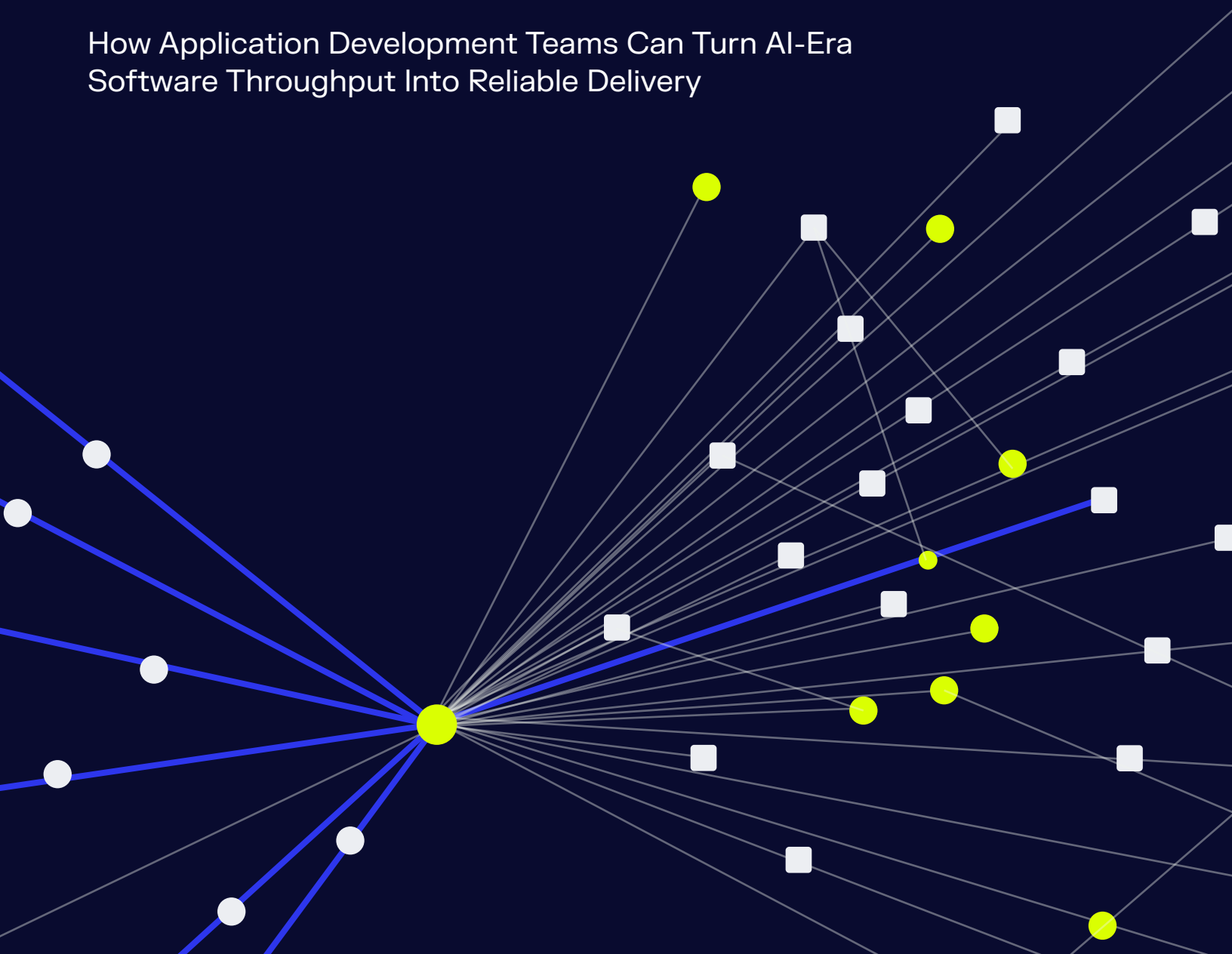




Modern Repository Security That Moves at Machine Speed

How Application Development Teams Can Turn AI-Era
Software Throughput Into Reliable Delivery



AI-assisted development is making it easier to create code, assemble applications, and introduce new software into the development workflow. For application development and platform leaders, a pressing question is whether or not that speed translates into more stable applications in production.

The answer depends on if teams can access the software they need without creating avoidable build instability, dependency rework, and release delays. A trusted repository control point helps make the fast path reliable: developers and pipelines can consume approved artifacts quickly, while malicious or policy-violating components are stopped before they disrupt delivery.

Sonatype Nexus Repository provides that foundation. It gives developers, CI/CD pipelines, and AI-assisted workflows a secure, scalable system of record for the packages, containers, binaries, models, and tools they build from. It centralizes artifact management, improves reliability and traceability, and applies protection and policy controls at the front door.

Executive Summary

- ▶ AI-assisted development is increasing the speed and volume of software consumption. The challenge is turning that activity into more applications successfully reaching production, not simply generating more code and dependency requests.
- ▶ Sonatype Nexus Repository gives development teams a reliable, supported path to the packages, containers, models, and internal artifacts they need, helping reduce build friction and improve delivery predictability.
- ▶ Open source malware has become an industrialized threat. Sonatype observed more than 464,650 new malicious packages in Q2 2026, bringing the cumulative total of known and blocked malicious packages to more than 1.8 million across major ecosystems.
- ▶ A repository is no longer just infrastructure for storing and caching artifacts. It is a control point for developer productivity, application stability, software supply chain governance, and operational resilience.
- ▶ Policy turns repository governance into an engineering advantage. Teams can automate recurring component decisions, provide developers with clear guidance, and apply consistent standards across formats, projects, and environments.
- ▶ As software decisions accelerate, governance must operate at the moment a component is chosen, not after it has disrupted a build, created rework, or entered an application.

The Repository Has Become a Strategic Application-Delivery Control Point

Repository management was once viewed primarily as a way to store binaries, proxy public registries, and make builds more reliable. Those functions remain fundamental. But the repository now sits directly in the path of modern application delivery.

It is where development teams can manage the inputs that shape applications:

- ▶ Open source packages and dependencies.
- ▶ Internally developed components and build artifacts.
- ▶ Container images and layers.

- ▶ Plugins, build tools, and automation assets.
- ▶ AI/ML models and related artifacts.
- ▶ Components selected by developers, CI/CD systems, and AI-assisted workflows.

As the threat landscape changes, so does the role of the repository. [Sonatype research](#) identified 3,430 malware advisories in 2025, a 3.69x increase over the pre-AI baseline of 931 advisories per year. Even after removing large-scale abuse campaigns that can inflate malware counts, the number of distinct malicious payloads increased by more than 50%. Attackers are not simply publishing more malware, they are using more techniques, delivery mechanisms, and approaches to compromise builds.



A repository is no longer just an operational system for storing and retrieving artifacts. It is a strategic control point for ensuring developers and pipelines can access the software they need through a reliable, governed path.

Sonatype Nexus Repository helps teams establish that foundation without forcing them to sacrifice speed or flexibility. Nexus Repository centralizes software assets, proxies externally sourced components, hosts internally produced artifacts, and provides a consistent way for teams to consume packages, containers, binaries, and AI/ML assets.

That centralization creates practical delivery advantages:

- ▶ **Improved build reliability.** Local caching and proxying reduce dependence on the availability and performance of public registries.
- ▶ **A consistent developer experience.** Teams can access approved tools and components through a supported path instead of navigating fragmented repositories and ad hoc download sources.
- ▶ **More predictable delivery.** Standardized artifact access reduces the inconsistencies that can interrupt builds, delay releases, or create unnecessary troubleshooting work.
- ▶ **Better traceability.** Artifact history helps teams understand what a component is, where it came from, when it entered the environment, and how it moved through software delivery workflows.
- ▶ **Operational resilience.** A centralized repository foundation can reduce disruption from upstream registry outages, changing availability, or inconsistent access across development environments.
- ▶ **A basis for automated governance.** When software enters through a known control point, organizations can apply policies consistently instead of relying on disconnected tools and manual processes.

For enterprise teams, that control has to work at scale. A repository strategy must provide a consistent path for development teams to consume trusted software while giving platform leaders the traceability, resilience, and governance needed to operate across teams, formats, environments, and locations — without adding friction to delivery.

Malicious Packages Turn a Development Decision Into an Incident

Open source software is foundational to application development, but public ecosystems are also an efficient distribution channel for attackers. A malicious package can reach developer workstations and CI/CD environments through the same trusted registries teams use every day.

The impact is not limited to a security event. A malicious package can interrupt a build, expose credentials, force an application team into recovery work, and delay software that was otherwise on track for production.

The scale of the problem continues to grow. Sonatype observed 464,650 new malicious packages in Q2 2026, bringing the cumulative total of known and blocked malicious packages to more than 1.8 million across major ecosystems. [Sonatype's 2026 State of the Software Supply Chain report](#) also shows how attackers are evolving their methods: in 2025, more than 99% of observed open source malware occurred on npm, and campaigns increasingly used multi-stage behavior designed to steal secrets, profile hosts, establish backdoors, download additional payloads, and propagate through local projects and registries.



Several trends make this difficult to address through traditional downstream processes alone:

- ▶ **Attackers exploit familiar development patterns.** Malicious packages may imitate plugins, framework extensions, build utilities, SDKs, wrappers, or migration tools, not just obvious misspellings of popular packages. A routine dependency decision can become a source of unexpected rework.
- ▶ **A trusted name is not a safety verdict.** A compromised maintainer account or hijacked project can turn a familiar component into a delivery mechanism for malicious code, creating release risk even for teams using established ecosystems.
- ▶ **Malware can be part of a larger attack chain.** The package may be only the initial stage, followed by credential theft, encrypted payloads, remote access, or further compromise. That can turn a single download into a developer-environment or CI/CD recovery effort.
- ▶ **Campaigns can spread at machine speed.** The Shai-Hulud campaign demonstrated how open source malware can use stolen credentials to rapidly propagate poisoned updates through developer environments and public package ecosystems, creating disruption across multiple projects and teams.

These threats are not well suited to manual review. No development organization can reasonably expect every engineer to inspect every package, source repository, container layer, or transitive dependency before use.

The more practical approach is to apply protection at the point where software enters the development environment.

Block Risk Before It Becomes Rework

The earliest point to prevent a harmful component from disrupting application delivery is when it is requested.

Once a malicious or policy-violating component has been downloaded, it can become embedded in a local project, a build cache, a container workflow, or a CI/CD pipeline. Even if a later-stage scan detects it, teams may then need to investigate exposure, identify affected builds, remove or replace the component, retest the application, and document the outcome.

That is costly, disruptive work, especially when the component could have been blocked before it entered the environment.

Sonatype Nexus Repository helps organizations apply malicious-component protection at the repository front door. Instead of treating all content from a public registry as equally acceptable, the repository can help stop known malicious components before they reach developer workstations, builds, or downstream applications.

This approach protects delivery flow. Developers receive fast access to components that meet organizational requirements, while components that are known to be malicious or unsuitable for use are stopped before developers invest time building on them.

When a risky component is allowed through



A developer, pipeline, or AI agent downloads the component.



The component becomes part of a project, build, or container.



A later tool or security team detects the problem.



Teams remove, replace, retest, and document the impact.



Delivery slows while the issue is resolved.

When it is blocked at the repository



The request is evaluated before the component enters the workflow.



The developer is prevented from consuming a known risky component.



The organization avoids downstream investigation and remediation work.



Teams can move to an approved or safer option sooner.



The safe path remains the fast path.

Blocking malicious components is therefore not just an application security capability. It is an application delivery capability. It helps prevent avoidable interruptions before they affect developer throughput, pipeline reliability, application stability, or release schedules.

Policy Makes Repository Governance Practical

Repository governance is the ability to make consistent, accountable decisions about what software can enter and move through development workflows. It gives teams a practical way to apply organizational standards at the point components are requested, rather than discovering exceptions or policy violations after a build has already begun.

Malware protection addresses an urgent category of risk: components intentionally designed to harm an organization. But development teams also need a repeatable way to govern the broader set of software-consumption decisions that affect delivery, quality, licensing, and operations.

That is the role of repository policy.

Policies allow organizations to define the conditions under which components can enter and move through development workflows. Instead of leaving individual teams to interpret the same risk, licensing, and quality questions repeatedly — or creating a generic approval queue — policy turns those standards into automated, consistent decisions.

A policy-driven repository strategy helps organizations define:

- ▶ Which components and versions are approved for use.
- ▶ Which packages or versions should be blocked.
- ▶ What security, license, quality, or operational criteria a component must meet.
- ▶ Which applications, repositories, or environments require stricter controls.
- ▶ How exceptions are documented, owned, scoped, and expired.

The goal is to make the trusted path the easy path: give developers and AI-assisted workflows fast access to what they need while making standards clear, exceptions controlled, and risky choices harder to carry forward.

For application development leaders, this means fewer inconsistent component decisions across projects, fewer late-stage surprises, and less time spent asking developers to interpret complex risk data. For platform engineers, it means controls that can be delivered through the tools and repositories developers already use. For security teams, it means governance can occur before a component is embedded in an application.

Effective policies should be:

- ▶ **Clear enough to automate.** Developers and pipelines need predictable outcomes, not ambiguous guidance.
- ▶ **Targeted to delivery needs.** Different repositories, formats, applications, and environments may require different controls.
- ▶ **Actionable for developers.** A block should explain the reason and point teams toward an approved path whenever possible.
- ▶ **Governed but flexible.** Organizations need a controlled way to document and time-box legitimate exceptions without creating permanent workarounds.
- ▶ **Continuously maintained.** Threat intelligence, component health, licensing needs, and application requirements change over time.

With the right policies in place, the repository becomes more than a record of software consumption. It becomes a governance control point that helps teams make reliable choices, preserve accountability, and keep delivery moving at scale.

Address the Downloads That Bypass the Repository

A repository strategy is most effective when teams use it consistently. But not every software request follows an approved repository path.

Developers may download components through command-line tools, scripts, personal workflows, or instructions generated by AI assistants. These direct downloads can bypass repository controls and create an inventory blind spot precisely where malicious packages can do the most damage: on a developer workstation or in a CI environment.

This is a software-consumption version of shadow IT and shadow AI. When an approved path is slow, difficult to use, or incomplete, teams find workarounds. The answer is not to make governance more restrictive. It is to provide a fast, trusted repository experience while extending protection to the paths that fall outside it.

The Sonatype integration with Zscaler helps address this gap. Zscaler provides internet access controls at the network edge, while Sonatype supplies package-level malware intelligence for requests to public open source registries. The integration can help block or quarantine suspicious packages before they reach a developer machine or CI pipeline, including downloads that do not pass through an internal repository.

This gives organizations an additional layer of protection while they improve adoption of their trusted repository workflow. It also helps application, platform, and security teams reduce unmanaged software intake that can destabilize delivery.

A Practical Program for Modern Repository Security

Every organization will have different architectures, development practices, and risk requirements. But a strong repository strategy generally begins with a few core principles.



CENTRALIZE BUILD ARTIFACTS

Create a reliable system of record for the artifacts teams consume, create, and promote. Centralization gives developers a fast, supported path to the software they need while creating visibility when teams need something outside the standard path.



APPLY PROTECTION AT INTAKE

Block known malicious components before they reach developers, build systems, or applications. Earlier prevention reduces the likelihood that teams will need to investigate, remediate, and retest software that should never have entered the environment.



EXTEND COVERAGE BEYOND THE REPOSITORY

Use complementary controls, such as the Sonatype integration with Zscaler, to address public-registry downloads that bypass approved repository workflows. This provides an added layer of protection while helping teams identify and reduce unmanaged software consumption.



ESTABLISH CLEAR POLICIES

Define what is permitted, restricted, or blocked across the organization. Policies should reflect actual application delivery, security, licensing, and operational requirements and should be paired with understandable developer guidance and governed exceptions.



DESIGN FOR DEVELOPER ADOPTION

The governed path needs to be practical. Give developers dependable access to the components, formats, and tools they need in a centralized repository. Measure and address the friction that drives direct downloads or other workarounds.



KEEP GOVERNANCE CURRENT

Repositories, policies, and threat conditions are not static. Review usage patterns, exceptions, blocked components, and policy outcomes regularly. The goal is not a one-time repository implementation; it is an operating model that evolves with changing dependencies, AI workflows, and application-delivery needs.

Build Faster With Software You Can Trust

The speed of modern application development depends on the ability to reuse software confidently. Developers, pipelines, and AI-enabled workflows need rapid access to the components that help them build. Application development leaders need that speed to translate into stable applications and dependable production delivery.

Sonatype Nexus Repository provides the secure, scalable system of record that brings those needs together. It centralizes the artifacts that power modern software delivery, helps block known malicious components at the point of entry, and enables teams to apply consistent policies across the software supply chain.

With a trusted repository foundation and added edge protection, organizations can reduce downstream rework, strengthen governance, and keep AI-assisted development speed from becoming operational risk.

Control what goes in. Trust what comes out.

Learn how Sonatype Nexus Repository can help your organization build, govern, and deliver software with greater speed and confidence.



Sonatype is the company that accelerates agentic software development with confidence. Trusted by thousands of enterprises and millions of developers, Sonatype helps organizations build with confidence by governing the open source, AI-generated, and third-party components that power modern software. As the steward of Maven Central and the company behind Nexus Repository, Sonatype provides unmatched visibility into how software is built, consumed, and secured — helping teams move faster, reduce risk, and ship software with confidence at AI scale. To learn more about Sonatype, please visit www.sonatype.com