LangChain and LangGraph: Critical Vulnerabilities in AI Orchestration

Serialization Flaws, Injection Chains, and Remediation Guidance

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-29

© 2026 Cloud Security Alliance. Unofficial AI-assisted Research.

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

- A coordinated disclosure by Cyera Research on March 27, 2026 ("LangDrained") [1] identified multiple high- and critical-severity vulnerabilities across LangChain and LangGraph, compounding a pattern of serious flaws documented throughout 2024 and 2025. Organizations building agentic AI applications on these frameworks face material risk until all affected components are patched. Separate but related research by Cyata ("LangGrinch") [2] provided the detailed technical analysis of CVE-2025-68664; Cyera's coordinated disclosure incorporated and extended that work.
- The highest-severity vulnerability in this analysis cycle, CVE-2025-68664 (CVSS 9.3), enables serialization injection that allows an attacker to leverage prompt injection to escalate into arbitrary code execution, secret extraction from environment variables, and instantiation of trusted internal classes. [2]
- LangGraph's stateful checkpoint persistence layer contains two independently exploitable vulnerabilities: a remote code execution flaw via deserialized checkpoint data (CVE-2025-64439) [3] and a SQL injection flaw in the SQLite checkpoint backend (CVE-2025-67644, CVSS 7.3). [4] These are particularly significant in enterprise agentic deployments where checkpoint stores accumulate sensitive conversation state.
- Prompt injection serves as an initial access vector that enables downstream exploitation of higher-severity vulnerabilities in multiple CVEs documented here, rather than functioning solely as a standalone risk.
- Patches are available for all documented vulnerabilities. Teams should treat the following upgrades as urgent remediation actions: `langchain-core >= 0.3.81` (CVE-2025-68664), `langchain-core >= 1.2.22` (CVE-2026-34070), `langgraph-checkpoint >= 3.0`, and `langgraph-checkpoint-sqlite >= 3.0.1`.

Background

LangChain is an open-source Python (and Go) framework for building applications powered by large language models. First released in late 2022, it rapidly became the dominant orchestration library for LLM-backed applications, providing abstractions for chaining prompts, managing memory, integrating tools, and interacting with external data sources. Its companion framework, LangGraph, extends LangChain's capabilities to support stateful, multi-actor agentic workflows modeled as directed graphs, with built-in support for persistent state checkpoints. Both frameworks are maintained by LangChain Inc. and, according to Cyera's LangDrained disclosure, are used across a wide range of enterprise deployments in sectors including financial services, healthcare, legal, and government. [1][10]

The frameworks' popularity makes their security posture a systemic concern. A production LangChain or LangGraph deployment commonly operates with access to database credentials, API keys, document stores, and email or calendar systems. An attacker who achieves code execution within such an environment does not merely compromise an application — they gain a foothold with access to the full set of integrations the agent was granted. The exploitation impact is therefore amplified beyond what a typical web application presents: code execution in an agent environment yields not just application access but access to every external system the agent was credentialed against.

The LangChain ecosystem is organized into several distinct packages that are versioned and patched independently: `langchain-core`, `langchain`, `langchain-community`, `langchain-experimental`, `langgraph`, `langgraph-checkpoint`, and `langgraph-checkpoint-sqlite`, among others. This modular package structure means that a security team patching `langchain` may inadvertently leave a vulnerable version of `langchain-community` or `langgraph-checkpoint` in place. Effective vulnerability management requires tracking each package separately.

Security Analysis

Serialization Injection and the "LangGrinch" Vulnerability (CVE-2025-68664)

The highest-severity vulnerability disclosed in this analysis cycle, CVE-2025-68664 (CVSS 9.3), is a serialization injection flaw in `langchain-core` that was publicly named "LangGrinch" by its discoverers at Cyata. [2][9][12] The flaw exists in the `dumps()` and `dumpd()` serialization functions, which fail to sanitize user-controlled dictionaries containing the reserved `lc` key. LangChain's internal

object model uses `lc` as a marker to identify trusted, native framework objects during deserialization. When an attacker injects content containing this key — most practically by steering an LLM agent via prompt injection to produce crafted structured output — the subsequent deserialization step interprets the attacker-controlled data as a trusted LangChain object. Cyata's original LangGrinch research was subsequently incorporated into Cyera's broader LangDrained coordinated disclosure, which covers this CVE alongside two additional vulnerabilities.

The exploitation chain is multi-stage and depends on context, but the documented impact includes three distinct consequences. First, when `secrets_from_env=True` was set (the prior default), the deserialization path could extract environment variables — including API keys and database credentials — and include them in the serialized output. Second, the flaw enables instantiation of arbitrary classes within the trusted namespaces `langchain_core`, `langchain`, and `langchain_community`, effectively giving an attacker access to any class constructor reachable from those packages. Third, in configurations using Jinja2 prompt templates, this class instantiation path can reach the template engine and achieve arbitrary code execution. The vulnerability received a CVSS score of 9.3 (Critical) and affected all versions of `langchain-core` prior to 0.3.81 and `langchain` versions 1.0.0 through 1.2.4. [2]

The patch introduced in `langchain-core 0.3.81` addresses the flaw by adding an `allowed_objects` allowlist that restricts which classes may be instantiated during deserialization, changing the default for `secrets_from_env` to `False`, and blocking Jinja2 template rendering from the deserialization path. Organizations that have deployed `langchain-core` or `langchain` without upgrading past these versions should treat remediation as an emergency response, particularly if their deployment processes any user-controlled input that reaches the serialization layer.

LangGraph Checkpoint Vulnerabilities

LangGraph's checkpoint persistence layer — the mechanism by which stateful agent workflows persist and resume conversation state — contains two independently exploitable vulnerabilities that together represent a serious risk to enterprise agentic deployments.

CVE-2025-64439 affects `langgraph-checkpoint`, the default checkpoint serialization library used across all LangGraph state backends. The `JsonPlusSerializer` component normally uses msgpack for serialization, but falls back to JSON mode when msgpack encounters illegal Unicode surrogate values. In JSON mode, a `_reviver` method dynamically imports and executes Python code based on the `id` field of any deserialized object typed as `"constructor"`. [3] An attacker who can persist a crafted payload into the checkpoint store — for example, by manipulating a message that the

agent will process and save — can cause arbitrary Python code execution during the next checkpoint load. The fix in `langgraph-checkpoint 3.0` introduces an allowlist that restricts which `id` paths are permissible during constructor deserialization.

CVE-2025-67644 is a SQL injection vulnerability in the SQLite checkpoint backend (`langgraph-checkpoint-sqlite < 3.0.1`). [4] The `SqliteSaver.list()` and `alist()` methods call an internal `_metadata_predicate()` function that uses Python f-strings to directly embed metadata filter keys supplied by the caller into SQL queries without parameterization. An attacker who controls these filter keys can execute arbitrary SQL statements against the checkpoint database. Given that checkpoint databases in production deployments may contain complete conversation histories, agent instructions, tool invocation records, and session tokens, successful exploitation could enable an attacker to read or exfiltrate conversation histories, agent instructions, and session tokens stored in the checkpoint database. The fix in `langgraph-checkpoint-sqlite 3.0.1` replaces the f-string SQL construction with parameterized queries.

Cypher and Database Injection via Prompt Amplification

CVE-2024-8309, disclosed in October 2024, represents a class of vulnerability that is conceptually distinct from traditional injection flaws. The `GraphCypherQAChain` component in `langchain-community` 0.2.5 translates natural-language questions into Cypher queries for graph databases via an LLM. Because the chain applied no validation or sandboxing to the LLM-generated query before execution, an attacker who could control the input prompt could inject Cypher syntax that caused the chain to execute unauthorized operations against the connected graph database. [5][11] The NVD/NIST assessment assigns a CVSS score of 9.8 for this vulnerability; the vendor CNA and GitHub Advisory assessments rate the severity at 4.9 Medium and 2.1 Low respectively, reflecting differing assessments of exploitability conditions. Teams should review the applicable advisory for their environment when prioritizing remediation.

This vulnerability is significant as a pattern rather than solely as an isolated flaw. It demonstrates that wherever a framework uses an LLM to generate code, queries, or commands that are then executed without an intermediate validation step, the LLM's output surface becomes an injection vector. The same pattern has appeared in SQL generation chains, shell command chains, and code execution chains across the LangChain ecosystem. The fix — an explicit `allow_dangerous_requests=True` opt-in — is a design acknowledgment that such chains should not be production-safe by default.

Unsafe Deserialization of Persistent Stores (CVE-2024-5998)

The FAISS vector store integration in `langchain-community` prior to version 0.2.4 called `pickle.loads()` directly on data read from disk via `FAISS.deserialize_from_bytes()`, without any integrity or trust validation of the source bytes. [6] Because Python's `pickle` module is an arbitrary code execution primitive — any pickled object can invoke `os.system()` or `subprocess` during deserialization — any attacker who could write to or substitute the FAISS index file could achieve persistent code execution on the host. This vulnerability is relevant in deployment scenarios where vector stores are loaded from shared filesystems, object storage buckets with insufficient access controls, or externally supplied model artifacts. The fix required callers to explicitly set `allow_dangerous_deserialization=True`, making the risk opt-in rather than the default.

XML External Entity Injection in EverNoteLoader (CVE-2025-6984)

CVE-2025-6984 is an XML External Entity (XXE) injection vulnerability in the `EverNoteLoader` component of `langchain-community`, with a CVSS score of 7.5 (High). [14] The loader parses EverNote export files (`.enex` format) without disabling external entity resolution in its XML parser. An attacker who supplies a maliciously crafted `.enex` file — for example through a shared document ingestion pipeline or a compromised document source — can cause the loader to make server-side requests or read local files from the host filesystem. XXE vulnerabilities of this class are well-understood to enable server-side request forgery, local file disclosure, and, in some configurations, denial of service. The vulnerability is patched in `langchain-community >= 0.3.27`.

Unsafe Code Execution in Experimental Components

The `langchain-experimental` package has accumulated a disproportionate density of vulnerabilities stemming from a common root cause: LLM-generated output being passed directly to Python's `eval()` or `exec()` built-ins without sanitization. CVE-2024-21513 documents this pattern in `VectorSQLDatabaseChain` versions 0.0.15 through 0.0.20, where all values returned by the database were passed to `eval()` prior to use, enabling an attacker to inject executable Python through database content. [7] The earlier CVE-2023-44467 in PALChain exhibits the identical pattern. The concentration of eval-based execution vulnerabilities in `langchain-experimental` may suggest insufficient security review prior to publication, or may reflect that the package's "experimental" designation was not consistently understood by downstream adopters as a signal of elevated security risk.

Organizations that have deployed any components from `langchain-experimental` in production environments should audit those deployments for exposure to eval-based execution chains, regardless of whether the specific vulnerable versions are in use.

Path Traversal in Prompt Configuration Loading

CVE-2026-34070, disclosed as part of the March 2026 LangDrained coordinated disclosure, affects the `load_prompt()` and `load_prompt_from_config()` functions in `langchain-core` prior to version 1.2.22. [1][13] These functions accept file paths embedded in deserialized configuration dictionaries and read from those paths without enforcing that the resolved path remains within an expected directory. An attacker who can influence the content of a prompt configuration — for example through a poisoned template repository or a misconfigured configuration management pipeline — can read arbitrary files from the host filesystem, subject only to a file-extension check. This class of path traversal vulnerability could enable an attacker to read sensitive files such as `.env` files, SSH private keys, TLS certificates, or Docker daemon configuration. The CVSS score of 7.5 reflects the requirement that the attacker must first achieve influence over the prompt configuration loading path.

GmailToolkit Indirect Prompt Injection (CVE-2025-46059)

CVE-2025-46059 documents an indirect prompt injection vulnerability in the GmailToolkit component of `langchain-ai` version 0.3.51 and prior. [8] In this attack, a malicious actor crafts an email message whose content contains adversarial instructions for the LLM agent. When an agent using GmailToolkit reads and processes the email, the injected instructions redirect the agent's behavior — potentially causing it to exfiltrate inbox contents, send unauthorized messages, or trigger downstream tool invocations. This vulnerability is illustrative of a broader class of risk in LangChain's tool integrations: any integration that causes an agent to process untrusted external content (emails, web pages, database records, calendar entries) is a potential indirect prompt injection surface.

Recommendations

Immediate Actions

Organizations currently running LangChain or LangGraph in production should treat the following package upgrades as urgent remediation actions that warrant emergency change procedures where their risk posture warrants it.

- Upgrade `langchain-core` to version `0.3.81` or later to address CVE-2025-68664 (serialization injection). [2] For deployments also exposed to CVE-2026-34070 (path traversal in prompt loading), upgrade to `langchain-core 1.2.22` or later. [13]
- Upgrade `langchain` to version `1.2.5` or later to address CVE-2025-68664 in the `langchain` package. [2]
- Upgrade `langchain-community` to version 0.3.27 or later, which addresses CVE-2025-6984 (XXE in EverNoteLoader) [14] and CVE-2024-5998 (pickle deserialization in FAISS). [6]
- Upgrade `langgraph-checkpoint` to version 3.0 or later to address CVE-2025-64439 (deserialization RCE in JsonPlusSerializer). [3]
- Upgrade `langgraph-checkpoint-sqlite` to version 3.0.1 or later to address CVE-2025-67644 (SQL injection in metadata filtering). [4]
- Audit all production deployments for use of `langchain-experimental` components, particularly any that use `PALChain` or `VectorSQLDatabaseChain`. These components have a documented history of executing LLM-generated code and should be replaced or isolated from production data.

Short-Term Mitigations

Beyond version upgrades, several architectural and configuration controls reduce the attack surface of LangChain and LangGraph deployments. Operators should audit all agent integrations that process external content — including email toolkits, web browsing tools, and document loaders — for indirect prompt injection exposure. An adversary-controlled email message, web page, or database record that will be summarized or acted upon by an agent should be treated as untrusted input, and agent architectures should include a validation step before any tool-use decision that originated from such content.

Operators should also review the access scope granted to agents. An agent that requires access to Gmail for inbox summarization should not hold credentials for database write operations, code deployment pipelines, or other high-impact systems. Least-privilege scoping of agent credentials limits the scope of access an attacker gains through successful exploitation: an agent credentialed only for inbox summarization cannot pivot to database write operations or deployment pipelines even after its LLM component is compromised. Similarly, checkpoint databases and vector stores should be stored with appropriate filesystem access controls and should not be world-readable or writable by unprivileged processes.

For teams using FAISS or other vector stores loaded from files, enforcing integrity verification — for example, cryptographic hash validation of index files before loading — provides a defense-in-depth control against deserialization attacks even when the framework's own protections are present.

Strategic Considerations

The pattern of vulnerabilities documented here reflects early design decisions made as LangChain was built for rapid adoption and developer flexibility. Several of the most severe flaws — serialization injection, unsafe deserialization, eval-based execution — appear to be consequences of early design decisions rather than isolated implementation errors. Whether these reflected deliberate tradeoffs or insufficient security review at the design stage, the effect has been compounding security debt as enterprise adoption grew and the frameworks were applied in high-stakes contexts their original design did not anticipate.

Security teams should conduct a formal evaluation of whether LangChain and LangGraph meet their organization's production security requirements, independent of the framework's utility during development. For the highest-risk deployments, teams may wish to consider whether a narrower, purpose-built integration — rather than a general-purpose orchestration framework with broad tool access — provides a more defensible attack surface.

Organizations should establish a dependency monitoring process specific to the LangChain ecosystem's multi-package structure. Because `langchain`, `langchain-core`, `langchain-community`, `langchain-experimental`, `langgraph`, `langgraph-checkpoint`, and `langgraph-checkpoint-sqlite` are versioned independently, a single `pip install langchain --upgrade` does not guarantee that all vulnerable components are addressed. Dependency tracking tools such as Dependabot, Snyk, or Renovate should be configured to monitor each package independently and to alert on new CVEs.

CSA Resource Alignment

This research note connects directly to several Cloud Security Alliance frameworks and publications relevant to AI system security.

MAESTRO (Multi-Agent Environment Security, Trust, Risk, and Operational Guidance) provides the threat modeling foundation for the agentic vulnerabilities documented here. The LangGraph checkpoint deserialization flaw (CVE-2025-64439) maps to MAESTRO's coverage of trust boundary

violations in stateful agent workflows, where an attacker who compromises the persistence layer gains the ability to manipulate agent state across sessions. The indirect prompt injection documented in CVE-2025-46059 and the Cypher injection in CVE-2024-8309 illustrate the MAESTRO threat model for tool-use manipulation, where an adversary redirects an agent's tool invocations through malicious content in the environment.

The CSA AI Organizational Responsibilities framework addresses the governance dimension of this disclosure. The modular package versioning of the LangChain ecosystem underscores the importance of the framework's guidance on AI component inventory — organizations cannot protect what they have not inventoried. The framework's emphasis on assigning explicit ownership for AI dependencies applies directly to the multi-package management complexity described here.

The CSA AI Infrastructure Control Matrix (AICM), which extends the Cloud Controls Matrix (CCM) to AI-specific controls, provides relevant controls in several domains. The deserialization and serialization vulnerabilities map to AICM controls governing data integrity and input validation in AI pipelines. The SQL injection in the checkpoint backend maps to AICM controls for secure data storage practices in AI systems. The prompt injection amplification pattern maps to AICM's coverage of adversarial input controls.

CSA Zero Trust guidance is directly applicable to the agent credential scoping recommendations in this note. Zero Trust principles — assume breach, verify explicitly, use least-privilege access — translate concretely to the practice of ensuring that LangChain agents are granted only the minimum necessary permissions and that credential scopes are bounded to the specific task the agent is performing.

References

[1] Cyera Research, "LangDrained: 3 Paths to Your Data Through the World's Most Popular AI Framework," Cyera, March 2026. https://www.cyera.com/research/langdrained-3-paths-to-your-data-through-the-worlds-most-popular-ai-framework

[2] Cyata, "LangGrinch: CVE-2025-68664 — Critical Serialization Injection in langchain-core," Cyata AI Blog, December 2025. https://cyata.ai/blog/langgrinch-langchain-core-cve-2025-68664/ — see also GitHub Advisory GHSA-c67j-w6g6-q2cm and NVD entry CVE-2025-68664.

[3] GitHub Security Advisory GHSA-wwqv-p2pp-99h5, "langgraph-checkpoint: RCE via JsonPlusSerializer Deserialization," 2025. https://github.com/advisories/GHSA-wwqv-p2pp-99h5 — see also NVD CVE-2025-64439.

[4] GitHub Security Advisory GHSA-9rwj-6rc7-p77c, "langgraph-checkpoint-sqlite: SQL Injection in SqliteSaver metadata filtering," 2025. https://github.com/langchain-ai/langgraph/security/advisories/GHSA-9rwj-6rc7-p77c — see also Snyk SNYK-PYTHON-LANGGRAPHCHECKPOINTSQLITE-14361682 and NVD CVE-2025-67644.

[5] GitHub Security Advisory GHSA-45pg-36p6-83v9, "langchain-community: Cypher Injection via GraphCypherQAChain," October 2024. https://github.com/advisories/GHSA-45pg-36p6-83v9 — see also NVD CVE-2024-8309. Note: NVD/NIST assessment is 9.8 Critical; vendor CNA assessment is 4.9 Medium; GitHub Advisory shows 2.1 Low.

[6] GitHub Security Advisory GHSA-f2jm-rw3h-6phg, "langchain-community: Pickle Deserialization of Untrusted Data in FAISS," 2024. https://github.com/advisories/GHSA-f2jm-rw3h-6phg — see also NVD CVE-2024-5998.

[7] NVD, "CVE-2024-21513: Arbitrary Code Execution in langchain-experimental VectorSQLDatabaseChain," National Vulnerability Database, 2024. https://nvd.nist.gov/vuln/detail/cve-2024-21513

[8] GitHub Security Advisory GHSA-3g6x-vq45-v2jv, "langchain-ai: Indirect Prompt Injection in GmailToolkit (CVE-2025-46059)," 2025. https://github.com/advisories/GHSA-3g6x-vq45-v2jv

[9] The Hacker News, "Critical LangChain Core Vulnerability Allows Remote Code Execution via Serialization Flaw," December 2025. https://thehackernews.com/2025/12/critical-langchain-core-vulnerability.html

[10] The Hacker News, "LangChain/LangGraph Flaws Expose Files and Data to Attackers," March 2026. https://thehackernews.com/2026/03/langchain-langgraph-flaws-expose-files.html

[11] Keysight Technologies, "CVE-2024-8309: Prompt Injection and Cypher Injection in LangChain," Keysight Network & Visibility Blog, August 2025. https://www.keysight.com/blogs/en/tech/nwvs/2025/08/29/cve-2024-8309

[12] Orca Security, "CVE-2025-68664: LangChain Serialization Flaw Analysis," Orca Security Blog, 2025. https://orca.security/resources/blog/cve-2025-68664-langchain-serialization-flaw/

[13] GitHub Security Advisory GHSA-qh6h-p6c9-ff54, "langchain-core: Path Traversal in load_prompt() functions (CVE-2026-34070)," March 2026. https://github.com/langchain-ai/langchain/security/advisories/GHSA-qh6h-p6c9-ff54

[14] GitHub Security Advisory GHSA-pc6w-59fv-rh23, "langchain-community: XML External Entity Injection in EverNoteLoader (CVE-2025-6984)," 2025. https://github.com/advisories/GHSA-pc6w-59fv-rh23 — see also NVD CVE-2025-6984.