



POST-DEPLOYMENT OPEN-SOURCE VULNERABILITY DEFENSE

From SBOM Inventory to the Production Endpoint

Whitepaper Audience: CISOs, Platform Engineering Leaders, DevSecOps Architects, Open-Source Program Offices, Government, and Regulated Enterprises

Executive Summary

Modern software is built from hundreds or thousands of open-source packages, dependencies, containers, and microservices. Security teams have invested heavily in static analysis, software composition analysis, container scanning, and pipeline gates. Those controls remain essential, but they answer a point-in-time question: was this software acceptable when it was built?

The harder problem appears after release. A package that had no known vulnerability at build time can become critically vulnerable days, months, or years later. Pre-deployment scans can also create large volumes of findings without showing which vulnerable package versions actually reached production. Post-deployment agent-based scanning can rediscover that inventory, but installing and maintaining agents across every server, container, cluster, endpoint, and environment can be **expensive and operationally heavy**.

Know what's vulnerable. Know exactly where it's running.

Ortelius connects the software inventory in Software Bill of Materials reports (SBOMs) with Helm and deployment metadata so teams can map an open-source package and version to the artifact, application, environment, and production endpoint where it is deployed.

Ortelius is an open-source post-deployment vulnerability management platform. It consumes SPDX and CycloneDX SBOMs, can use Syft when an SBOM does not exist, preserves build-to-deploy relationships, and continuously evaluates deployed package inventory against OSV.dev. When a new CVE appears, teams can rapidly determine whether they use the affected package, which artifact contains it, which application depends on it, and where that exact version is running.

Finding a CVE Is Easy. Finding Where It Is Running Is Hard.

Traditional DevSecOps controls are optimized for the build and release process. They help prevent known vulnerabilities from entering software, but they cannot predict vulnerabilities that have not yet been disclosed. The application may not change after deployment; the threat does.

This creates a visibility gap. Organizations often know what they intended to deploy and what scanners reported at build time, but they may not retain a continuously queryable relationship between open-source package versions and the endpoints that ultimately received them.

Don't Rediscover Your Software. Remember What You Deployed.

Your CI/CD pipeline already knows what software was built and where it was deployed. Ortelius captures and preserves that evidence using a digital twin. Instead of starting a new discovery exercise every time a CVE is published, Ortelius maintains the software relationships that were created as software moved through the delivery pipeline.

The SBOM identifies the open-source packages and versions inside the software. Helm charts and deployment metadata identify the artifact, environment, and endpoint receiving that software. Ortelius connects the two.

The operational question

An SBOM tells you what is inside the software. Deployment evidence tells you where that software went. Ortelius combines both so security teams can move from CVE identification to production impact.

When a New CVE Drops, Ortelius Answers Four Questions

Do we use it? Search the SBOM inventory for the affected package and version.

Which artifact is impacted? Map the package and dependency to the exact component or artifact version.

Where is that version running? Use Helm and deployment metadata to connect that artifact to deployed environments and endpoints.

What needs attention first? Provide package, version, artifact, deployment, endpoint, and ownership context so teams can prioritize real exposure.

Collect. Map. Monitor. Pinpoint.

1. Collect the Software Inventory

Ortelius consumes standard SBOM formats including SPDX and CycloneDX. If an SBOM does not exist, Ortelius can use the open-source Syft tool to generate one. Different teams can keep the development and security tools they already use while Ortelius provides a common inventory of packages, dependencies, components, and application versions.

2. Map Packages to Production

This is where Ortelius goes beyond basic SBOM storage. It maintains relationships between packages, component versions, artifacts, applications, deployments, environments, and endpoints. The result is a software digital twin: a living model of how software was assembled and where each release was deployed.

3. Monitor for Newly Disclosed Vulnerabilities

Software does not stop becoming vulnerable after it ships. Ortelius continuously cross-references the packages in its software inventory against OSV.dev. The SBOM becomes more than a compliance artifact; it becomes the foundation for continuous post-deployment vulnerability monitoring. Ortelius is designed to surface newly disclosed vulnerability relationships within roughly 10 minutes of reporting.

4. Pinpoint the Blast Radius

When a vulnerability is disclosed, knowing the package name is only the beginning. Ortelius traces the affected package through the delivery chain so teams can identify the exact artifact, application, deployment, and endpoint that may require action.

Instead of: CVE → Start Investigation

Ortelius provides: CVE → Affected Package + Version → Artifact → Application → Deployment → Endpoint

A Digital Twin for the Live Software System

A software digital twin is a continuously maintained representation of the relationships that make up a deployed application. Ortelius builds this model from evidence already produced by software delivery: SBOMs, source and build metadata, artifact information, Helm charts, and deployment records.

Because the relationships are captured as software moves through CI/CD, Ortelius does not need to rediscover the entire software inventory from each production endpoint every time a new vulnerability appears. The model preserves build-to-deploy lineage and makes package-to-runtime relationships searchable over time.

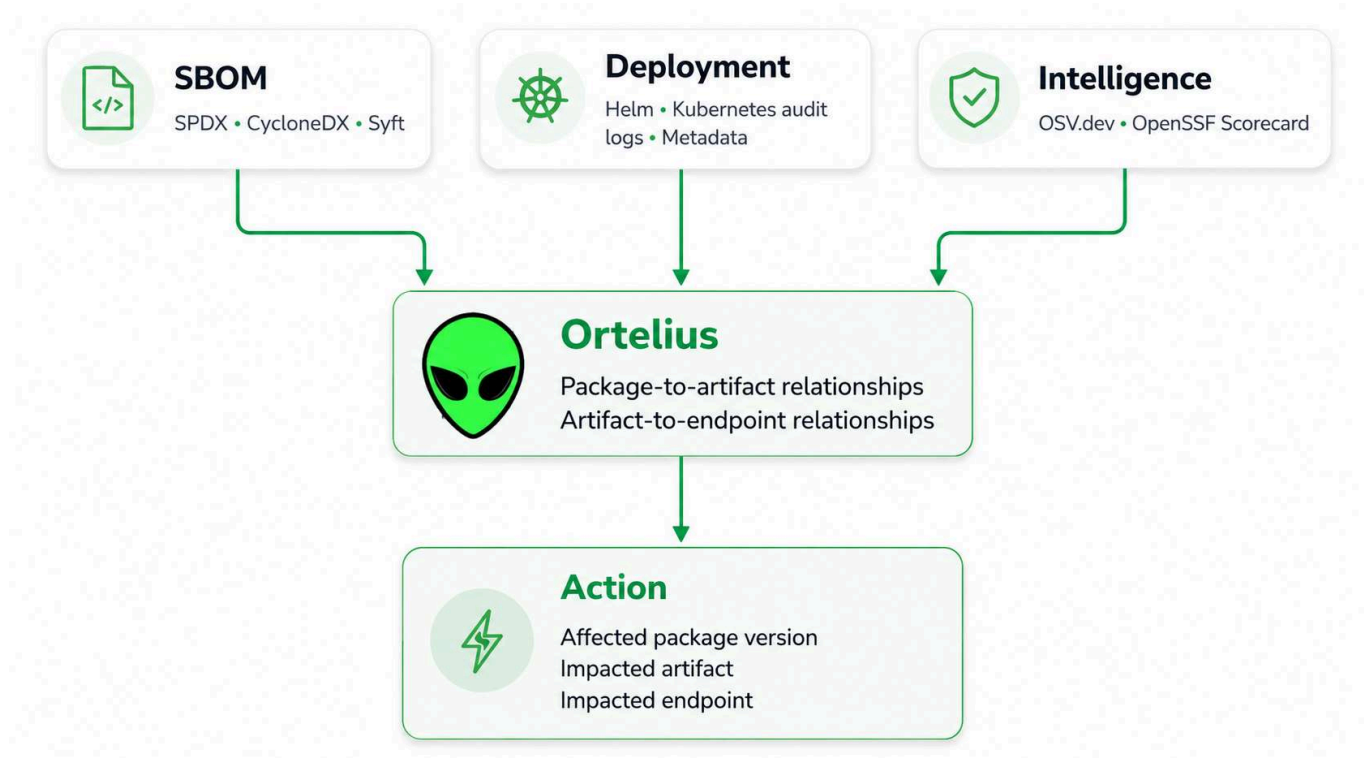


Figure 1. Ortelius connects SBOM inventory, deployment evidence, and vulnerability intelligence to map vulnerable package versions to production endpoints.

End-to-End Lineage and Provenance

Ortelius can trace software from source and CI/CD workflows to artifacts and deployed runtime locations while preserving SBOM provenance and release lineage. That evidence supports incident response, change governance, audit, and the practical question security teams face during a critical disclosure: what is actually affected in production?

Continuous Security and Compliance After Deployment

Compliance is often evaluated at discrete points in the software pipeline, but software risk changes after deployment. Ortelius keeps **security context attached** to the software that is actually running so organizations can maintain evidence and prioritize live-system exposure rather than relying solely on historical scan results.

OpenSSF Scorecard Context

OpenSSF Scorecard provides signals about the security practices of open-source projects. Ortelius can **federate Scorecard information** with component and application inventory so teams can identify where deployed systems depend on open-source projects with weaker security posture and incorporate that information into risk decisions.

Support for NIST and Continuous Authorization

Ortelius is designed to support software inventory, SBOM reporting, build-to-deploy traceability, continuous vulnerability monitoring, and evidence needed by organizations working with NIST 800-53, NIST 800-218, NIST 800-171, Zero Trust programs, and continuous Authority to Operate (cATO) processes. Ortelius does not replace an organization's governance or authorization process; it supplies **continuously maintained security evidence** that can support it.

From Detection to Remediation

Visibility alone does not reduce risk. Ortelius gives development, security, and platform teams the context needed to act: the affected package and version, the artifact that contains it, the application that consumes it, the deployment where it is running, and the endpoint that may require remediation.

That context can feed existing issue, pull-request, CI/CD, and operational workflows. Ortelius is intended to **augment existing security and delivery investments**, not replace them. Pre-deployment tools reduce the chance of shipping known problems; Ortelius closes the loop by continuously watching the software after it has shipped.

Conclusion

Pre-deployment security asks: Was the software safe when it was built?

Post-deployment vulnerability defense asks: Is the software affected by a newly disclosed vulnerability now—and exactly where is that software running?

Ortelius turns SBOMs into operational security data. By combining software inventory with Helm and deployment metadata, Ortelius maintains the relationships between open-source packages and production endpoints. It then continuously monitors that inventory for newly disclosed vulnerabilities so teams do not have to rediscover their software estate every time the threat landscape changes.

The result is a clearer and more actionable post-deployment security model: collect the software inventory, map it to the runtime environment, monitor it continuously, and pinpoint the production systems that actually need attention.

About Ortelius

Ortelius is an open-source project focused on post-deployment vulnerability management and software supply chain visibility. It connects SBOMs, artifacts, applications, deployment metadata, environments, and endpoints to provide a living view of where open-source package versions are running and how newly disclosed vulnerabilities affect live systems.

Ortelius supports SPDX and CycloneDX SBOMs, can generate missing SBOMs with Syft, continuously correlates package inventory with OSV.dev, and federates security context such as OpenSSF Scorecard data. The project is built for teams that need to understand open-source risk from code to cloud—and, most importantly, after deployment.

Learn more: ortelius.io | github.com/ortelius | docs.ortelius.io