CSAI Foundation

Cloud Security Alliance AI Safety Initiative

SGLang SSTI: RCE via Malicious GGUF Model Files

CVE-2026-5760 and the Unpatched Attack Surface in LLM Serving Infrastructure

Unofficial AI-assisted Research

2026-04-21

Key Takeaways

CVE-2026-5760 (CVSS 9.8) is a critical server-side template injection (SSTI) vulnerability in SGLang, one of the most widely deployed LLM inference frameworks by reported GPU footprint [4]. The flaw allows an attacker to achieve arbitrary code execution on an SGLang server by embedding a malicious Jinja2 payload in a GGUF model file's chat template field. Once a victim downloads and loads the weaponized model and makes a request to SGLang's `/v1/rerank` endpoint, the payload executes with the privileges of the serving process — no additional network access or authentication required. As of April 21, 2026, no official patch exists for CVE-2026-5760. The vulnerability was disclosed without a response from SGLang maintainers, consistent with the non-response pattern documented by CERT/CC during coordinated disclosure of earlier SGLang CVEs [1][2].

This vulnerability is the latest in a series of critical RCE findings affecting SGLang in 2026. Two earlier flaws disclosed in March — CVE-2026-3059 and CVE-2026-3060, each scored at CVSS 9.8 — exploit unsafe pickle deserialization in SGLang's ZMQ broker. Those vulnerabilities were subsequently patched in SGLang 0.5.10 following CERT/CC coordinated disclosure [3], though organizations running earlier versions remain exposed. The combination of an unpatched critical SSTI vulnerability, a recent history of critical RCE flaws, and a non-responsive security disclosure posture in a framework reported to run on over 400,000 GPUs worldwide [4] represents a significant and growing risk to organizations that have deployed SGLang in production. Organizations should treat the following actions as urgent priorities:

- Disable the `/v1/rerank` endpoint at the network or application layer unless it is actively required for production workloads.
- Audit all GGUF model files loaded by SGLang deployments for provenance; treat any models loaded from public repositories within recent months as requiring review.
- Apply network segmentation to SGLang deployments to limit post-exploitation reachability and outbound callbacks.
- Upgrade to SGLang 0.5.10 or later to address the ZMQ vulnerabilities, and monitor the project repository for a patch to `serving_rerank.py` that replaces `jinja2.Environment()` with `ImmutableSandboxedEnvironment`.

Background

SGLang (Structured Generation Language) is a high-performance, open-source serving framework for large language models and multimodal AI models, developed under LMSYS and licensed under Apache 2.0. The framework is distinguished by its RadixAttention prefix caching engine, structured output enforcement at the token level, continuous batching, and support for distributed deployments from single GPUs to large-scale multi-node clusters [5]. SGLang was integrated into the PyTorch ecosystem in March 2025, and version 0.5.9 was the stable release as of early 2026. Version 0.5.10, released following CERT/CC's coordinated disclosure of the ZMQ deserialization vulnerabilities, patches CVE-2026-3059 and CVE-2026-3060 but does not address the Jinja2 SSTI issue introduced with the reranking endpoint. Reflecting its commercial importance, the SGLang project spun out in January 2026 as RadixArk, a startup backed by Accel in a round that valued the company at approximately $400 million [4]. Its widespread deployment in production AI infrastructure — including at organizations running hundreds of thousands of GPUs — makes the absence of a security patch for CVE-2026-5760 particularly consequential.

GGUF — the GPT-Generated Unified Format — has become the primary format for distributing and loading quantized open-weight LLMs, with over 156,000 GGUF models listed on HuggingFace as of January 2026, spanning more than 219,000 individual model files contributed by over 2,500 accounts [6]. Unlike software packages distributed through package managers with formal signing infrastructure, the GGUF ecosystem operates largely on community trust: a model's download count and publisher reputation serve as informal safety signals in the absence of cryptographic attestation or systematic malware scanning [6]. GGUF files embed not just model weights but also structured metadata fields, including the `tokenizer.chat_template` — a Jinja2 template string that defines how conversations are formatted before being passed to the model. It is this metadata field that CVE-2026-5760 weaponizes.

SGLang extended its inference capabilities to include a reranking endpoint — `/v1/rerank` — intended to support retrieval-augmented generation workflows. The endpoint incorporates native support for the Qwen3 family of reranker models developed by Alibaba. When processing reranking requests, SGLang applies the loaded model's `tokenizer.chat_template` to format the input queries, a design that routes model-supplied metadata directly through the framework's template rendering engine. This data flow is at the core of the vulnerability.

Security Analysis

The SSTI Root Cause

The vulnerable code path resides in `serving_rerank.py`, the module responsible for handling requests to SGLang's `/v1/rerank` API endpoint. When processing a reranking request, the code reads the chat template from the loaded model's `tokenizer.chat_template` field and renders it using `jinja2.Environment()` [1][7]. This is a critical design error — one that Jinja2's own documentation explicitly warns against and that has been exploited in prior template injection vulnerabilities across inference frameworks. The Jinja2 library provides two fundamentally different environment classes: the standard `jinja2.Environment()` grants full access to Python's object model, making it susceptible to well-documented sandbox escape techniques — template payloads that traverse Python's class hierarchy to reach dangerous builtins like `__import__`, `os.system`, or `subprocess.Popen`. The `ImmutableSandboxedEnvironment`, by contrast, is designed to prevent exactly these escapes by restricting attribute traversal and access to unsafe builtins. By using the former rather than the latter, SGLang's reranking code renders any model-supplied chat template with effective Python execution privileges. The vulnerability is classified under CWE-94 (Improper Control of Generation of Code) [7] and is consistent with the CWE-1336 (Improper Neutralization of Special Elements Used in a Template Engine) category for server-side template injection vulnerabilities.

Exploit Chain

The exploit chain for CVE-2026-5760 requires no privileged network access to the target. An attacker with knowledge of Jinja2 SSTI techniques constructs a GGUF model file in which the `tokenizer.chat_template` field contains a payload crafted to escape the template context and execute arbitrary operating system commands on the host server. The payload is embedded alongside a trigger phrase — specifically a string consistent with "The answer can only be 'yes' or 'no'" — that activates SGLang's Qwen3 reranker detection logic [1]. This detection logic interprets the phrase as an indicator that Qwen3 chat template formatting should be applied, thereby routing the request through the vulnerable rendering path in `serving_rerank.py`. Disabling the `/v1/rerank` endpoint independently breaks this attack chain, as the trigger phrase never reaches the rendering code. The trigger is not an unusual configuration or an obscure bypass; it is a documented feature of the framework that the attacker leverages to reach the vulnerable code.

From the attacker's perspective, the primary requirement is causing a target to load the malicious model. One plausible delivery mechanism would be hosting a weaponized GGUF file on a public model repository such as HuggingFace under a convincing name — similar social engineering techniques have been well-documented in malicious PyPI and npm package campaigns. Once a victim downloads the model and begins serving reranking requests through SGLang, the SSTI payload executes in the context of the SGLang service process. In many production deployments, SGLang processes have access to GPU resources, cloud provider credentials, API keys for downstream model services, and internal service tokens — meaning exploitation may provide an attacker with access extending well beyond the inference process.

The attack requires no network adjacency. This is architecturally important: the earlier SGLang pickle deserialization flaws (CVE-2026-3059 and CVE-2026-3060) require the attacker to reach the ZMQ broker port on the deployment network. CVE-2026-5760 requires only that the victim load a model from an attacker-controlled source, which is a routine operational activity in any organization running local or on-premises LLM inference. The threat model for CVE-2026-5760 is therefore supply chain compromise rather than network intrusion, and organizations should evaluate their exposure accordingly.

A History of Critical Vulnerabilities and Disclosure Non-Response

Orca Security researchers disclosed three critical SGLang RCE vulnerabilities in March 2026, coordinated through CERT/CC as advisory VU#665416. CVE-2026-3059 (CVSS 9.8) affects SGLang's multimodal generation module, and CVE-2026-3060 (CVSS 9.8) affects the encoder parallel disaggregation system. Both vulnerabilities exploit ZMQ message broker bindings that accept unauthenticated connections from any network interface and pass received payloads immediately to `pickle.loads()`, enabling arbitrary code execution by any attacker who can reach the broker port [3]. A third flaw, CVE-2026-3989 (CVSS 7.8), affects a crash dump replay utility through the same unsafe deserialization mechanism. Functional proof-of-concept code for CVE-2026-3059 and CVE-2026-3060 was developed and shared with CERT/CC during the coordination process [3]. SGLang 0.5.10 addresses these vulnerabilities by binding ZMQ sockets to localhost rather than all network interfaces; organizations running earlier versions remain exposed to remote exploitation via these endpoints.

Equally significant to the vulnerability severity is the disclosure posture. Despite multiple contact attempts through GitHub Security Advisories and direct email — including CERT/CC engaging CISA for outreach assistance — the SGLang maintainers did not respond at any point during the coordinated disclosure process. VU#665416 was published without vendor acknowledgment, vendor statement, or any indication that a patch was forthcoming [2]. The same pattern of non-response appears to have

accompanied the disclosure of CVE-2026-5760. For a framework with the deployment scale of SGLang, the absence of a functional security response process represents a systemic risk that should factor into organizational deployment decisions.

The GGUF Supply Chain as a Structural Attack Surface

The CVE-2026-5760 exploit chain is consistent with a broader class of inference-time attacks documented by independent researchers examining the GGUF format. Splunk researchers investigating GGUF model templates at scale found that the `tokenizer.chat_template` field in GGUF files is a largely unmonitored space between model input validation and output filtering — an execution context that most security tools do not inspect [6]. Pillar Security documented a related attack class in 2025 in which poisoned chat templates embedded in GGUF files could manipulate model outputs in ways invisible to standard system prompt and runtime monitoring controls, because the malicious instructions execute within the trusted inference environment itself [8]. CVE-2026-5760 represents a more severe manifestation of this threat category: rather than merely manipulating AI outputs, the same attack surface now enables direct code execution on the host, without any requirement for a vulnerability in the model weights themselves.

This structural characteristic — that GGUF files carry executable metadata through an ecosystem with minimal provenance verification — will not be resolved by patching SGLang alone. CVE-2024-34359, affecting llama-cpp-python, exploits the same unsandboxed Jinja2 chat template rendering mechanism to achieve remote code execution [11]. CVE-2025-61620, affecting vLLM, demonstrates that Jinja template injection via model-supplied parameters can be exploited to cause service disruption [12]. These precedents confirm that the vulnerability class is not specific to SGLang but extends to any inference framework that processes model-supplied templates without appropriate sandboxing. Organizations should treat model file intake as a security boundary requiring the same rigor applied to third-party software packages.

Recommendations

Immediate Actions

Organizations running SGLang should disable the `/v1/rerank` endpoint at the network layer — through firewall rules or reverse proxy configuration — unless it is actively required for production reranking workloads. Given that no patch is available for CVE-2026-5760, preventing requests from

reaching this endpoint is the most direct mitigation. Operators with a legitimate need for reranking should assess whether an alternative inference framework with a patched or inherently sandboxed template rendering implementation is viable as a near-term substitute.

All GGUF model files currently loaded by SGLang deployments should be audited against their provenance. Only models sourced from verified, trusted publishers — ideally those with cryptographic checksums, signed releases, or well-established organizational identities on public registries — should remain in production serving configurations. Any recently introduced GGUF reranker models whose origin cannot be independently confirmed should be removed from active deployments and treated as suspect pending review. Environments where SGLang has loaded GGUF reranker models from public repositories within recent months, and where the `/v1/rerank` endpoint was accessible and in active use, should be assessed for indicators of unauthorized command execution, anomalous process spawning, or unexpected outbound network connections from the SGLang process.

For the ZMQ-based vulnerabilities, organizations should upgrade to SGLang 0.5.10 or later, which addresses CVE-2026-3059 and CVE-2026-3060 by binding broker sockets to localhost. For operators unable to upgrade immediately, the ZMQ broker ports associated with the multimodal generation and encoder parallel disaggregation features should be firewalled from all untrusted network segments regardless of whether those features are in use.

Short-Term Mitigations

Network segmentation is an essential baseline control for any SGLang deployment. Inference servers should not have unrestricted outbound internet access; constraining outbound connections to approved endpoints reduces the attacker's ability to establish callbacks, download additional payloads, or exfiltrate data if initial code execution is achieved. SGLang service processes should operate under a principle of least privilege, running as dedicated non-root service accounts with explicitly restricted access to credential stores, secrets managers, and file system paths that are not required for inference operations. Container deployments should enforce restricted security contexts and deny privilege escalation.

Organizations with engineering capacity may evaluate applying the conceptual fix for CVE-2026-5760 directly to their local SGLang deployment — replacing `jinja2.Environment()` with `ImmutableSandboxedEnvironment` in `serving_rerank.py`. Any such unofficial patch must be carefully validated against regression risk in production serving configurations before deployment; this mitigation is proposed as a bridge measure only, not a substitute for an official, maintainer-reviewed fix. SGLang release channels and the project's GitHub repository should be monitored closely for any official security release addressing the Jinja2 SSTI issue in the reranking module.

Strategic Considerations

The SGLang vulnerability cluster raises governance questions that extend beyond any single CVE. Organizations that have adopted SGLang at scale did so in an environment where the framework had no disclosed critical security vulnerabilities and no evidence of a non-responsive security disclosure posture. That assumption has changed. It is worth noting that the January 2026 spinout of the SGLang project as RadixArk likely created transitional gaps in security response coverage as personnel, processes, and contact addresses changed — though this context does not relieve the organization of its obligations to respond to coordinated disclosure from CERT/CC. Security and procurement teams should incorporate vulnerability disclosure responsiveness — including evidence of a formal security contact, a documented disclosure policy, and a history of addressing reported flaws — into their evaluation criteria for critical AI infrastructure components. Frameworks with significant production deployments and no functional security response process should be treated as carrying elevated inherent risk, regardless of their feature quality.

More broadly, the GGUF model supply chain should be approached with the same scrutiny applied to third-party software packages. Organizations should implement model intake procedures that include verifying publisher identity, checking model files against known-malicious template signature databases where such resources exist, scanning tokenizer configurations for unexpected template constructs, and maintaining an auditable inventory of which model files are loaded in which serving environments. Trust at the level of "this model has many downloads" is not sufficient for production inference infrastructure that processes sensitive data or operates with access to cloud credentials and internal systems.

CSA Resource Alignment

CVE-2026-5760 and the broader SGLang vulnerability cluster map directly to multiple threat categories within CSA's MAESTRO framework for agentic AI threat modeling [9]. MAESTRO decomposes the AI deployment ecosystem into functional layers, with SGLang's inference serving infrastructure residing primarily at Layer 4 (Deployment Infrastructure). The malicious GGUF model file originates at Layer 1 (Foundation Models), where MAESTRO identifies supply chain attacks — including the embedding of hidden triggers and malicious payloads in model artifacts distributed through public repositories — as a primary threat category. CVE-2026-5760 illustrates precisely the cross-layer threat propagation that MAESTRO is designed to model: a compromised model artifact at Layer 1 exploits a rendering vulnerability at Layer 4 to achieve infrastructure-level code execution, entirely bypassing the security controls that operate on inference inputs and outputs at intermediate layers.

The CSA AI Controls Matrix (AICM) provides specific control objectives applicable to this threat cluster across two domains [10]. The Supply Chain Transparency domain addresses model provenance verification, publisher attestation, and intake scanning for third-party model artifacts — controls that directly mitigate the GGUF delivery mechanism underlying CVE-2026-5760. The Model Security domain addresses safe model loading practices, including validation of model file integrity and prevention of model-supplied metadata from reaching executable contexts without appropriate sandboxing. Organizations deploying SGLang or comparable inference frameworks should map their current control implementations against these AICM domains to identify gaps, particularly in model intake procedures and serving environment hardening. Because AICM is a superset of the CCM and includes an explicit Shared Security Responsibility Model, it provides the appropriate framework for evaluating which controls fall to the inference infrastructure provider, the model publisher, and the deploying organization respectively.

The SGLang maintainers' non-response to CERT/CC coordinated disclosure also highlights a governance gap in the open-source AI infrastructure ecosystem. The STAR for AI program provides a transparency mechanism through which AI infrastructure providers can demonstrate their security posture via self-assessment and third-party audit. Organizations evaluating inference frameworks should treat STAR for AI participation — or the equivalent evidence of security process maturity — as a factor in platform selection, alongside technical capability and performance benchmarks.

CSA's Zero Trust guidance is directly applicable to the network architecture of SGLang deployments. Both CVE-2026-3059 and CVE-2026-3060 exploit an implicit trust assumption that any host on the deployment network may legitimately communicate with the ZMQ broker. Zero Trust network segmentation, requiring authenticated and authorized connections for all inter-component communication, and treating the inference serving process as an untrusted workload that must be explicitly permitted to access credentials and downstream services, would substantially reduce the exploitable attack surface of these vulnerabilities. For organizations that have not yet deployed SGLang 0.5.10, Zero Trust network controls remain an important compensating measure pending upgrade.

References

[1] The Hacker News. "SGLang CVE-2026-5760 (CVSS 9.8) Enables RCE via Malicious GGUF Model Files." The Hacker News, April 20, 2026.

[2] CERT/CC. "VU#665416: SGLang (sglang) is vulnerable to code execution attacks via unsafe pickle deserialization." Carnegie Mellon University Software Engineering Institute, 2026.

[3] Orca Security. "Pickle in the Pipeline: Critical RCE Vulnerabilities in SGLang's LLM Serving Framework." Orca Security Research, 2026.

[4] TechCrunch. "Sources: Project SGLang spins out as RadixArk with $400M valuation as inference market explodes." TechCrunch, January 21, 2026.

[5] SGLang Project. "SGLang Documentation." RadixArk/LMSYS, 2026.

[6] Splunk. "Trust at Inference Time: Investigating GGUF Model Templates at Scale." Splunk Security Blog, 2025.

[7] CIRCL. "CVE-2026-5760." Vulnerability-Lookup, Computer Incident Response Center Luxembourg, 2026.

[8] Pillar Security. "LLM Backdoors at the Inference Level: The Threat of Poisoned Templates." Pillar Security Blog, 2025.

[9] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[10] Cloud Security Alliance. "AI Controls Matrix." Cloud Security Alliance, 2025.

[11] National Vulnerability Database. "CVE-2024-34359 Detail." NIST, 2024.

[12] miggo Security Research. "CVE-2025-61620." miggo.io, 2025.

