SonarQube Hunter Agent

The AI security agent that hunts logic flaws, and why it matters for your customers.

Not all vulnerabilities are bugs in your code.

Attackers don't need to break your code, they can take advantage of how the logic behaves in ways you didn’t intend.

What is SonarQube Hunter Agent

Hunter Agent is a high-fidelity AI security agent, native to SonarQube, that reasons through code the way a human security researcher works through a manual audit. It runs as an on-demand background agent, uncovering the broken access control, business logic, and authentication flaws that pattern-based scanners are not built to catch, and it surfaces confirmed findings directly as SonarQube issues rather than in a separate tool or portal.

Built on the Sonar Foundation Agent, based on the harness used in a leading SWE-bench agent, with a model-agnostic Foundation Agent underneath it. It runs a structured, multi-step playbook, Analyze, Explore, Validate Exploitability, and Synthesize, and it is language agnostic across the 40+ languages SonarQube Cloud supports.

What does it solve

Scanners read what your code does. They cannot tell whether it should. The worst flaws live in the gap between what code is meant to do and what it actually does. For years, that gap was covered by people: manual code review by senior engineers, quarterly or annual penetration tests, bug bounty programs, and pre release security audits.

That model is running out of runway. AI agents now write more than 40% of committed code, AppSec review talent is scarce and expensive, point-in-time pentests leave gaps between runs, and release velocity has outpaced what manual review can keep up with. Hunter Agent runs that manual review depth reasoning on every project, automatically, closing the gap before it turns into a breach.

What it finds, and why it matters

Every category below has already caused a headline breach in the last two years, not a hypothetical, a real incident with a real cost.

Broken access control
IDOR, missing auth, privilege escalation
64M applicants exposed, McDonald's, 2025

Business logic flaws
Workflow bypass, rate limit abuse, repeated actions
$9M drained in one exploit, Yearn Finance, 2025

Auth & session issues
Session fixation, missing MFA, weak recovery, CSRF
192.7M people affected, Change Healthcare, 2024

How it works, and its benefits

Hunter Agent runs as an on-demand background agent, not a one-time scan. Point it at a project and it takes over from there, reasoning through code the way a security researcher would, validating what it finds, and delivering confirmed issues where your team already works.

Four steps, zero setup

1. Connect
Point it at any SonarQube Cloud project. Nothing to install.

2. Reason
The security playbook follows code paths and identifies flows like a manual auditor.

3. Validate
A subagent confirms exploitability across feasible attack paths, removing false positives.

4. Raise
Confirmed flaws land in SonarQube Cloud as issues, tagged and ready to triage.

Why teams choose Hunter Agent

- **Find what traditional scanners cannot**
  Detects logic flaws like broken access control and IDOR before attackers do.
- **No new workflow**
  No new dashboard, no context switching. Findings arrive as SonarQube issues, ready to triage.
- **High precision**
  Every flagged finding has been verified: what surfaces is a confirmed risk worth investigating, not noise to sort through.
- **Independent by design**
  A different methodology reviews the code than the one that wrote it: auditable, consistent, repeatable.

How Hunter Agent complements SAST

SAST is built to catch flaws in how code is written. It cannot tell whether a request should be allowed. SonarQube's SAST engine finds structural flaws through pattern-matching and taint analysis, tracing data from untrusted sources to sensitive sinks in your first-party code. Hunter Agent picks up exactly where that design boundary sits.

- **What SAST is built to catch**
  Pattern, data-flow based and syntactic vulnerabilities like SQL injection, XSS, hardcoded secrets. These are flaws in how code is constructed. SAST is designed to identify them fast, consistently and at scale
- **What SAST is not designed to catch**
  Whether the logic itself should exist like broken access control, IDOR, privilege escalation. Nothing about these flaws looks wrong to a pattern matcher—the code runs exactly as written. Catching them requires reasoning about intent, roles and workflows the way an attacker would.
- **What Hunter Agent catches**
  It traces code paths and identifies flows the way a security researcher would, reasoning about what should be allowed to do what, and confirms each finding before it surfaces.

©2026, SONARSOURCE SÀRL