

KONVU VS CHECKMARX

Checkmarx detects and triages. Konvu proves and fixes.

Keep Checkmarx One for detection, CxQL, the IDE and AI Triage, which proposes Not Exploitable for a reviewer to confirm. Its Exploitable Path covers Python, Java, JavaScript and C#, from the most recent full SAST scan, for packages called directly, not through reflection. Add Konvu for evidence per condition, a reproduced exploit and a shipped fix, and for Go, Ruby, PHP and Rust.

Konvu is not a scanner. It reads results your pipeline sends; you mark them in Checkmarx.

AT A GLANCE

	CHECKMARX	KONVU
Question it answers	Is the finding reachable, exploitable and unmitigated?	Can an attacker use this finding in this deployment?
Reachability languages	Exploitable Path (SCA): Python, Java, JavaScript, C#. Full SAST scan, direct calls, no reflection.	Java/Scala/Kotlin, Python, Ruby, JS/TS, C#, Go, PHP, Rust; builds its own dependency graph
What it checks	AI Triage: reachability, exploitability, masking and sanitization. Exploitable Path: call path to the package	SCA: installed, environment, reachable, attacker-controllable, with evidence. Both: sandbox reproduction
Verdict	Confirmed, or Proposed Not Exploitable for a reviewer to keep or change	SCA: Exploitable, Needs input. SAST: Reproduced, Hardening. Both: False positive, Inconclusive
Fixes	Fix PRs for GitHub Code Repository Integration projects, in-platform remediation for other SCMs	Draft PRs that run your tests, or a coding-agent plan; exploit re-run on the fix
Write-back	States, notes and change log in Checkmarx One	Your team marks it in Checkmarx. Jira tickets on Enterprise
Pricing	Quote per package: modules, deployment model, developers in scope. AI Triage runs use credits	\$2,000/month for 400 confident verdicts; inconclusive runs free. API and CLI on Enterprise

Where Checkmarx is stronger

- Detection across code, dependencies, secrets, IaC, APIs, containers and AI components.
- AI Triage with reviewer sign-off on every dismissal.
- CxQL queries and Developer Assist in the IDE.
- SaaS, self-hosted or air-gapped deployment.

Where Konvu fits

- Dismissals that need evidence for each condition, including attacker-controlled input and configuration.
- Go, Ruby, PHP, Rust, Scala and Kotlin dependencies.
- Findings reproduced, and fixes verified against the exploit.
- Findings from other scanners as well as Checkmarx One.

RUNNING BOTH

Checkmarx keeps the record. Konvu adds proof.

01

Pipeline sends results

The pipeline posts results to the Konvu API (Enterprise), or you upload SAST exports.

02

Konvu decides

Evidence per condition, and the exploit reproduced in a sandbox.

03

Your team marks it

Your team marks each result in Checkmarx, with Konvu's evidence in the note.

Detection is solved. The decision is not.

Full comparison with every source: konvu.com/compare/konvu-vs-checkmarx

Konvu docs: docs.konvu.com

Sources checked October 13, 2026: docs.checkmarx.com (AI Triage, Exploitable Path, triaging SAST results, Feedback Apps), checkmarx.com (platform, Triage & Remediation, Assist, pricing, blog, press releases), docs.konvu.com, konvu.com/pricing, konvu.com/integrations/checkmarx.