TECHNICAL GUIDE

Chainguard

The new standard for software supply chain security

Introduction

The axios attack in March 2026 was not a nation-state breach of a government system. UNC1069, a state-sponsored threat group independently attributed by Google Threat Intelligence Group and Microsoft Threat Intelligence, didn't penetrate a classified network or exploit a zero-day in federal infrastructure. Instead, they social-engineered a single open source maintainer, obtained an npm access token, and published two backdoored versions of one of the most widely used JavaScript packages on the internet. For roughly three hours, axios, present in an estimated 80 percent of cloud and code environments, silently deployed a cross-platform remote access trojan on every system that pulled a routine update.

Any build pipeline pulling from public registries faces the exact same exposure. The attack vector isn't the perimeter. It's the dependency.

That attack arrived alongside the emergence of Mythos, the first publicly disclosed frontier model to demonstrate autonomous exploitation of Common Vulnerabilities and Exposures (CVEs) and long dormant zero-days at expert scale. The implication is direct: the AI models available to engineering teams to accelerate development are, in adversarial hands, also available to find and exploit vulnerabilities at machine speed.

This is the new threat environment for software supply chain security. Reactive security cannot withstand an adversary operating at this pace and level of sophistication. The only viable defense is to eliminate vulnerabilities and shield malware attack points before poisoned artifacts reach the software development lifecycle (SDLC).

Software supply chain security has become a baseline expectation across the industry. What has changed is the speed and sophistication of the adversary. This guide addresses what it takes to actually meet that bar: understanding the threat landscape, a practical and secure AI software supply chain, and how Chainguard helps engineering and security teams protect the full SDLC so they can build for their business rather than manage a CVE backlog.

The threat landscape has three vectors, and all three are accelerating

Software supply chain attacks

The axios attack is a clear, recent example of how threat actors approach the open source supply chain and why commercial organizations are in the blast radius.

UNC1069 spent approximately two weeks social-engineering axios's primary npm maintainer, Jason Saayman, through an elaborate fake company impersonation on Slack and Microsoft Teams. The attacker obtained a long-lived npm access token and published two backdoored versions: axios@1.14.1 and axios@0.30.4. The malicious dependency, plain-crypto-js@4.2.1, had been pre-staged 18 hours earlier specifically to avoid "brand-new package" detection rules. The payload, WAVESHAPER.V2, was a cross-platform remote access trojan (RAT) that executed via a post-install script with no user interaction required and deleted its own dropper after delivery.

Axios has over 100 million weekly downloads and is present in an estimated 80 percent of cloud and code environments. The exposure window was approximately three hours. Even so, Huntress confirmed over 135 compromised endpoints, and Google Threat Intelligence Group assessed that the full scope of the breach remained unknown at the time of publication.

> **"North Korean hackers have deep experience with supply chain attacks, which they've historically used to steal cryptocurrency. Given the popularity of the compromised package, we expect it will have far-reaching impacts."**
>
> **John Hultquist**
> CHIEF ANALYST, GOOGLE THREAT INTELLIGENCE GROUP

For engineering teams, open source drives meaningful efficiency by not having to build commonly used artifacts from scratch and is key to enabling faster delivery. The key is to ensure those artifacts are hardened and trusted because in modern software development, dependencies in public registries are the attack surface.

AI-powered vulnerability exploitation

Anthropic's Mythos Preview is the first publicly disclosed frontier model to demonstrate autonomous CVE exploitation at expert scale. Its existence changes the economics of attack in a fundamental way: vulnerabilities that previously required skilled, time-intensive human exploitation can now be identified and exploited programmatically, at machine speed, across a wide target surface.

The implication for engineering and security teams is direct. Adversaries now have access to the same class of frontier models that development programs use to accelerate delivery. AI-accelerated code generation without a hardened foundation is also an AI-accelerated expansion of the attack surface. For every hour AI tooling saves an engineer writing code, it also helps an attacker find the unpatched CVE in the dependency that code pulls in. Machine-speed development demands machine-speed supply chain security, and machine-speed security cannot be achieved through manual patching and triage.

These AI models will eventually be available to anyone, meaning that organizations have to think about protecting themselves not just against organized, nation-state-level threat actors, but also "average joes" who can use AI tools to operate at a scale much larger than one person.

Nation-state AI capability targeting commercial infrastructure

Alongside Mythos, nation-state frontier models with autonomous vulnerability-exploitation capabilities are an emerging threat to commercial organizations. In adversarial hands, models in this class can be directed at critical business systems, cloud infrastructure, and production software at any organization, not just government targets.

Mythos is the first model publicly confirmed to do this well. It will not be the last. The pace of frontier model development means this threat only accelerates, and the gap between a model's release and its deployment in offensive operations continues to compress.

Taken together, these three vectors describe a consistent pattern: adversaries are targeting the open source supply chain because it is the path of least resistance into commercial environments. The question for every security and engineering team is not whether the software it builds on will be targeted. It's whether the foundation was hardened before the attack arrived.

What a secure AI software supply chain looks like

Securing the AI software supply chain has become far more consequential. Best practice has a clear definition. Every artifact in the SDLC should be built to the following standard.

- **Minimal:** Contains only the components strictly required to run the application, reducing dependencies, image size, and attack surface. No shells, package managers, or extras that attackers exploit.
- **Zero CVE:** Shipped with no known vulnerabilities at publish time, and kept at zero through continuous rebuilds. Vulnerabilities are not patched reactively; artifacts are rebuilt from source on a cadence that outpaces the exploit window.
- **Built from source:** Compiled from verified upstream source code in a Supply-chain Levels for Software Artifacts (SLSA) Level 3-compliant build environment, rather than repackaging opaque binaries. The package an engineer pulls is verifiably the same one built from reviewed, approved source code, with no opportunity for tampering in transit.
- **Compiled with hardened flags:** Binaries built with strong compiler hardening options to mitigate memory safety bugs and common exploit techniques.
- **Without install-time scripts:** Packages and libraries published without install-time scripts that run arbitrary code on developer or build machines. This is the vector WAVESHAPER.V2 used. Eliminating it closes the entire class of attack.
- **SLSA Level 3 provenance:** Artifacts built in a hardened, isolated environment with cryptographic provenance, signed Software Bills of Materials (SBOMs), and verifiable signatures. Every artifact is auditable from build to deployment.

Executing this consistently at scale across the full SDLC protects against the three main attack classes: known vulnerabilities (CVEs and Known Exploited Vulnerabilities), unknown vulnerabilities (zero-day attacks), and malware. Building and maintaining this in-house, at the pace of AI-powered development, is extraordinarily hard. The adversary's attack pace is also accelerating. These two realities make a purpose-built, continuously maintained supply chain foundation a necessity for any organization building production software.

How Chainguard makes it safe to build with AI and protects against it

Chainguard secures the software supply chain at the stages of the SDLC where untrusted code can enter an operational environment. Every container, package, library, action, agent skill, and virtual-machine image is built from source in the Chainguard Factory, verified with full provenance, and never pulled from a public registry. This approach integrates seamlessly into your current SDLC and proactively secures against attacks, vulnerabilities, and malware, so engineers can build faster without triaging security risks.

Safe dependencies: plan and build

Chainguard products: Chainguard Libraries, Chainguard Agent Skills, Chainguard Containers, Chainguard OS Packages, Chainguard Actions

Every engineer and AI agent constantly reaches for open source dependencies. Under the current status quo, each of those pulls is a potential WAVESHAPER delivery vector, a moment when a compromised or typosquatted package can enter the build pipeline.

HOW CHAINGUARD HELPS

Because Chainguard Libraries never builds packages with post-install scripts, an attack like axios has no path in. This is structural prevention: the attack mechanism doesn't exist in Chainguard-built artifacts. During the axios attack window, Chainguard Libraries for JavaScript had 83 safe axios versions available throughout.

This protection applies across developer or agent builds: language libraries (Chainguard Libraries), verified AI agent instruction sets (Chainguard Agent Skills), base container images (Chainguard Containers), OS-level dependencies (Chainguard OS Packages), and CI/CD workflows (Chainguard Actions).

CI/CD integrity: build, test, and deploy

Chainguard products: Chainguard Actions, Chainguard OS Packages, Chainguard Containers

During the build, test, and deploy stages, the pipeline builds, tests, and deploys the application. The threat at this stage is the build process itself being weaponized: a tampered action, a malware ghost release, a compromised build environment, or a development tool that becomes the delivery mechanism for an attack.

A compromised build pipeline can halt a release process entirely. Pipeline integrity is directly tied to delivery timelines and business continuity.

HOW CHAINGUARD HELPS

Chainguard Actions is built from source in a minimal, hardened build environment that runs daily with the same provenance guarantees as the artifacts it produces. Together with Chainguard OS Packages and Chainguard Containers, it closes the two most exploited layers of the CI/CD pipeline: the actions that automate it and the operating system that runs them. Version tags cannot be manipulated to redirect a pipeline to attacker-controlled code. The class of attack in which the CI/CD pipeline itself becomes a malware delivery mechanism is eliminated.

Hardened foundations: deploy and run

Chainguard products: Chainguard Containers and Chainguard VMs

Hardened foundations secure the critical stages when an application is pushed to operational environments and exposed to external threats. The threat at this stage is exploitable surface area in the runtime: malware in a base image, CVE-laden OS layers, oversized images carrying shells, package managers, and binaries that attackers exploit.

HOW CHAINGUARD HELPS

Chainguard Containers and Chainguard VMs are rebuilt daily, with no shells, package managers, or extras. Every foundational artifact required by a mission system ships with a signed SBOM and a verifiable build attestation. Attack surface is minimized before deployment.

The results are measurable. Chainguard delivers 97.6 percent fewer CVEs than industry alternatives. Chainguard has remediated 88,000+ CVEs across its catalog, saving engineering teams 352,000+ hours of remediation work. Those hours are engineering capacity and business value, not overhead. Zero-CVE host layer coverage extends to both cloud-hosted and on-premises deployments. Malware that bypasses CVE scanning has no path in because no untrusted binary was ever accepted into the artifact.

For teams navigating compliance requirements like SOC 2, Payment Card Industry Data Security Standard (PCI DSS), or ISO 27001, Chainguard's build provenance, signed Software Bills of Materials (SBOMs), and continuously verified artifacts provide the evidence base that makes continuous compliance achievable, not a documentation exercise that consumes engineering cycles before every audit.

Policy-governed repositories: trust at machine speed

With the Chainguard Repository, platform and security teams define trust once and enforce it everywhere. Security and compliance teams define rules as code: which projects are allowed, which versions are in bounds, which compliance frameworks apply. The Chainguard Repository enforces those decisions consistently across every human engineer and every AI agent running in the environment.

Engineers and agents pull from a pre-approved, continuously verified source. Security and platform teams get full provenance, deterministic behavior, and visibility when something drifts out of policy. This is how organizations maintain security governance as development velocity increases. Policy is defined once, enforced everywhere, and auditable at any point, whether that's for an internal review, a compliance assessment, or a post-incident investigation.

Trust without toil or mission disruption

Migrating to Chainguard's trusted open source artifacts is a drop-in replacement, not a multi-year modernization program of its own.

For container images, The Guardener automates migration. It points at an existing Dockerfile and build context, maps Debian, Ubuntu, or Alpine packages to Chainguard equivalents, converts images into modern multi-stage builds, iterates layer by layer until the migrated image behaves like the original, and outputs a ready-to-run Dockerfile alongside a migration report showing before-and-after changes in image size, CVE count, and filesystem composition. What used to require engineers rewriting Dockerfiles one by one becomes a repeatable, automated process.

The same drop-in pattern applies across the full catalog. Chainguard Libraries requires pointing a package manager to the Chainguard Libraries endpoint instead of public registries like npm, PyPI, or Maven Central. Chainguard Actions requires swapping GitHub Actions version tag references for pinned, immutable Chainguard Actions references. Chainguard OS Packages moves OS dependencies onto a hardened, continuously verified foundation. Chainguard Agent Skills replaces ad-hoc prompts with verified, policy-governed AI context.

Engineers keep using the same tools, CLI utilities, and workflows they rely on today. The result is a standardized, continuously verified open source foundation across containers, libraries, actions, and agent context, without a disruptive migration.

Why Chainguard

No other company secures more of the software supply chain by delivering threat-resistant open source artifacts with verifiable integrity. Chainguard's catalog includes 2,400+ container images, 600+ Federal Information Processing Standards (FIPS)-validated images, and 172,000+ version tags, supported by Chainguard OS, a purpose-built Linux distribution designed for continuous software integration and delivery, and the Chainguard Factory, an agentic build system that continuously drives every package toward its ideal state: zero CVEs, latest version, fully tested.

Customers building on this foundation include Anduril, OpenAI, Canva, Fortinet, Hewlett Packard Enterprise, Snap Inc., and Snowflake, organizations that require both development velocity and security guarantees.

The Chainguard founding team brings over 40 years of collective experience in software supply chain security, including the creation of Sigstore, the SLSA framework, Tekton, and distroless. This is the organization that defined the standards the industry is now working to meet, and that built the production system capable of enforcing them at scale.

Get started

Engineering and security teams ready to build on a trusted open source foundation can learn how Chainguard fits into their environment.

Contact us to discuss your team's requirements.