LLM evaluation components such as promptfoo enable automated quality control of LLM responses on the Shakudo platform.

SHAKUDO

LLM10
Model Theft

Definition

Attackers may gain unauthorized access to model parameters, through reverse engineering, or stealing training data and training algorithms.

OWASP Suggestion

1. Watermark framework for lifecycle
2. Rate limit API calls
3. Implement strong monitoring and auditing
4. Automate MLOps deployment with governance

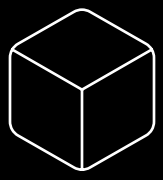
Shakudo Comments

1. In most cases for LLM applications, the model and algorithms are very similar and most applications require fine-tuning. We also often using standard techniques.
2. Shakudo is deployed completely with your infrastructure and provides RBAC to all components within the platform. We treat protecting model artifacts and parameters in the same way as proprietary data and environment components.

SHAKUDO

Recommendations for Container Scanning Tools

- We use **Artifact Registry scanner** prior to shipping any images or software to customers. Artifact registry scanner detects package-level and language-level vulnerabilities. We ensure no images are shipped with critical or high vulnerabilities, and we make the results of our scans available to customers [link]
- **Trivy**, an open source container scanner AquaSecurity, with vulnerability scanning for container images, file systems, and Git repositories, and configuration issues. Trivy comes with Shakudo's built-in Harbor registry as the default in-cluster scanner. [GitHub link]
- **Clair**, an open source container scanner by quay, also available on Shakudo's built-in Harbor registry. [GitHub link]



Recommendation for Kubernetes environment security scanning tools

- **kube-bench** is an open source tool by AquaSecurity that checks whether Kubernetes is deployed securely by running the checks documented in the CIS Kubernetes Benchmark, and provide remediation suggestions. It can be deployed with Trivy (see next section). [GitHub link]
- **kubeaudit** scans Kubernetes clusters against common security controls and detects common misconfigurations of kubernetes resources [GitHub link]

SHAKUDO

About Shakudo

Shakudo creates compatibility across the best-of-breed data tools for a more reliable, performant, and cost effective data and AI operating system. As an operating layer on top of your cloud Shakudo allows you to pick the best-of-breed data tools for your needs, while providing a platform with fully automated DevOps experience. This combines the best of both worlds in data stack practices so you can focus on delivering business value with data.

Shakudo is the most **easy, secure, cost-effective, scalable** way to bring the most advanced data and AI tools to your data. Find out more at shakudo.io.

Contact

For further information or to discuss how Shakudo can assist in enhancing your security measures for Large Language Models, please do not hesitate to contact us:

Email: info@shakudo.io

Phone: (437) 783-5683