NETRISE

Anthropic Software Supply Chain Risk Blast Radius

Prepared by: Michael Scott, Co-Founder & CTO, March 6, 2026

For additional information: thomas.pace@netrise.io

Executive Summary

On March 4, 2026, the Pentagon designated Anthropic and its products a supply-chain risk, effective immediately. Anthropic disclosed the March 4 letter on March 5, and broader public reporting followed on March 6. Public reporting and Anthropic's own statements make clear that the dispute is not about a newly disclosed software vulnerability. It is about control, terms of use, and whether a frontier AI supplier can refuse certain uses of its models. That is precisely why this episode matters. The hard problem is not merely deciding whether a supplier is in or out. The hard problem is understanding, quickly and defensibly, where that supplier sits inside the software, container, and deployment estate that organizations already depend on.

That is the operational gap this report addresses. Using internal NetRise platform data, a current cross-ecosystem snapshot, plus container SBOM analysis, Helm rendering, and GitHub repository analysis, we mapped not only official Anthropic packages but also the direct and transitive packages that depend on them. The result is a practical blast-radius view across the software supply chain.

The findings are substantial. We identified 476 official Anthropic package names spanning 3,434 package versions across seven ecosystems. We then identified 127,441 direct or transitive dependent package versions. In downstream artifacts, we found six affected container repositories in the public cohorts we analyzed, representing 203,705,108 Docker Hub pulls at the time of analysis, six affected Helm charts in a targeted Kubernetes cohort, and 19 affected repositories across two selected GitHub cohorts totaling 130 repositories and 1,286,463 GitHub stars. We also cross-referenced the findings against internal platform inventory data and identified affected assets across enterprise appliance images, development platforms, self-hosted AI interfaces, MCP server binaries, and AI framework packages. The pattern is not evenly distributed. Generic infrastructure images were largely clean. The concentration appeared higher in AI application layers, orchestration stacks, MCP tooling, agent frameworks, and workflow products.

The policy implication is straightforward: when an organization faces a future designation, ban, urgent directive, or supplier offboarding event, it should not have to assemble this picture manually. It should be able to press a key and obtain a cross-layer blast-radius analysis grounded in package versions, image digests, Helm manifests, source repositories, and direct or transitive dependency lineage.

What Happened

The public timeline moved quickly.

On February 26, 2026, Anthropic CEO Dario Amodei stated that Claude was already extensively deployed across the Department of War and other national security agencies, including classified environments, and said the company had been threatened with offboarding and a supply-chain-risk designation if it did not remove two safeguards: restrictions on mass domestic surveillance and fully autonomous weapons use.

On February 27, 2026, Anthropic published a second statement after Secretary Pete Hegseth publicly said he was directing the Department to designate Anthropic a supply-chain risk. Anthropic argued that the move was unprecedented for a U.S. company and said it would challenge such a designation in court. The same day, Senator Edward Markey publicly called the threatened action reckless and unprecedented and urged Congress and the Department of War to reverse course.

On March 4, 2026, the Department designated Anthropic a supply-chain risk. On March 5, 2026, Anthropic said it had received the March 4 letter confirming the designation. On March 6, 2026, broader public reporting described the Pentagon's position as effective immediately.

This is an unusual fact pattern. Under 10 U.S.C. § 3252, "supply chain risk" is framed in terms of adversarial sabotage, maliciously introduced function, or other subversion of a covered system. Public reporting and the parties' own statements indicate that the present dispute instead centers on model-use restrictions and procurement authority. That distinction matters. It means buyers, operators, and software suppliers are now confronting a supply-chain-risk event that is operationally real regardless of how the underlying legal and policy dispute is ultimately resolved.

What We Found

This analysis was designed to answer a simple operational question:

If Anthropic becomes restricted, where is it likely to show up?

We did not limit the search to packages named anthropic or @anthropic-ai/*. We first built the official Anthropic package inventory, then expanded the scope to direct and transitive dependents. We then matched that package universe against container SBOMs, rendered Helm charts, and pinned GitHub repository artifacts.

The downstream artifact counts in this report should be read as findings within selected cohorts, not global totals for the public internet. The container results come from a mix of popularity-ranked and targeted public image cohorts. The Helm results come from targeted deployment-chart cohorts. The GitHub results come from one broad star-and-growth cohort and one popular MCP Registry cohort. Those are useful operational slices, but they are not an attempt to enumerate every public repository, image, or chart.

Key Metrics

Area	Result
Official Anthropic package names	476
Official Anthropic package versions	3,434
Direct or transitive dependent package names	9,819
Direct or transitive dependent package versions	127,441
Affected public container repositories in analyzed cohorts	6
Combined Docker Hub pulls for affected container repos in analyzed cohorts	203,705,108
Affected Helm charts in analyzed targeted cohort	6 of 6
Affected GitHub repositories across analyzed cohorts	19 of 130
Combined GitHub stars across affected repositories in analyzed cohorts	1,286,463
Affected repositories in analyzed popular MCP Registry cohort	6 of 30
Affected internal platform assets	15

Ecosystem-Level Exposure

The dependency picture is broad, but not uniform. NPM and PyPI dominate the dependent set. We found 79,586 affected NPM package versions and 47,373 affected PyPI package versions, with much smaller counts in Maven and Cargo. That is consistent with where current AI application development is happening: JavaScript and Python ecosystems, often with deep framework and plugin chains that obscure provenance.

Container Findings

The important container result is not simply that Anthropic-related components exist in public images. It is where they exist.

The top-50 generic Docker Hub `library/*` baseline produced zero affected images. That is useful because it shows this is not a broad contamination problem across common base infrastructure. The risk concentrates in the public image cohorts we analyzed for application-layer products that package AI capabilities directly or indirectly.

Across the public image cohorts analyzed here, we identified six affected container repositories:

- `n8nio/n8n` with 191,776,692 pulls
- `flowiseai/flowise` with 6,014,283 pulls
- `mintplexlabs/anythingllm` with 2,636,199 pulls
- `langflowai/langflow` with 2,255,859 pulls
- `langchain/langsmith-backend` with 598,870 pulls
- `litellm/litellm` with 423,205 pulls

Some of those images include official Anthropic packages directly. Others carry packages that depend on Anthropic components, such as `@langchain/anthropic`, `@browserbasehq/stagehand`, `langchain-anthropic`, and `pydantic-ai`. In other words, the blast radius is not limited to "products that intentionally use Anthropic." It includes products whose functional stack or optional integrations pull Anthropic packages in through a dependency chain.

Helm and Kubernetes Findings

> The Helm result is even more operationally significant than the raw container result. Containers tell you what exists. Helm charts tell you what is likely to be deployed.

In the targeted Artifact Hub cohort built from applications already implicated by the container analysis, all six charts rendered and all six were affected:

- `n8n` from `community-charts`
- `n8n` from `open-8gears`
- `flowise`
- `anything-llm`
- `langflow-ide`
- `langflow-runtime`

That matters because it bridges the gap between software composition and deployment reality. It shows these are not just image-level curiosities. They are being packaged for Kubernetes deployment in forms that operators can install today.

GitHub Findings

We analyzed two distinct GitHub slices. The first was a broader set of 100 repositories selected for raw star count and recent growth rather than an AI-only label. Thirteen were affected, representing 1,252,916 GitHub stars within that cohort. This is not a claim that only 13 GitHub repositories are affected overall. It is a statement about that selected high-visibility slice.

Notable affected repositories included:

- `openclaw/openclaw`
- `Significant-Gravitas/AutoGPT`
- `langflow-ai/langflow`
- `langchain-ai/langchain`
- `open-webui/open-webui`
- `firecrawl/firecrawl`

Eleven of the 13 carried official Anthropic packages directly. Two were dependent-only hits. That distinction matters for response. Direct package presence is usually easier to explain and remediate. Transitive dependency exposure is where organizations lose time, because the affected component is often several layers removed from the product team that owns the code.

We then added a second GitHub slice focused on page 1 of GitHub's MCP Registry, which is effectively a curated popularity view of MCP servers. Six of the 30 most visible MCP repositories in that cohort were affected:

- `oraios/serena`

- `wonderwhy-er/DesktopCommanderMCP`
- `makenotion/notion-mcp-server`
- `stripe/ai`
- `apify/apify-mcp-server`
- `vercel/next-devtools-mcp`

That second result matters because it pushes the finding out of general AI application code and into the model tooling ecosystem itself. Most of those MCP hits carry official Anthropic packages directly rather than only indirect framework dependencies. In practice, that means future supplier restrictions are likely to surface not only in end-user AI applications, but in the tools engineers use to connect models to browsers, terminals, documentation, SaaS platforms, and developer workflows.

Internal Platform Findings

The public artifact results were reinforced by internal NetRise Platform inventory data. Without naming customers or individual tenants, the internal data identified 15 affected assets across six separate inventory groups. Twelve of those assets contained official Anthropic packages directly. Three were indirect-only findings.

The important point is not any single product name. It is the classes of assets where the exposure showed up:

- large enterprise appliance and virtual-appliance images
- platform SBOMs for developer or AI-enabled services
- self-hosted AI web interface bundles
- MCP server binaries and containerized tool connectors
- AI framework packages and wheel distributions

That internal view also sharpened two operational conclusions. First, the exposure is not confined to obvious AI demos or developer sandboxes. It can appear in large bundled enterprise images with thousands of components and substantial existing vulnerability backlogs. Second, MCP-related assets were a recurring class. Internal inventory surfaced multiple MCP server binaries carrying official Anthropic packages or close dependents, which is consistent with the separate GitHub MCP registry findings.

The internal data also confirmed that version discipline matters. Ecosystem and version differences prevented false positives. For example, some packages with familiar names were not confirmed dependents in the relevant ecosystem, and some versions predated the Anthropic dependency entirely. That is exactly why blast-radius work has to be version-specific rather than name-only.

Why This Is Hard in Practice

This incident is a case study in why software supply-chain visibility remains incomplete even when agencies collect more artifacts than they did a few years ago.

First, the problem is multi-layered. Package registries, source repositories, containers, and Kubernetes deployment artifacts each describe only part of reality. A package graph can tell you what may be included. An SBOM can tell you what was packaged into an image. A Helm chart can tell you what is likely to be deployed. A repository can tell you what developers intended to build. Operationally useful visibility comes from correlating all four.

Second, transitive dependencies are where the time goes. A direct dependency on `anthropic` is easy to spot. A dependency path such as `pydantic-ai -> pydantic-ai-slim -> anthropic or @langchain/anthropic -> @anthropic-ai/sdk` is much easier to miss without graph analysis.

Third, identifiers drift. Packages move by version. Containers move by digest. Tags like `latest` are mutable. Repositories move by commit. Helm defaults do not always reflect production overrides. If the analysis is not pinned to exact versions, digests, and commit SHAs, it becomes hard to defend in front of a contracting officer, a CISO, or an inspector general.

Fourth, SBOM availability is inconsistent. Some GitHub repositories expose useful dependency-graph SBOMs. Some do not. Some projects declare dependencies in lockfiles. Others only list ranges in manifests. Some deployment artifacts expose image references cleanly. Others require rendering or environment-specific values. That is why "just ask for an SBOM" is necessary but not sufficient.

Fifth, time matters. The set of affected artifacts changes as packages are updated, containers are rebuilt, and repositories cut new releases. A static spreadsheet assembled after the fact is not a response capability.

Likely Prevalence By Artifact Type

Based on the evidence gathered here, organizations should expect different prevalence patterns across different software supply-chain artifacts.

Source repositories are likely to show the highest visible prevalence. Developers declare SDKs, MCP packages, and framework integrations in manifests and lockfiles, and those declarations are relatively easy to recover. That is why the analyzed GitHub cohorts produced both the broad high-star findings and the separate MCP registry findings.

Containers show lower apparent prevalence than source repositories, but the distinction is important. Generic infrastructure images were largely clean. The affected images clustered in AI-enabled applications, orchestration products, developer tooling, and agent-facing services. In other words, once a product meaningfully exposes model functionality, Anthropic-related packages or dependents become much more likely.

Helm charts are likely to have lower standalone prevalence, but high conditional prevalence. If an affected application image has an actively maintained deployment chart, the chart is often affected as well because it is primarily a transport for the same image. The chart analysis here reflected exactly that pattern.

Packaged enterprise assets are where the operational surprise tends to be. Large appliances, virtual appliances, platform bundles, and binary distributions can carry AI-related components deep inside the image even when the product is not marketed first and foremost as an AI tool. That is what makes internal inventory and artifact scanning necessary. Public package and repo analysis will not fully capture that layer.

The practical expectation is straightforward. Future supplier-related events will probably not look evenly distributed across the stack. They will likely cluster in four places: developer-facing model tooling, MCP and agent connectors, AI-enabled application platforms, and bundled enterprise products that have quietly incorporated those capabilities.

What Organizations Should Take From This

The lesson is not that Anthropic is uniquely risky. The lesson is that a modern software supplier can become a mission problem overnight, and most organizations still cannot answer the first-order questions quickly enough.

When a future supplier issue emerges, organizations should be able to answer the following in minutes, not weeks:

1. What official software components from the named supplier are in scope?
2. What direct and transitive dependents also enter scope?
3. Which source repositories, build artifacts, containers, and deployment charts contain them?
4. Which of those artifacts are highest impact by operational footprint, pull volume, deployment prevalence, or mission criticality?
5. What is the exact lineage that introduced the component, and what is the most direct remediation path?

That is what blast-radius analysis should mean in practice.

This is also where existing federal guidance is already pointing. EO 14028 and OMB Memorandum M-22-18 pushed agencies toward secure software development attestations and, where needed, artifacts such as SBOMs. NIST's SSDF and its software supply-chain guidance gave agencies a framework for evaluating software producers. CISA has continued to refine SBOM guidance and, with NSA and international partners, has emphasized that SBOMs are valuable because they support automation and operational decision-making. The DoD's 2025 Cyber DT&E Guidebook goes further and plainly advises practitioners to seek visibility into all software, libraries, and firmware through software and firmware BOMs.

But the present event shows the limit of artifact collection by itself. Organizations do not merely need more SBOMs. They need a way to turn supplier names, packages, repositories, containers, charts, and dependency graphs into an evidence-backed operational picture on demand.

What a One-Keystroke Response Capability Looks Like

If an organization wants to be ready for the next episode, the requirement should be simple: a response team should be able to name a supplier or component and immediately produce a cross-layer blast-radius report.

In practical terms, that is the approach NetRise is taking. NetRise Turbine establishes a binary-verified software asset inventory across containers, applications, operating systems, firmware, and other compiled artifacts, which answers the first question: what is actually present in the software that an organization buys, ships, and runs. NetRise Provenance adds the second layer: who is behind those components, which maintainers and organizations are involved, what trust and hygiene signals surround the project, and how risk propagates through dependency and reverse-dependency graphs.

That combination matters because supplier events are not always classic vulnerability events. Sometimes the trigger is a sanctioned maintainer, a malicious contributor, a policy restriction, a country-of-origin concern, or a sudden supplier offboarding decision. In each of those cases, the operational question is the same: where does this code, project, maintainer, or organization appear across our portfolio, and where does it ultimately run. A system that combines verified asset inventory with provenance and dependency analysis can answer that question directly instead of forcing teams to reconstruct it by hand from disconnected tools.

Starting from an SBOM, filesystem, container image, or other compiled artifact, that workflow can move directly to affected packages, maintainers, dependency paths, asset classes, and policy impact. That is the practical difference between collecting supply-chain data and operating on it.

That capability should:

- enumerate official components across ecosystems
- expand to direct and transitive dependents
- match those components against source repositories, release artifacts, container images, deployment manifests, and compiled enterprise assets
- attach maintainer, organization, advisory, and project-health context to the affected components where relevant
- preserve exact evidence, including package versions, image digests, chart versions, and commit SHAs
- rank findings by operational relevance rather than raw count alone
- support policy decisions for both builders and buyers, including build gating, procurement review, and third-party risk response
- export a report suitable for acquisition, legal, security, and mission stakeholders

The benefit is straightforward. The same workflow can support an incident response team trying to scope immediate exposure, a procurement team trying to understand whether a vendor portfolio crosses a policy line, or a program office trying to determine which classes of systems have to be prioritized first. In other words, the response capability should not stop at software composition. It should connect software composition to deployment reality, and it should connect both to people, policy, and trust.

Bottom Line

The Anthropic designation is the immediate headline. The deeper story is the continuing need for fast, defensible visibility into the full software supply chain.

Our analysis shows why. The relevant blast radius does not stop at a package registry. It propagates into containers, Kubernetes charts, widely used source repositories, MCP tooling, and packaged enterprise assets. In the public cohorts analyzed here, the affected footprint alone includes 127,441 dependent package versions, six affected public container repositories with more than 203 million pulls, six affected deployment charts in the targeted Helm cohort, and 19 affected GitHub repositories across 130 selected repositories with more than 1.28 million stars. The internal platform data added a second layer of confirmation by showing the same pattern inside real packaged assets, especially enterprise appliance images and MCP-related binaries.

That is the policy and operational lesson. The next time a supplier becomes restricted, compromised, sanctioned, or otherwise unacceptable for continued use, the organizations that move fastest will be the ones that can identify the blast radius in one motion, preserve the

evidence, understand who and what are behind the affected components, and prioritize remediation immediately.

Anthropic is the case study. The gap is structural.

Sources

Public sources

- Anthropic, "Statement from Dario Amodei on our discussions with the Department of War" (February 26, 2026): https://www.anthropic.com/dow
- Anthropic, "Statement on the comments from Secretary of War Pete Hegseth" (February 27, 2026): https://www.anthropic.com/news/statement-comments-secretary-war
- Anthropic, "Where things stand with the Department of War" (March 5, 2026): https://www.anthropic.com/news/where-stand-department-war
- Defense News / Associated Press, "Pentagon says it is labeling Anthropic a supply chain risk 'effective immediately'" (March 6, 2026): https://www.defensenews.com/news/pentagon-congress/2026/03/06/pentagon-says-it-is-labeling-anthropic-a-supply-chain-risk-effective-immediately/
- Senator Edward J. Markey, "Markey Demands Immediate Congressional Action to Reverse DOD Designation of Anthropic a Supply Chain Risk" (February 27, 2026): https://www.markey.senate.gov/news/press-releases/markey-demands-immediate-congressional-action-to-reverse-dod-designation-of-anthropic-a-supply-chain-risk
- 10 U.S.C. § 3252, "Requirements for information relating to supply chain risk": https://www.govinfo.gov/link/uscode/10/3252
- OMB Memorandum M-22-18, "Enhancing the Security of the Software Supply Chain through Secure Software Development Practices" (September 14, 2022): https://www.whitehouse.gov/wp-content/uploads/2022/09/M-22-18.pdf
- NIST, "Secure Software Development Framework (SSDF) Version 1.1" (SP 800-218): https://www.nist.gov/publications/secure-software-development-framework-ssdf-version-11-recommendations-mitigating-risk
- NIST, "Software Supply Chain Security Guidance Under Executive Order (EO) 14028 Section 4e": https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity/software-cybersecurity-producers-and
- CISA, "Software Bill of Materials (SBOM)": https://www.cisa.gov/sbom
- CISA / NSA / international partners, "A Shared Vision of Software Bill of Materials (SBOM) for Cybersecurity" (September 3, 2025):

https://www.cisa.gov/resources-tools/resources/shared-vision-software-bill-materials-sbom-cybersecurity

- Department of War, "Cyber Developmental Test and Evaluation Guidebook" Version 3.0 (June 2025): https://www.cto.mil/wp-content/uploads/2025/08/Cyber-DTE-Guidebook-V3-June2025_Final_87e72e.p