LangChain/LangGraph: Critical Flaws in the AI Dev Stack

Path Traversal, Unsafe Deserialization, and SQL Injection Across the Industry's Most Widely Deployed LLM Application Frameworks

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-27

Key Takeaways

- Three distinct vulnerabilities have been disclosed in LangChain and LangGraph — collectively downloaded more than 84 million times in the most recent reporting week — exposing enterprise filesystem data, API credentials, and database contents to unauthorized access [1].
- CVE-2025-68664 (CVSS 9.3 per GitHub CNA; 8.2 per NVD independent analysis), the most severe of the three, enables deserialization of untrusted data to leak API keys and environment secrets; this flaw was previously disclosed in December 2025 under the name "LangGrinch" and apparently not fully remediated across all package versions until the current release cycle [1][7][8].
- CVE-2026-34070 (CVSS 7.5) introduces a path traversal flaw in LangChain's prompt-loading subsystem, allowing an attacker to read arbitrary files from the host filesystem without authentication or access controls [1].
- CVE-2025-67644 (CVSS 7.3) permits SQL injection through unsanitized metadata filter keys in LangGraph's SQLite checkpoint implementation, enabling arbitrary query execution against the underlying database [1][9].
- Patches are available for all three vulnerabilities; organizations should prioritize upgrading to langchain-core ≥1.2.22, langchain-core 0.3.81/1.2.5, and langgraph-checkpoint-sqlite 3.0.1 respectively, and conduct an immediate audit of any environment variables or credentials accessible to LangChain or LangGraph processes [1][8][9].
- These disclosures arrive in the immediate aftermath of CVE-2026-33017 in Langflow, which was weaponized within 20 hours of public disclosure, suggesting that threat actors are actively monitoring AI framework vulnerability feeds and operationalizing exploits rapidly [1] [2].

Background

LangChain and LangGraph have become foundational infrastructure for a large share of enterprise AI development. LangChain provides the core abstractions — chains, agents, memory, and retrieval — that allow developers to compose LLM-powered applications from modular components, while LangGraph extends that model into stateful, graph-based agent workflows capable of coordinating multi-step reasoning across tool invocations and persistent memory states. Together with the LangChain-Core library, these packages are downloaded at extraordinary scale: LangChain, LangChain-Core, and LangGraph received approximately 52 million, 23 million, and 9 million downloads respectively in a single recent week [1]. That aggregate reach means that a single exploitable flaw anywhere in the stack carries systemic consequence — not just for individual deployments, but for the upstream trust chains, API credentials, and enterprise data pipelines those deployments touch.

The security profile of LLM application frameworks differs in important ways from that of traditional middleware. A conventional web application framework is typically granted access to its own application data and, through controlled ORM layers, its associated database. LangChain and LangGraph deployments, by design, aggregate access to a much broader set of sensitive resources: LLM provider API keys (for OpenAI, Anthropic, Google, and others), RAG vector database credentials, SQL and document store connection strings, environment variables containing cloud provider tokens, and the full text of every conversation processed through the system. An attacker who achieves data exfiltration from a LangChain host therefore gains leverage not just over a single application, but potentially over every connected AI service, cloud account, and downstream integration the deployment manages.

Security researcher Vladimir Tokarev of Cyera disclosed the three vulnerabilities on March 27, 2026, characterizing them as "three independent paths" that allow an adversary to "drain sensitive data from any enterprise LangChain deployment" [1][10]. The disclosure accompanied the release of patched package versions across all three affected components.

Security Analysis

CVE-2026-34070: Path Traversal in the Prompt-Loading API

The first vulnerability, rated CVSS 7.5, resides in `langchain_core/prompts/loading.py` and exploits the absence of path validation in LangChain's prompt template loading mechanism. When an application uses LangChain's API to load prompt templates from the local filesystem, the framework accepts caller-supplied paths without restricting traversal sequences. An attacker capable of influencing the path value — whether through direct API access, an injection attack on a higher-level application layer, or a malicious tool invocation within an agent workflow — can escape the intended directory scope and read arbitrary files accessible to the LangChain process [1].

The practical impact of this flaw is significant because LangChain processes in many production deployments execute with broad filesystem access, particularly in containerized environments where secrets are mounted or written to predictable paths. In those configurations, this traversal enables an attacker to enumerate and exfiltrate environment files, application configurations, and credential materials without triggering authentication or authorization checks. In agent architectures where the LangChain process is granted tool-calling capabilities over local resources, the traversal may compound with other tool permissions to extend the exfiltration surface further. Patches addressing CVE-2026-34070 are included in langchain-core version 1.2.22 and later [1].

CVE-2025-68664: Unsafe Deserialization Enabling Credential Theft

The highest-severity of the three disclosures, CVE-2025-68664, carries a CVSS score of 9.3 as assigned by the GitHub CNA — a score reflecting critical severity. NVD's independent analysis of the same vulnerability yields a score of 8.2 (HIGH), and both values appear in the official record [7][8]. Regardless of which scoring authority an organization references, the vulnerability warrants urgent attention: it reflects a deserialization weakness that enables leakage of API keys and environment secrets through attacker-crafted data structures. The vulnerability arises when LangChain deserializes inputs without adequate type restriction or integrity validation, permitting an adversary to supply a payload that causes the deserialization routine to read and transmit credential values from the process's environment [1][7][8].

This vulnerability is notable for its disclosure history. Cyera previously identified the same underlying weakness in December 2025 under the informal name "LangGrinch," and the flaw appears to have been partially addressed at that time without complete remediation across all maintained package versions [1]. The March 2026 disclosure represents a second iteration of coordinated vulnerability management for the same class of flaw, suggesting that the underlying deserialization architecture may warrant more comprehensive redesign rather than point-in-time patching. Concurrent patch releases targeting both langchain-core 0.3.81 (the 0.x maintenance branch) and langchain-core 1.2.5 (the 1.x branch) indicate that the vulnerability was present across both actively maintained major versions, broadening the affected population considerably [1][8].

The credential exfiltration risk is particularly acute in enterprise LangChain deployments, where environment variable injection is the standard method for supplying provider API keys. In deployments where environment variables supply LLM provider API keys, database connection strings, and cloud provider tokens — a common pattern in production configurations — exploitation of CVE-2025-68664 can yield access to a broad set of sensitive credentials. The actual exposure surface depends on which secrets are loaded into the affected process environment, the scope of permissions attached to those credentials, and whether secrets management tooling is used to inject credentials ephemerally rather than persistently.

CVE-2025-67644: SQL Injection in LangGraph's Checkpoint Implementation

The third vulnerability, CVE-2025-67644, rated CVSS 7.3, affects LangGraph's SQLite-backed checkpoint system. LangGraph uses checkpoints to persist agent state across steps and sessions, and the SQLite implementation accepts metadata filter keys from callers to query checkpoint records. These keys are incorporated into SQL queries without adequate sanitization, creating a classical SQL injection pathway through which an attacker can execute arbitrary queries against the underlying database [1][9].

The affected component — the `langgraph-checkpoint-sqlite` package — may appear narrowly scoped, but the checkpoint store is designed to persist multi-turn conversation histories, intermediate agent reasoning traces, retrieved document contents, and cached tool outputs across steps and sessions. In deployments that leverage LangGraph's full stateful capabilities, this data may be present in the checkpoint store and subject to unauthorized access, modification, or deletion through an injection attack. The implications range from conversation history disclosure to tampering with the agent's persistent memory in ways that influence future behavior. The patch, released in langgraph-checkpoint-sqlite version 3.0.1, addresses the injection pathway through query parameterization [1][9].

A Maturing Attack Surface for AI Frameworks

The simultaneous disclosure by a single researcher of three distinct vulnerability classes — path traversal, deserialization, and SQL injection — across two closely related packages indicates the depth of scrutiny that LangChain is now receiving in targeted professional security audits. Such focused single-researcher audits, covering multiple vulnerability categories across related components, are a pattern that historically precedes more widespread adversarial attention once a platform reaches critical mass adoption. This trajectory mirrors what was observed with traditional web frameworks in the mid-2010s, when systematic security audits of platforms such as Ruby on Rails and Apache Struts — as those frameworks scaled to enterprise adoption — surfaced clusters of foundational vulnerabilities and preceded sustained adversarial targeting.

The recurrence of CVE-2025-68664 after an earlier December 2025 disclosure compounds this concern. Incomplete remediation of high-CVSS deserialization vulnerabilities — a class with an established track record of weaponization — indicates that the development teams managing these frameworks may benefit from structured security programs, including architectural threat modeling, automated dependency scanning integrated into CI/CD pipelines, and coordinated vulnerability disclosure processes with defined SLA expectations. Related academic research examines adjacent attack surfaces — including system prompt exploitation in LLM agent configurations [3] and injection techniques in MCP-based frameworks [4] — indicating that the broader LLM application security research community is active in mapping these threat surfaces. Security teams should monitor this research landscape alongside framework-specific vulnerability disclosures, as findings from adjacent protocol and framework research frequently inform techniques that are later adapted to other targets. Research specifically examining how database-layer attacks combine with prompt injection in RAG architectures [5] and how browser-based AI agents can be defended against indirect prompt injection [6] further illustrates the multi-layer threat model emerging for AI application stacks.

The proximity in time to the active exploitation of Langflow CVE-2026-33017 — weaponized within 20 hours of public disclosure — reinforces that organizations cannot rely on obscurity or delayed attacker uptake as a risk mitigation strategy for AI framework vulnerabilities [2]. Threat actors monitoring AI infrastructure disclosure feeds have demonstrated the capability to operationalize exploits rapidly, and the credential-rich nature of LangChain deployments makes them high-value targets worth the investment.

Recommendations

Immediate Actions

Organizations should treat CVE-2025-68664 (CVSS 9.3 per CNA / 8.2 per NVD) as requiring emergency response given the credential exfiltration potential and its prior incomplete remediation. CVE-2026-34070 (7.5) and CVE-2025-67644 (7.3) warrant expedited patch cycles, with prioritization based on deployment exposure and the presence of compensating controls. The specific upgrade targets are: langchain-core 1.2.22 or later for CVE-2026-34070; langchain-core 0.3.81 (for 0.x deployments) or 1.2.5 (for 1.x deployments) for CVE-2025-68664; and langgraph-checkpoint-sqlite 3.0.1 for CVE-2025-67644 [1][8][9]. Where package upgrades are blocked by dependency constraints or internal approval processes, organizations should implement compensating controls immediately — particularly for CVE-2025-68664 — including removing unnecessary environment variables from LangChain process environments and rotating any API keys that may have been accessible.

Following patching, incident response teams should review application logs for anomalous path requests in prompt-loading calls (indicators of CVE-2026-34070 exploitation), unexpected deserialization errors or environment variable access patterns (CVE-2025-68664), and malformed metadata filter values in LangGraph checkpoint queries (CVE-2025-67644). Given that CVE-2025-68664 was previously known under the "LangGrinch" designation, teams should consider whether earlier exploitation activity in their environments went undetected during the December 2025–March 2026 window when the flaw existed in a partially-patched state. While no exploitation during this window has been publicly reported as of this publication, the combination of elevated CVSS score, credential-theft impact, and partial-only remediation makes the possibility worth investigating as a precautionary measure.

- Upgrade all three affected packages to patched versions per the schedule above [1][8][9]
- Rotate API keys and environment secrets accessible to LangChain or LangGraph processes as a precautionary measure
- Audit package lock files and transitive dependencies for outdated langchain-core and langgraph-checkpoint-sqlite versions
- Review deployment logs for evidence of exploitation attempts predating the patch cycle

Short-Term Mitigations

Beyond the immediate patch cycle, organizations should apply the principle of least privilege to LangChain and LangGraph process environments. The credential-aggregation risk exposed by CVE-2025-68664 is structurally reduced when LLM application processes are granted only the API keys and secrets they require for their specific function, with separate processes handling separate integrations. Secret management platforms — HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, or equivalent — should be used to supply credentials at runtime through secure injection mechanisms rather than static environment variable files, limiting the credential exposure window for any given process execution.

For LangGraph deployments using the SQLite checkpoint backend, teams should evaluate whether the more operationally complex but inherently safer alternative checkpoint backends (PostgreSQL-backed, or in-memory for stateless deployments) better match their threat model. Where SQLite checkpoints remain in use, input validation on metadata filter keys should be added at the application layer as defense-in-depth even after patching, since parameterization at the library level does not protect against injection through intermediate application layers that construct filter values from untrusted sources.

Network segmentation for LangChain inference and orchestration hosts can reduce the blast radius of path traversal exploitation. Restricting outbound connections from LangChain processes to only the endpoints required for LLM API calls and approved integrations limits an attacker's ability to exfiltrate retrieved filesystem data to external collection infrastructure.

Strategic Considerations

The LangChain and LangGraph vulnerability disclosures are a signal, not an isolated event. As LLM application frameworks accumulate the scale and adoption that once characterized web application servers and container orchestration platforms, they are likely to attract the sustained, systematic security research that historically follows critical mass. Security teams should establish a dedicated watch function for LangChain ecosystem vulnerability disclosures, including monitoring the GitHub advisories feeds for the `langchain-ai` organization, the PyPI advisory database, and researchers working in the LLM application security space.

More broadly, the three vulnerability classes exposed here – path traversal, deserialization, and SQL injection – are foundational categories that decades of secure development guidance have identified as preventable through consistent application of well-understood controls: input validation and path canonicalization, type-safe deserialization with explicit allowlists, and parameterized query construction. Their presence in widely deployed AI application frameworks suggests that security-by-design practices may not yet be fully embedded in the development lifecycle of the LLM tooling ecosystem, whether due to growth velocity, resource constraints, or the inherent complexity of the expanding attack surface. Organizations integrating LangChain or LangGraph into critical business processes should include these frameworks in their SAST and DAST testing pipelines, and should evaluate whether vendor security posture is a factor in framework selection decisions for future AI infrastructure investments.

Academic research examining how LLM agent configuration itself constitutes an attack surface [3] and how database-layer attacks can combine with prompt injection in RAG architectures [5] – along with defensive research on protecting browser-based AI agents against indirect prompt injection [6] – points toward a threat landscape where AI application vulnerabilities will increasingly combine multiple attack vectors rather than presenting as isolated flaws. Security architecture for AI applications should be designed with this multi-layer threat model in mind.

CSA Resource Alignment

These vulnerabilities have direct relevance to several active CSA frameworks and initiatives.

The **MAESTRO (Multi-Agent Environment, Security, Trust, Risk, and Orchestration)** framework provides a structured threat model for agentic AI systems specifically applicable to LangChain and LangGraph deployments. MAESTRO's focus on inter-agent trust relationships and the security of orchestration layers maps directly to the risk that CVE-2025-67644 and CVE-2025-68664 pose in multi-agent architectures where a compromised orchestration component can propagate credential theft or data manipulation across all subordinate agents. Organizations should apply MAESTRO's threat enumeration process to their LangChain and LangGraph agent graphs to identify the highest-impact exploitation paths within their specific deployment topologies.

The **AI Infrastructure and Control Matrix (AICM)**, CSA's superset of the Cloud Controls Matrix adapted for AI workloads, addresses controls in the domains of supply chain security, runtime isolation, and secret management that are directly implicated by these vulnerabilities. AICM control domains covering least-privilege access for AI processes, credential lifecycle management for LLM provider integrations, and software composition analysis for AI development dependencies should be reviewed and gap-assessed against the compensating controls described above.

The **CSA AI Organizational Responsibilities** guidance establishes accountability structures for AI security incidents, which are relevant here because the recurrence of CVE-2025-68664 after an earlier partial disclosure raises questions about vendor security accountability and the adequacy of coordinated disclosure processes for high-severity AI framework vulnerabilities. Organizations procuring AI development tooling should reference this guidance when establishing contractual expectations around disclosure timelines and patch SLAs with framework providers.

CSA's **Zero Trust Architecture** guidance applies to the network segmentation and identity controls recommended above. The zero-trust principle of assuming breach is particularly important for LangChain and LangGraph hosts given their credential-aggregation role: even where perimeter defenses are in place, runtime controls should be designed on the assumption that an AI orchestration process may at any time be executing attacker-influenced code paths or serving as a pivot point for lateral credential exfiltration.

A previous CSA AI Safety Initiative research note addressing CVE-2026-33017 in Langflow covers the broader pattern of rapid weaponization of AI platform vulnerabilities and provides complementary guidance on incident response and threat intelligence for this ecosystem [2].

References

[1] Tokarev, Vladimir (Cyera) / The Hacker News. "LangChain, LangGraph Flaws Expose Files, Secrets, Databases in Widely Used AI Frameworks." The Hacker News, March 27, 2026. https://thehackernews.com/2026/03/langchain-langgraph-flaws-expose-files.html

[2] Cloud Security Alliance AI Safety Initiative. "Langflow RCE CVE-2026-33017: Exploited Within 20 Hours." CSA Research Note, March 25, 2026. /output/white-papers/CSA_research_note_CVE-2026-33017-langflow-AI-pipeline-RCE-20260325.md

[3] Litvak, Ron. "The System Prompt Is the Attack Surface: How LLM Agent Configuration Shapes Security and Creates Exploitable Vulnerabilities." arXiv:2603.25056, March 2026. https://arxiv.org/abs/2603.25056

[4] Shen, Yulin; Pan, Xudong; Hong, Geng; Yang, Min. "Invisible Threats from Model Context Protocol: Generating Stealthy Injection Payload via Tree-based Adaptive Search." arXiv:2603.24203, March 2026. https://arxiv.org/abs/2603.24203

[5] Wang, Haozhen; Liu, Haoyue; Zhu, Jionghao; Wang, Zhichao; Guo, Yongxin; Tang, Xiaoying. "PIDP-Attack: Combining Prompt Injection with Database Poisoning Attacks on Retrieval-Augmented Generation Systems." arXiv:2603.25164, March 2026. https://arxiv.org/abs/2603.25164

[6] Lan, Qianlong; Kaul, Anuj. "The Cognitive Firewall: Securing Browser Based AI Agents Against Indirect Prompt Injection Via Hybrid Edge Cloud Defense." arXiv:2603.23791, March 2026. https://arxiv.org/abs/2603.23791

[7] National Vulnerability Database. "CVE-2025-68664 Detail." NIST NVD, 2025. https://nvd.nist.gov/vuln/detail/CVE-2025-68664

[8] GitHub Security Advisory. "GHSA-c67j-w6g6-q2cm: LangChain Unsafe Deserialization." GitHub Advisory Database, 2025. https://github.com/advisories/GHSA-c67j-w6g6-q2cm

[9] GitHub Security Advisory. "GHSA-9rwj-6rc7-p77c: LangGraph SQL Injection in Checkpoint SQLite." GitHub Advisory Database, langchain-ai/langgraph, 2025. https://github.com/langchain-ai/langgraph/security/advisories/GHSA-9rwj-6rc7-p77c

[10] Cyera Research Labs. Researcher profile: Vladimir Tokarev, Senior Security Researcher. https://www.cyera.com/research