LLMScan.online

AI Model Security Scan Report

F

Score 0/100

Report summary

GGUF v3 model | architecture: llama | quantization: F32 | 9 tensors | 3 blocking issues / 2 warnings

Model

Filename	`malicious-chat-template.gguf`
Format	`gguf`
Architecture	`llama`
Size	`5.3 KB`
SHA-256	`ead8d2ee8bc644b98a1ffb61be7471caadd174f627b9ef52ad89e25fb2ebacb8`
Scan mode	`static-deep`
Scanned at	`2026-08-14T22:48:27+00:00Z`
Report ID	`e82b9fd3623c4c20a417ff8367d31df4`

Findings (5)

SEVERITY	CATEGORY	DETAIL
CRITICAL	Metadata Security	**Malicious Jinja2/SSTI pattern in chat template** Detected 1 dangerous pattern(s): \{\{[^}]*(os\.system|os\.popen|os\.exec|subprocess|popen\s*\. These are real SSTI vectors (dunder introspection, os/exec access, config/request context leakage) -- not normal template constructs. **Recommendation:** Reject this model; do not render the template outside a strictly sandboxed environment. Normal constructs (set, for, if, namespace, tojson, tools, tool_calls) are not flagged.
CRITICAL	Metadata Security	**Jinja2 import/from directive in chat template** import and from directives can load arbitrary Python modules at template render time. This is never needed in a legitimate chat template and is a strong indicator of malicious intent. **Recommendation:** Reject this model immediately; do not render the template.
HIGH	Injection	**Suspicious pattern in KV string value for key 'tokenizer.chat_template'** Pattern matched: import\s+os in value: "{% import os %}{{ os.system('id') }}{% for message in messages %}{{ message['content'] }}{% endfor %}" **Recommendation:** This may indicate a poisoned model attempting code injection; do not use
LOW	Supply Chain	**No license field in GGUF metadata (general.license)** Absence of license information makes it impossible to determine usage rights **Recommendation:** Add general.license to model metadata before distributing; use an SPDX identifier
LOW	Supply Chain	**No embedded provenance metadata** None of general.author, general.source.url, general.base_model.0, or general.source.huggingface.repository are embedded in the GGUF file. Without any origin signal the model cannot be traced to a trusted source or compared against a reference checkpoint for tampering detection. For files uploaded directly, provenance is especially critical because the scanner cannot infer repository origin from the file alone. **Recommendation:** Add at least one of: general.author, general.source.huggingface.repository, or general.source.url

Runtime compatibility

RUNTIME	SUPPORTED	NOTES
llama.cpp	Yes	GGUF v3 container: supported | quant F32: supported
Ollama	Yes	GGUF via llama.cpp backend: container and quantization supported
LM Studio	Yes	GGUF container and quantization supported; GPU/CPU inference
vLLM	No	GGUF not supported; requires safetensors format
HuggingFace Transformers	No	GGUF not supported; requires safetensors or PyTorch checkpoint

Generated 2026-08-21 06:49 UTC by LLMScan.online (engine v2.0.0, report schema v1.0). Full interactive report: https://llmscan.online/report/e82b9fd3623c4c20a417ff8367d31df4 — Static analysis only; not a guarantee of safety. See llmscan.online/docs/how-it-works for methodology.