OFFLOAD SECURITY

Source Code Protection

Security Architecture Whitepaper

How customer repositories and source code are handled, protected, and deleted

Version 1.0 · July 2026
Classification: Customer-Shareable

Contents

1. Executive Summary....3
2. Platform and Deployment Overview....3
3. Source Code Lifecycle During a Scan....3
4. What Is — and Is Not — Retained....4
5. Repository Access Model....5
6. Credential and Secrets Protection....5
7. Tenant Isolation, Retention, and Deletion....6
8. Scanning Toolchain and Intelligence Sources....7
9. AI Features and Customer Code....8
10. Compliance and Certifications....8
11. Summary....8

1. Executive Summary

The Offload platform provides application and cloud security scanning — static analysis (SAST), software composition analysis (SCA), secrets detection, infrastructure-as-code (IaC) analysis, and SBOM generation — over customer source code repositories. This document describes, at an engineering level, how customer source code is obtained, processed, protected, and deleted throughout that lifecycle.

The design follows four principles:

- **Ephemeral source access.** Repositories are cloned into an isolated, scan-scoped workspace, scanned locally, and deleted when the scan finishes — on success and on failure. The full source tree is never written to a database or object store.
- **Minimal retention.** Only scan findings and small code excerpts (a few lines around each finding, so engineers can triage in context) are retained — under configurable retention windows and hard-delete APIs.
- **Least-privilege access.** Read-only repository access is sufficient for all scanning. Write permissions are requested only for optional, customer-enabled features such as automated fix pull requests.
- **No third-party exposure.** Scanning runs entirely on platform infrastructure. Customer code is never used to train AI/ML models and is never shared with third parties; optional AI features use the customer's own model-provider API keys.

2. Platform and Deployment Overview

Offload is a multi-tenant security platform composed of a FastAPI application tier, asynchronous scan workers, and a MongoDB data tier. It is offered in two deployment models:

Model	Where scanning runs	Where data resides
SaaS	Offload-managed infrastructure	Offload-managed, tenant-isolated data stores
On-premises / self-hosted	Customer's own environment (containerized deployment)	Entirely within the customer's environment — source code and scan data never leave the customer's network

Customers with strict data-residency or code-egress requirements can adopt the on-premises model, in which the entire scanning pipeline described in this document executes inside their own perimeter.

3. Source Code Lifecycle During a Scan

3.1 Acquisition

When a scan is initiated, the platform performs a shallow git clone (depth 1, single branch, no tags) of the target repository into an isolated workspace directory created exclusively for that scan. Code is

transferred directly from the git provider to the scanning workspace over TLS; it is not proxied through any third-party service. Alternatively, customers may upload a ZIP archive of their code, which is extracted into the same kind of isolated, scan-scoped workspace.

3.2 Clone hardening

The clone operation itself is hardened against both malicious repository content and credential leakage:

- Git hooks are disabled and local file-protocol transports are blocked (a hostile repository cannot execute code on the scanning host through git mechanisms).
- The git process runs with a minimal environment and interactive prompts disabled, so host secrets cannot leak into the scan context.
- Clone targets are validated against SSRF and private-address guards before any network connection is made.
- Operations are time-boxed; access tokens are redacted from all logs and error messages.
- ZIP extraction is protected against path-traversal, with archive size limits enforced during upload.

3.3 Scanning

All scanners run locally against the workspace on platform infrastructure (or customer infrastructure, in on-premises deployments). No source code is transmitted to external scanning services. Section 8 lists the toolchain.

3.4 Automatic deletion

Every scan path — repository clone, ZIP upload, and fix-PR generation — deletes its workspace in a finally-style cleanup that executes on success and on failure alike. Uploaded ZIP archives are likewise deleted after extraction and scanning. The full source tree therefore exists on scanning infrastructure only for the duration of the scan (typically minutes).

4. What Is — and Is Not — Retained

Retained after a scan	Never retained
Scan metadata: repository name, branch, commit SHA, timestamps, tools run	The cloned source tree
Findings: rule, severity, file path, line numbers, triage state	Uploaded ZIP archives
Code excerpts: up to ±6 lines around each finding (≈13 lines max per finding)	Build artifacts or binaries
SBOM component inventories and license analysis	Repository history beyond the scanned commit

Retained after a scan	Never retained
Generated reports and triage/audit history	Any code from files larger than 2 MB or binary files (excluded from excerpt capture)

The code excerpts exist so that engineers can review a finding in context without re-opening the repository. Excerpt capture is bounded: at most a few lines around the finding, individual lines truncated at 500 characters, and binary or oversized files skipped entirely. These excerpts are stored with the finding records under the retention and deletion controls described in Section 7.

5. Repository Access Model

Read-only access is sufficient for all scanning functionality. The platform supports two GitHub integration mechanisms (plus Bitbucket token support):

Capability	Integration	Permission required	Access level
Repository listing, cloning, scanning	Personal access token	repo scope (or fine-grained: contents & metadata read)	Read-only
Repository scanning via GitHub App	GitHub App	contents: read, metadata: read	Read-only
Optional: PR check runs & inline annotations	GitHub App	checks: write	Write (opt-in)
Optional: PR summary comments	GitHub App	pull_requests: write	Write (opt-in)
Optional: automated fix pull requests	Personal access token	repo push access; pushes only to new offload/fix/* branches — never the base branch	Write (opt-in)
Optional: CI release-gate commit status	GitHub Action	statuses: write — uses the workflow's own GITHUB_TOKEN, not a stored credential	Write (opt-in)

Customers who need scanning only can grant read-only credentials and every scanning feature will work. Write-level permissions are needed solely for the optional feedback features listed above, each of which is enabled explicitly by the customer.

6. Credential and Secrets Protection

- **Encryption at rest.** All stored integration credentials — git access tokens, GitHub App private keys and webhook secrets, cloud provider credentials, and AI provider API keys — are encrypted with

Fernet authenticated symmetric encryption (AES-128-CBC with HMAC-SHA256 integrity protection) before being written to the database. Key rotation is supported.

- **Decryption on use only.** Credentials are decrypted in memory at the moment of use and are never returned by configuration APIs; administrative views expose only whether a credential is set.
- **Log redaction.** Access tokens are masked in clone URLs, log lines, and stored error messages.
- **Webhook authenticity.** Inbound GitHub webhooks are verified with HMAC-SHA256 signatures and rejected (fail-closed) when no webhook secret is configured.
- **Transport security.** All communication with git providers and external APIs uses TLS.

Storage-level encryption of the database and disks (encryption at rest for scan results and metadata) is provided by the hosting environment; in on-premises deployments this is under the customer's direct control.

7. Tenant Isolation, Retention, and Deletion

7.1 Multi-tenant isolation

Every scan document, finding, report, and credential is scoped to the owning tenant (team). All read and write paths filter strictly on the tenant identifier, and per-tenant uniqueness constraints are enforced at the index level. One tenant's scan data — including code excerpts — is never visible to another tenant.

7.2 Retention windows

Scan results are subject to configurable retention windows enforced by scheduled retention jobs and TTL policies. Defaults (configurable per deployment):

Data category	Default retention
Code scan results, including findings and code excerpts	180 days
SBOM / dependency analysis reports	180 days
Cloud & container scan results	90 days
Generated reports	180 days
Platform audit logs	365 days

Retention for code-scan data is enforced unconditionally — records past the window are removed regardless of scan state, so the retention bound on stored code excerpts holds even for interrupted scans.

7.3 Deletion on request

- **Per-scan deletion.** Any scan report or SBOM report can be permanently hard-deleted through the product or API. Deletion removes the entire scan record, including all findings and embedded code excerpts.
- **Integration removal.** Git connections (and their encrypted tokens) can be removed at any time.
- **Tenant offboarding.** Deleting a tenant runs a platform-wide cascade that hard-deletes all of the tenant's scan data — code scan results, code excerpts, SBOM reports, triage history, analysis records, and stored git credentials — alongside its cloud, container, and Kubernetes security data.

Customers may request complete deletion of their data at any time; the mechanisms above make that deletion permanent rather than a soft-hide.

8. Scanning Toolchain and Intelligence Sources

For transparency, the scanners executed against customer code and the vulnerability intelligence sources feeding results are listed below. All tools run locally within the scan workspace.

Function	Tooling
Static analysis (SAST)	OpenGrep, Bandit; SonarQube (optional)
Dependency analysis (SCA)	osv-scanner, with malicious-package and typosquat detection and import-level reachability analysis
Secrets detection	Gitleaks
Infrastructure-as-code	Checkov
SBOM generation	Syft — CycloneDX JSON, SPDX JSON, syft-json; ingestion of CycloneDX (JSON/XML) and SPDX (JSON)
Container image scanning	Trivy, Grype

Intelligence source	Use
OSV.dev	Primary open-source vulnerability advisories
NVD and distribution advisories (via Trivy/Grype databases)	CVE data and CVSS scoring; GitHub Security Advisories arrive through these aggregations
OpenSSF malicious-packages dataset	Known-malicious dependency detection
CISA Known Exploited Vulnerabilities (KEV)	Active-exploitation flagging
FIRST.org EPSS	Exploit-probability scoring for prioritization

9. AI Features and Customer Code

The scanning pipeline itself involves no AI services: no code is sent to any model provider as part of a scan. Optional AI-assisted features (for example, suggested fixes for a finding, or natural-language scan summaries) operate under the following controls:

- **Explicit invocation.** AI features run only when a user invokes them on a specific finding or report.
- **Customer-controlled providers.** Requests are sent directly over TLS to the model provider (Anthropic, OpenAI, or Google) configured with the customer's own API key. These providers do not train on API traffic under their standard API terms.
- **Minimal payload.** Only the finding and its small code excerpt are included — never the repository.
- **No training, ever.** Offload does not use customer code to train or fine-tune any model.
- **No cross-tenant caching.** The platform's AI response cache is restricted, by design, to generic tenant-agnostic content; responses containing customer-specific context are never cached or reused across tenants.

10. Compliance and Certifications

[TO BE COMPLETED BY OFFLOAD SECURITY — state current certification status, e.g. SOC 2 Type II report availability under NDA, ISO 27001 scope, GDPR/DPDP posture and DPA availability, penetration-testing cadence, and responsible-disclosure policy.]

11. Summary

Customer question	Answer
Is our source code stored on the platform?	No. Code is cloned to an ephemeral scan workspace and deleted when the scan finishes, including on failure. Only findings and small per-finding code excerpts are retained.
Is stored data encrypted?	Credentials are application-layer encrypted (Fernet/AES); scan data is protected by storage-level encryption of the hosting environment; all transport uses TLS.
Is read-only repository access enough?	Yes, for all scanning. Write permissions are only for optional, opt-in feedback features.
Is our code used for AI training or shared with third parties?	Never. Optional AI features use the customer's own provider keys, and providers do not train on API traffic.
Can our data be deleted?	Yes — per-scan hard deletes, integration removal, automatic retention windows, and a full tenant-offboarding cascade.

For questions about this document, or to request additional security documentation, contact security@offloadsecurity.com.