SCANOSS

SIOS

Managing AI-generated code risks:

How SIOS turns the unknown into a process

KEY TAKEAWAYS

From speculation to evidence

AI-generated code risk moved from unmeasurable to data-driven and manageable.

Snippet-level detection in the pipeline

Winnowing fingerprinting catches OSS in AI-modified code that traditional SCA tools miss.

Human review focused where it counts

A file-size threshold routes only lower-confidence cases to manual verification.

THE SITUATION

AI-generated code created a compliance blind spot

SIOS Technology's PSSL division delivers open source–focused services—technical support, system integration, and product development—helping organisations deploy and operate open source technologies reliably in production. Internally, SIOS also builds its own software products using **Python, TypeScript, and Bicep**, primarily within a **monorepo** architecture.

Open source governance at SIOS is a shared responsibility. Engineers identify potential issues during development; and compliance is verified before release through internal processes. It is treated as both a legal obligation and a quality standard essential to maintaining customer trust.

Before adopting SCANOSS, open source management depended heavily on declared dependencies—such as those listed in package.json—and this worked well when dependency manifests accurately reflected the codebase. But the adoption of generative AI coding tools introduced a critical gap: open source code fragments could now enter the repository without ever appearing in a dependency file.

Engineering increasingly found themselves asking questions that traditional tooling couldn't answer: could AI-generated code include fragments from copyleft-licensed projects? The concern became concrete when they realised that even when generative AI modifies code — changing variable names, reorganising structure — the underlying OSS origin remains. Traditional tools, which rely on whole-file matching, cannot detect this. That gap triggered the search for a new approach.

THE SOLUTION

Snippet-level detection integrated into the CI/CD pipeline

SIOS adopted SCANOSS to analyse source code at the snippet level, a capability that proved essential for catching AI-generated code, which often differs only slightly from its original source.

The deciding factor was **SCANOSS's Winnowing-based fingerprinting (wfp)**, which detects partial matches and modified code fragments rather than requiring whole-file matches. This made it possible to identify situations where AI-generated code still originated from an open source project, even after variable names had been changed or structure had been reorganised. The open source nature of the detection logic also mattered: because the methodology is not a black box, the team could explain and justify results with confidence across teams — not just assert that the tool had flagged something.

SCANOSS was integrated directly into the CI/CD pipeline via GitHub Actions, ensuring that potential open source matches were surfaced early in development before late-stage compliance reviews.

Focusing analysis with SBOM ingestion

SIOS implemented SBOM ingestion as part of their workflow. By comparing declared open source dependencies against code actually present in the repository, the process highlights segments that aren't explained by declared dependencies. These segments—copied fragments, manually integrated snippets, or AI-generated code—become the focus of analysis, allowing teams to direct attention where it matters most rather than rescanning the entire codebase.

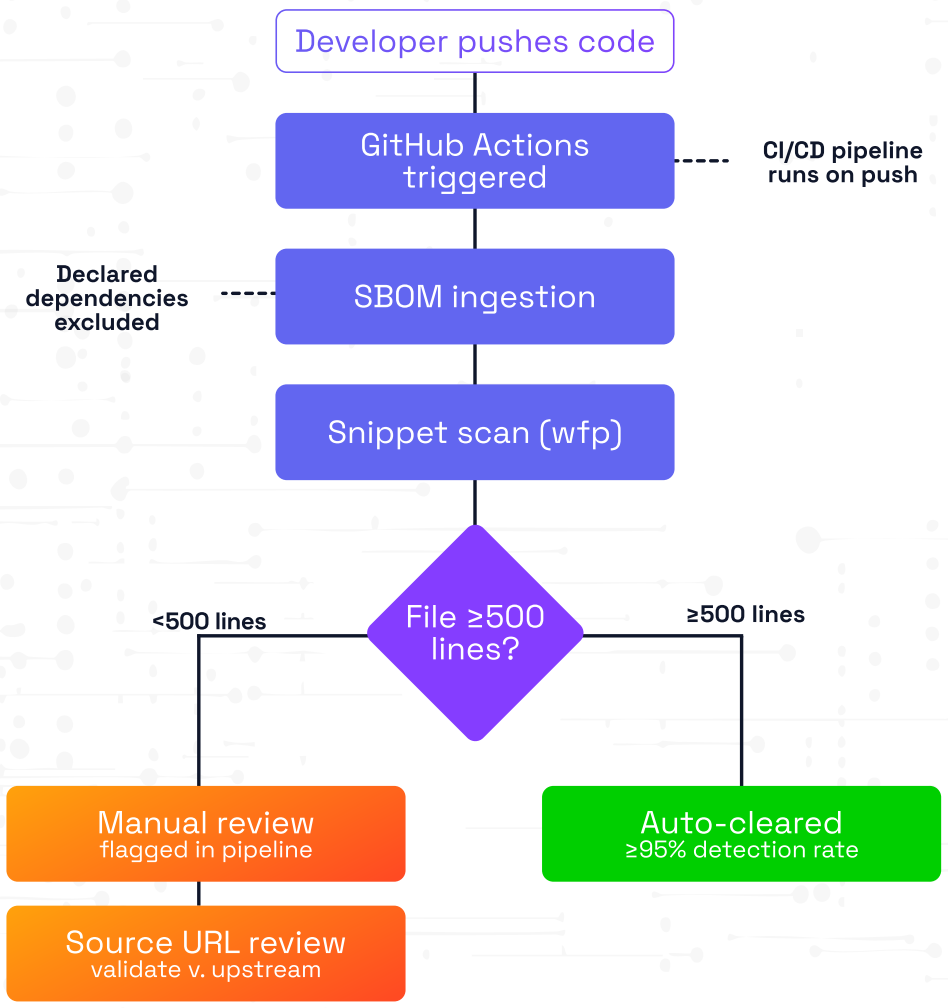
A practical threshold for human review

During internal validation, SIOS observed that detection accuracy correlates with file size: for files of 1,000 lines or more, accuracy exceeded 95%. For smaller files, the probability of misattribution increased noticeably—in one test involving a **442-line file, accuracy dropped to approximately 23%.**

Rather than treating this as a limitation, the team used the insight to design a targeted workflow rule: files **smaller than 500 lines automatically trigger a manual review** warning in the CI/CD pipeline. This ensures lower-confidence cases receive human verification without burdening developers unnecessarily on larger, higher-confidence files.

Source urls as an investigation tool

One of the most practical outputs for SIOS is the **Source URL included in SCANOSS results.** When a match appears, **engineers can immediately compare their local code against the upstream project** to determine whether the match represents genuine derivation or coincidental similarity, turning what was previously a speculative conversation into a direct, evidence-based review.



THE OUTCOME

from speculation to data-driven decision-making

Before SCANOSS, the risks introduced by AI-generated code were impossible to measure. Developers could only speculate about whether generated code might carry open source obligations.

After deployment and internal validation, SIOS documented the following:

- 95.8%–97.9% detection accuracy for files containing more than 1,000 lines
- 80%–99% detection accuracy when AI had made minor modifications such as variable name changes
- A clearly defined manual review threshold for smaller files, based on observed accuracy patterns rather than guesswork

The practical outcome is a two-tier workflow: automated detection handles the majority of scenarios, while targeted human review addresses edge cases. Engineering no longer debate whether AI introduces unacceptable risk—they can now identify precisely where automation is reliable and where human verification is required.

SCANOSS gave SIOS something more valuable than a compliance tool: a shared, measurable basis for responsible decisions about how generative AI is used in software development.

> **The risks associated with code development using generative AI changed from an invisible state to a manageable one.**