PENTESTERRA

CODE ANALYSIS

The Dangerous Part Is Rarely the Source Code

It is a postinstall script, a leaked token, a Git hook, an unpinned action, a hidden instruction in a config file. Pentesterra analyses all of it before you push - without your code ever leaving your machine.

Local
Source code never leaves the machine

Pre-push
IDE plugin, CLI, or CI/CD gate

Correlated
Code risk mapped to runtime exploitability

WHAT GETS ANALYSED

Supply Chain and Dependencies

Every dependency is checked against CVE and CISA KEV data, malicious-package intelligence, and typosquat patterns.

- → CVE / KEV mapping with exploitability context
- → Known-malicious and typosquatted package detection
- → Postinstall / lifecycle-script and install-behaviour review
- → SBOM generation and SARIF export

Secrets and Credential Flow

Secrets are found by pattern and entropy, then fingerprinted with a SHA-256 hash - never the value itself.

- → Pattern + entropy secret detection, values never transmitted
- → Token liveness and blast-radius assessment
- → Proxy-redirect and env-exposure credential flow checks
- → Commit-message and shell-history scanning

Repository, CI/CD, and IaC

Malicious repositories rarely look malicious in the source.

- → Git hook / config / filter / history inspection
- → CI/CD workflow injection and secret-exfil patterns
- → Terraform / IaC misconfiguration checks
- → SVN history exposure scanning

AI Toolchain and Logic

The developer surface now includes AI tooling - flagged and mapped to the OWASP LLM Top 10.

- → MCP / IDE-extension / agent-config risk
- → Prompt injection in .md / .json / .csv / .yaml data files
- → torch.load RCE, .pth runtime-hook, RAG SSRF checks
- → SAST-lite logic flaws and API-route auth issues

TRIAGE THAT ACTUALLY HOLDS UP

Push Gate, Not Just a Finding List

Every scan computes a server-side push gate - red, yellow, or green - so a developer knows instantly whether a push is safe.

- → A known-exploited (KEV) finding is always red, no exceptions
- → Dev-only, non-critical CVEs are excluded from the gate so it doesn't cry wolf
- → The gate decision is computed once, server-side, and reused everywhere

Composite Risk Score, Weighted by Real Exploitability

Not a flat count of findings - a weighted score that reflects what actually matters.

- → KEV-listed vulnerability: 30 points, malicious package: 25 points
- → Critical SAST finding: 22 points, exposed secret: 20 points, IDOR: 15 points
- → Dev-only findings are weighted at 0.2x so noise doesn't dominate the score

A Status That Never Silently Downgrades

Every finding is a persistent record, not a fresh guess on every scan.

- → Machine status - detected, verified, exploited - only ever moves forward
- → Malicious packages and confirmed threats are auto-verified on first sight
- → Diff mode shows exactly what's new, resolved, or unchanged since the last scan

Built for How Developers Actually Work

Triage that fits into review, not a separate security backlog nobody opens.

- → One-click "Not an issue" dismissal opens a pre-filled false-positive request
- → Suppressions can be scoped per-project or org-wide, with optional approval workflow
- → A plain-English AI summary turns findings into an action list, no extra LLM call needed

FAQ

Does my source code leave my machine?

No. The local agent collects only structured metadata - dependency types, paths, secret fingerprints (SHA-256, never the value), and endpoint metadata - and sends a redacted payload to the cloud engine. Source code never leaves the machine.

Where does it run?

As a standalone IDE plugin for VS Code, Cursor and Windsurf that a developer runs before pushing, as a CLI, or wired into CI/CD for automated gate enforcement. No pipeline integration is required to start.

How is this different from a normal SAST or SCA tool?

Most tools look at one layer - source code, or dependency versions. Pentesterra also inspects install-time behaviour, repository configuration, CI/CD, IaC, secrets flow, and AI toolchain risk, and correlates code-level risk with production exploitability from the rest of the platform.

What happens to a finding I've already triaged once I fix and re-scan?

Its machine status only ever moves forward - detected, then verified, then exploited - never silently back down, so a finding your team already investigated doesn't quietly reappear as new. Diff mode shows exactly what changed between scans.

Is there a free option?

Yes. DevGuard, which delivers this capability at the IDE and CLI level, is free to try with no CI/CD integration or account changes required.

Catch it before it merges

Pre-push code and supply chain analysis, privacy-first.

Request a Demo

info@pentesterra.com · pentesterra.com/code-analysis