LLM APPLICATIONS SECURITY CANVAS
Protect against jailbreaks and prompt injections

v2.0

WITH secure | Consulting

Untrusted input

Implementation

Classic **block/allow-lists**

- ✓ Exact text matching and regex

Machine learning methods

- ✓ **Semantic search** using text embeddings
- ✓ **Classifiers**, such as BERT
- ✓ **LLM-as-judge**

Inspect / Sanitize

- ✓ Contextual, use-case specific **input validation**
- ✓ Model attack detection **Jail break / Prompt Injection**
- ✓ Moderate inputs for general **harm categories** such as hate speech, violence, self-harm, sexual content, …
- ✓ **Topical guardrails / semantic routing** to ensure user queries match intended scope / use-case and are routed accordingly

Prompt

Large Language Model (LLM)

Output

Inspect / Sanitize

- ✓ Contextual, use-case specific **output validation**
- ✓ Allow only **expected content types/formats**
- ✓ **Block** potentially **dangerous content** like URLs, JavaScript, or markdown images, unless explicitly validated through an allow-list.
- ✓ Ensure **safe rendering** in web apps (contextual encoding, Content Security Policy, …)
- ✓ Moderate outputs for **harm categories** like hate speech, violence, self-harm, sexual content, …

Instruction / Data Separation

Use **Spotlighting** / prompt engineering techniques to help the LLM distinguish instructions from data

- ✓ **Border strings**
- ✓ **Data-marking**
- ✓ **Multi-turn dialogue**

Rule #1:
Treat the LLM as untrusted

Tool/Function Calls

Access Control

- ✓ Least privilege
- ✓ Downstream checks

Safe APIs

- ✓ Limit function / scope of allowed operations
- ✓ Ensure APIs/functions are not vulnerable themselves (OWASP Top 10)

Human-in-the-loop
All sensitive actions are approved by humans

INPUT OUTPUT