1. Only fail the build when a fix is available

**Challenge:** Developers waste time working on security vulnerabilities with no fix, or of low severity.

**Solution:** Reduce the noise by failing the build for high-severity thresholds, and only for when a fix is available.

Help developers embrace security and handle failures without dismissing them as noise and false positives. Empower them with tools that are helpful and actionable to their workflows.

2. Find, categorize, and prioritize projects based on their business impact

**Challenge:** When trying to manage thousands of different projects, each with thousands of security issues, it’s easy to get lost or become overwhelmed. Even finding a project can become a more difficult task, not to mention understanding whether the project is among those that need your urgent attention.

**Solution:** Organize your projects in a standardized way by associating business impact and tech stack metadata to them. This makes it easier to search and find projects, as well as focusing your efforts on specific projects based on their deployment environment, lifecycle stage, priority score and any other characteristics that are important to your organization.

3. Prioritize the most urgent and fixable issues with pull requests

**Challenge:** Developers genuinely want to fix security issues but if you keep spamming them with non-actionable reports, false positives, or opening pull requests for every vulnerability—including low-severity ones or a development dependency—they will quickly lose interest in solving this pile of issues.

**Solution:** Reduce the overall risk by prioritizing first the most significant security vulnerabilities which can be fixed. Drive developers to act on the most important security vulnerabilities—those that have a mature exploit in the wild and are fixable.

4. Customize license compliance policies

**Challenge:** You struggle with legal requirements, such as obtaining licenses and copyrights for all of your open source packages across the org, but even further, putting guard rails to ensure developers don’t violate license and copyright laws unknowingly which oftentimes creep up via indirect dependencies.

**Solution:** Snyk helps you address enterprise security compliance issues by providing you an exportable report of licenses usage across all of your projects, and setting your own custom compliance policies which are enforced via continuous integration such as status checks on pull requests.

5. Prioritize vulnerabilities reachable by your own code*

**Challenge:** Developers are frustrated with false positive alerts for vulnerabilities in dependencies that are actually not being used in their code or in the upstream package.

**Solution:** Snyk uses call graph analysis to determine if a security vulnerability in a third-party open source library you’re using is actually reachable from your own code. This helps you prioritize and fix higher for those vulnerabilities!

---

*SUPPORTED FOR JAVA MAVEN AND GRADLE PROJECTS FROM THE CLI, YOU CAN USE IT AS FOLLOWS: snyk test --reachable

**@liran_tal**
Node.js Security WG & Developer Advocate at Snyk