

Find what's exploitable. Fix it 10x faster.

Legit's native scanners — SAST, SCA, secrets, IaC, and pipeline — cut through alert noise with reachability analysis and AI-aware detection. Teams stop triaging theoretical risk and start fixing what's actually exploitable, wherever it lives: in the code, the dependencies, the pipeline, or the build.

Go fast. We've got you.



SAST



SCA



SECRETS



IAC



PIPELINE

Signal, not volume.



SCA

Reachability cuts through dependency noise

Legit maps every vulnerable dependency to the code paths that actually call it, so a CVE buried in an unused transitive package stops competing with the one your application really executes.

Full instruction path

Traces direct and transitive dependencies end to end, so the real introduction point is visible.

Reachability

Confirms the vulnerable function is actually called before it reaches a developer's queue.

Container correlation

Combines with container scanning to show full exposure across code and image layers.

Issues > 5106837703

Critical CVE-2025-48734 • commons-beanutils:commons-beanutils 1.9.4

Vulnerable dependencies were detected. It is strongly recommended to remediate it

Snooze Closing options Set assignee

Vulnerable dependency commons-beanutils:commons-beanutils@1.9.4

Description Apache Commons Improper Access Control vulnerability

Manifest file war/pom.xml

Fixed versions 1.11.0

Reachability Reachable

Reachable usage

```
<> .../util/ReflectionUtils.java
77 return PropertyUtils.getProperty(o, p);
```

Dependency path

Introduced through

Manifest file: war/pom.xml

org.jenkins-ci.main:jenkins-war@2.440-SNAPSHOT



SAST

Purpose-built detection for AI-generated code

Native scanning tuned for how AI writes code — LLM and agentic patterns, insecure defaults inherited from generated scaffolding, and the repeated flaws that arrive at machine speed.

Broad language support

One scanner across the stack — Java, Python, JavaScript/TypeScript, Go, C#, Ruby, PHP, C/C++.

AI-specific risk

Detects risk introduced by AI usage itself — LLM calls, agentic patterns, generated defaults.

Agentic remediation

Opens a fix PR for the developer instead of handing back another ticket to triage.

Critical

LLM Output Code Injection

SAST

LLM output passed to exec/eval without validation, enabling arbitrary code execution on the server.

Snooze

Closing options

Set assignee

```
value = ast.literal_eval(content)
```

Risk Score

90

Context

Using AI

Exposing API

Data Categories

--

Security Frameworks

--

External Services

Anthropic

OpenAI

Findings

<> llm_output_vulnerabilities.py

See code

```
39 result = eval(response.choices[0].message.content)
```

Rule ID

AdvancedRules.Python.LLM_Output_Code_Injection.python_llm_rule-output-code-injection-openai



SECRETS

Catch secrets before they ever reach a public repo

Legit scans every commit, branch, and CI/CD log for live credentials, API keys, and tokens — then checks validity in real time, so teams triage what's actually exploitable, not just what matches a pattern.

95%+

false-positive reduction, validated in secret scanning.

Validity checking

Confirms whether a detected secret is live, expired, or already rotated, so triage starts with what's exploitable.

Pre-commit and pre-push blocking

Stops a secret from ever reaching a shared branch, with real-time enforcement at the point of push.

Full history coverage

Scans commit history and CI/CD logs, not just the latest diff, so legacy exposure gets found too.



IAC

Fix misconfiguration before it is ever provisioned

Legit analyzes infrastructure-as-code files — Terraform, Helm, CloudFormation and more — to identify security misconfigurations, compliance gaps, and insecure defaults before cloud resources reach production.

Terraform

Helm

Kubernetes

Docker

CloudFormation

1,197

built-in IaC policies ship enabled — 218 high and critical — across AWS, Azure, GCP, Kubernetes, and Docker.



PIPELINE

Security built into the build, not bolted on after

Legit extends native coverage into the CI/CD layer itself — pipeline configuration, build infrastructure, and pre-receive hooks — so risk introduced by the software factory gets caught before it ships.

CI/CD risk detection

Flags misconfigurations and risky settings across GitHub Actions, Jenkins, GitLab CI, and other build systems.

PR checks & gated builds

Runs as a status check on every pull request, surfacing findings inline and blocking merges on critical issues.

COVERAGE

Full coverage, one platform.

SCANNER	WHAT IT CATCHES	WHERE IT RUNS
SAST	Vulnerabilities in first-party and AI-generated code, including LLM and agentic-specific patterns	Native, continuous, PR checks
SCA	Vulnerable and outdated open-source dependencies, with reachability analysis	Native, continuous, PR checks
Secrets	Live credentials, API keys, and tokens, with validity checking	Code, commit history, CI/CD logs
IaC	Misconfigured cloud and infrastructure resources, caught before provisioning	Terraform, CloudFormation, Kubernetes manifests
Pipeline	Build misconfiguration, pipeline risk, and policy violations	System level, PR level

ALSO BUILT IN



RCA

Root cause analysis clears the backlog in one fix

Findings collapse to their shared origin, so one change closes a cluster of alerts instead of a hundred tickets that all trace back to the same dependency.

The screenshot displays a dashboard for a root cause analysis. At the top, it says "Fix root cause dependency to resolve 273 dependent issues" and "A dependency used in the source code creates multiple issues across the SDLC." Below this are sections for "Ticket" (Create ticket), "Assignees" (Set assignee), "Detected at" (07/13/2025 04:01 AM), "Tags" (Add tags), "Source tools", "Compliance" (SSDF, PCI DSS, +1), and "Legit". The main content area is divided into three columns: "Root cause" (Dependency introducing multiple issues, with a tree diagram showing a root node branching into multiple child nodes), "Impact" (Issues: 273, Assets: 1, with a bar chart showing counts for C: 28, H: 140, M: 70, L: 35), and "How to fix" (Upgrade dependency 2024.02.0, with a mail icon and "Learn more" link). Each column has a "See details" button at the bottom.



License risk detection

Flags open-source license types and violations alongside security risk.



Broad language coverage

Java, Python, JavaScript/TypeScript, Go, C#, Ruby, PHP, and C/C++.



Code-to-cloud context

Connects findings to business criticality and a precise developer ownership model.



Compliance framework mapping

Tags findings against frameworks like SSDF and PCI DSS for audit-ready reporting.

Go fast. We've got you.

Connect your SCM and Legit starts scanning — no complex rollout, no agents to babysit, no backlog of theoretical risk to sort through first.