

Your SAST finds thousands of issues. Your team fixes a handful.

Detection was never the bottleneck. Triage is.

Xygeni AI SAST finds vulnerabilities and malicious code in your own code, then triages, explains and fixes them with AI, including the findings of the SAST you already run.

100% true positive rate on the OWASP Benchmark for the SQL injection and XSS categories.

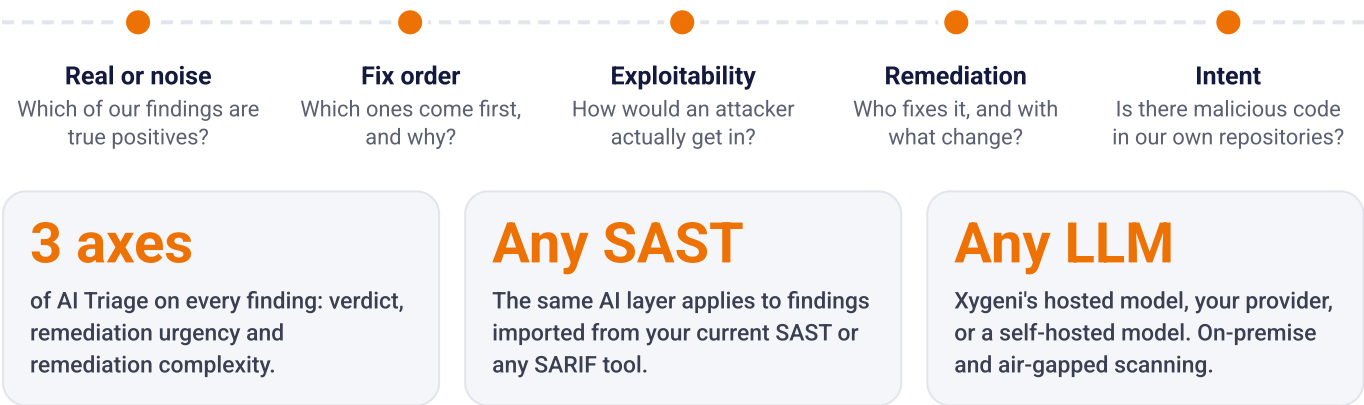
THE PROBLEM

Static analysis that nobody acts on is not a security control.

SAST programs drown in findings. Developers stop trusting the results, backlogs age and gates get bypassed. AI-assisted coding raises the volume of code, and supply-chain attacks hide malicious logic inside first-party code, where classic SAST never looks for intent. Replacing the incumbent tool is expensive and political, so the noise stays.

The answer is not another scanner. It is a deterministic engine you can trust, an AI layer that decides what is real and what comes first, and fixes that arrive as pull requests, on your findings and on the ones your current SAST already produces.

FIVE QUESTIONS MOST TEAMS CANNOT ANSWER TODAY



ONE PRODUCT, THREE STEPS

Detect. Triage. Fix.



Detect with determinism. **Triage** with AI. **Fix** with pull requests.

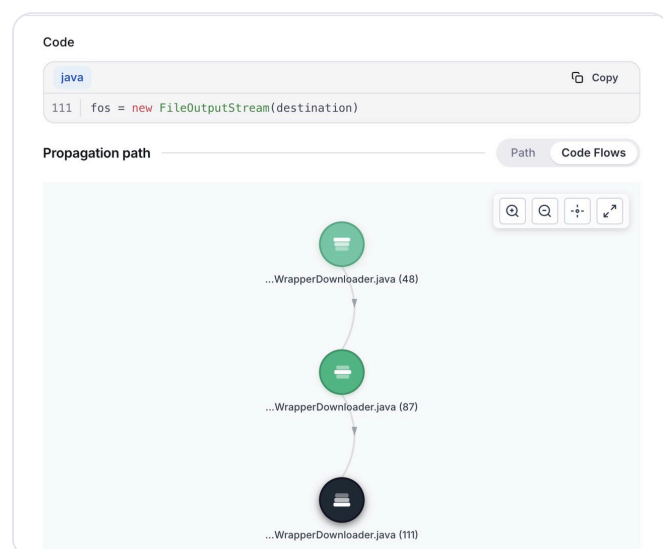
Without Xygeni	With Xygeni
Thousands of findings, no clear order	Verdict, urgency and complexity per finding
"Why is this exploitable?" debated in tickets	Source-to-sink explanation in the finding
Fixes stall in the backlog	Fix pull requests per file, in bulk
Replace your SAST to cut the noise	Keep it: its findings get the same AI layer

DETECT · NATIVE ENGINE

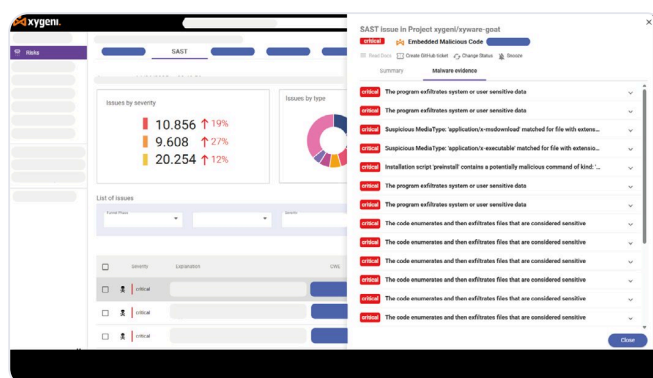
Find the flaws, and the intent, in your own code

Deterministic taint analysis, malicious code detection, broad and growing language coverage.

Xygeni AI SAST analyzes source code before build or deployment with a native inter-procedural taint engine that follows untrusted input from source to sink. It also looks for intent: backdoors, obfuscated execution and data exfiltration hidden in first-party code, the patterns supply-chain attacks rely on.



Code Flow: the propagation path of a path traversal finding, from the command-line argument to the file write.



Embedded malicious code: each finding lists the evidence, such as exfiltration of sensitive data.

- Injection and XSS:** SQL, command, code, LDAP and XPath injection, plus reflected and stored XSS.
- Weak cryptography:** weak algorithms, fixed IVs, insecure randomness, and PQC-aware crypto rules.
- Templates and mobile:** JavaScript inside HTML, EJS and PHP templates, plus Swift and Kotlin.
- Deserialization, XXE and SSRF:** insecure parsers and object handling across languages.
- Malicious code (CWE-506):** backdoors, trojans, obfuscated execution, data exfiltration and registry tampering.
- Languages:** native taint analysis for Java, JavaScript / TypeScript, Python, C#, Go, PHP, Kotlin, Swift and HTML, plus curated rule packs for C/C++, Rust, Scala, Dart, Objective-C and VB.NET.

OWASP Top 10 2025

OWASP ASVS

OWASP MASVS

CWE

Custom detectors

Incremental scans

For the AppSec lead: findings tied to the exact path an attacker would follow, not a pattern match.

TRIAGE · AI LAYER

From thousands of findings to the few worth fixing first

Three-axis AI Triage, funnels and memory. On your findings and on everyone else's.

AI Triage reads each finding in its code context and returns a verdict, a remediation urgency and a remediation complexity, with the reasoning behind them. Funnels use those results to remove AI-confirmed false positives before a human looks, and AI Memories capture why the team muted, accepted or fixed something, so the next decision starts from it.

SAST issue in Project demoxygeni/WebGoat

Critical Improper limitation of a pathname to a restricted directory **CWE-22** Actions

AI triage detected a real vulnerability

- True positive
- Urgency: Planned
- Complexity: Trivial

The flow is present: main receives the externally controlled CLI value args[0], constructs baseDirectory with new File(args[0]), and uses baseDirectory.getAbsolutePath() as the...

Show more

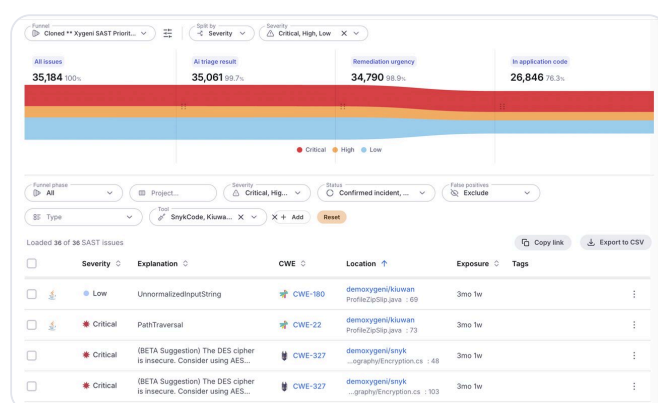
Automatically generated with AI and may contain errors. Please review.

Explanation Improper limitation of a pathname to a restricted directory

Type path resolution

Where main

AI Triage on a path traversal finding: true positive, planned urgency, trivial complexity, with its reasoning.



Findings imported from other SAST tools, triaged and prioritized in the same funnel as Xygeni's.

- Verdict:** potential true positive, potential false positive or needs review, with a reasoning trace.
- Complexity:** from trivial to hard, so effort can be planned before work starts.
- AI Memories:** past decisions captured at global or project scope and reused by AI remediation.
- Urgency:** immediate, next sprint, planned or backlog, from impact and exploitability.
- AI stages in the funnel:** AI Triage result and remediation urgency, on by default for new projects.
- Third-party findings:** results imported from other SAST tools and any SARIF source get the same triage.

Single or bulk triage

On demand

Overridable verdicts

Labeled AI output

AI advises. Deterministic engines decide. Triage runs on demand or in bulk, every verdict carries its reasoning and can be overridden, and AI output is labeled. What blocks a build is still decided by the engine and your guardrails.

For the CISO: a backlog the team can actually burn down, with an audit trail for every decision.

FIX · REMEDIATION

Explanations developers trust. Fixes they only have to review.

AI Explanation, AI Auto-fix and bulk fix pull requests, from the platform to the IDE.

Developers push back on findings without context. AI Explanation reasons over the complete code flow to show how an attacker gets in, how to reproduce it and how to validate the fix. AI Auto-fix then proposes the change and opens a pull request after review, one finding at a time or in bulk from the Remediation View.

Path attack explanation

The reported write operation, `fos = new FileOutputStream(destination)`, receives a destination derived from the command-line argument `args[0]` without a demonstrated restriction to an approved base directory. This is a path-traversal/path-control finding: an actor able to control the launcher arguments may direct the downloader to write its downloaded content outside the intended project location, subject to the process's filesystem permissions. The rule documentation identifies this as CWE-22, where externally controlled paths can lead to unintended file reads or writes.

How the vulnerability occurs

- `main(String[] args)` receives external command-line input through `args[]` at line 48. The analysis identifies `args[0]` as the untrusted source.
- At line 50, the application constructs `new File(args[0])`, assigning attacker-influenced input to `baseDirectory`.
- The application obtains `baseDirectory.getAbsolutePath()` at line 78. Converting a path to an absolute path does not establish that it belongs to a trusted or restricted directory.
- At line 87, the resulting `outputFile` is passed to `downloadFileFromURL(url, outputFile)` as the `destination` parameter.
- Inside `downloadFileFromURL`, line 111 opens the supplied `destination` for writing with `new FileOutputStream(destination)`. No path canonicalization-and-containment check is shown between the externally influenced argument and the filesystem write.

AI attack-path explanation: how the vulnerability occurs, step by step, and the exploit scenario.

Remediation View: fixes grouped by file, with complexity, effort and estimated time, and one action to create fix PRs.

- Explanation over the full flow:** every source to the sink, cached per scan and reused by triage and fixes.
- Remediation View:** issues grouped by file, with complexity, effort and estimated time.
- Xygeni Bot:** remediation on every PR or on a schedule, filtered by severity, language and rule.
- Fix Vulnerability:** a reviewed diff and a pull request, never applied without consent.
- Bulk fix PRs** across SAST, SCA, IaC, CI/CD and Code Quality from one view.
- In the IDE:** findings on save, Code Flow, AI Explanation and fixes, plus an MCP server for AI coding agents.

Correlated

SAST findings joined to the API endpoint that reaches them and to DAST evidence.

Governed

Policies, guardrails and PR gating, with mute rules that require a justification.

Sovereign

SaaS, on-premise or air-gapped scanning, with any LLM, including self-hosted models.

For VP Engineering: fewer blocked pull requests, and fixes that arrive ready to review.

WHY XYGENI FOR AI SAST

Detect with determinism. Triage with AI. Fix with pull requests.

Xygeni AI SAST combines a native taint engine and malicious code detection with an AI layer that triages, explains and fixes, on its own findings and on the findings of the tools you already run.

Deterministic detection

A native taint engine with an explainable propagation path. AI never decides what blocks a build.

Three-axis triage

Verdict, urgency and complexity per finding, so the backlog becomes a plan.

Intent, not just bugs

Malicious code in first-party code, detected next to vulnerabilities.

Fixes as pull requests

AI Auto-fix, Remediation View and Xygeni Bot, from one finding to the whole backlog.

Why security leaders choose Xygeni

Proven in production

Banco Serfinanza reports around 90% less time identifying, prioritizing and assigning findings; Fintonic, a 90% reduction in prioritization time.

One platform

Code, open source, secrets, IaC, pipelines, running apps and AI in one inventory, one risk model and one workflow.

Extend, don't replace

Xygeni's AI layer triages, explains and fixes findings from the security tools you already run.

Built for regulated environments

European vendor. SaaS, on-premise or air-gapped scanning, with any LLM, including self-hosted models.



The AI SAST that finds what matters, explains why, and fixes it

Detect. Triage. Fix. One platform.

[Request a PoC](#)

[Book a demo](#)

- Keep your current SAST: Xygeni triages its findings
- SaaS, on-premise or air-gapped scanning
- xygeni.io

RECOGNIZED IN 2026



Hot Company in Application Security Posture Management 2026

Global InfoSec Awards · Cyber Defense Magazine