Secret Scanning Triage SOP

GitHub Advanced Security (GHAS)

Security Engineering

January 19, 2026

Document Control

Version: 1.0 **Last Updated:** January 19, 2026
Owner: Security Engineering (GHAS Admins) **Applies To:** All private/org repositories

Contents

1 Purpose & Scope

This Standard Operating Procedure (SOP) defines how the organization triages, remediates, and documents **GitHub Advanced Security (GHAS) Secret Scanning** alerts. It covers alerts raised by push protection (on push), on pull requests (PR checks), and retro/background scans across the commit history. Goals:

- Prevent credential leaks from reaching default branches and releases.
- Respond rapidly based on impact-driven **SLAs**.
- Ensure consistent evidence, auditability, and reporting.

2 Roles & Access

Developers	Remove secrets from code/PRs, rotate/revoke credentials, add remediation notes.
Repo Admins / Code Owners	Review/approve remediation PRs, enforce repo policy, coordinate fixes.
Security (GHAS Admins)	Define policy, tune patterns, manage org/repo settings, audit and reporting.
Access	Only users granted *read security alerts* can view repo alerts. Org-level security managers can view/manage alerts across repositories.

3 Where Alerts Come From

- **Push Protection (on push):** Blocks commits that contain secrets; committer must remove the secret or request an override with justification.
- **PR Checks:** Secrets surfaced during pull request validation to shift left.
- **Retro/Background Scans:** Scans find secrets in history or inactive branches.

4 Severity Levels & SLAs

Severity	Description	SLA
P0	Active production credential (cloud root/admin, DB admin, payment keys).	Contain within **1 hour**; rotate immediately.
P1	Privileged non-production credential or elevated risk.	**24 hours**.
P2	Low-impact, expired, or test credential.	**3 business days**.

5 End-to-End Triage Workflow

1) Acknowledge & Assign

Auto-assign to **Committer + Code Owners**; Security added as watcher. If push protection blocked the push, the committer must remediate or submit an exception with justification (recorded by GitHub).

2) Verify the Finding

Confirm the item is truly a credential. Prefer provider/partner patterns; treat custom patterns carefully to avoid noise.

3) Contain

- Remove the secret from the PR/branch immediately.
- **Rotate/revoke** the exposed credential with its provider (treat as compromised).

4) Eradicate

If the secret reached history, open a task to purge or rewrite history (e.g., `git filter-repo`) and **invalidate** any still-valid token.

5) Recover & Harden

Tune detection to prevent recurrence:

- Add or refine **custom patterns** (regex) for proprietary secrets.
- Test new patterns in the GitHub UI to reduce false positives before publishing org-wide.

6) Document & Disposition

In the alert record, add remediation notes and set status:

- **Resolved**: Secret removed, rotated, and (if needed) history cleaned.
- **Dismissed**: Select a reason (Section 7) and add a brief comment; dismissed alerts remain visible for audit.

6 Notification Routing

Principles

Notify the fewest people who can act, escalate only when needed, and avoid alert fatigue by tuning notification preferences.

Push Protection Block	Committer (required), Code Owners, Security channel (informational). If an override is requested, page Security.
PR Alert	PR Author and Code Owners (action), Security (watch).
Background/Retro Finding	Repo Admin (owns remediation), Security (tracker).
Channels	GitHub notifications (per-user), plus the incident channel (e.g., Slack/Email) for P0/P1.

7 Dismissal Reasons Policy

When dismissing an alert, you **must** choose a reason and add a short comment. Accepted reasons:

- **False positive**: Matches the pattern but not a credential (include 1-line proof).
- **Test/intentional fixture**: Dummy/test key per test policy (link to policy).
- **Expired/Revoked**: Verified invalid; include ticket/ID of revocation.
- **Duplicate**: Same secret already tracked under a different alert (link it).
- **Compensating control**: Allowed only with Security approval and documented control.

Never dismiss an active/valid credential

Dismissal must not be used to avoid remediation of a valid secret. Treat all valid credentials as compromised until rotated or revoked.

8 Exceptions (Push Protection Allow-List)

If a committer overrides a push-protection block, they must provide a justification; identity and rationale are recorded by GitHub. Security reviews all exceptions daily and may revoke allow-list entries at any time.

9 Reporting & Audit

- **Weekly**: Security reviews opened/closed counts by repo and *dismissal reasons*; verify push-protection exceptions.
- **Monthly**: Pattern tuning—add org-level patterns, retire noisy ones; test regex in UI before publishing.
- **Quarterly**: Retro scan review for inactive repositories and long-lived branches.

10 Quick-Action Checklist (On-Call)

1. Identify severity (**P0**/**P1**/**P2**).
2. Contain: remove secret and rotate/revoke immediately.
3. Hunt for other occurrences (full repo/history search).
4. Document remediation notes; link revocation evidence.
5. Disposition: **Resolve** or **Dismiss** with reason.
6. Tune patterns if applicable; test before rolling out org-wide.

Appendix A — Triage Notes Form

Alert ID: _____ **Repository:** _____ **Branch/PR:** _____
Committer: _____ **Assignee:** _____ **Severity (P0/P1/P2):** ____

Containment: Secret removed? (✓/✗) Rotation/Revocation ticket: _____

Eradication: History cleaned? (✓/✗) Method: _____

Disposition: Resolved / Dismissed (Reason: _____)

Notes/Evidence: _____

Follow-ups: Pattern tuning / Training / Control update _____

Appendix B — Exception Request (Push Protection Override)

Requester: _____ **Repo/Branch:** _____ **Date:** _____

Justification: _____

Risk Assessment (Security): _____

Decision: Approved / Denied **Reviewer:** _____ **Expiry:** _____

Appendix C — Glossary

GHAS: GitHub Advanced Security. **PR**: Pull Request. **SLA**: Service Level Agreement.