

AI supply chain security: Best practices cheat sheet

Traditional [software supply chain security](#) focuses on protecting components like source code, dependencies, tools, and processes in order to prevent [supply chain attacks](#). This security model is built around the “software factory”: developer machines, source control systems, CI/CD pipelines, build infrastructure, artifact repositories, and deployment platforms. Together, these elements form a trusted pipeline where artifacts are assumed safe once they pass through approved tools, runners, and workflows. But AI has drastically altered this landscape.



While AI systems don't replace the traditional software factory, they extend it significantly. Traditional components in the factory still matter, yet they're no longer the only assets that define how software is built and operated. As a result, AI has introduced new supply chain attack vectors that can't be ignored.

This cheat sheet provides foundational information about supply chain security, takes a close look at how software supply chain security is evolving in the AI era, and provides hands-on advice for protecting your systems.

What is a supply chain attack?

A supply chain attack is a security incident where threat actors compromise trusted upstream elements to reach downstream targets. According to [CISA](#), common supply chain attack examples include compromised open-source dependencies, poisoned updates, and attacks on code-signing mechanisms using stolen keys. [Industry best practices](#) for securing traditional supply chains include SAST and SCA scanning, hardened CI/CD environments, and strict access controls.

But because AI workflows introduce new first-class artifacts and processes, they also open the door to new types of supply chain attacks that these traditional best practices can't catch. These include poisoned training datasets and embeddings, tampered or backdoored model weights, malicious prompt templates, compromised agent tools, unsafe model serialization formats, and attacks delivered through AI-powered IDEs or orchestration frameworks.

What is the AI supply chain attack surface?

With the advent of AI, training datasets, pre-trained and fine-tuned models, prompts, embeddings, experiment environments, and inference pipelines now sit alongside source code and all other established components.

And as development teams adopt LLM frameworks like LangChain, LlamaIndex, and CrewAI, along with AI-powered tools and [IDE plugins](#), the supply chain expands further. It now involves agent orchestrators, prompt templates, memory stores, and runtime decision layers.

In other words, AI supply chain security must explicitly account for:

- Training datasets and enrichment services
- Models, weights, configurations, and registries
- IDEs, notebooks, and experiment environments
- Metadata and inference workflows
- Public model hubs and third-party AI components

Common AI supply chain threats

Next, let's look at the biggest risks in your AI supply chain and how attackers take advantage of them.

1 Training and runtime environment threats

Vulnerabilities in GPU drivers, container runtimes (e.g., Docker or containerd), or ML frameworks pave the way for a whole new class of software supply chain attacks. If an attacker gains entry to privileged accounts or exploits misconfigured runtime permissions, they can access proprietary training logic and valuable hardware resources. For example, they might reuse expensive CPU/GPU capacity for unauthorized workloads.

2 Open-source AI library risks

AI introduces a brand-new set of dependencies, such as tokenizers, embedding utilities, and popular SDKs for AI applications and agent building. A vulnerability or malicious injection in these packages can propagate silently through your processes and training/inference pipelines. In effect, a minor library issue can turn into a large-scale trust failure across multiple environments.

3 Model artifact vulnerabilities

Unsafe serialization formats like PyTorch's `.pt` allow for [remote code execution \(RCE\)](#) the second you load the model via `torch.load()`. If an attacker tampers with weights, config files, or checkpoint data, they can degrade performance or trigger specific unsafe outputs without any indication that the model is behaving abnormally.

4 Threats to third-party integrations

AI integrations usually rely on API calls to external providers that facilitate direct LLM access or connect AI with systems like vector databases or specialized tokenizers. Each integration is a [new surface for potential attacks](#): If these third-party services are compromised or lack strong authentication and authorization, your prompts and corporate data could be exposed.

5 Tool chaining abuse in agent frameworks

In agentic frameworks, a simple prompt injection or poisoned input can trigger toxic flows. For example, an attacker could manipulate an agent to read data from an internal database, summarize it, and then post the result to a specific endpoint. Since that might look like exactly the intended behavior, traditional SAST/SCA tools won't catch the issue.

6 Exposed endpoints

Even if your backend is a fortress, endpoints that expose access to your models or AI processes can still leak secrets. Through [side-channel attacks or model inversion prompts](#), threat actors can infer sensitive training data, PII, or other business-critical information.

Why AI supply chain threats are difficult to detect

As we've seen, AI greatly increases the attack surface due to all the new components and processes entering the software factory. Still, the challenge isn't just safeguarding these new assets; blind spots are an equally pressing issue.

In general, AI systems introduce opaque risks because their inner workings aren't fully visible and are incredibly complex. Part of the problem comes down to the core functionality of AI: AI's outputs are typically probabilistic rather than deterministic, meaning it's normal to see variability in responses. While that's great for flexibility, response variability allows malicious behavior to fly under the radar. And when an attack is detected, it's often tricky to study, replicate, and trace back to AI due to its varying behavior.

These factors make supply chain threat assessment much more complex, increasing the risk that compromised models or AI-assisted processes will cause serious problems in downstream systems (including across multiple tenants or cloud environments).

Top 10 AI supply chain security best practices

Now that we've covered the risks, let's dive into how to address them. In this section, we'll explore the ten most essential AI supply chain security best practices.

1 Maintain an AI bill of materials (AI-BOM)

As a rule of thumb, if you can't trace an artifact's lineage, it doesn't belong in any AI process.

An [AI-BOM](#) tracks every AI component in your system—models, datasets, pipelines, endpoints, and dependencies (including agent definitions, model metadata, training datasets, and experiment notebooks)—to create a chain of trust for the full AI lifecycle. By serving as an [approved inventory](#), an AI-BOM prevents dynamic loading and execution of untrusted tools.

2 Secure the AI development pipeline

Securing your AI development pipeline is all about containment and visibility:

- Treat third-party model weights like untrusted binaries, loading them in isolated, low-privilege sandbox environments to mitigate RCE risks.
- Extend your [SCA scanners](#) beyond just AI libraries to cover GPU drivers and ML-specific container runtimes.
- Practice aggressive secret rotation and [enforce hardened RBAC](#) so that a single leaked credential won't give an attacker access to your AI workflows.

3 Manage third-party risk

Managing third-party AI risk requires continuous oversight. Start with rigorous vendor assessments before onboarding, then map transitive dependencies to detect upstream compromises hidden in nested SDKs.

To prevent issues that do occur from spiraling, rely on strict token limits, API call rate limits, and financial caps. These controls help mitigate wallet exhaustion attacks and abusive LLM usage.

4 Apply zero trust principles in AI-centric software

When it comes to AI, you can't ever assume trust. That's particularly true when integrating with third-party AI-related vendors who will handle and manage your data.

Your best bet is to implement a [zero trust architecture](#) that's built on least privilege, continuous authentication, and strict authorization across all AI components in the software factory. Zero trust treats every model, vendor, vector database, and agent as a potential threat vector (which is always a great mindset for preventing supply chain attacks).

5 Sign model artifacts cryptographically

Cryptographically signing model artifacts makes the AI supply chain verifiable and enforceable. By following the [OpenSSF Model Signing \(OMS\)](#) specification, you can sign model weights, configurations, tokenizers, preprocessing assets, and auxiliary model files as a single, immutable unit.

Forcing signature verification during deployment in the CI/CD pipeline lets you block any release where cryptographic hashes don't match the signed attestation. That way, only artifacts coming from trusted pipelines reach production.

6 Scan agent workflows like source code/configuration code

When working with AI agent workflows, some configuration files (e.g., `agents.yaml` and `tasks.yaml` in CrewAI) should be treated as executable logic because they specify how an agent behaves.

Consider these configuration files part of your codebase: Scan them for over-privileged tools, unsafe shell access, or mutable goals that could be hijacked. Also, apply [policy as code](#) to set guardrails around orchestration logic so that agents only operate within well-defined security boundaries.

7 Harden semantic search

Most AI applications rely on RAG to deliver more accurate, contextual results. To keep RAG pipelines resilient, strengthen the semantic information retrieval layer. Start by storing source documents in "write once, read many" (WORM) formats to prevent stealthy tampering.

Another top tip? Before feeding retrieved context to AI, pass it through a secondary LLM or a deterministic filter to catch injected instructions, irrelevant content, or subtle data manipulation.

8 Protect memory systems used in RAG

Beyond passive storage, vector stores and memory databases are also key infrastructure elements in AI pipelines. After all, poisoned or misleading entries in long-term memory can gradually introduce severe behavior drift.

To protect yourself, implement [anomaly detection](#) within your vector store to flag suspicious clusters. Also, practice memory trust scoring to rank entries by provenance and reliability, apply expiration policies to limit long-lived influence, and isolate memory namespaces across tenants or agents.

9 Ensure continuous visibility

Dedicated software supply chain tools help you maintain visibility into ever-evolving AI systems by:

- Tracking API queries to catch model inversion or data exfiltration attempts
- Mapping AI assets, identities, and exposure in a unified view to show how components interact and where risks may propagate
- Using automated inventories to uncover [shadow AI](#) risks and surface unauthorized elements entering the software factory
- Monitoring external integrations to prevent blind spots

10 Perform AI red teaming

AI red teaming isn't just about triggering hallucinations; it's also about testing how malicious actors could exploit trusted AI tooling within your software factory. Make sure to cover these critical test cases:

- Simulate attacks targeting AI-powered IDE plugins, agents, and copilots across developer machines, CI pipelines, and training workflows.
- Check whether [adversarial prompts](#) can coerce models into exposing credentials, environment variables, secrets, or sensitive data.
- Assess behavior when an agent is granted developer- or pipeline-level privileges, including access to internal tools, APIs, and artifact registries.
- Examine how your AI supply chain reacts to the presence of tampered external tools, poisoned registries, or compromised model artifacts.

Strengthen AI supply chain cybersecurity with Wiz

By now, it's clear: Securing the AI supply chain is an iterative journey that requires a unified approach across models, data, LLM-powered development tools, configurations, agent frameworks, and LLM vendors.

That's where Wiz comes in.

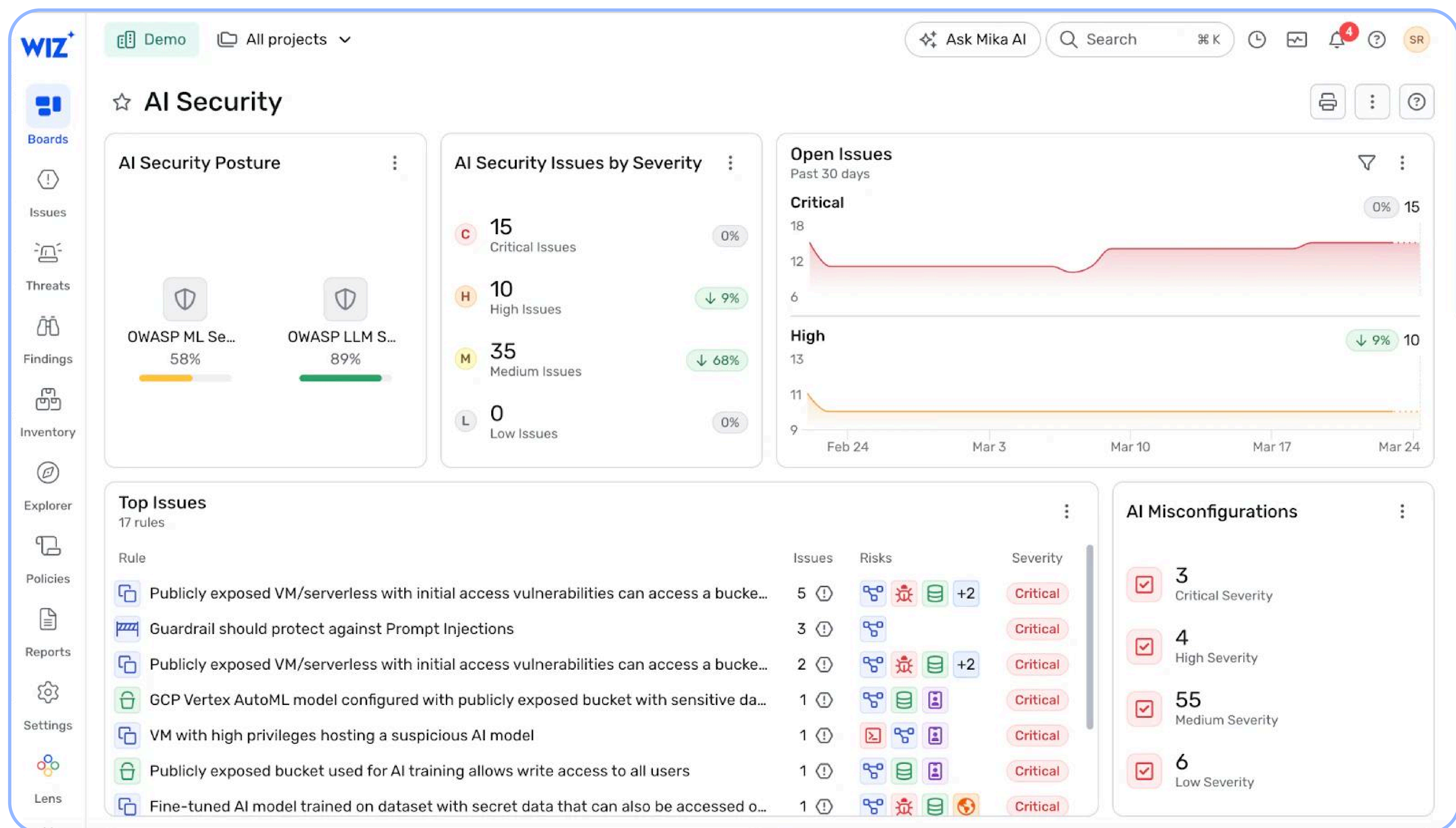


Figure 1: Wiz gives you an at-a-glance view of your AI security posture

Wiz is an all-in-one security platform that equips you with everything you need to secure cloud, on-prem, hybrid, and AI-specific assets. [Wiz Code's AI supply chain security](#) features include:

- **AI-BOM generation:** Wiz automatically inventories models, datasets, and pipelines to eliminate shadow AI and track asset provenance.
- **Attack path analysis:** The Wiz Security Graph reveals how misconfigurations or identities could be abused to exfiltrate sensitive training data.
- **Code-to-cloud tracing:** Wiz links exposed AI services back to the specific IaC or code repositories that introduced them for rapid remediation.
- **Cross-platform visibility:** With Wiz, you can view risks and misconfigurations across AWS, Azure, GCP, and managed services like Vertex AI and Hugging Face from a single pane of glass.

Ready to harden your AI factory?

[Request a demo](#) to see how Wiz can secure your models, data, and infrastructure today.

Get a Demo