LibVulnWatch: A Deep Assessment Agent System and Leaderboard for Uncovering Hidden Vulnerabilities in Open-Source AI Libraries

Zekun Wu [*] [1 2] **Seonglae Cho** [*] [1 2] **Umar Mohammed** [1] **Cristian Munoz** [1] **Kleyton Costa** [1] **Xin Guan** [1] **Theo King** [1] **Ze Wang** [1 2] **Emre Kazim** [1 2] **Adriano Koshiyama** [1 2]

Abstract

Open-source AI libraries are foundational to modern AI systems but pose significant, underexamined risks across security, licensing, maintenance, supply chain integrity, and regulatory compliance. We present LibVulnWatch, a graph-based agentic assessment framework that performs deep, source-grounded evaluations of these libraries. Built on LangGraph, the system coordinates a directed acyclic graph of specialized agents to extract, verify, and quantify risk using evidence from trusted sources such as repositories, documentation, and vulnerability databases. LibVulnWatch generates reproducible, governance-aligned scores across five critical domains, publishing them to a public leaderboard for longitudinal ecosystem monitoring. Applied to 20 widely used libraries, including ML frameworks, LLM inference engines, and agent orchestration tools, our system covers up to 88% of OpenSSF Scorecard checks while uncovering up to 19 additional risks per library. These include critical Remote Code Execution (RCE) vulnerabilities, absent Software Bills of Materials (SBOMs), licensing constraints, undocumented telemetry, and widespread gaps in regulatory documentation and auditability. By translating high-level governance principles into practical, verifiable metrics, LibVulnWatch advances technical AI governance with a scalable, transparent mechanism for continuous supply chain risk assessment and informed library selection.

1. Introduction

As AI systems are adopted in high-stakes settings, there is growing pressure to ensure that governance objectives are grounded in technical practice. Yet a gap remains between policy intent and the engineering work required to meet it, limiting oversight and increasing the risk of unintended harms (Reuel et al., 2025). *Technical AI Governance* aims to address this challenge by developing tools and methods that enable safe, verifiable, and accountable deployment.

A critical yet underexamined area is the AI software supply chain. Open-source libraries and frameworks underpin most machine learning systems but introduce legal, security, maintenance, and regulatory risks that are often missed by conventional assessment tools (Wang et al., 2025; Alevizos et al., 2024), which typically provide surface-level checks and lack a holistic, in-depth view of these vulnerabilities.

We present LIBVULNWATCH, an agent-based system aligned with key Technical AI Governance capacities—*Assessment*, *Security*, *Operationalisation*, and *Ecosystem Monitoring* (Reuel et al., 2025). The system uses LangGraph to coordinate LLM agents for structured evaluations across five domains: licensing, security, maintenance, dependency management, and regulatory compliance. Each finding is supported by verifiable evidence from repositories, advisories, and documentation.

Results are published to a public Hugging Face leaderboard[1] to support ongoing monitoring. Applied to 20 widely used AI libraries—covering ML frameworks, inference tools, and agent systems—LIBVULNWATCH shows:

- Up to **88% coverage** of OpenSSF Scorecard checks;
- **Up to 19 additional risks** per library, including RCEs, missing SBOMs, and compliance gaps;
- **Governance-aligned, reproducible scores** for transparent comparison and decision-making.

LIBVULNWATCH helps operationalize governance principles through technical evaluation, supporting more informed and accountable use of open-source AI infrastructure.

[*]Equal contribution [1]Holistic AI [2]University College London. Correspondence to: Adriano Koshiyama <adriano.koshiyama@holisticai.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]The leaderboard and full assessment reports will be made publicly available on Hugging Face upon paper acceptance.

2. Related Work

While adversarial inputs and data poisoning are well-studied, system-level vulnerabilities in AI pipelines are gaining attention (Wang et al., 2025). Wang et al. analyzed 529 LLM supply chain issues, exposing flaws in application and serving components. Chen et al. found recurring bugs in frameworks like TensorFlow and PyTorch (Chen et al., 2023). LLM-based vulnerability detection has been explored (Zhou et al., 2024), though false positives remain a concern. Supply chain threats such as dependency confusion and package hijacking are detailed by Ladisa et al. and Ohm et al.(Ladisa et al., 2023; Ohm et al., 2020). Tools like OpenSSF Scorecard assess project hygiene(Zahan et al., 2023), but focus on surface metrics and lack deep vulnerability analysis. LangChain and LangGraph support multi-agent workflows (LangChain AI, 2025a;b), forming the basis for several assessment pipelines. Our system extends this infrastructure with domain-specific agents focused on software risk. This supports goals in technical AI governance around traceability, compliance, and oversight (Raji et al., 2020; Reuel et al., 2025). It complements efforts like the AI Incident Database (McGregor, 2021) and bug bounty platforms (e.g., Huntr (Huntr, 2024)) by enabling public, ongoing evaluation of AI libraries.

3. Methodology

Our approach operationalizes key Technical AI Governance capacities via a multi-stage, agentic evaluation pipeline and integrated search. This section details the pipeline's architecture, risk assessment framework, evaluation protocol, and benchmarking procedures.

3.1. Governance-Aligned Risk Assessment Framework

We define a comprehensive risk assessment framework adapted from established open-source and AI risk taxonomies. It encompasses five governance-relevant domains: License Analysis (assessing legal terms, commercial use, patent grants, compliance); Security Assessment (evaluating known vulnerabilities, security history, patching, policy adherence); Maintenance Indicators (analyzing release frequency, contributor activity, project governance, issue resolution); Dependency Management (examining SBOM availability, transitive risks, supply chain controls); and Regulatory Considerations (reviewing documentation for GDPR/AI Act alignment, explainability, and audit readiness). Each domain is operationalized as a distinct assessment module within the agentic workflow, guided by engineered prompts enforcing key concept coverage and quantifiable metric extraction.

3.2. Agentic, Graph-Based Assessment Workflow

Our system employs a structured, agentic workflow implemented as a DAG using a modern agent orchestration framework. For more detailed implementation and experimental settings, please refer to Appendix A.1. This ensures modularity, consistency, and parallelism. Integrating the LLM, structured data handling, and search capabilities (utilizing standard web search APIs like Google Search API, with support for local RAG), the workflow comprises:

- **Planning:** An initial agent generates a detailed assessment plan adhering strictly to the five core risk domains and formulates initial research queries based on the target library.
- **Iterative Section Synthesis:** For each planned section (corresponding to a risk domain), a dedicated agent iteratively performs targeted searches against authoritative sources (official documentation, security databases, repository metadata using specialized query patterns), aggregates evidence, and synthesizes assessments. This synthesis process is strictly governed by prompts enforcing:
  - **Structured Reporting:** Each risk factor is reported in a predefined table with the factor, observed data, risk rating, justification, and a proposed control.
  - **Quantification Mandate:** Agents must extract specific, verifiable metrics (e.g., counts, dates, versions) instead of qualitative summaries.
  - **Evidence Requirement:** All key claims must be supported by direct citation links to the source evidence.
  - **Missing Information:** Agents must indicate when information is unavailable; such gaps raise the risk rating by default. A quality check enforces completeness and, if unmet, triggers a loop with refined queries (up to a fixed depth) to improve evidence or fill gaps.
- **Report Compilation:** A final agent synthesizes individual section findings into a consolidated report, including an executive summary adhering to a defined structure (risk dashboard, emergency issues, prioritized controls, mitigation strategy).
- **Public Reporting and Ecosystem Monitoring:** The finalized report is programmatically published to a public leaderboard. This facilitates *Ecosystem Monitoring* and accountability by dynamically ranking libraries by Trust Score and highlighting key risks. We follow responsible disclosure practices for any new, non-public vulnerabilities identified during the assessment.

3.3. Library Selection and Evaluation Protocol

We evaluated 20 diverse open-source AI libraries spanning the AI lifecycle, selected for representative coverage (see Table 1 for list and scores). Libraries were chosen from three key functional categories: Core ML/DL Frameworks include PyTorch (Paszke et al., 2019), TensorFlow (Abadi et al., 2016), ONNX (ONNX, 2025), Huggingface Transformers (Wolf et al., 2020) and JAX (Bradbury et al., 2025); LLM Inference & Orchestration tools include TensorRT (NVIDIA, 2025), LlamaIndex (Liu, 2022), SGLang (Zheng et al., 2023), vLLM(Kwon et al., 2023), LangChain (LangChain AI, 2025a), and Text Generation Inference (Hugging Face, 2025); and AI Agent Frameworks include Browser Use (Müller & Žunič, 2024) CrewAI (CrewAI, 2025), MetaGPT(Zhang & colleagues, 2024), LangGraph (LangChain AI, 2025b), SmolAgents (Roucher et al., 2025), Stagehand (Browserbase, 2025), Composio (Composio, 2025), Pydantic AI (Pydantic, 2025), and Agent Development Kit (Google, 2025) . Selection aimed for diversity in function, community size, maturity, and impact. Each library underwent the full protocol; results are public.

3.4. Assessment Metrics and Scoring

We employ a 1-5 numerical scale for risk rating within each of the five governance-relevant domains outlined above (Section 3.1), where 1 indicates High Risk, 3 Medium Risk, and 5 Low Risk. As detailed in the workflow description (Section 3.2), each rating requires justification tied to specific, verifiable evidence thresholds defined in the prompts. **The detailed criteria defining Low Risk (Score 5), Medium Risk (Score 3), and High Risk (Score 1) for each domain are provided in Appendix A.3.** Intermediate scores (2 or 4) may be assigned based on the agent's assessment when evidence suggests a risk level between these defined thresholds. Critically, the absence of necessary information for assessment (e.g., no public security policy or SBOM) on any key risk factor is also explicitly defined as a High Risk indicator (Score 1). The overall Trust Score provides a composite measure by aggregating the five domain scores (Li, Se, Ma, De, Re):

$$\text{Trust}(l) = \frac{1}{5} \sum_{d \in \{Li, Se, Ma, De, Re\}} d(l)$$

.

3.5. Baseline Benchmarking and Novelty Analysis

We use the OpenSSF Scorecard (Zahan et al., 2023) as a baseline to evaluate our agent. This involves identifying the main repository, running the Scorecard, and comparing its output with our agent's report to derive two key metrics:

- **Baseline Alignment(%)**: The percentage of relevant Scorecard checks addressed in the agent's report, calculated as Coverage $= \frac{\text{\# matched checks}}{\text{\# applicable checks}} \times 100$.
- **Novelty Yield (#)**: The number of unique, meaningful issues identified by the agent but missed by the Scorecard, defined as Yield = # unique agent-only findings.

4. Results

We assessed 20 open-source AI libraries—including core ML/DL frameworks, LLM inference/orchestration tools, and agent frameworks—using our agent system. Benchmarking against the OpenSSF Scorecard (Zahan et al., 2023), we identified risk areas and demonstrated the unique strengths of agent-based evaluation. Example reports can be found in Appendix A.5, A.7, A.8, A.9, and A.6.

4.1. Benchmarking and Alignment Analysis

We benchmarked our agentic system against the OpenSSF Scorecard to evaluate alignment and identify unique contributions. Table 1 presents key metrics defined in Section 3—Baseline Alignment (overlap with Scorecard checks) and Novelty Yield (unique findings)—across all evaluated libraries, grouped by functional category and including category averages. While observed Baseline Alignment for most libraries ranged from 55% to 88%, indicating substantial overlap, the agentic system consistently surfaced a significant Novelty Yield (typically 5-13 unique findings per library) not captured by baseline tools. Notably, agentic reports sometimes missed formal contributor declarations, CI testing evidence, binary artifact identification, and explicit security testing policies flagged by the baseline, suggesting areas for agent refinement or continued explicit checks. Examples of critical risks identified by the agentic process beyond conventional automated scans, contributing to Novelty Yield, include:

- Complex RCEs from insecure defaults or subtle data processing flaws.
- Systemic SBOM absence and supply chain/transitive dependency risks.
- Pervasive regulatory/privacy compliance gaps (GDPR, HIPAA, AI Act).
- Widespread lack of governance mechanisms (audit trails, explainability, privacy controls).
- Undocumented telemetry/data collection (e.g., in one AI agent framework).
- Potential patent risks from unclear/insufficient licensing for core ML algorithms.

Table 1. Baseline Alignment and Novelty Yield Across Libraries

Library	Baseline Alignment (%)	Novelty Yield (#)
Core ML/DL Frameworks	***77.1***	*6.8*
PyTorch	88.2	8
JAX	61.1	12
Tensorflow	72.2	5
ONNX	87.5	5
Huggingface Transformers	76.5	4
LLM Inference & Orchestration	*73.7*	*7.8*
TensorRT	68.8	5
LlamaIndex	82.4	7
SGLang	73.3	5
vLLM	73.3	7
LangChain	72.2	19
Text Generation Inference	72.2	6
AI Agent Frameworks	*76.2*	***9.1***
Browser Use	88.2	7
CrewAI	71.4	13
MetaGPT	57.1	7
LangGraph	77.8	7
SmolAgents	73.3	9
Stagehand	83.3	6
Composio	68.8	5
Pydantic AI	88.2	10
Agent Development Kit	77.9	7

4.2. Aggregated Domain Risk Findings and Patterns

Detailed library-by-library trust scores across the five primary domains and the composite Trust Score are provided in the Appendix (Table 2). Analysis of these scores reveals notable patterns across the evaluated libraries and risk domains. Aggregate Trust Scores varied by category, with Core ML/DL frameworks generally scoring higher than newer AI Agent frameworks, potentially reflecting greater maturity. Common weaknesses were observed across the ecosystem, particularly in:

- **Dependency Management:** Widespread absence of SBOMs hindering transparency, poorly managed transitive dependencies, and lack of automated vulnerability scanning were common.
- **Regulatory Considerations:** Significant gaps existed regarding comprehensive documentation for GDPR/HIPAA/AI Act compliance and features for model explainability or audit logging.
- **Security:** Many libraries exhibited vulnerabilities like RCEs, unsigned releases, and insecure CI/CD pipelines, with newer frameworks often lacking mature disclosure policies.
- **License Analysis:** While often permissive, nuanced risks like potential patent issues or conflicts with restrictive licenses (e.g., AGPL) were found, and formal patent grants were frequently missing.
- **Maintenance Indicators:** Established libraries showed robust core maintenance, but patterns of unmaintained sub-projects or less transparency/slower resolution in newer frameworks posed risks.

5. Discussion and Limitation

Our findings indicate that while many evaluated AI libraries are technically sophisticated, they often exhibit significant deficits in enterprise readiness, particularly concerning the common weaknesses identified in supply chain security and regulatory preparedness detailed in Section 4.2. Proactive vulnerability management also appears underdeveloped across the ecosystem. The agentic approach effectively identified critical risks missed by standard checks, with high Novelty Yield often pointing to complex vulnerabilities (RCEs, supply chain flaws, license risks, governance gaps) requiring contextual understanding.

Benchmarking against OpenSSF Scorecard contextualizes these findings. The substantial Baseline Alignment observed (detailed in Section 4.1) confirms that the agent recognizes standard indicators, while variations across library categories are insightful. Higher alignment in mature Core ML/DL frameworks likely reflects better documentation accessibility. The observation that newer AI Agent Frameworks showed higher alignment than LLM Inference tools might stem from recent community scrutiny. A library's functional niche seems more predictive of its risk profile than complexity alone.

5.1. Limitation

Despite its strengths, our approach has limitations. The agent did not consistently identify all baseline Scorecard factors (e.g., binary artifacts, contributor agreements), suggesting areas for refinement. Assessments are limited by public information availability and represent a snapshot (May 2025) of rapidly evolving libraries. Assessment quality depends on underlying LLM capabilities and potential biases; prompt sensitivity could influence results. Computational resources for deep analysis might pose scalability challenges for continuous monitoring. Finally, the pervasive lack of documentation on regulatory compliance, privacy, and explainability across the ecosystem inherently limits assessment depth in these critical domains.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

Alevizos, V., Papakostas, G. A., Simasiku, A., Malliarou, D., Messinis, A., Edralin, S., Xu, C., and Yue, Z. Integrating artificial open generative artificial intelligence into software supply chain security. In *2024 5th International Conference on Data Analytics for Business and Industry (ICDABI)*, pp. 200–206. IEEE, October 2024. doi: 10.1109/icdabi63787.2024.10800301. URL http://dx.doi.org/10.1109/ICDABI63787.2024.10800301.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al. Jax: composable transformations of python+ numpy programs. https://github.com/google/jax, 2025. Accessed: 2025-05-12.

Browserbase. Stagehand: The production-ready framework for ai browser automations. https://github.com/browserbase/stagehand, 2025. Accessed: 2025-05-12.

Chen, J., Liang, Y., Shen, Q., Jiang, J., and Li, S. Toward Understanding Deep Learning Framework Bugs. *ACM Transactions on Software Engineering and Methodology*, 32(6):135:1–135:31, 2023.

Composio. Composio: Production-ready toolset for ai agents. https://github.com/ComposioHQ/composio, 2025. Accessed: 2025-05-12.

CrewAI. CrewAI: Fast and flexible multi-agent automation framework. https://github.com/crewAIInc/crewAI, 2025. Accessed: 2025-05-12.

Google. Agent Development Kit: An open-source, code-first python toolkit for building, evaluating, and deploying sophisticated ai agents with flexibility and control. https://github.com/google/adk-python, 2025. Accessed: 2025-05-12.

Hugging Face. Text Generation Inference: Large language model text generation inference. https://github.com/huggingface/text-generation-inference, 2025. Accessed: 2025-05-12.

Huntr. Huntr: The world's first bug bounty platform for ai/ml. https://huntr.dev, 2024. Accessed: 2025-05-12.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023. URL https://arxiv.org/abs/2309.06180.

Ladisa, P., Plate, H., Martinez, M., and Barais, O. SoK: Taxonomy of Attacks on Open-Source Software Supply Chains. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1509–1526, 2023.

LangChain AI. LangChain: Large language model application framework. https://github.com/langchain-ai/langchain, 2025a. Accessed: 2025-05-12.

LangChain AI. LangGraph: An open-source ai agent orchestration framework. https://docs.langchain.com/docs/langgraph, 2025b. Accessed: 2025-05-12.

Liu, J. LlamaIndex, 11 2022. URL https://github.com/jerryjliu/llama_index.

McGregor, S. Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database. In *Proceedings of the Thirty-Third AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 2021.

Müller, M. and Žunič, G. Browser use: Enable ai to control your browser, 2024. URL https://github.com/browser-use/browser-use.

NVIDIA. TensorRT: High-performance deep learning inference on nvidia gpus. https://github.com/NVIDIA/TensorRT, 2025. Accessed: 2025-05-12.

Ohm, M., Plate, H., Sykosch, A., and Meier, M. Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks. In *Proceedings of the 17th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pp. 23–43, 2020.

ONNX. ONNX: Open neural network exchange. https://github.com/onnx/onnx, 2025. Accessed: 2025-05-12.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information*

Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Pydantic. Pydantic AI: shim to use pydantic with llms. https://github.com/pydantic/pydantic-ai, 2025. Accessed: 2025-05-12.

Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., et al. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* 2020)*, pp. 20–31, 2020.

Reuel, A., Bucknall, B., Casper, S., Fist, T., Soder, L., Aarne, O., Hammond, L., Ibrahim, L., Chan, A., Wills, P., Anderljung, M., Garfinkel, B., Heim, L., Trask, A., Mukobi, G., Schaeffer, R., Baker, M., Hooker, S., Solaiman, I., Luccioni, S., Rajkumar, N., Moës, N., Ladish, J., Bau, D., Bricman, P., Guha, N., Newman, J., Bengio, Y., South, T., Pentland, A., Koyejo, S., Kochenderfer, M., and Trager, R. Open problems in technical AI governance. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=1nO4qFMiS0. Survey Certification.

Roucher, A., del Moral, A. V., Wolf, T., von Werra, L., and Kaunismäki, E. ‘smolagents‘: a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

Wang, S., Zhao, Y., Liu, Z., Zou, Q., and Wang, H. SoK: Understanding Vulnerabilities in the Large Language Model Supply Chain. *arXiv preprint arXiv:2502.12497*, 2025.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Zahan, N., Kanakiya, P., Hambleton, B., Shohan, S., and Williams, L. Openssf scorecard: On the path toward ecosystem-wide automated security metrics. *IEEE Security & Privacy*, 21(6):76–88, 2023. doi: 10.1109/MSEC.2023.3279773.

Zhang, f. n. o. f. b. and colleagues. Metagpt: Meta programming sota autonomous multi-agent cooperative llm workflows. *arXiv preprint arXiv:2404.14496*, 2024. URL https://arxiv.org/abs/2404.14496.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., and Sheng, Y. Sglang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2023. URL https://arxiv.org/abs/2312.07104.

Zhou, X., Cao, S., Sun, X., and Lo, D. Large Language Model for Vulnerability Detection and Repair: Literature Review and the Road Ahead. *ACM Transactions on Software Engineering and Methodology*, 2024. to appear.

A. Appendix

A.1. Implementation Details

Our implementation was inspired by the Open Deep Research repository[2]. We redesign the graph design and defined domain-specific prompts based on relevant knowledge to guide the evaluation. All experiments used `gpt-4.1-mini`. OpenSSF Scorecard (Zahan et al., 2023) checks were run on the primary GitHub repository of each target library. We used the Google Search API for evidence retrieval. The automated workflow begins with high-level search-based planning, followed by domain-specific iterative retrieval until sufficient evidence is gathered for each of the five domains. These are processed in parallel to generate draft findings, which are combined into a full report including an executive summary. The report is then validated by identifying the main GitHub repository, running the Scorecard, and comparing outputs using an LLM. Figure 1 shows the full workflow, including parallel domain evaluation, retrieval loops, synthesis, and benchmarking.

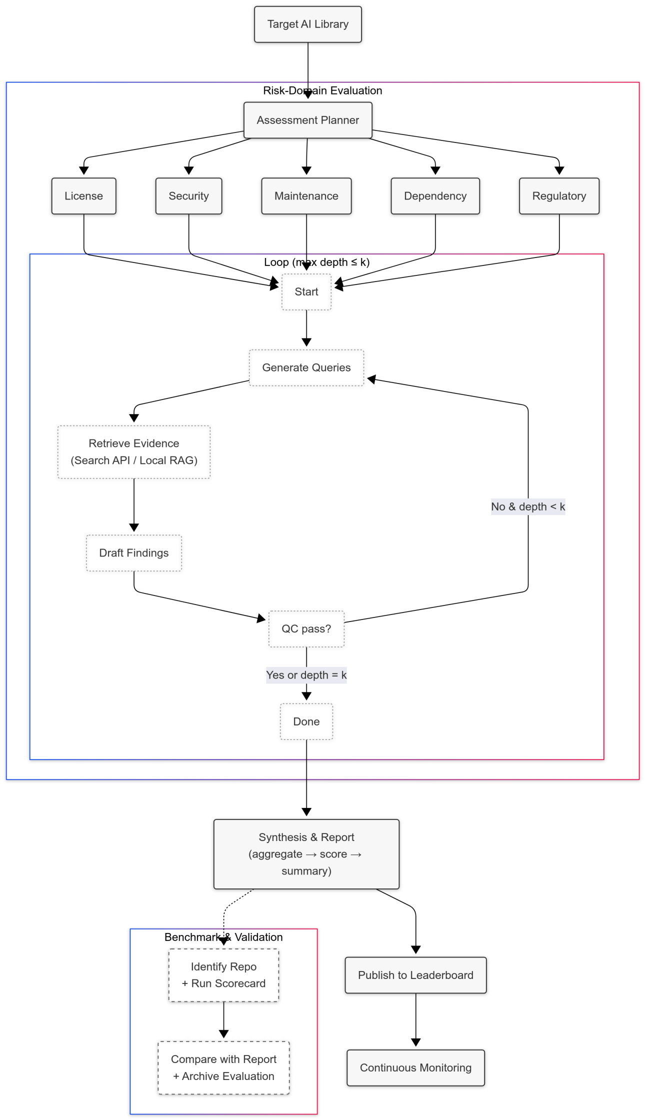


Figure 1. Workflow of the automated agent. Each risk domain (License, Security, Maintenance, Dependency, Regulatory) runs in parallel, with controlled-depth evidence retrieval and drafting. The results are synthesized into a report, benchmarked using the OpenSSF Scorecard, and then published with monitoring.

[2]https://github.com/langchain-ai/open_deep_research

A.2. Detailed Risk Assessment Scores

Table 2 presents the aggregated trust scores for each library across the five primary risk domains and the composite Trust Score, as delineated in Section 3. Scores are normalized to a 1–5 scale (higher scores indicate lower risk). This quantitative overview offers a comparative snapshot of specific strengths and vulnerabilities prevalent throughout the AI library ecosystem. The table also includes group averages for the Trust Score.

Table 2. Detailed Risk Assessment Scores Across Libraries and Domains (Li: License, Se: Security, Ma: Maintenance, De: Dependency, Re: Regulatory, Trust: Trust Score; Scale: 1-5, higher is better)

Library	Li	Se	Ma	De	Re	Trust
Core ML/DL Frameworks						*13.0*
PyTorch	5	1	3	1	3	13
JAX	5	3	4	1	1	**14**
Tensorflow	5	1	3	1	3	13
ONNX	5	1	3	1	1	11
Transformers	5	1	4	1	3	**14**
LLM Inference & Orchestration						*11.8*
TensorRT	5	1	5	1	3	**15**
LlamaIndex	5	1	3	1	3	13
SGLang	5	1	3	1	1	11
vLLM	3	1	4	1	1	10
LangChain	5	1	1	1	3	11
Text Generation Inference	5	1	3	1	1	11
AI Agent Frameworks						*11.4*
CrewAI	5	1	3	1	1	11
MetaGPT	5	1	5	1	1	13
LangGraph	1	1	3	1	3	9
SmolAgents	5	1	1	1	1	9
Stagehand	5	3	1	1	1	11
Composio	1	1	5	1	3	11
Browser Use	5	1	4	1	3	**14**
Pydantic AI	5	1	3	1	1	11
Agent Development Kit	5	3	4	1	1	**14**

A.3. Detailed Risk Rating Criteria

The risk scoring within each domain is anchored by the following criteria derived from the agent system prompts:

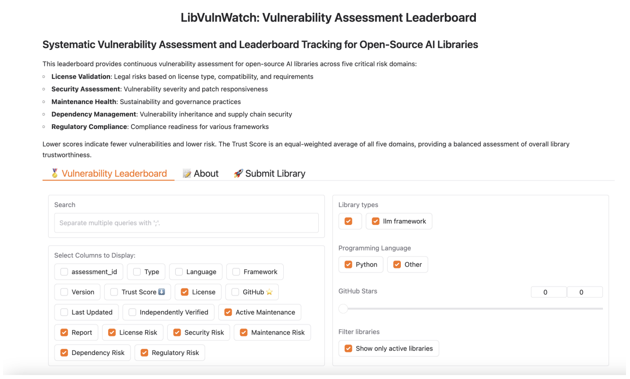
- **Low Risk (Score 5)** is associated with:
  - *License:* Permissive (e.g., MIT, Apache 2.0, BSD) with clear terms and compatibility.
  - *Security:* No CVEs in the past 24 months, a robust security policy, and rapid fixes (e.g., <7 days).
  - *Maintenance:* More than 10 active contributors, monthly or more frequent releases, and prompt issue response (e.g., <24 hours).
  - *Dependencies:* SBOM available, fewer than 20 direct dependencies, and evidence of automatic updates.
  - *Regulatory:* Clear compliance documentation and a complete audit trail.
- **Medium Risk (Score 3)** is associated with:
  - *License:* Moderate restrictions or unclear patent provisions.
  - *Security:* 1-3 minor CVEs in the past 12 months, a basic security policy, and moderate response times (e.g., 7-30 days).
  - *Maintenance:* 3-10 active contributors, quarterly releases, and issue response times of 1-7 days.
  - *Dependencies:* Partial SBOM, 20-50 direct dependencies, and some transitive visibility.

  - *Regulatory:* Incomplete compliance documentation or partial audit readiness.

- **High Risk (Score 1)** is associated with:
  - *License:* Restrictive terms (e.g., GPL/AGPL), incompatible terms, or other legal concerns.
  - *Security:* Critical or multiple CVEs, a missing security policy, or slow response times (e.g., >30 days).
  - *Maintenance:* Fewer than 3 active contributors, infrequent releases (e.g., >6 months), or poor issue response.
  - *Dependencies:* No SBOM, more than 50 direct dependencies, or known vulnerable transitive dependencies.
  - *Regulatory:* Missing compliance documentation or failure to meet essential regulations.

As noted in the Methodology, the absence of critical information on any key risk factor is also typically rated as High Risk (Score 1), and intermediate scores (2 or 4) can be assigned based on the agent's nuanced assessment between these thresholds.

A.4. Leaderboard Pictures



LibVulnWatch: Vulnerability Assessment Leaderboard

Systematic Vulnerability Assessment and Leaderboard Tracking for Open-Source AI Libraries

This leaderboard provides continuous vulnerability assessment for open-source AI libraries across five critical risk domains:

- **License Validation**: Legal risks based on license type, compatibility, and requirements
- **Security Assessment**: Vulnerability severity and patch responsiveness
- **Maintenance Health**: Sustainability and governance practices
- **Dependency Management**: Vulnerability inheritance and supply chain security
- **Regulatory Compliance**: Compliance readiness for various frameworks

Lower scores indicate fewer vulnerabilities and lower risk. The Trust Score is an equal-weighted average of all five domains, providing a balanced assessment of overall library trustworthiness.

Search: Separate multiple queries with ';'.

Select Columns to Display: ☐ assessment_id ☐ Type ☐ Language ☐ Framework ☐ Version ☐ Trust Score [✓ License] ☐ GitHub ☐ Last Updated ☐ Independently Verified [✓ Active Maintenance] [✓ Report] [✓ License Risk] [✓ Security Risk] [✓ Maintenance Risk] [✓ Dependency Risk] [✓ Regulatory Risk]

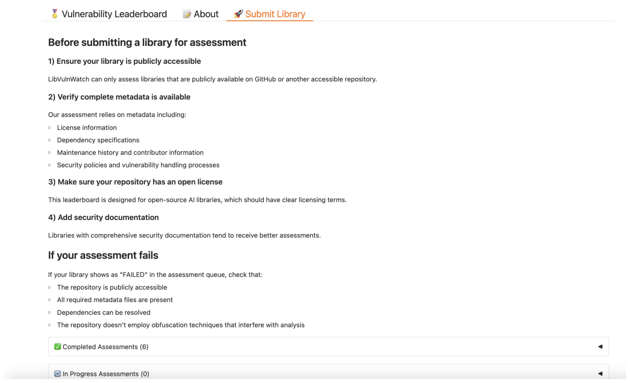
Library types: [✓] [✓ llm framework]

Programming Language: [✓ Python] [✓ Other]

GitHub Stars: 0 0

Filter libraries: [✓ Show only active libraries]

Figure 2. One of LibVulnWatch's Leader board



Before submitting a library for assessment

1) Ensure your library is publicly accessible

LibVulnWatch can only assess libraries that are publicly available on GitHub or another accessible repository.

2) Verify complete metadata is available

Our assessment relies on metadata including:

- License information
- Dependency specifications
- Maintenance history and contributor information
- Security policies and vulnerability handling processes

3) Make sure your repository has an open license

This leaderboard is designed for open-source AI libraries, which should have clear licensing terms.

4) Add security documentation

Libraries with comprehensive security documentation tend to receive better assessments.

If your assessment fails

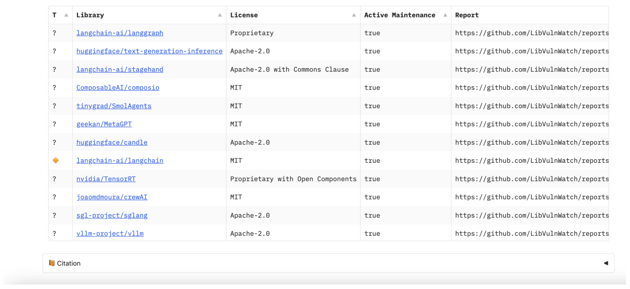
If your library shows as "FAILED" in the assessment queue, check that:

- The repository is publicly accessible
- All required metadata files are present
- Dependencies can be resolved
- The repository doesn't employ obfuscation techniques that interfere with analysis

✅ Completed Assessments (6)

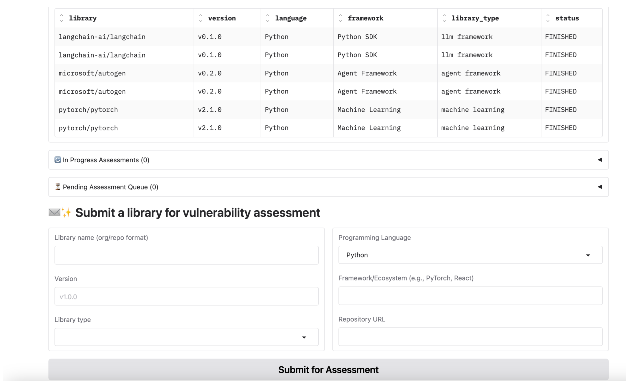
In Progress Assessments (0)

Figure 3. One of LibVulnWatch's Leader board



T	Library	License	Active Maintenance	Report
?	langchain-ai/langgraph	Proprietary	true	https://github.com/LibVulnWatch/reports
?	huggingface/text-generation-inference	Apache-2.0	true	https://github.com/LibVulnWatch/reports
?	langchain-ai/stagehand	Apache-2.0 with Commons Clause	true	https://github.com/LibVulnWatch/reports
?	ComposableAI/composio	MIT	true	https://github.com/LibVulnWatch/reports
?	tinygrad/SmolAgents	MIT	true	https://github.com/LibVulnWatch/reports
?	geekan/MetaGPT	MIT	true	https://github.com/LibVulnWatch/reports
?	huggingface/candle	Apache-2.0	true	https://github.com/LibVulnWatch/reports
🔶	langchain-ai/langchain	MIT	true	https://github.com/LibVulnWatch/reports
?	nvidia/TensorRT	Proprietary with Open Components	true	https://github.com/LibVulnWatch/reports
?	joaomdmoura/crewAI	MIT	true	https://github.com/LibVulnWatch/reports
?	sgl-project/sglang	Apache-2.0	true	https://github.com/LibVulnWatch/reports
?	vllm-project/vllm	Apache-2.0	true	https://github.com/LibVulnWatch/reports

Figure 4. One of LibVulnWatch's Leader board



library	version	language	framework	library_type	status
langchain-ai/langchain	v0.1.0	Python	Python SDK	llm framework	FINISHED
langchain-ai/langchain	v0.1.0	Python	Python SDK	llm framework	FINISHED
microsoft/autogen	v0.2.0	Python	Agent Framework	agent framework	FINISHED
microsoft/autogen	v0.2.0	Python	Agent Framework	agent framework	FINISHED
pytorch/pytorch	v2.1.0	Python	Machine Learning	machine learning	FINISHED
pytorch/pytorch	v2.1.0	Python	Machine Learning	machine learning	FINISHED

In Progress Assessments (0)

Pending Assessment Queue (0)

Submit a library for vulnerability assessment

Library name (org/repo format):

Version: v1.0.0

Library type:

Programming Language: Python

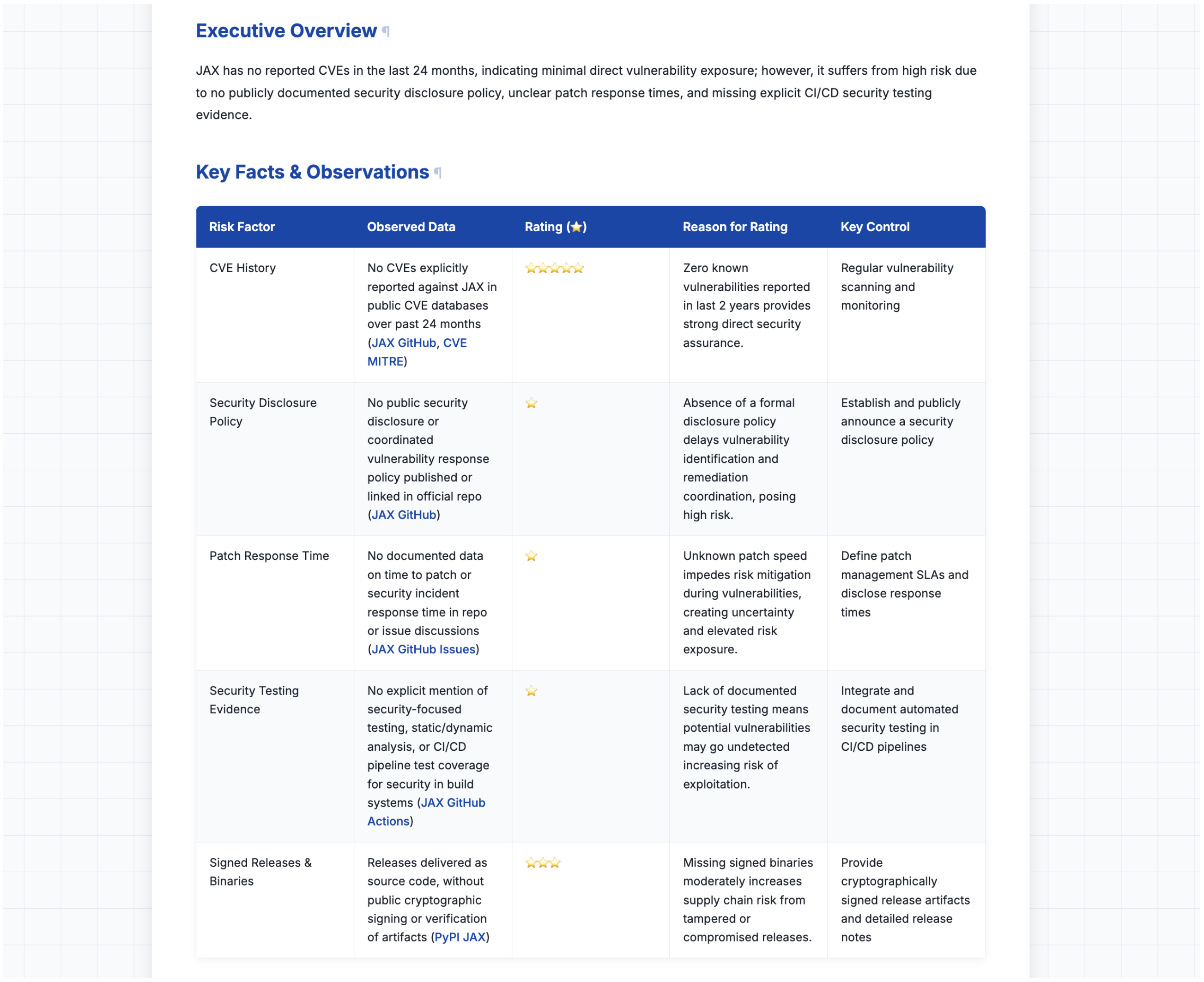
Framework/Ecosystem (e.g., PyTorch, React):

Repository URL:

Figure 5. One of LibVulnWatch's Leader board

A.5. License Case Study: LangGraph

Our LibVulnWatch system identified that while LangGraph specifies an MIT license in its repository, a more comprehensive analysis revealed connections to LangChain's Terms of Use that affect its licensing status. This demonstrates the system's capability to analyze complex licensing relationships beyond source code, providing a more holistic assessment than the OpenSSF Scorecard, which primarily considers repository-level licensing information.



Executive Overview ¶

JAX has no reported CVEs in the last 24 months, indicating minimal direct vulnerability exposure; however, it suffers from high risk due to no publicly documented security disclosure policy, unclear patch response times, and missing explicit CI/CD security testing evidence.

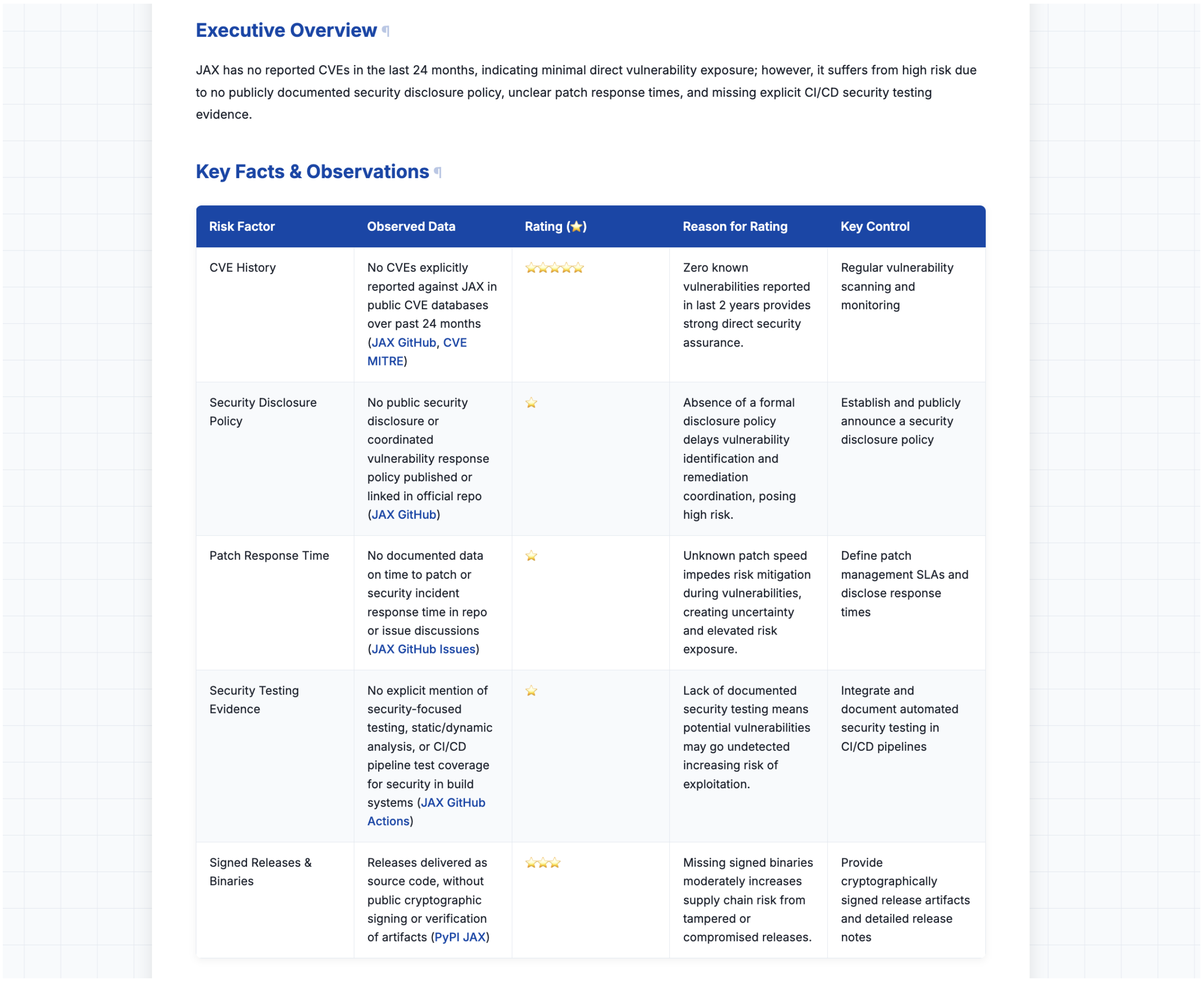
Key Facts & Observations ¶

Risk Factor	Observed Data	Rating (⭐)	Reason for Rating	Key Control
CVE History	No CVEs explicitly reported against JAX in public CVE databases over past 24 months (JAX GitHub, CVE MITRE)	⭐⭐⭐⭐⭐	Zero known vulnerabilities reported in last 2 years provides strong direct security assurance.	Regular vulnerability scanning and monitoring
Security Disclosure Policy	No public security disclosure or coordinated vulnerability response policy published or linked in official repo (JAX GitHub)	⭐	Absence of a formal disclosure policy delays vulnerability identification and remediation coordination, posing high risk.	Establish and publicly announce a security disclosure policy
Patch Response Time	No documented data on time to patch or security incident response time in repo or issue discussions (JAX GitHub Issues)	⭐	Unknown patch speed impedes risk mitigation during vulnerabilities, creating uncertainty and elevated risk exposure.	Define patch management SLAs and disclose response times
Security Testing Evidence	No explicit mention of security-focused testing, static/dynamic analysis, or CI/CD pipeline test coverage for security in build systems (JAX GitHub Actions)	⭐	Lack of documented security testing means potential vulnerabilities may go undetected increasing risk of exploitation.	Integrate and document automated security testing in CI/CD pipelines
Signed Releases & Binaries	Releases delivered as source code, without public cryptographic signing or verification of artifacts (PyPI JAX)	⭐⭐⭐	Missing signed binaries moderately increases supply chain risk from tampered or compromised releases.	Provide cryptographically signed release artifacts and detailed release notes

Figure 6. LangGraph License Analysis from the Generated Report

A.6. Security Case Study: JAX

The system correctly identified that JAX had no CVEs for 2 years and pointed out, after examining GitHub Action links, that it lacks an explicit security CI workflow.



Executive Overview ¶

JAX has no reported CVEs in the last 24 months, indicating minimal direct vulnerability exposure; however, it suffers from high risk due to no publicly documented security disclosure policy, unclear patch response times, and missing explicit CI/CD security testing evidence.

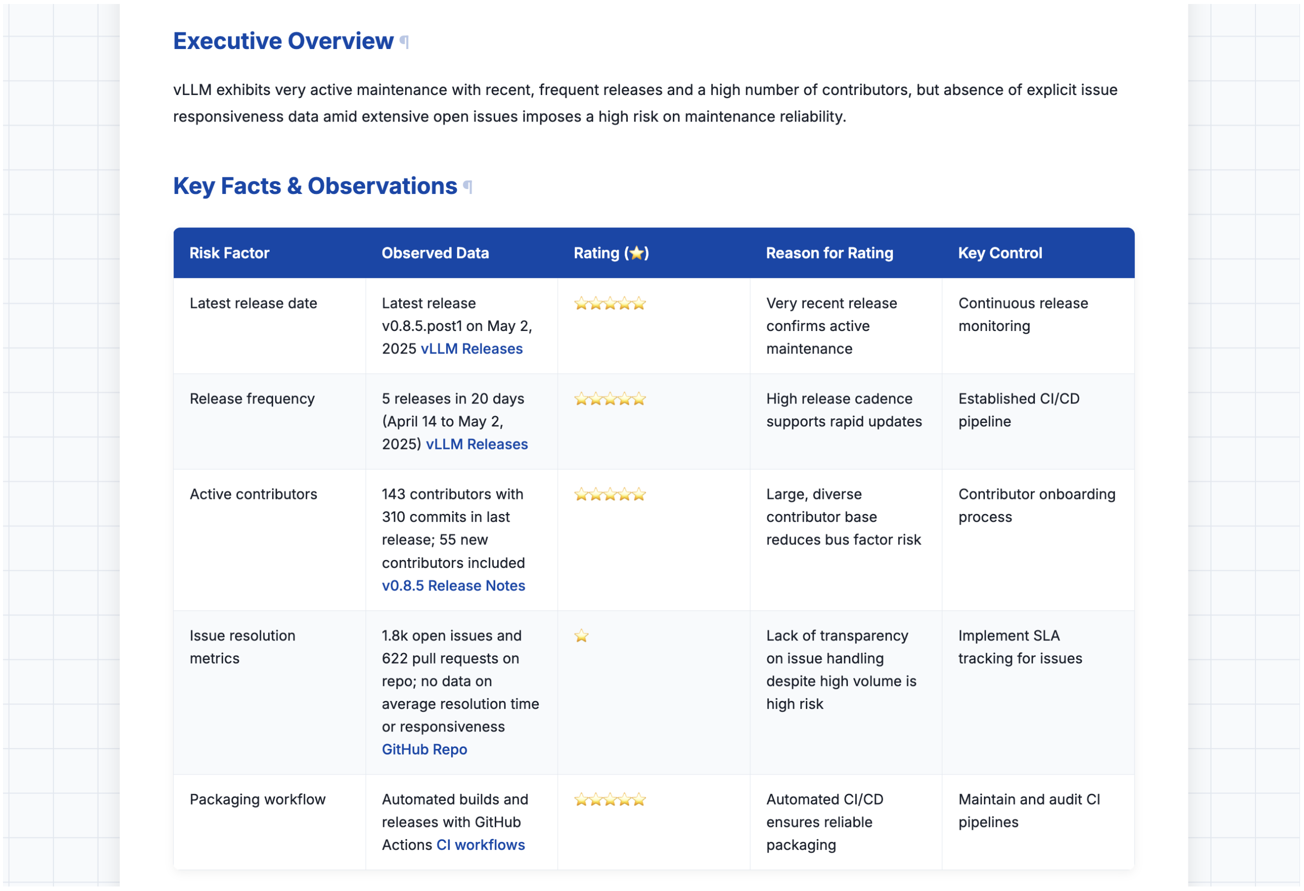
Key Facts & Observations ¶

Risk Factor	Observed Data	Rating (⭐)	Reason for Rating	Key Control
CVE History	No CVEs explicitly reported against JAX in public CVE databases over past 24 months (JAX GitHub, CVE MITRE)	⭐⭐⭐⭐⭐	Zero known vulnerabilities reported in last 2 years provides strong direct security assurance.	Regular vulnerability scanning and monitoring
Security Disclosure Policy	No public security disclosure or coordinated vulnerability response policy published or linked in official repo (JAX GitHub)	⭐	Absence of a formal disclosure policy delays vulnerability identification and remediation coordination, posing high risk.	Establish and publicly announce a security disclosure policy
Patch Response Time	No documented data on time to patch or security incident response time in repo or issue discussions (JAX GitHub Issues)	⭐	Unknown patch speed impedes risk mitigation during vulnerabilities, creating uncertainty and elevated risk exposure.	Define patch management SLAs and disclose response times
Security Testing Evidence	No explicit mention of security-focused testing, static/dynamic analysis, or CI/CD pipeline test coverage for security in build systems (JAX GitHub Actions)	⭐	Lack of documented security testing means potential vulnerabilities may go undetected increasing risk of exploitation.	Integrate and document automated security testing in CI/CD pipelines
Signed Releases & Binaries	Releases delivered as source code, without public cryptographic signing or verification of artifacts (PyPI JAX)	⭐⭐⭐	Missing signed binaries moderately increases supply chain risk from tampered or compromised releases.	Provide cryptographically signed release artifacts and detailed release notes

Figure 7. JAX Security Analysis from the Generated Report

A.7. Maintenance Case Study: vLLM

The system analyzed vLLM's recent GitHub contributions and issues to identify maintenance trends.



Executive Overview ¶

vLLM exhibits very active maintenance with recent, frequent releases and a high number of contributors, but absence of explicit issue responsiveness data amid extensive open issues imposes a high risk on maintenance reliability.

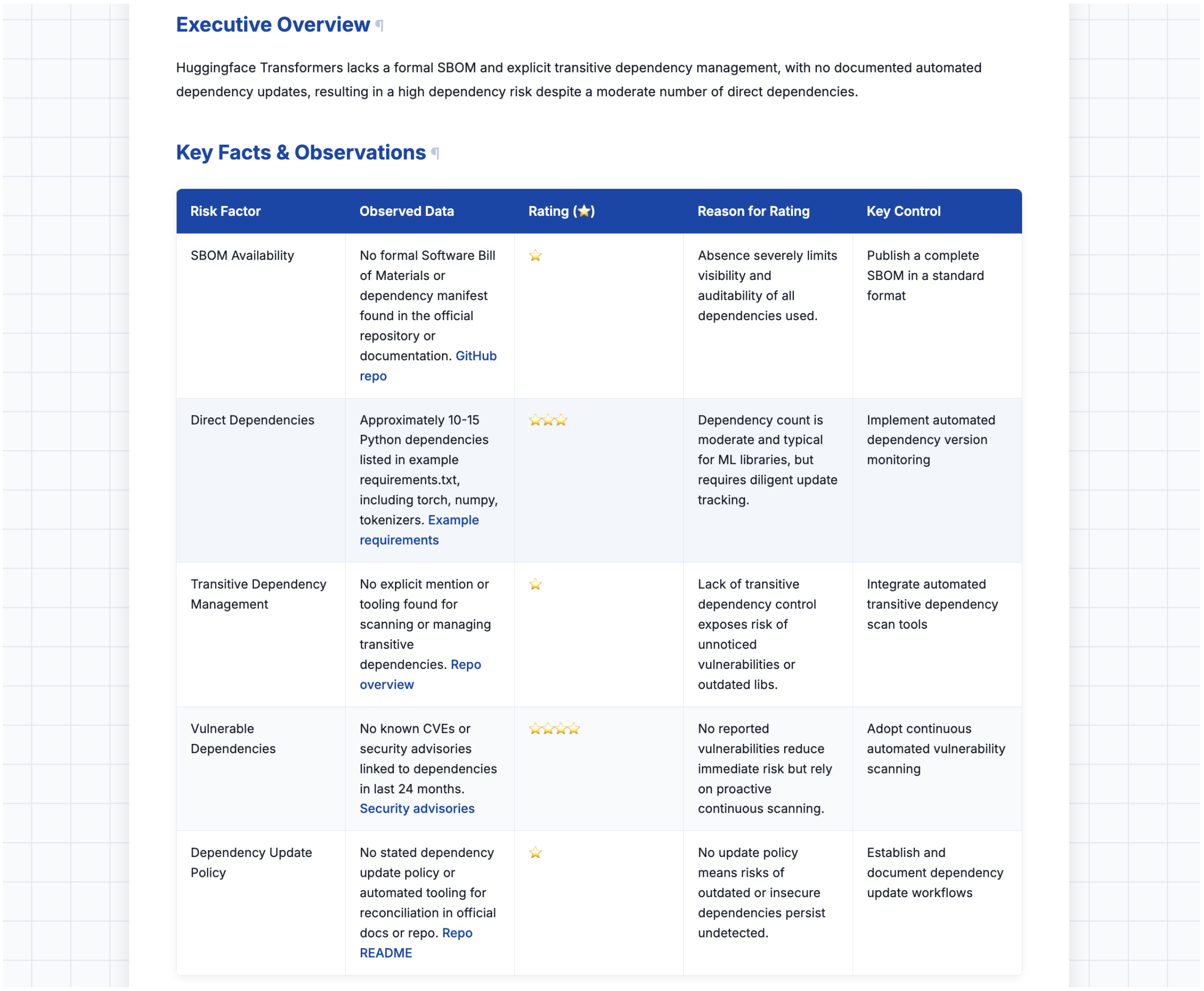
Key Facts & Observations ¶

Risk Factor	Observed Data	Rating (⭐)	Reason for Rating	Key Control
Latest release date	Latest release v0.8.5.post1 on May 2, 2025 vLLM Releases	⭐⭐⭐⭐⭐	Very recent release confirms active maintenance	Continuous release monitoring
Release frequency	5 releases in 20 days (April 14 to May 2, 2025) vLLM Releases	⭐⭐⭐⭐⭐	High release cadence supports rapid updates	Established CI/CD pipeline
Active contributors	143 contributors with 310 commits in last release; 55 new contributors included v0.8.5 Release Notes	⭐⭐⭐⭐⭐	Large, diverse contributor base reduces bus factor risk	Contributor onboarding process
Issue resolution metrics	1.8k open issues and 622 pull requests on repo; no data on average resolution time or responsiveness GitHub Repo	⭐	Lack of transparency on issue handling despite high volume is high risk	Implement SLA tracking for issues
Packaging workflow	Automated builds and releases with GitHub Actions CI workflows	⭐⭐⭐⭐⭐	Automated CI/CD ensures reliable packaging	Maintain and audit CI pipelines

Figure 8. vLLM Maintenance Analysis from the Generated Report

A.8. Dependencies Case Study: Huggingface Transformers

The system examined Huggingface's dependencies, SBOM, and policies for a comprehensive assessment.



Executive Overview ¶

Huggingface Transformers lacks a formal SBOM and explicit transitive dependency management, with no documented automated dependency updates, resulting in a high dependency risk despite a moderate number of direct dependencies.

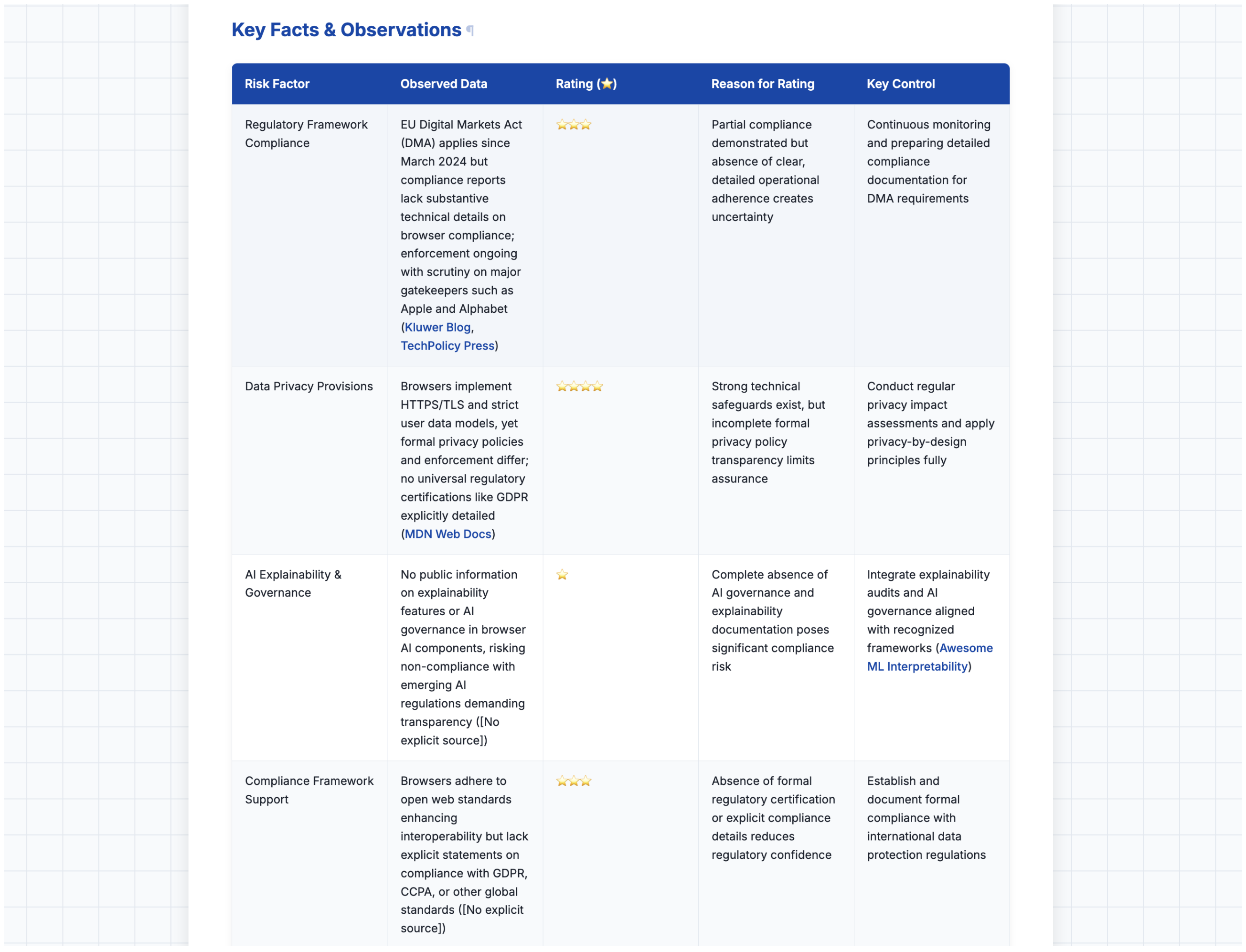
Key Facts & Observations ¶

Risk Factor	Observed Data	Rating (⭐)	Reason for Rating	Key Control
SBOM Availability	No formal Software Bill of Materials or dependency manifest found in the official repository or documentation. GitHub repo	⭐	Absence severely limits visibility and auditability of all dependencies used.	Publish a complete SBOM in a standard format
Direct Dependencies	Approximately 10-15 Python dependencies listed in example requirements.txt, including torch, numpy, tokenizers. Example requirements	⭐⭐⭐	Dependency count is moderate and typical for ML libraries, but requires diligent update tracking.	Implement automated dependency version monitoring
Transitive Dependency Management	No explicit mention or tooling found for scanning or managing transitive dependencies. Repo overview	⭐	Lack of transitive dependency control exposes risk of unnoticed vulnerabilities or outdated libs.	Integrate automated transitive dependency scan tools
Vulnerable Dependencies	No known CVEs or security advisories linked to dependencies in last 24 months. Security advisories	⭐⭐⭐⭐	No reported vulnerabilities reduce immediate risk but rely on proactive continuous scanning.	Adopt continuous automated vulnerability scanning
Dependency Update Policy	No stated dependency update policy or automated tooling for reconciliation in official docs or repo. Repo README	⭐	No update policy means risks of outdated or insecure dependencies persist undetected.	Establish and document dependency update workflows

Figure 9. Huggingface Transformers Dependencies Analysis from the Generated Report

A.9. Regulatory Case Study: Browser Use

The analysis linked browser agent library characteristics to the EU AI Act with relevant supporting references.



Key Facts & Observations ¶

Risk Factor	Observed Data	Rating (⭐)	Reason for Rating	Key Control
Regulatory Framework Compliance	EU Digital Markets Act (DMA) applies since March 2024 but compliance reports lack substantive technical details on browser compliance; enforcement ongoing with scrutiny on major gatekeepers such as Apple and Alphabet (Kluwer Blog, TechPolicy Press)	⭐⭐⭐	Partial compliance demonstrated but absence of clear, detailed operational adherence creates uncertainty	Continuous monitoring and preparing detailed compliance documentation for DMA requirements
Data Privacy Provisions	Browsers implement HTTPS/TLS and strict user data models, yet formal privacy policies and enforcement differ; no universal regulatory certifications like GDPR explicitly detailed (MDN Web Docs)	⭐⭐⭐⭐	Strong technical safeguards exist, but incomplete formal privacy policy transparency limits assurance	Conduct regular privacy impact assessments and apply privacy-by-design principles fully
AI Explainability & Governance	No public information on explainability features or AI governance in browser AI components, risking non-compliance with emerging AI regulations demanding transparency ([No explicit source])	⭐	Complete absence of AI governance and explainability documentation poses significant compliance risk	Integrate explainability audits and AI governance aligned with recognized frameworks (Awesome ML Interpretability)
Compliance Framework Support	Browsers adhere to open web standards enhancing interoperability but lack explicit statements on compliance with GDPR, CCPA, or other global standards ([No explicit source])	⭐⭐⭐	Absence of formal regulatory certification or explicit compliance details reduces regulatory confidence	Establish and document formal compliance with international data protection regulations

Figure 10. Browser Use Regulatory Analysis from the Generated Report