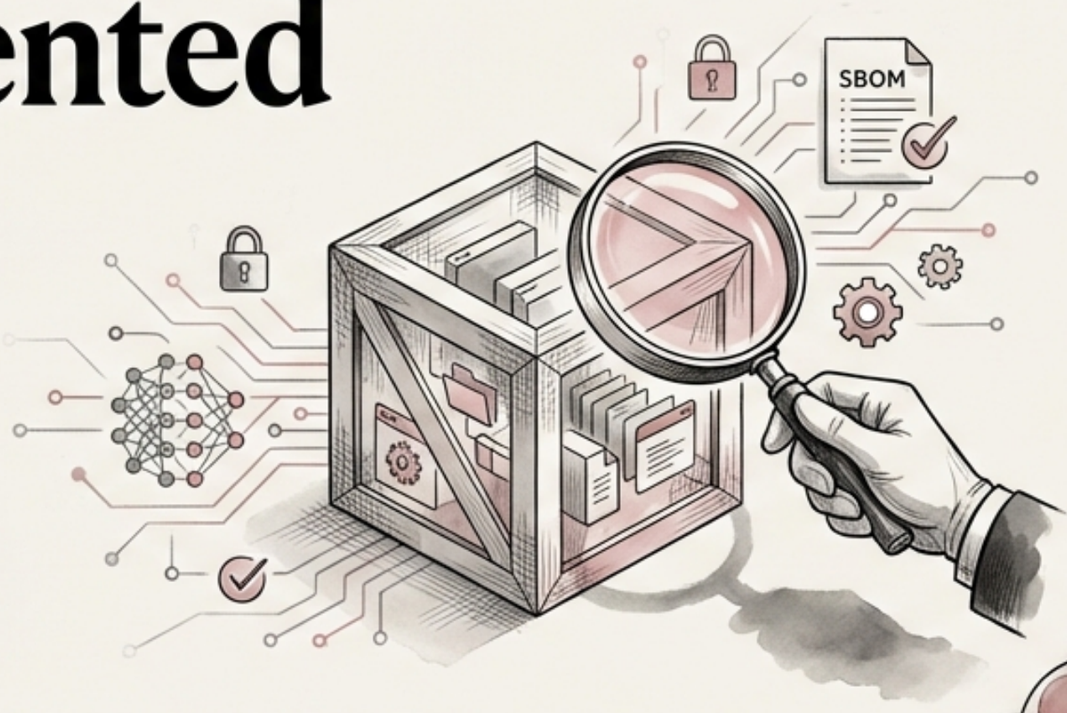
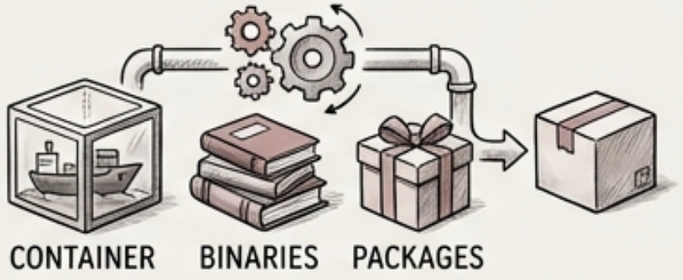
Module 6.2: Artifact Integrity & SBOM — Verifying What You Ship in AI-Augmented Development

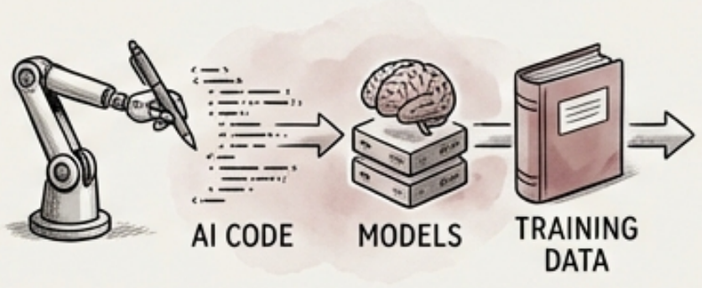


Securing the AI-Augmented Software Supply Chain: Introduction to Artifact Integrity



- Software artifacts, including container images, binaries, and packages, represent the final output of your build pipeline.

- Maintaining artifact integrity from build to deployment is crucial for security and reliability.



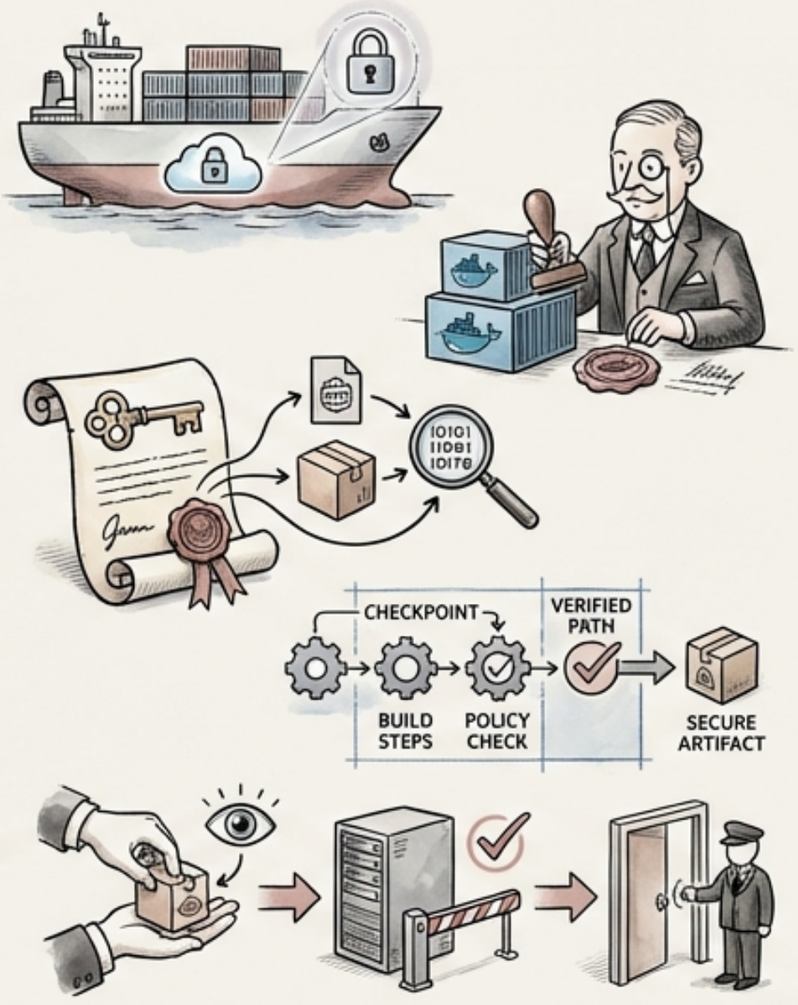
- For AI-augmented teams, this extends to include AI-generated code, models, and training data.

- Compromised artifacts can lead to significant vulnerabilities and supply chain attacks.

- This module will explore techniques for ensuring the security and integrity of your software artifacts.

Artifact Signing and Verification Techniques: Ensuring Provenance and Trust

- **Cosign (Sigstore):** Enables keyless signing of container images using OIDC identity for simplified security.
- **Notary v2:** Facilitates Docker Content Trust, ensuring the integrity and authenticity of Docker images.
- **GPG signing:** A traditional method for signing packages, binaries, and release artifacts to verify their origin.
- **In-toto:** Provides supply chain layout verification, ensuring artifacts have followed required build steps and policies.
- Verification must occur at every consumption point, including registry pull, deployment, and runtime admission control.

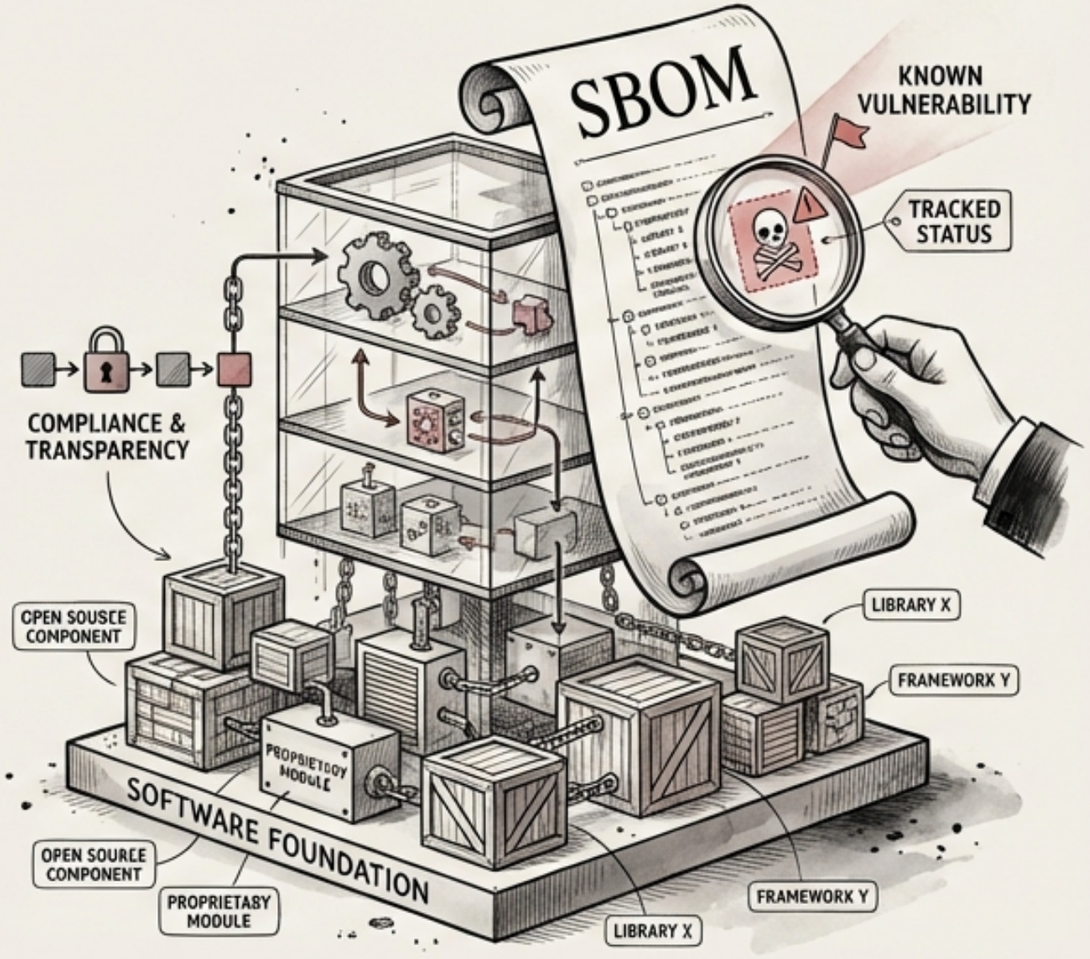


Software Bill of Materials (SBOM): A Comprehensive Inventory of Your Software

A Software Bill of Materials (SBOM) is a formal record containing the details and supply chain relationships of various components used in building software.

SBOMs enable transparency, vulnerability management, and compliance within the software supply chain.

SBOMs are critical for identifying known vulnerabilities within your dependencies, and tracking their status over time.

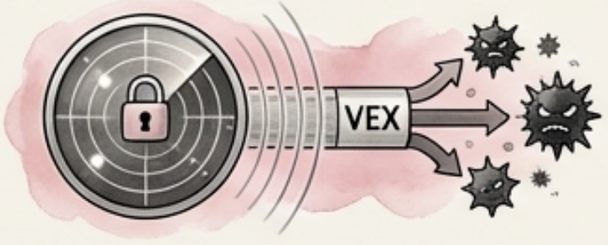


SBOM Formats: SPDX and CycloneDX – Choosing the Right Standard

SPDX 2.3: An **ISO standard** SBOM format that enjoys **broad ecosystem support** and **wide adoption.**

CycloneDX 1.5: An **OWASP** (Open Web Application Security Project) standard, focused on **security** with **VEX** (Vulnerability Exploitability Exchange) support.

SPDX is generally preferred for its **comprehensive coverage** and **wide-spread industry acceptance.**

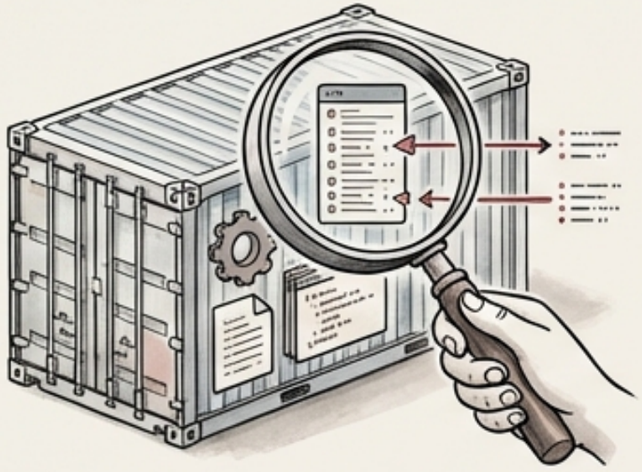


CycloneDX excels in **security-focused** applications due to its **advanced VEX** capabilities.

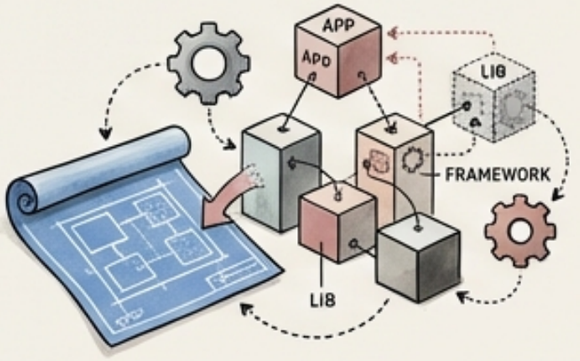
SBOM Content: Key Elements for Effective Dependency Management

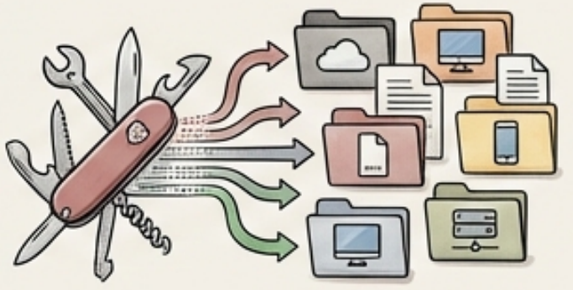
- **Component Name:** The unique identifier for the software component (e.g., library name).
- **Version:** The specific version of the identified component.
- **Supplier:** The organization or individual that created or supplied the component.
- **Hash:** A cryptographic hash of the component for integrity verification.
- **License:** The license under which the component is distributed.

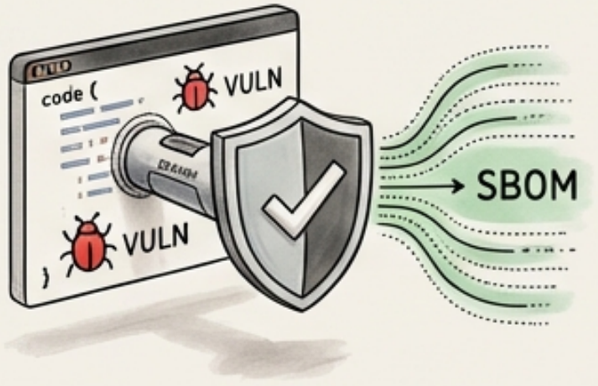
SBOM Generation Tools: Automating the Creation of Software Bills of Materials



- **Syft:** A popular tool for generating SBOMs for container images, providing deep insights into their contents.
- **cdxgen:** Focused on creating SBOMs for applications, helping to identify application-level dependencies.
- **Microsoft SBOM Tool:** An all-purpose tool for generating SBOMs for various project types.
- **Trivy:** A **vulnerability scanner** that also has SBOM generation capabilities, allowing simultaneous vulnerability detection and SBOM creation.
- Consider using **multiple tools** to ensure comprehensive SBOM coverage.

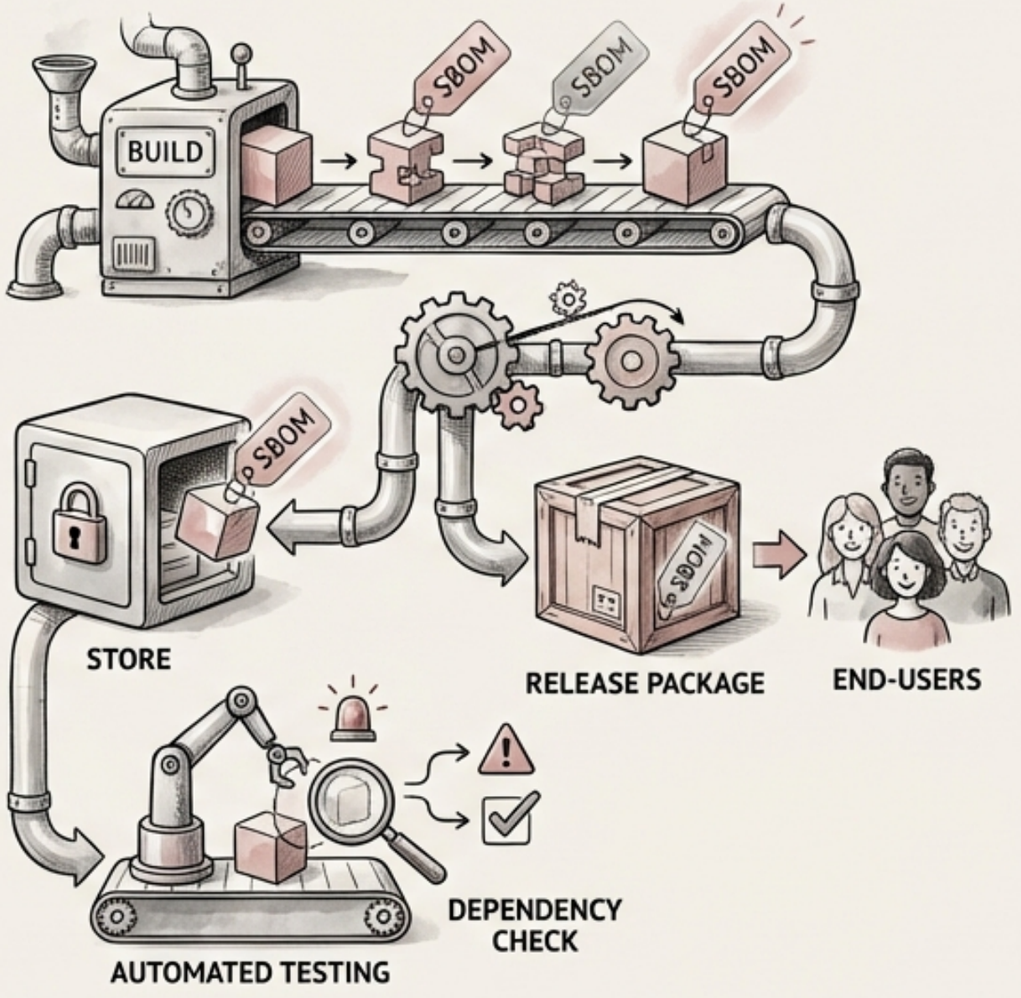






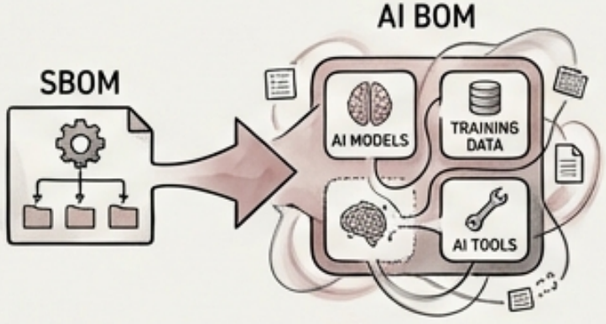
SBOM Generation Strategy: Integrating SBOMs into Your CI/CD Pipeline

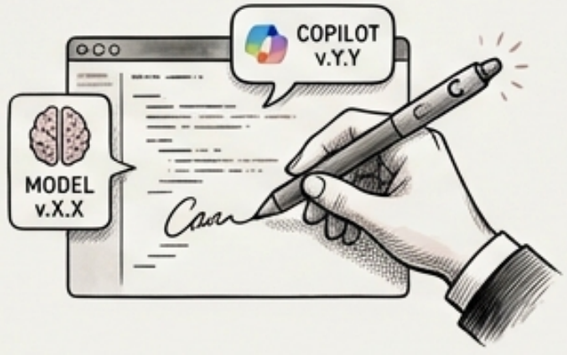
- Generate SBOMs at every build within your CI pipeline for maximum coverage and up-to-date dependency information.
- Store the generated SBOMs alongside your artifacts in a secure and accessible location.
- Include SBOMs within release packages to provide transparency to downstream users.
- Integrate SBOM generation into automated testing to detect dependency issues early.



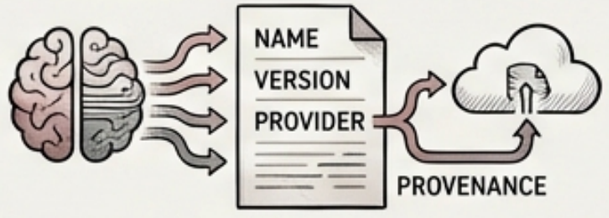
AI Bill of Materials (AI BOM):

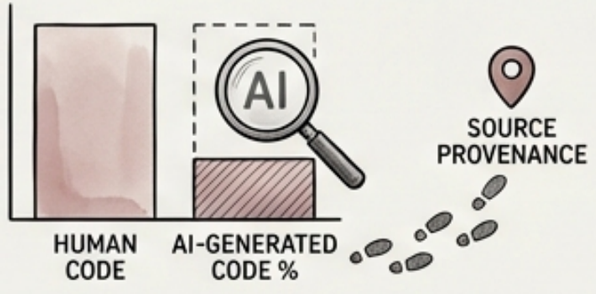
Extending SBOM for AI Components



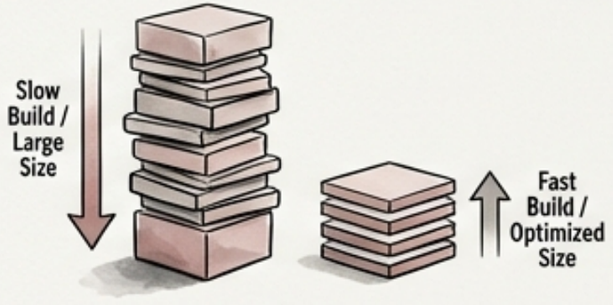


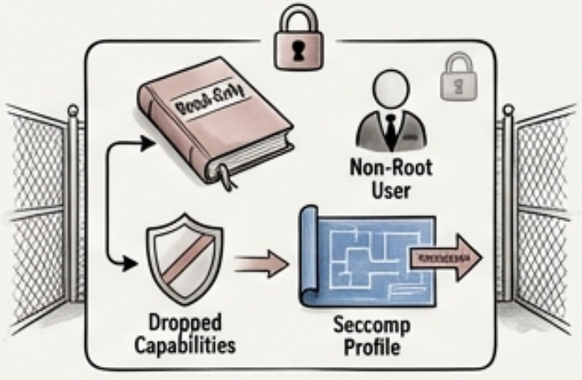
- Traditional SBOMs primarily focus on software dependencies, neglecting AI-specific components.
- AI Bill of Materials (AI BOM) extends the SBOM concept to include details about AI models, training data, and AI tools.
- This includes AI models (name, version, provider, training data provenance).
- Also AI tools used in development (Copilot version, model version).
- AI-generated code percentage and provenance also need to be tracked.



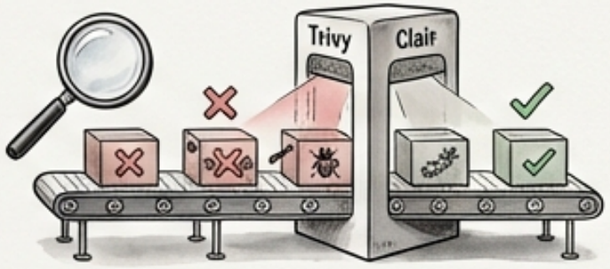


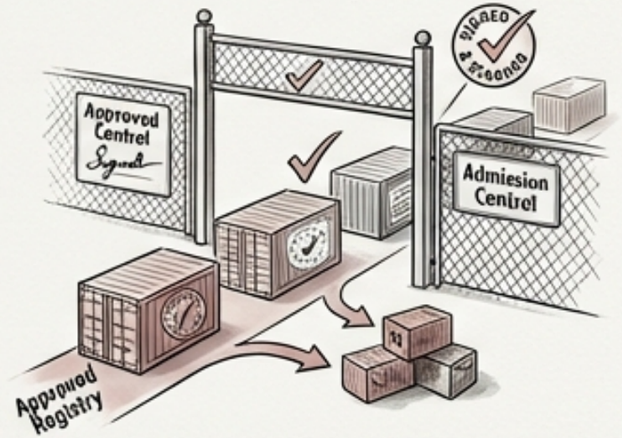
Container Image Security: Best Practices for Building Secure Images





- **Base image selection:** Use minimal, verified base images such as distroless, Alpine, or Chainguard to reduce the attack surface.
- **Image scanning:** Scan for OS and application vulnerabilities before pushing images to the registry using tools like Trivy or Clair.
- **Layer optimization:** Minimize the number of layers in your images to improve build times and reduce image size.
- **Runtime security:** Configure read-only filesystem, non-root user, dropped capabilities, and seccomp profiles to enhance runtime security.
- **Admission control:** Implement policies to only deploy images that are signed, scanned, and from approved registries.



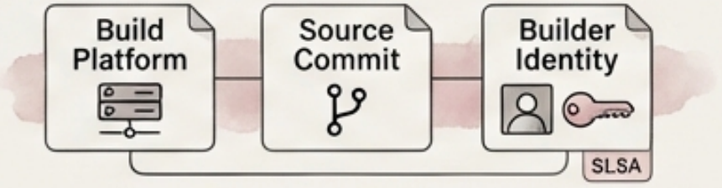


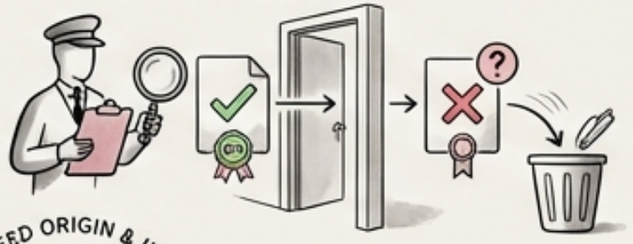
Artifact Registry Security: Protecting Your Centralized Artifact Store

- Registry Access Control: Implement granular access control with separate push and pull permissions, scoped to individual teams or repositories.
- Vulnerability Scanning: Enable automated vulnerability scanning on image push and continuous monitoring for new CVEs.
- Retention Policies: Define policies to keep a specific number of image versions (e.g., N versions), automatically delete untagged images, and preserve release artifacts.
- Immutability: Enforce immutability for production tags to prevent tag-based supply chain attacks.
- Geo-Replication: Implement geo-replication for availability and disaster recovery purposes.

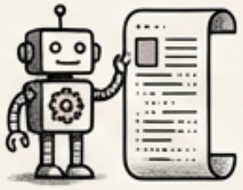
Provenance and Attestation: Verifying the Origin and Integrity of Artifacts

- **Build provenance** provides cryptographic proof of where, when, how, and by whom an artifact was built.
- **SLSA provenance format:** A standardized attestation including build platform, source commit, and builder identity.
- **Attestation storage:** Attach attestations to artifacts in the registry (Cosign attestations) or store them in a transparency log (Rekor).
- **Verification:** Consumers must verify provenance before deploying artifacts, rejecting those without valid attestations.
- Using provenance and attestation provides a strong guarantee about the origin and integrity of your artifacts.

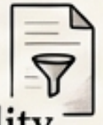


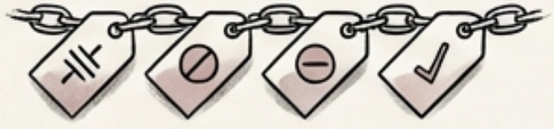


Vulnerability Exchange (VEX): Reducing False Positives in Vulnerability Management

- 
  VEX documents are machine-readable statements indicating whether a product is affected by a specific vulnerability.

- 

  VEX provides context beyond the presence of a vulnerable component in an SBOM, clarifying actual exploitability. 

- 
  - STATUS VALUES: include poversiva, AFFECTED, NOT_AFFECTED vulnerability is actually reachable within the product.

- 

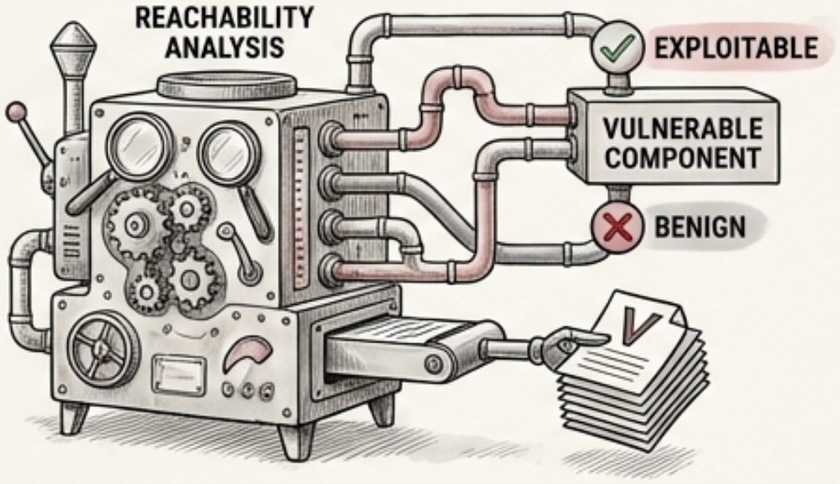
  - PURPOSE: Reduce false positive noise by clarifying whether a vulnerability is actually reachable within the product.

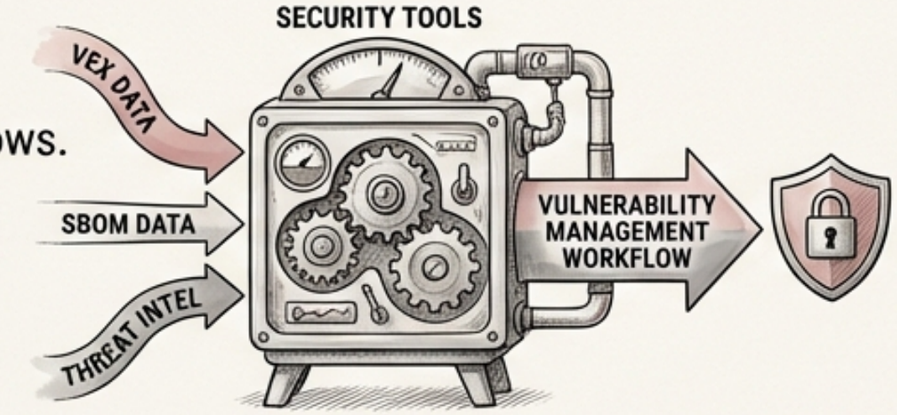
- STATUS VALUES: include AFFECTED, NOT_AFFECTED (with justification), UNDER_INVESTIGATION, FIXED.

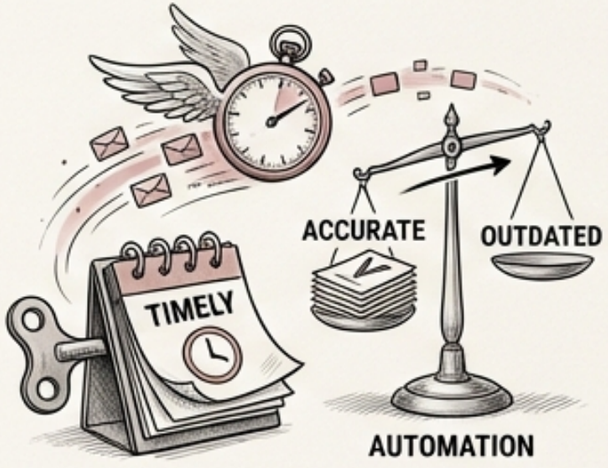
- 
  INTEGRATION: CycloneDX VEX, CSAF VEX, OpenVEX are the standard formats for VEX documents.

Automating VEX Generation and Distribution

- Generate VEX from reachability analysis tools to determine if a vulnerable component is actually exploitable.
- Distribute VEX alongside the SBOM to provide comprehensive vulnerability context.
- Integrating VEX into existing security tools streamlines vulnerability management workflows.
- Automate VEX generation to ensure timely and accurate vulnerability information.

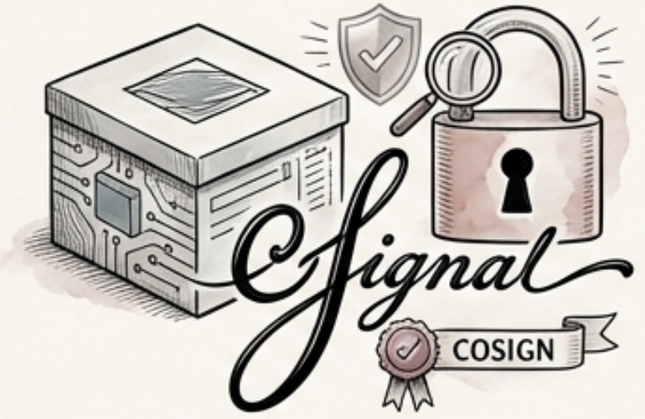


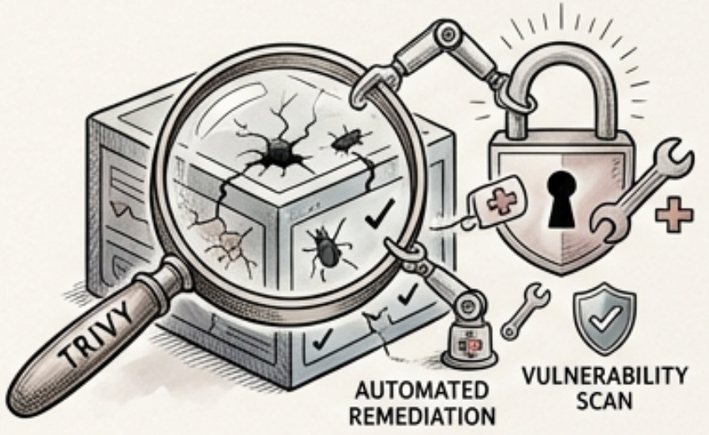


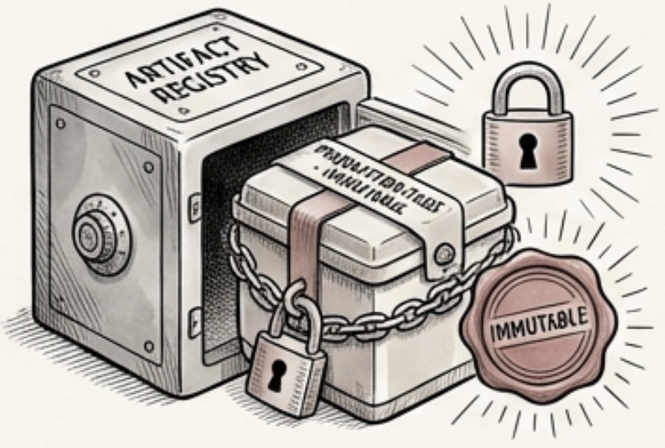


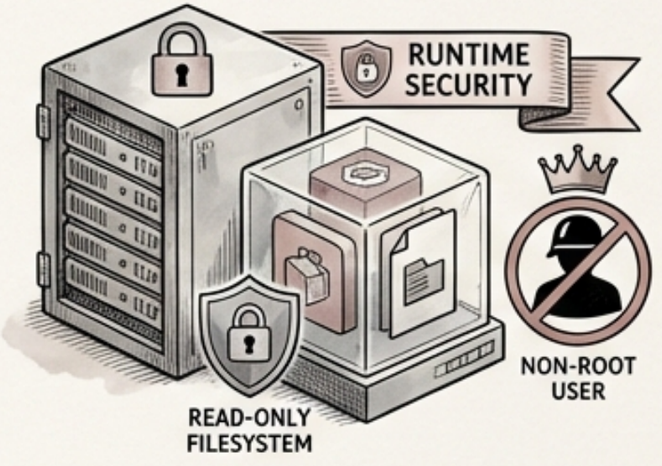
IMPLEMENTING ARTIFACT INTEGRITY: A PRACTICAL CHECKLIST FOR DEVELOPERS

- Integrate SBOM generation into your CI/CD pipeline using tools like Syft or cdxgen.
- Sign your container images using Cosign for keyless signing and enhanced security.
- Scan container images for vulnerabilities using tools like Trivy and implement automated remediation processes.
- Enforce immutability for production tags in your artifact registry.
- Implement runtime security measures like read-only filesystems and non-root users for containerized applications.









Securing Your AI-Augmented Future: Continuous Vigilance is Key

- Artifact integrity is critical for maintaining a secure and reliable software supply chain, especially with AI integration.
- SBOMs, AI BOMs, signing, attestation, and VEX are essential tools for ensuring trust and transparency.
- Automate security practices within your CI/CD pipeline for continuous monitoring and protection.
- Stay informed about emerging threats and best practices in software supply chain security.
- Continuous vigilance and proactive security measures are essential for mitigating risks and ensuring the integrity of your software.

Thank You

- Questions?