GitHub Advanced Security (GHAS) Secret Scanning

Standard Operating Procedure & Quick-Start Guide

Prepared for: Security & Platform Engineering

October 31, 2025

Purpose. This document defines a pragmatic, auditable way to enable, route, triage, and maintain GitHub Advanced Security *Secret Scanning* across the organization. It includes a quick-start checklist, operating procedures, and ready-to-use configuration.

Executive Summary

- **Minimize alert fatigue:** scope visibility to responders who must act (org-/repo-level).
- **Catch issues early:** enable *Push Protection* so secrets are blocked before they land.
- **Reduce noise safely:** use `.github/secret_scanning.yml` with conservative `paths-ignore`.
- **Detect internal tokens:** add org-level custom patterns; allow repo overrides when necessary.
- **Faster response:** send alerts to Teams/Slack/PagerDuty via a lightweight GitHub Actions workflow.
- **Accountability:** require dismissal reasons; review periodically for compliance and tuning.

1 Scope & Roles

Role	Area	Responsibilities
Security Engineering	Policy & Tuning	Org-level enablement, custom patterns, dismissal governance, quarterly tuning.
Platform/DevEx	Tooling & Routing	Notification integrations (Teams/Slack), sample workflows, templates.
Repo Maintainers	Project Setup	Adopt org defaults, minimal overrides, triage participation.
Incident Response	Response	Containment, key rotation, partner validity checks, post-incident review.

2 Quick-Start Checklist

1. **Enable** Secret Scanning and **Push Protection** org-wide.
2. Create `.github/secret_scanning.yml` with *minimal* `paths-ignore`; avoid broad exclusions.
3. Define **org-level custom patterns** for internal tokens; document samples and test regex in the UI.
4. Grant a **Security Team** permissions to triage/manage alerts; scope contributor visibility appropriately.

5. Wire **Teams/Slack/PagerDuty** notifications via GitHub Actions and a channel/webhook.

6. **Require dismissal reasons**; review monthly in Security Overview; remediate recurring root causes.

3 Operating Procedure (SOP)

3.1 Enablement

a) Org Owners enable Secret Scanning and Push Protection at the organization level.

b) Security Engineering publishes an org-wide baseline `.github/secret_scanning.yml`.

3.2 Notification Routing

a) Platform team deploys a reusable workflow (see Listing 2) posting alerts to chosen channels.

b) Each repo references the reusable or local workflow; secrets for webhooks are stored in GitHub Secrets.

3.3 Triage & Response

a) Alert received → **Acknowledge** in channel; assign an on-call/responder.

b) **Validate** via provider/partner checks when available (is the key active?). Disable/rotate immediately if active.

c) **Containment:** revoke/rotate token, purge from code if necessary, update IaC/secrets manager.

d) **Dismissal policy:** only use allowed reasons (see Section 4); justification is *required*.

e) **Post-incident:** add tests/rules to prevent recurrence; capture learnings; adjust patterns/exclusions.

3.4 Maintenance Cadence

- **Monthly:** review dismissal reasons & recurring patterns; refine custom patterns or exclusions.
- **Quarterly:** org-wide health check (coverage, false positive rate, MTTA/MTTR); re-validate routing.

4 Dismissal Governance

Allowed dismissal categories (examples):

- **Pattern Test/Example:** dummy token in docs or sample files (prefer moving to `*.sample`).
- **Revoked/Inactive:** provider confirms token is inactive; evidence linked in comment.
- **False Positive:** string matches regex but is not a credential; include proof (e.g., format contract).

Every dismissal must include a meaningful justification and, when applicable, an evidence link or ticket.

5 Configuration Reference

5.1 `.github/secret_scanning.yml` (baseline)

Listing 1: `.github/secret_scanning.yml` baseline

```yaml
# Scope carefully; start small and review quarterly
paths-ignore:
  - "deployment/**"        # example: generated configs or manifests
  - "**/*.sample"          # example: sample files with fake secrets

validity-checks: true       # attempt provider validation when possible

# Optional: ignore branches/tags if needed (be conservative)
branches-ignore:
  - "experiment/**"
tags-ignore:
  - "archive-*"
```

5.2 Teams Notification Workflow (example)

Listing 2: GitHub Actions workflow: Post signal to Microsoft Teams

```yaml
name: Secret scan alert to Teams
on:
  workflow_dispatch: {}
  pull_request:
    types: [opened, reopened, synchronize]
    branches: [main]

jobs:
  notify:
    runs-on: ubuntu-latest
    steps:
      - name: Post to Teams
        uses: fjogeleit/http-request-action@v1
        with:
          url: ${{ secrets.TEAMS_WEBHOOK_URL }}
          method: POST
          data: |
            {
              "text": "Secret scanning signal: PR #${{
              ↪  github.event.pull_request.number }} in ${{ github.repository }}
              ↪  needs review."
            }
```

Security notes. Store all webhooks/tokens in *org- or repo-level* GitHub Secrets. Restrict who can read or update these secrets. Prefer a dedicated channel and rotate webhook URLs when staff changes.

6 Appendix

Change Log Template

- v1.0 — Initial publication; org baseline, routing workflow, dismissal policy.