resolved.

The Backporting (Mini) Blueprint

What Security Scanners Miss, and How Engineering Teams Can Take Back Control

A no-nonsense guide to using backporting in open source security - so you can fix vulnerabilities without breaking everything.

Today, the only way to fix a vulnerability in an open-source dependency is to upgrade the package version to one that contains the fix, usually the latest and greatest. But dependency upgrades often introduce breaking changes. Backporting changes that. This guide shows you how.

The Mess We're In

We live in an Open-Source world. But what happens when a vulnerability is discovered? Security teams tell you to fix it, but this dependency is disruptive:

- The new version brings in API changes and behavior shifts, forcing you to change our own code
- What seems like a small version bump in one library can cascade into a chain reaction of changes across your codebase
- It is almost impossible to vet all the new version of the upgraded version and the impact on your codebase

The real problem:

The real problem: Dependency upgrades are manual, costly, and bring in uncontrolled risk. No wonder developers refrain from doing them at scale, the scale security requires.

You waste time upgrading dependencies

You face impossible choices between security and stability

Your deployment cycles can not accommodate upgrades in the pace you need

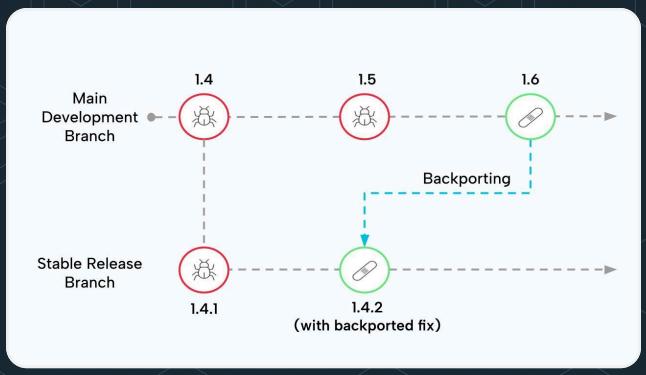
Engineering get burned on mundane work

> Organizations spend over **5% of their R&D** budget on dependency upgrades to **fix only 10%** of their security goals. (Resolved Security research)

What is Backporting, Really?

Definition:

Backporting is the process of taking a fix or a feature that was made in a newer version of a software project and applying it to an older version. This is commonly done in software development for the sake of stability, security, or compatibility - especially when users can't upgrade to the latest version due to compatibility or operational constraints.



Examples:

- **Linux vendors** like Red Hat, Ubuntu, and Debian have been doing it for decades. They routinely backport security patches into older package versions they support - that's why the same version of OpenSSL can be secure on both RHEL 7 and RHEL 9, even if it looks old.
- **Large tech** companies often maintain internal forks of critical open source libraries, applying patches independently of the upstream maintainers.

Why We Believe in Backporting

Fix precisely
Target the exact vulnerability, not whole dependency chains

Reduce test cycles
Smaller changes = fewer regressions

Ship faster
No waiting for full upgrade validation

Bridge security-dev gap
"Yes we fixed it, no we didn't upgrade"

Real-world impact metrics:

80% reduction
in security-related deployment delays

MTTR
improved from days/weeks to hours

Engineering
efforts spent on dependency management reduced to minutes

> Backporting transforms your approach from "update everything and pray" to "solve it with surgical precision with minimal risk."

Implementing Backporting with Resolved

Core Steps

STEP 1

Map & Identify

Resolved scans your codebase and provides a secure twin for every vulnerable dependency - no manual work needed.

STEP 2

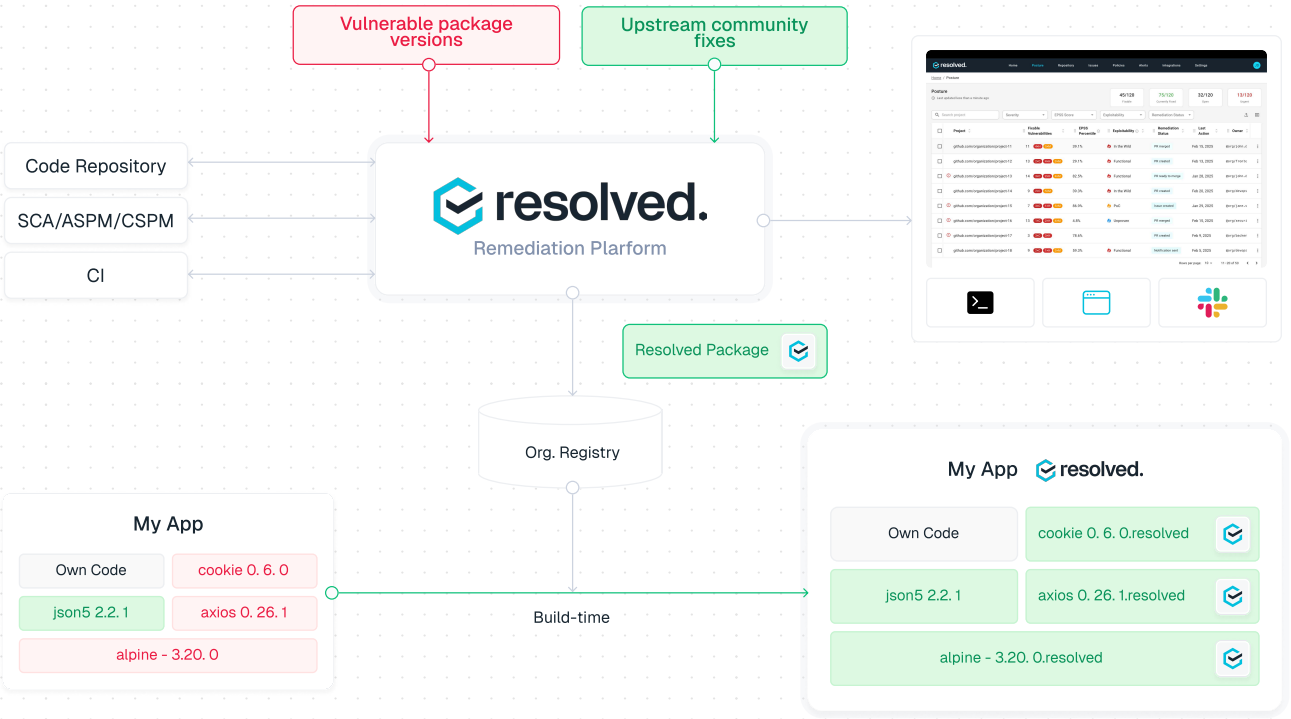
Fix without Breaking

Automated workflows seamlessly update the application to use secure versions of packages, while maintaining full compatibility.

STEP 3

Ship with Confidence

Test and deploy your application with Resolved secure twins, knowing your code is both secure and stable.



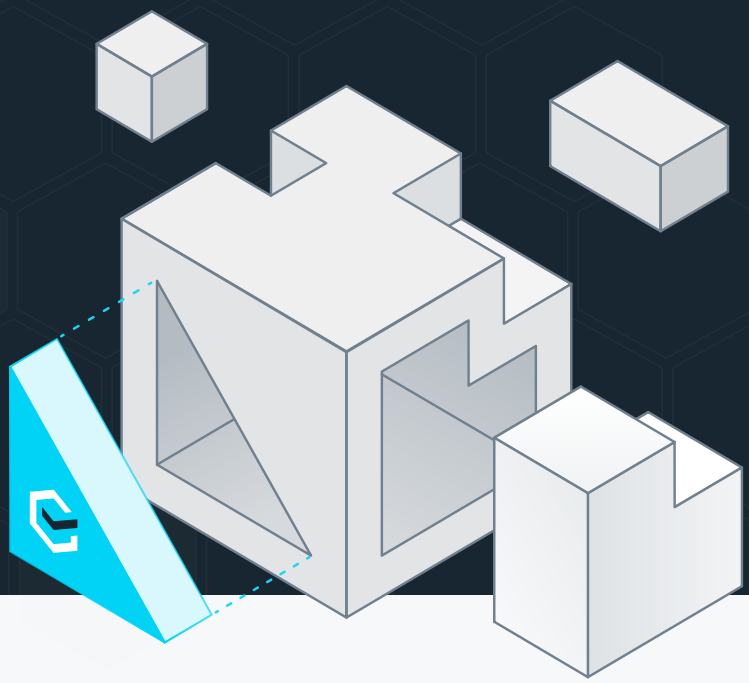
Jacob Avidar
VP R&D and CISO

PLAXIDITYX

Handling open source risks was a constant uphill battle - no matter our efforts, we couldn't get ahead. With Resolved Security what felt impossible is now automatic and seamless, finally allowing us to manage vulnerabilities at scale.

resolved.

Finding vulnerabilities is easy; fixing them is hard.



Resolved Open-source Security Platform automatically fixes vulnerabilities without slowing development. Finding issues is easy - fixing them without breaking things is what matters. Our solution eliminates remediation overhead with no upgrades required and zero release disruption. We deliver: near-zero engineering effort, scalable automation, and a validated approach that satisfies both security compliance and engineering stability requirements.

The Bottom Line

Backporting fundamentally changes your security posture by separating crucial fixes from disruptive upgrades. Engineering teams regain control of their deployment cycles while still addressing critical vulnerabilities. The real power comes from precision - surgical fixes that target exactly what matters without the overhead of full dependency chain updates.

Stop letting version numbers dictate your security response. Fix what matters, when it matters, how it matters.

Contact our engineering team at hello@resolvedsecurity.com to schedule a technical consultation.