

Safeguarding the Software Supply Chain: Best Practices for Nexus Repository

Executive Summary

Nexus Repository isn't just a development convenience—it's a linchpin of your organization's software supply chain security. Misconfigured or poorly governed repositories have become a prime vector for malicious attacks and compliance failures. This guide outlines key practices to ensure your Nexus Repository environment mitigates risk, supports governance, and scales securely. Each practice is backed by Sonatype's real-world data and experience securing tens of thousands of software pipelines.

Why Nexus Repository Matters More Than Ever

Your artifact repository is the gateway through which every third-party component enters your development ecosystem. As software supply chain threats evolve, repositories must move from passive storage to active enforcement.

Data Point: In 2024, Sonatype found that 1 in 8 open source downloads contained a known vulnerability. Worse, 10% of those were intentionally malicious.

Modern DevOps depends on a repository being available, secure, and governed. Treat it like you would production databases or CI/CD platforms—as mission-critical infrastructure.

1. Eliminate Shadow Downloads

Pulling dependencies directly from public registries was once the fastest way to get developers up and running, and most teams accepted the trade-off. But today, attackers actively target those registries with poisoned packages, and unmonitored sourcing has become one of the most common entry points for supply chain compromise. Leading organizations now mandate that every download routes through a governed repository to maintain visibility and control.

Risk:	Best Practice:	Benefit:
<p>Developers and CI pipelines fetching components directly from public sources (e.g., npm, PyPI) bypass security controls, increasing the chance of malware, license violations, and audit gaps. Sonatype's Repository Health Check data shows that 50% of unprotected repositories already contain one or more pieces of malware. This demonstrates how quickly uncontrolled downloads can compromise an environment.</p>	<ul style="list-style-type: none"> ✓ Ensure all downloads route through Nexus to eliminate shadow risk and streamline auditing. ✓ Protect build environments by restricting direct external registry access. ✓ Enable Firewall and Health Check in Nexus to vet new components. 	<p>Full visibility and control over third-party risk. Prevents zero-day malware and compliance violations before they enter the build.</p>

2. Secure Access with Zero-Trust Controls

Relying on the network perimeter to determine who could access components was once the accepted model. But with credential leaks and insider threats now routine, zero-trust controls have become the standard way organizations safeguard critical artifacts. Enforcing SSO and role-based access is no longer optional—it's the expected baseline for security and compliance.

Risk:

Weak access control and anonymous users expose critical artifacts and prevent auditability.

Best Practice:

- ✓ Secure access by limiting users to authenticated, least-privilege roles.
- ✓ Integrate with enterprise SSO or LDAP.
- ✓ Enforce least privilege: no broad admin rights; assign deploy permissions per team.
- ✓ Minimize credential risk through token-based authentication with rotation.

Benefit:

Reduces insider threat, ensures audit trails, and simplifies compliance.

Data Point:

Organizations with centralized governance over software components are significantly more likely to meet regulatory and audit readiness thresholds. Implementing RBAC and SSO helps streamline these controls.

3. Make Backups and Recovery a Compliance Priority

Simply keeping a backup used to be considered good enough. But with ransomware, cloud outages, and compliance audits now routine, tested recovery processes are the standard organizations are measured against. Regulators and boards alike expect not just backups, but documented, validated recovery procedures that prove business continuity under pressure.

Risk:

A corrupted or unavailable repository halts development, breaks builds, and could permanently lose proprietary components.

Best Practice:

- ✓ Automate nightly backups of blob stores and the database.
- ✓ Store backups offsite (cloud storage, secondary region).
- ✓ Test restore process annually; simulate failure scenarios.
- ✓ Document and periodically validate recovery steps.

Benefit:

Business continuity, proof of resilience during audits, and protection against ransomware.

Data Point:

47% of organizations lack a tested recovery plan for their artifact repository. Those that do recover 2.5x faster after an outage.

4. Design for High Availability

Running a single-node repository was once acceptable, with occasional downtime tolerated as part of the job. Today, uninterrupted access to components is a fundamental requirement—pipelines, developers, and compliance processes all depend on it. High-availability architectures are now the standard expectation for eliminating single points of failure and ensuring continuous delivery.

Risk:	Best Practice:	Benefit:
A single-node Nexus instance creates a critical single point of failure.	<ul style="list-style-type: none">✓ Support uninterrupted availability with high-availability Nexus architecture.✓ Use an external PostgreSQL database and shared blob store.✓ Separate nodes across infrastructure to avoid cascading failure.	Zero downtime during maintenance or node failure. Developers stay productive, pipelines stay green.

Data Point:

Resilient infrastructure, including HA configurations, is a key trait of mature software supply chain organizations. While downtime reduction varies, leaders prioritize redundancy and availability to meet SLA expectations.

5. Implement Policy Enforcement at the Repository Level

Manual reviews and occasional scans once passed as adequate safeguards against risky components. But with the pace of open source adoption and the volume of vulnerabilities discovered daily, manual approaches can no longer keep up. Automated policy enforcement at the repository level is now the standard for maintaining compliance and preventing security risks before they spread downstream.

Risk:	Best Practice:	Benefit:
Components with critical CVEs or forbidden licenses (e.g., GPL) can slip into production unnoticed.	<ul style="list-style-type: none">✓ Embed policy governance by integrating Nexus with Lifecycle/IQ for proactive enforcement.✓ Automate policy enforcement: block, quarantine, or warn on violations.✓ Periodically review policy violations and remediate.	Maintains legal compliance and prevents security risks from compounding downstream.

Data Point:

Teams using automated policy enforcement and governance tooling remediate vulnerable components 2.7x faster than those relying solely on manual reviews.

6. Treat SBOMs as First-Class Artifacts

Until recently, software bills of materials (SBOMs) were viewed as optional documentation, produced only when specifically requested. But with regulations like EO 14028 and rising customer demands for transparency, SBOMs have become a standard requirement for every production release. Treating them as first-class artifacts is now the norm for demonstrating software integrity and compliance.

Risk:

Regulatory frameworks (e.g., EO 14028) increasingly demand clear software provenance. Without SBOMs, proving compliance or responding to incidents is slow and error-prone.

Best Practice:

- ✓ Generate SBOMs using CycloneDX or other supported tooling.
- ✓ Store SBOMs in a dedicated Nexus repository.
- ✓ Automate SBOM generation for each release pipeline.

Benefit:

Builds regulatory trust, accelerates incident response, and increases transparency with customers and auditors.

Data Point:

SBOM coverage is now a top-five audit item for U.S. federal suppliers (NTIA 2024).

7. Use Cleanup and Retention Policies Strategically

Letting repositories grow without limits was once viewed as harmless, with storage costs seen as negligible. Today, that approach not only wastes budget but also leaves outdated and vulnerable components accessible long after their usefulness has expired. Automated cleanup and retention policies are now a standard safeguard against both inefficiency and risk.

Risk:

Unbounded repository growth increases storage costs and keeps vulnerable components accessible.

Best Practice:

- ✓ Reduce storage cost and security exposure using automated retention policies.
- ✓ Separate repositories by purpose: long-term releases vs. short-lived builds.
- ✓ Back up before cleanup; monitor deletion logs.

Benefit:

Lower storage spend, faster backups, reduced attack surface.

Data Point:

Customers implementing cleanup policies reduced blob store growth by 60% year-over-year (Sonatype support data).

8. Optimize Repository Performance and Organization

In many environments, repository design was historically left to evolve organically, with ad hoc structures tolerated as long as builds succeeded. But as teams scale and pipelines multiply, those inefficiencies turn into latency, developer frustration, and operational bottlenecks. Deliberate organization and performance tuning are now expected to keep delivery flowing at enterprise speed.

Risk:

Poor structure increases latency and developer friction.

Best Practice:

- ✓ Simplify dependency resolution by consolidating endpoints via repository groups.
- ✓ Avoid excessive proxies per group.
- ✓ Separate snapshots from releases.
- ✓ For Docker: use subdomain routing, not legacy ports.

Benefit:

Faster builds, simpler CI configuration, and better scaling.

Data Point:

Organizations using recommended grouping and routing practices saw a 3x reduction in average artifact resolution time.

9. Promote Artifacts Instead of Rebuilding

Rebuilding artifacts for every environment was once common practice, accepted as part of the delivery process. But this approach introduces variability, erodes traceability, and undermines confidence that testing matches production. Today, promoting validated artifacts across stages is the standard for ensuring consistency and maintaining a verifiable chain of custody.

Risk:

Rebuilding artifacts in each environment introduces variability and breaks traceability. This undermines testing and auditing, especially when artifacts differ between dev, staging, and production.

Best Practice:

- ✓ Use hosted repositories for each lifecycle stage (e.g., staging, QA, prod).
- ✓ Preserve release integrity by promoting verified artifacts between environments.
- ✓ Avoid rebuilding from source after initial validation — promote instead.

Benefit:

Ensures the artifact tested is the artifact released. Promoting artifacts preserves auditability, eliminates inconsistencies, and aligns with secure software release practices.

Data Point:

Promoting binaries rather than rebuilding them is a foundational DevSecOps pattern used by organizations with mature software delivery pipelines (State of the Software Supply Chain 2023).

10. Enforce Immutable Artifact Versions

Allowing artifacts to be overwritten or re-published under the same version once seemed like a harmless convenience. In practice, it breaks reproducibility, obscures what was actually deployed, and complicates incident response. Immutability has become the standard expectation for maintaining trust, preserving SBOM accuracy, and meeting regulatory demands for verifiable software.

Risk:	Best Practice:	Benefit:
Mutable artifacts (e.g., re-publishing version 1.0.0) break build reproducibility and make it impossible to verify what was actually deployed.	<ul style="list-style-type: none">✓ Ensure traceable releases by enforcing immutable artifact versions.✓ Enforce versioning policies within CI/CD (e.g., Semantic Versioning).✓ Limit snapshot usage to early-stage development only.	Immutability strengthens trust in your release artifacts, supports SBOM traceability, and simplifies incident response and compliance reviews.

Data Point:

Immutable artifacts are essential to maintaining SBOM integrity, according to guidance aligned with U.S. Executive Order 14028 and NTIA's SBOM Minimum Elements.

Maturity Benchmarks

Understanding adoption is important—but so is knowing what “good” looks like. These maturity benchmarks help executive leaders evaluate their organization’s progress across critical Nexus Repository practices.

Practice Area	Average Adoption	Good Adoption	Best-in-Class Adoption
Proxy Usage	<30% of builds use Nexus	31–90% adoption across teams	100% of builds route through Nexus
Repository Group Usage	Inconsistent use of groups	Major repos use groups	All repos grouped with fallback routing
Access Governance	Ad hoc roles and logins	SSO + RBAC partially adopted	Centralized RBAC via SSO; least privilege enforced
Policy Enforcement	Manual scanning or ad hoc reviews	Lifecycle/Firewall on critical repos	Full Lifecycle or Firewall coverage of all proxy repos
Retention Coverage	Minimal cleanup applied	High-churn repos managed	All active repos governed by retention policy
SBOM Adoption	SBOMs created occasionally	SBOMs for critical apps only	SBOMs published for every production release

These benchmarks support roadmap planning and help leadership track progress toward consistent, scalable DevSecOps maturity.

Executive Operational Adoption Metrics

To sustain secure and compliant repository operations, security leaders should track adoption through the following operational metrics. These metrics provide insight into the effectiveness of DevSecOps policy enforcement and help identify systemic weaknesses before they turn into security incidents.

Proxy Usage Rate

Measure the percentage of builds and developer environments retrieving dependencies exclusively through Nexus Repository rather than directly from public registries. Low usage indicates a risk of shadow downloads, compliance drift, and lack of governance.

Repository Group Coverage

Assess how consistently teams use repository groups (e.g., maven-public, npm-group) rather than configuring direct access to individual hosted or proxy repositories. This reduces complexity, enforces consistent policy, and simplifies failover planning.

RBAC & SSO Coverage

Track the proportion of users authenticated via SSO and governed by role-based access control. Full adoption ensures centralized governance, facilitates onboarding/offboarding, and supports audit trails.

Policy Firewall Coverage

Evaluate the percentage of proxy repositories protected by Nexus Firewall. This helps quantify your exposure to known vulnerabilities and unvetted third-party risk.

Retention Policy Coverage

Identify what portion of hosted and proxy repositories are governed by automated cleanup or retention policies. This prevents uncontrolled blob store growth and reduces the long-tail exposure to outdated or vulnerable components.

SBOM Adoption Rate

Track how many production releases are accompanied by a Software Bill of Materials (SBOM) stored in Nexus. This is increasingly a compliance expectation under regulations like EO 14028 and is essential for proving provenance.

Together, these adoption metrics enable executive stakeholders to evaluate the strength of repository governance, identify areas of improvement, and report confidently on the organization's DevSecOps maturity.



Questions to Ask Your Team

Are we enforcing all downloads through Nexus?

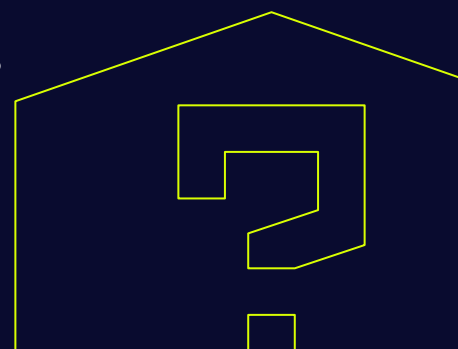
Do we have role-based access controls tied to SSO?

When was our last Nexus recovery drill?

Do we generate and store SBOMs per release?

How often do we scan Nexus components for policy violations?

Are HA and backup strategies tested and monitored?



Closing Note

Treating Nexus Repository as critical infrastructure isn't optional—it's a requirement in today's regulatory and threat landscape. These practices deliver not just technical hygiene, but strategic advantages: faster recovery, stronger compliance posture, and a demonstrably safer software supply chain.

Backed by Sonatype's platform and intelligence data, Nexus becomes more than a repo manager—it becomes your security enforcer.

Data sourced from Sonatype's State of the Software Supply Chain (2023–2024), internal customer benchmarks, and industry reports where noted.



Sonatype is the leader in secure software development built on open source and AI. As the maintainers of Maven Central and creators of Nexus Repository, Sonatype has spent two decades pioneering how the world manages and secures open source software — making Sonatype the trusted authority for modern software supply chains. With unmatched open source visibility and a unified product suite built for modern software development, Sonatype gives enterprises the intelligence and automated governance they need to harness the full potential of open source and AI. Sonatype handles the complexity behind the scenes: guiding component and model selection, blocking harmful malicious code, automating dependency and vulnerability management, and ensuring faster, more reliable builds — so developers spend more time on innovation and less time on remediation and rework. Trusted by more than 15 million developers, Sonatype helps power secure, modern software development at nearly 2,000 global organizations including 70% of the Fortune 100. To learn more about Sonatype, please visit www.sonatype.com.