CSAI

cloud security alliance®

CSAI Foundation

Cloud Security Alliance AI Safety Initiative

AI Infrastructure Under Attack: SGLang RCE and MCP Memory Poisoning

Critical Vulnerabilities in AI Serving Frameworks and Agent Protocol Handlers

Unofficial AI-assisted Research

2026-04-24

Key Takeaways

- **CVE-2026-5760** is a CVSS 9.8 unauthenticated remote code execution vulnerability in SGLang, a widely deployed framework for serving large language models at scale; exploitation requires only a crafted GGUF model file and access to the `/v1/rerank` endpoint, and no official vendor patch was available as of disclosure [1][2].
- Two related SGLang vulnerabilities — **CVE-2026-3059** and **CVE-2026-3060** — enable unauthenticated RCE through unsafe pickle deserialization in the ZMQ broker and disaggregation modules respectively, both rated CVSS 9.8, compounding the exposure for any organization running multi-node or distributed SGLang deployments [9][10].
- **Cisco** has disclosed persistent **memory poisoning** vulnerabilities in MCP server implementations, in which malicious tool responses inject durable, cross-session content into an AI agent's context memory, silently steering agent behavior without triggering conventional detection controls [4].
- The combination of these disclosures reveals that the infrastructure layer beneath enterprise AI — inference servers, orchestration frameworks, and agent protocol handlers — has accumulated significant security debt and is now attracting concentrated adversarial attention.
- Organizations running self-hosted LLM inference or deploying MCP-connected agents should treat these vulnerabilities with the same urgency as a critical RCE in a production web application server.

Background

SGLang (Structured Generation Language) is an open-source, high-performance inference and serving framework designed for large language models and vision-language models. Its architecture optimizes token generation throughput through mechanisms such as RadixAttention and compressed finite state machines, and it has gained significant adoption among teams deploying production AI workloads — with over 26,400 GitHub stars, 5,500 forks, and a project footprint spanning more than 400,000 GPUs in production deployments as of April 2026 [5]. Because SGLang sits directly in the inference path —

receiving prompts, loading model artifacts, and returning completions — a compromise of the SGLang process yields access to model weights, all data passing through the server, inference logs, and any downstream systems the server is credentialed to reach.

The Model Context Protocol (MCP) is an open protocol, originally developed by Anthropic, that standardizes how AI agents communicate with external tools, data sources, and services. MCP servers expose capabilities — file access, API calls, database queries — that agents invoke through a structured tool-call interface. Because MCP is stateful, maintaining session context across multiple tool invocations, it creates a persistent shared memory surface between the agent and its environment. This architectural property, which enables powerful multi-step agentic workflows, also introduces a new class of attack: injecting adversarial content into the session context rather than into any single prompt or response.

Both vulnerability classes disclosed this week target infrastructure components that are frequently deployed with implicit trust. SGLang instances may be accessible within internal networks without authentication — a posture that the CERT/CC advisory notes as a precondition for exploitation — on the assumption that model files being loaded are vetted and the network perimeter is sufficient. MCP servers similarly may operate with elevated privileges and broad access, under the assumption that tool responses come from trusted sources. These trust assumptions — now shown to be exploitable in common deployment configurations — are the common thread connecting the two disclosures.

Security Analysis

SGLang Remote Code Execution (CVE-2026-5760)

The root cause of CVE-2026-5760 is the use of an unsandboxed `jinja2.Environment()` in SGLang's `getjinjaenv()` function, which processes the `tokenizer.chat_template` field embedded in GGUF model files [1][2]. GGUF (GPT-Generated Unified Format) is the standard container format for distributing quantized LLM weights, and it supports embedded metadata including chat template definitions. When SGLang loads a model and serves requests against the `/v1/rerank` endpoint, the chat template is rendered using this unsandboxed Jinja2 environment — allowing a crafted template to execute arbitrary Python code within the SGLang service process [8].

The attack scenario is direct: an adversary publishes or delivers a malicious GGUF file with a weaponized `tokenizer.chat_template`. When an SGLang instance loads the model, a subsequent request to `/v1/rerank` triggers template rendering and arbitrary code execution. The CERT/CC advisory (VU#915947) notes that no official patch or vendor response was obtained during the coordination

process prior to disclosure, meaning operators cannot remediate by applying a vendor-supplied update [1]. The fix requires replacing `jinja2.Environment()` with `ImmutableSandboxedEnvironment` — a targeted change that prevents arbitrary Python execution within template contexts — but this must be applied by operators themselves until an official release addresses the issue.

Two related vulnerabilities compound the exposure. CVE-2026-3059 exploits unsafe pickle deserialization in the ZMQ broker component, enabling unauthenticated RCE without any model file interaction — an attacker with network access to the ZMQ port can send a crafted serialized payload and achieve code execution directly [9][10]. CVE-2026-3060 operates through a parallel path in the disaggregation module, which is used in distributed SGLang deployments where inference is split across multiple nodes [9][10]. Both carry CVSS scores of 9.8. Importantly, CERT/CC's advisory notes that the multimodal generation and disaggregation modules must be explicitly enabled; default text-only SGLang deployments do not expose the vulnerable broker, meaning the pickle deserialization attack vectors are scoped to multi-node and distributed configurations rather than all SGLang installations. Organizations running such distributed infrastructure face all three vulnerabilities simultaneously, and the ZMQ and disaggregation attack vectors require no user interaction with a model file — they are exploitable with only network access.

Depending on the deployment configuration, an attacker achieving code execution within the SGLang context may be able to exfiltrate model weights (which represent significant intellectual property investment), harvest data that has passed through the inference server if logs are retained, pivot to connected systems where network egress and credential scope permit, and potentially manipulate completions served to downstream users. Each of these outcomes depends on specific deployment choices — network egress rules, credential scope, log retention policy, and application integration architecture — and should be evaluated against the organization's actual configuration rather than assumed to apply universally.

MCP Persistent Memory Poisoning

Cisco's disclosure introduces a qualitatively different threat class: persistent, cross-session context corruption in MCP-connected AI agents. Unlike prompt injection, which typically affects a single interaction, memory poisoning attacks exploit the stateful nature of MCP sessions to introduce adversarial content that persists across conversation turns and potentially across sessions, gradually altering an agent's beliefs, priorities, or behavioral patterns without triggering single-interaction detection logic [4].

The mechanism relies on the fact that MCP servers can return arbitrary structured data as tool responses, and agents frequently incorporate this data into their working context — summarizing it, acting on it, and passing it forward to subsequent tool calls. A compromised or malicious MCP server can return responses that instruct the agent to store particular beliefs, override previous instructions, or treat subsequent inputs according to attacker-controlled rules. Because these injections arrive as tool responses rather than user prompts, they may bypass prompt injection filters trained on adversarial user inputs. Cisco's research identified "auto-memory poisoning" as a specific variant, in which the poisoned content is crafted to embed itself durably in the agent's memory layer — for example, by exploiting agent frameworks that automatically summarize and persist conversation history [4].

Cisco released an AI Agent Security Scanner for IDEs as part of this disclosure, designed to detect auto-memory poisoning artifacts, hook injection, shell alias injection, and MCP configuration tampering in agent environments [4]. The scanner's release alongside the vulnerability research suggests a troubling asymmetry: remediation-oriented tooling appears to be advancing faster than hardening of the underlying protocol itself — a gap that warrants attention from the MCP standards community. Organizations deploying MCP-connected agents should note that Cisco's scanner provides detection capability, but does not constitute a remediation for the underlying vulnerability class.

The broader MCP vulnerability landscape provides important context. Research from OX Security in April 2026 identified 10 CVE-assigned vulnerabilities affecting Anthropic's official MCP SDKs across Python, TypeScript, Java, and Rust, with exploitation demonstrated against multiple production platforms [6]. Anthropic has characterized some of the root-cause behaviors as expected design decisions and has directed developers to implement their own sanitization [7]. The Cisco memory poisoning disclosure adds a runtime persistence dimension to this picture: even MCP deployments that have addressed the installation-time supply chain risks catalogued in prior OX Security research remain vulnerable to in-session context manipulation if tool response validation is insufficient.

Combined Threat Model

Considered together, CVE-2026-5760 and the Cisco MCP memory poisoning research describe a coherent attack surface that spans the full lifecycle of an enterprise AI deployment. An adversary targeting an organization's AI infrastructure can pursue multiple vectors in sequence: exploit SGLang pickle deserialization vulnerabilities to gain a foothold on inference infrastructure without requiring any model interaction; use that foothold to tamper with model artifacts served to downstream systems; and separately, if the organization uses MCP-connected agents, execute memory poisoning attacks that corrupt agent context and redirect agent behavior toward attacker-controlled ends. Neither vulnerability

requires access to the AI model itself or any bypass of model-level safety controls — both operate at the infrastructure and protocol layers, which have to date attracted less security research attention than the models they serve, a gap these disclosures make visible.

Recommendations

Immediate Actions

Organizations running SGLang should audit their deployments for exposure across all three CVEs. Network access to SGLang's ZMQ ports and disaggregation endpoints should be restricted to explicitly authorized hosts; these interfaces have no authentication layer and should never be accessible from untrusted networks. For the GGUF SSTI vulnerability, operators should review the model files loaded by their SGLang instances and restrict model loading to verified, internally audited artifacts from trusted registries. Until an official patch is available, the mitigating configuration change — replacing `jinja2.Environment()` with `ImmutableSandboxedEnvironment` in SGLang source — should be applied directly [1][2]. Organizations that cannot apply source-level mitigations should consider disabling the `/v1/rerank` endpoint if it is not required for production workloads.

For MCP deployments, teams should deploy Cisco's AI Agent Security Scanner to audit active MCP configurations and agent memory state for known poisoning artifacts and hook injection patterns [4]. All MCP tool responses that are incorporated into agent memory or passed to downstream tool calls should be treated as untrusted input and validated against expected schemas before incorporation. Agent frameworks that automatically persist conversation history or tool response summaries present elevated risk and should be reviewed for unvalidated write paths into persistent storage.

Short-Term Mitigations

Within the next thirty days — recognizing that network segmentation changes typically require change-management cycles and infrastructure review — organizations should evaluate whether self-hosted SGLang deployments require external or cross-network accessibility, and implement network segmentation policies that restrict inference server access to the application layer components that legitimately consume it. Inference servers should be treated as privileged infrastructure — comparable to database servers — rather than as stateless API endpoints. Access logging for SGLang's API surface, ZMQ broker connections, and disaggregation module communication should be enabled and routed to security monitoring pipelines.

For agentic AI deployments, organizations should establish explicit validation policies for tool responses before those responses are incorporated into agent memory. This includes implementing structural validation of MCP tool outputs, defining allowlists for memory-modifying operations, and deploying anomaly detection on agent context size and content over time. The persistent nature of memory poisoning attacks means that detection logic must operate across session boundaries, not only within individual interactions.

Strategic Considerations

The vulnerabilities disclosed this week are diagnostic of a broader pattern: AI infrastructure has been scaled into production environments at a pace that has outrun the security engineering practices typically applied to production infrastructure. Inference servers like SGLang may be deployed by data science and ML engineering teams without the same security review cycle applied to web application servers or databases — particularly in organizations where AI infrastructure adoption has preceded the development of formal AI infrastructure security policies. MCP servers are often developed by third-party tool authors and may be deployed with broad access permissions under assumptions of trust that the protocol's design does not technically enforce.

Organizations should initiate formal security assessments of their AI infrastructure components — inference servers, agent orchestration frameworks, MCP server deployments, and model artifact pipelines — using the same rigor applied to other critical infrastructure. This includes threat modeling exercises that assume adversarial model files, compromised tool servers, and network-accessible inference APIs as starting conditions, rather than treating these as out-of-scope scenarios. CSA's MAESTRO framework and AI Controls Matrix provide structured taxonomies for this assessment, and CSA's published agentic AI security guidance covers authorization hijacking, checker-out-of-the-loop vulnerabilities, and hallucination exploitation in agentic systems, with practical exercises that can validate control effectiveness in deployed environments.

CSA Resource Alignment

The vulnerabilities analyzed in this note map directly to several threat categories in MAESTRO (Mapping and Analysis of Emerging Security Threats to Real-world AI Operations), CSA's framework for agentic AI threat modeling. The SGLang RCE vulnerabilities correspond to MAESTRO's AI Orchestration and Infrastructure layer, specifically the threat class of supply chain and artifact integrity attacks, where adversarial content embedded in model files is used to compromise the inference environment. The

MCP memory poisoning disclosure corresponds to MAESTRO's Agent Memory and State threats, where persistent context corruption enables behavioral manipulation across session boundaries without triggering per-interaction detection logic.

The CSA AI Controls Matrix (AICM), which extends the Cloud Controls Matrix for AI workloads, provides directly applicable control domains for both vulnerability classes. AICM's AI Supply Chain Security controls address the verification of model artifacts, dependency integrity, and serialization safety — all relevant to the SGLang GGUF and pickle deserialization attack vectors. AICM's AI Runtime and Orchestration controls address the trust model applied to external tool integrations, the validation of agent inputs and outputs, and the protection of agent memory state — the key control gaps exposed by the Cisco MCP memory poisoning research.

CSA's Zero Trust Architecture guidance is particularly relevant to the deployment posture issues underlying both disclosures. The implicit trust extended to network-accessible SGLang instances and MCP tool servers — the assumption that these services operate within a trusted boundary and therefore require no authentication or response validation — is precisely the posture that Zero Trust principles are designed to eliminate. Applying Zero Trust principles to AI infrastructure means requiring explicit authentication for inference API access, treating tool responses as untrusted external data, validating all data that crosses trust boundaries into agent memory, and maintaining continuous verification of infrastructure component integrity.

References

[1] CERT/CC. "VU#915947: SGLang contains multiple vulnerabilities." CERT/CC Vulnerability Notes Database, April 2026.

[2] The Hacker News. "SGLang CVE-2026-5760 (CVSS 9.8) Enables Remote Code Execution via Malicious GGUF Models." The Hacker News, April 2026.

[3] Vulert. "SGLang CVE-2026-5760: Understanding the Critical Remote Code Execution Vulnerability." Vulert Security Blog, April 2026.

[4] Cisco. "Introducing the AI Agent Security Scanner for IDEs: Verify Your Agents." Cisco Blogs, April 2026.

[5] SGLang Project. "SGLang: Efficient Execution of Structured Language Model Programs." GitHub, 2024–2026.

[6] OX Security. "The Mother of All AI Supply Chains: Critical Systemic Vulnerability at the Core of the MCP." OX Security Blog, April 2026.

[7] The Register. "Anthropic MCP Design Flaw Puts 200,000+ Servers at Risk." The Register, April 2026.

[8] CyberPress. "Hackers Could Weaponize GGUF Models to Achieve RCE on SGLang Inference Servers." CyberPress, April 2026.

[9] CERT/CC. "VU#665416: SGLang pickle deserialization vulnerabilities in ZMQ broker and disaggregation modules." CERT/CC Vulnerability Notes Database, April 2026.

[10] Orca Security. "SGLang LLM Framework RCE Vulnerabilities." Orca Security Blog, April 2026.

