OFFLOAD SECURITY

Code Security

Module Whitepaper

SAST, dependencies, secrets, IaC, SBOM, and license compliance — governed from commit to release

Version 1.0 · July 2026 · Classification: Customer-Shareable

1. Overview

The Code Security module scans repositories and uploaded code for vulnerabilities across every layer — static code analysis, open-source dependencies, hardcoded secrets, infrastructure-as-code, and software supply chain — and turns the results into governed, auditable release decisions with fix automation and full triage lifecycle.

2. Key Capabilities

- Static analysis (SAST) via OpenGrep and Bandit, with optional SonarQube integration; findings carry code context so reviewers see the vulnerable lines in place.
- Dependency scanning (SCA) backed by OSV.dev and Grype, enriched with CISA KEV known-exploited flags and EPSS exploit-probability scores on every finding.
- Software supply-chain protection: known-malicious package detection (OpenSSF MAL advisories) and typosquat detection, both able to block the release gate.
- Import-level reachability analysis that classifies dependency findings as reachable or not-imported to cut noise (advisory — it never weakens the gate).
- Secrets detection with Gitleaks (~150 curated rules plus entropy analysis), fingerprinted and triageable.
- Infrastructure-as-code scanning with Checkov: Terraform, CloudFormation, Kubernetes manifests, Dockerfiles, and Helm.
- SBOM generation (Syft, CycloneDX) and upload analysis (CycloneDX JSON/XML, SPDX), with six-family license classification, policy engine, and generated NOTICE/attribution files.
- Deterministic, versioned release gates enforceable as required GitHub commit statuses; PR summary comments and check runs via a GitHub App.
- Automated fix pull requests: patches pushed to a dedicated fix branch — never the base branch — with a PR opened for review.
- Governed triage lifecycle: risk acceptance requires owner, justification, and expiry (expired acceptances re-block); false positives require reviewer and evidence; stable fingerprints persist decisions across re-scans.
- AI-assisted analysis and fix suggestions on individual findings, using the customer's own model-provider API key.
- GitHub and Bitbucket integration with encrypted tokens; ZIP upload for air-gapped code; CI/CD endpoints with scoped API keys.

3. How It Works

Repositories are shallow-cloned into an isolated, scan-scoped workspace with hardened git settings; scanners run locally and the workspace is deleted when the scan completes, on success or failure. Only

findings and small per-finding code excerpts are retained — the full source tree is never stored. (The companion whitepaper, 'Source Code Protection', covers this lifecycle in depth.)

Findings from every scanner are normalized, fingerprinted, and de-duplicated, then flow through the governed triage lifecycle. Release-gate evaluation is deterministic and policy-versioned: the same scan against the same policy always yields the same pass/review/fail decision with the blocking rule identifiers recorded — an audit-defensible answer to 'why did this ship?'.

4. Data Handling & Security

Aspect	How it is handled
Credential protection	All connection credentials are encrypted at rest with Fernet authenticated encryption (AES-128-CBC + HMAC), decrypted in memory only at the moment of use, and never returned by APIs.
Tenant isolation	Every record is scoped to the owning team; all reads and writes filter on the tenant identifier, enforced by database indexes.
Retention & deletion	Results are subject to configurable retention windows; data can be hard-deleted per scan and is removed by the tenant-offboarding cascade.
Deployment	Available as SaaS or fully self-hosted / on-premises — in the on-premises model no security data leaves your environment.

5. One Platform, One Risk View

Code findings share the platform's unified vulnerability queue, mint license and supply-chain violations into the enterprise risk register, provide evidence for compliance controls, and appear alongside cloud, container, and Kubernetes findings in a single per-repository report — one risk view from source code to runtime.