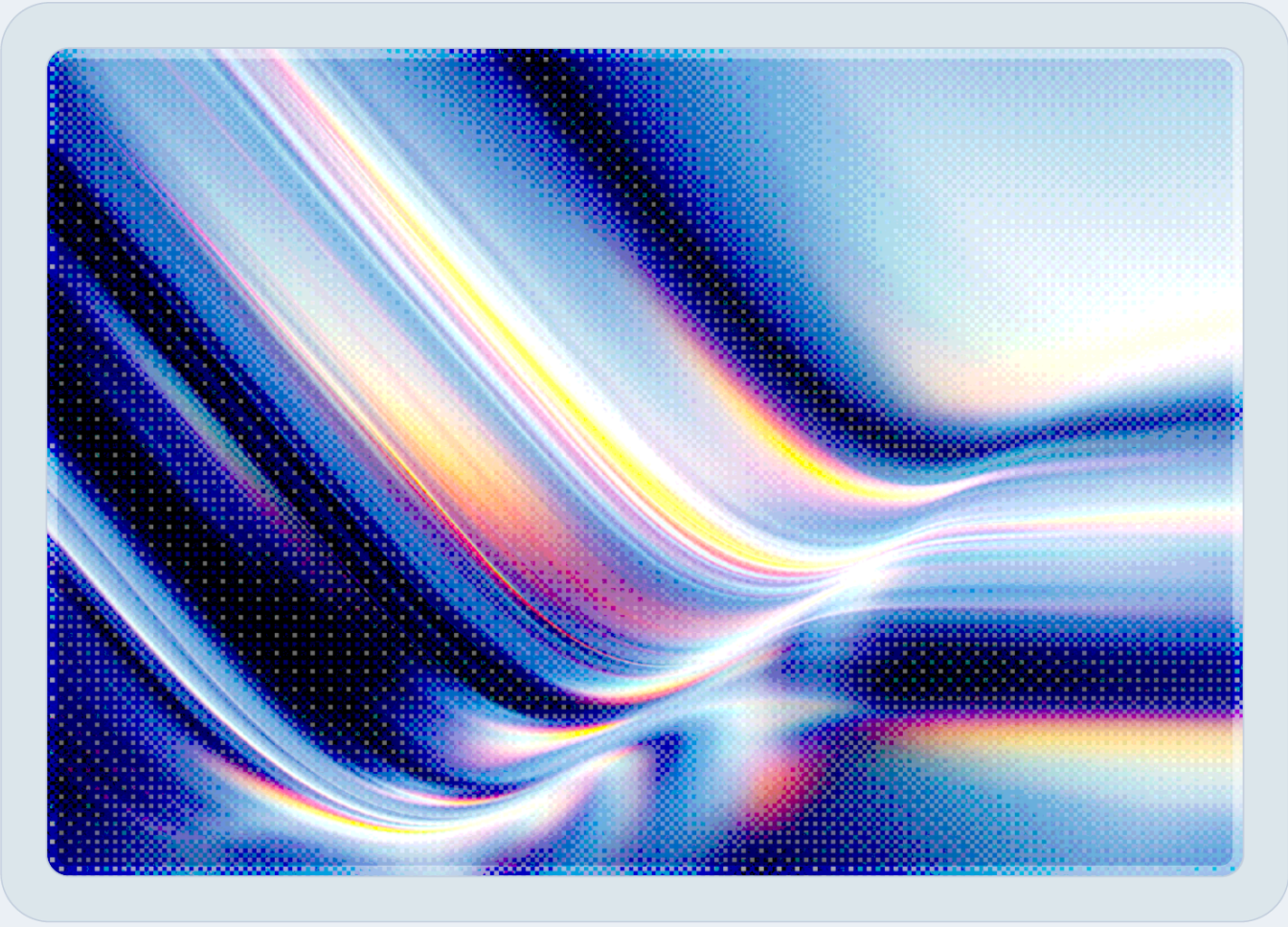
CSAI Foundation | Cloud Security Alliance

LangGraph Checkpoint RCE: SQL Injection to Code Execution

Chained Exploitation via SQLite Filter Injection and Unsafe Deserialization in AI Agent Frameworks

2026-06-12

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

A cluster of critical vulnerabilities disclosed between November 2025 and March 2026 demonstrates that the stateful memory infrastructure underpinning AI agents carries the same classical attack surface as any enterprise database application. The highest-complexity finding, and the one most likely to result in full host compromise, is a two-stage exploitation chain: an SQL injection vulnerability (CVE-2025-67644, CVSS 7.3) in LangGraph's SQLite checkpointer allows an attacker to manipulate which stored agent states are returned, and a separate deserialization flaw (CVE-2025-64439, CVSS 7.4) in LangGraph's `JsonPlusSerializer` converts that attacker-controlled data into remote code execution on the host server [1][2]. Organizations running LangGraph with user-controllable history queries and SQLite or Redis backends are exposed to this chain.

Three additional vulnerabilities in the broader LangChain framework compound the picture. A serialization injection flaw (CVE-2025-68664, CVSS 9.3) can expose LLM API keys and environment secrets to prompt-injection-aware attackers [3], and a path traversal vulnerability (CVE-2026-34070, CVSS 7.5) enables arbitrary file access through LangChain's prompt-loading API [4]. Taken together, these findings confirm that AI agent frameworks have not, by default, inherited the security controls that enterprise software engineers regard as baseline. Organizations operating LangGraph deployments should patch immediately and treat agent state persistence as a security-critical component.

Background

LangGraph is a stateful AI agent orchestration framework developed by LangChain, Inc., widely used for building multi-step, multi-actor AI workflows. Unlike stateless LLM API calls, LangGraph agents maintain persistent state through a checkpointing mechanism that serializes conversation history, intermediate reasoning steps, tool outputs, and metadata to a backing store — typically SQLite for local or small-scale deployments, or Redis for distributed production use. This checkpointing capability is the feature that enables LangGraph's core value proposition: agents that can pause, resume, and branch across complex, long-running tasks.

The attack surface implications of this design are significant. Because checkpoints represent the accumulated context of every tool call, credential exchange, and data access an agent has performed, a compromised checkpoint store is not merely a data breach — it is a complete operational record of what

the agent can do and what it has touched. LangGraph PyPI packages have accumulated approximately 46.5 million monthly downloads as of mid-2026, per Check Point Research's analysis [6], and production deployments span major financial institutions, technology firms, and professional services organizations, per LangChain's own adoption reporting [5]. The framework's rapid adoption means that vulnerabilities in its state persistence infrastructure carry enterprise-scale blast radius.

The vulnerabilities discussed in this note were identified through independent research by Check Point Research and Cyera Research. Check Point documented the SQL injection-to-deserialization RCE chain in their analysis of LangGraph's checkpoint handling [6]. Cyera Research disclosed three related vulnerabilities across the LangChain family through coordinated disclosure processes that concluded between December 2025 and March 2026 [3]. Patches exist for all documented vulnerabilities; a persistent concern at this stage of the disclosure cycle is that organizations deploying LangGraph through indirect dependencies or custom forks may not be aware of or have applied the patches — a pattern common to rapidly-adopted open-source frameworks.

Security Analysis

The RCE Exploitation Chain

While CVE-2025-68664 carries a higher base CVSS score (9.3), the most operationally severe risk for organizations hosting their own LangGraph infrastructure is the RCE chain formed by linking CVE-2025-67644 and CVE-2025-64439 — a combination that grants full host access rather than the partial data exposure enabled by a standalone secret exfiltration vulnerability.

The entry point is CVE-2025-67644, a SQL injection flaw in `langgraph-checkpoint-sqlite` versions prior to 3.0.1. The vulnerability resides in the `SqliteSaver._metadata_predicate()` method, which constructs SQL WHERE clauses for filtering agent history queries. While query *values* are properly parameterized, filter *keys* are interpolated directly into the SQL expression via f-strings without sanitization. An attacker who controls metadata filter keys — for example, through an API endpoint that exposes `get_state_history()` with user-supplied parameters — can inject payloads such as `"x') OR '1'='1"` to bypass metadata filtering entirely and retrieve checkpoint records they should not have access to [2]. In multi-tenant deployments, this access control bypass means one tenant can retrieve another's agent conversation history.

The second stage is CVE-2025-64439, a remote code execution vulnerability in `langgraph-checkpoint` versions prior to 3.0.0, classified under CWE-502 (Deserialization of Untrusted Data). LangGraph's `JsonPlusSerializer` normally serializes checkpoint payloads using msgpack. However, prior to version 3.0, when msgpack encounters illegal Unicode surrogate values during serialization, the library silently falls back to a JSON serialization mode. This JSON mode supports a constructor-style deserialization format: objects carrying `"type": "constructor"` cause the deserializer to dynamically import and execute Python functions specified in the payload's `"id"` field. Because the deserializer applied no allowlist before version 3.0, a malicious checkpoint payload can invoke arbitrary Python functions — including `os.system()` — when the checkpoint is loaded [1][7][8][10].

The chain requires two conditions: attacker-controlled data must be present in the checkpoint store — either submitted through the application's normal agent input flow or accessible via a multi-tenant store — and a SQL injection entry point must exist that delivers that data to the deserializer outside its authorized scope. When both conditions are met, an attacker can trigger deserialization of the malicious checkpoint payload to execute arbitrary code. The result is full server compromise, with access to LLM API keys, connected database credentials, conversation history, and any network resources reachable from the agent host [6].

CVE-2025-68664: Serialization Injection and Secret Exposure

CVE-2025-68664 (CVSS 9.3, CWE-502) affects LangChain's core serialization pipeline in `langchain_core/load/load.py`. The `loads()` and `load()` functions treat any dictionary carrying the field `"lc": 1` as a legitimate LangChain object during deserialization. User-controlled data that passes through serialization round-trips — including LLM response metadata in streaming flows via `astream_events()` and `astream_log()`, and data stored in `RunnableWithMessageHistory()` — is not escaped before being evaluated as a LangChain object. When the `Reviver` class processes such an object with `secrets_from_env` set to `True` (the default), it resolves environment variable references contained in the payload, effectively exfiltrating any secret accessible to the process [3].

The exploitation path combines prompt injection with this deserialization flaw: a crafted LLM response body containing a `"lc": 1` structure propagates through the serialization layer and causes the framework to read and potentially surface environment variables. Patches in langchain-core 1.2.5 introduced initial mitigations; version 1.2.22 introduces a full remediation for the reported attack path and exposes an allowlist configuration option.

CVE-2026-34070: Path Traversal in Prompt Loading

CVE-2026-34070 (CVSS 7.5, CWE-22) exposes a path traversal vulnerability in `langchain_core/prompts/loading.py`. The `load_prompt()` and `load_prompt_from_config()` functions accept external configuration files specifying template paths, example paths, and prefix/suffix values. None of these fields are validated against a base directory, allowing traversal sequences such as `../../../../.docker/config.json` to resolve to arbitrary file paths. Applications that accept external prompt configuration from users or from external orchestration sources are vulnerable to unauthorized file access, with Docker configurations, Kubernetes manifests, and `.env` files among the most sensitive targets [3][4]. The fix, available in langchain-core 1.2.22, was disclosed in March 2026, roughly 90 days after the December 2025 report date (the CVE identifier reflects its 2026 assignment date).

Structural Pattern: Classic Flaws in New Infrastructure

The set of vulnerabilities documented here — SQL injection from missing input sanitization on dictionary keys, deserialization RCE from unsafe fallback handling, path traversal from absent boundary validation — are not novel attack classes. They are the same categories that have defined enterprise application security for decades, now appearing in a new substrate. The distinguishing feature of AI agent frameworks is that this substrate has been deployed rapidly, at scale, carrying exceptionally sensitive data — API keys, multi-turn conversation histories, credentials to connected business systems — in a context where the engineering disciplines required to prevent classical injection and deserialization vulnerabilities have not yet been consistently applied. That all four CVEs in this cluster were addressed through coordinated disclosure confirms an active security response capability at LangChain; the challenge is extending that capability into proactive vulnerability prevention at the code-review and design stages. Cyera Research noted that "each vulnerability exposes a different class of enterprise data: filesystem files, environment secrets, and conversation history" [3] — an apt summary of what is at stake when this engineering gap persists.

Recommendations

Immediate Actions

Organizations running LangGraph with SQLite or Redis checkpointing should upgrade to the patched package versions without delay. The minimum required versions are langgraph-checkpoint-sqlite 3.0.1 or later (addressing CVE-2025-67644), langgraph-checkpoint 3.0.0 or later (addressing CVE-2025-64439), and langgraph 1.0.10 or later for the broader framework. LangChain applications should be upgraded to langchain-core 1.2.22 or later to address both CVE-2025-68664 and CVE-2026-34070 in a single update. For CVE-2025-68664 specifically, set `secrets_from_env=False` in serialization configurations as an interim measure while upgrades are being tested.

Audit all application surfaces that expose `get_state_history()`, `list()`, or `alist()` checkpoint query methods with parameters derived from user input, external API calls, or agent-generated data. Any such endpoint is a potential SQL injection entry point for CVE-2025-67644 and should be isolated, blocked from accepting external filter key parameters, or removed until patching is confirmed; rate-limiting may reduce abuse opportunity but does not prevent exploitation.

Short-Term Mitigations

After applying patches, treat the checkpoint store as a privileged data tier rather than an application convenience layer. Checkpoint databases should not be accessible from the public network, should require authentication for all access, and should be scoped to the minimum network footprint needed for the application to function. For SQLite-backed local deployments, apply filesystem-level access controls to the checkpoint database file. For Redis-backed deployments, enforce Redis authentication and TLS in transit, and restrict the Redis instance to the local network segment.

Implement input validation at the API layer for any application that accepts metadata filter parameters and passes them to LangGraph checkpoint queries. Until custom validation is in place, reject filter keys that contain SQL metacharacters, parentheses, or quote characters. This is a defensive layer, not a substitute for the patch.

For LangChain prompt-loading functionality, validate all external prompt configuration paths against an explicit allowlist of permitted directories before passing them to `load_prompt()` or `load_prompt_from_config()`. Do not accept template paths from user-supplied input without this boundary.

Strategic Considerations

The vulnerability cluster disclosed against LangGraph and LangChain points to a pattern that will recur as the AI agent framework ecosystem matures. Frameworks that began as developer convenience tools are now deployed as enterprise infrastructure handling privileged credentials, customer data, and access to critical business systems. The security practices appropriate to that positioning — systematic code review for injection and deserialization vulnerabilities, a security-specific dependency update process, and threat modeling for data flows through stateful persistence layers — have not yet been consistently applied across the ecosystem [12].

Organizations building AI agent infrastructure should require the same dependency management rigor for AI framework packages that they apply to other enterprise middleware [11]. This means tracking framework dependencies in software bill of materials tooling, subscribing to security advisories from framework maintainers, and establishing a defined SLA for patching critical and high severity CVEs in agent framework components. Organizations with LangGraph deployments where the checkpoint store holds credentials or sensitive customer data should consider threat modeling that exercises the agent's state persistence path as an attack surface, not just the model inference path.

CSA Resource Alignment

The LangGraph vulnerability chain maps directly to Layer 3 (Agent Frameworks) of the CSA MAESTRO threat modeling framework for agentic AI [9]. MAESTRO identifies input validation flaws, supply chain attacks, and deserialization vulnerabilities in agent development toolkits as representative Layer 3 threats. The SQL injection to RCE chain demonstrated by CVE-2025-67644 and CVE-2025-64439 is a concrete example of exactly this threat pattern: a flaw in the framework's internal data handling, reachable through attacker-controlled input to the agent's state management API.

Layer 2 (Data Operations) of MAESTRO captures the downstream consequence: once the checkpoint store is compromised, data poisoning, exfiltration, and tampering against the agent's persisted knowledge base become accessible attack objectives. The secrets exposure risk from CVE-2025-68664 represents cross-layer impact, reaching from agent framework infrastructure into the operational environment.

CSA's AI Controls Matrix (AICM) addresses checkpoint and state persistence security through controls in the AI Supply Chain and AI Data Governance domains. Organizations implementing AICM should ensure that state persistence stores — SQLite databases, Redis instances, and similar checkpointing backends — are included in the asset inventory and subject to the same access control, encryption, and vulnerability management requirements as other data stores handling sensitive information. CSA's Zero Trust

guidance applies directly to the network architecture question raised by these vulnerabilities: checkpoint stores should not be implicitly trusted because they reside on an internal network, and the traffic paths by which agent frameworks read and write checkpoint data should be subject to identity-based access policy.

References

[1] LangChain AI. "RCE in 'json' mode of JsonPlusSerializer — GHSA-wwqv-p2pp-99h5." GitHub Security Advisories, November 2025.

[2] mbanyamer. "CVE-2025-67644 LangGraph 3.0.1 SQLite Checkpoint SQL Injection (community PoC)." GitHub, 2025.

[3] Cyera Research. "LangDrained: 3 Paths to Your Data Through the World's Most Popular AI Framework." Cyera, March 2026.

[4] The Hacker News. "LangChain, LangGraph Flaws Expose Files, Secrets, Databases in Widely Used AI Frameworks." The Hacker News, March 2026.

[5] LangChain. "State of Agent Engineering." LangChain, 2025.

[6] Check Point Research. "When Your AI Agent's Memory Becomes a Security Liability." Check Point Blog, 2026.

[7] SentinelOne. "CVE-2025-64439: LangGraph Deserialization RCE Vulnerability." SentinelOne Vulnerability Database, 2025.

[8] NIST National Vulnerability Database. "CVE-2025-64439 Detail." NVD, November 2025.

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

[10] CyberPress. "Critical Flaw in LangGraph Allows Remote Code Execution via Deserialization." CyberPress, 2025.

[11] BeyondScale. "LangChain LangGraph Security: 2026 CVE Hardening Guide." BeyondScale, 2026.

[12] OWASP. "OWASP Top 10 for Agentic Applications." OWASP, 2025.

