GHAS Secret Scanning — Exam / Cheat Sheet

GitHub Advanced Security (GHAS) focus: features, setup, workflow, and exam-ready facts.

Core Idea

Secret scanning detects sensitive values (API keys, tokens, credentials) across your repository content and history. With **push protection**, supported secrets can be blocked *before* they land in the repo.

1 When Scans Run

- **On push**: recommended; push protection can block secrets pre-merge.
- **On pull request**: checks run and raise alerts before merge to default branch.
- **Historical & ongoing**: newly added commits are scanned; full history can be scanned by GitHub.

2 Scope & Licensing

- **Public repositories**: baseline provider coverage available.
- **Private/Internal repositories**: enable via **GitHub Advanced Security (GHAS)** to unlock full features (custom patterns, push protection controls, and additional options).

3 Enablement & Where to Click

Organization level

1. `Settings` → `Security` → `Code security and analysis`.
2. Enable: **Secret scanning**, **non-provider patterns**, **validity checks**, **push protection**.

Repository level

- Inherits org defaults; can be overridden by repo administrators/owners as permitted.

4 Push Protection (Developer Experience)

- Blocks pushes containing supported secret formats; developer sees a helpful message and remediation guidance.
- Distinct from *branch protection*; it prevents the push itself rather than failing a PR check post-push.

5 Patterns & Coverage

- **Provider/partner patterns**: well-known token formats (cloud, SaaS, etc.).
- **Non-provider patterns**: generic patterns for likely secrets (e.g., private keys, DB strings).
- **Custom patterns**: define and test your regex at org or repo scope; use dry-run/testing tools to reduce noise. (Capacity limits apply; prefer org-level for reuse.)

6 Repo Configuration: `.github/secret_scanning.yml`

Place a configuration file to tune allowlists and optional repo-local custom patterns.

```yaml
# .github/secret_scanning.yml
version: 1

# Allowlist paths or literal values to suppress known benign matches.
allowlist:
  paths:
    - "docs/examples/**"
    - "test/fixtures/**"
  secrets:
    - "DUMMY-KEY-DO-NOT-USE"

# Optional: repo-scoped custom patterns (consider org-level for reuse)
custom_patterns:
  - name: "Acme Internal Token"
    regex: "ACME-[A-Za-z0-9]{32}"
    # (optional) specify "max_issuers", "keywords", etc., if supported in your UI.
```

7 Alert Triage Workflow

1. **Assess** the alert: secret type, location (commit/PR/branch), exposure blast radius.
2. **Remediate**: revoke/rotate at the provider; remove/replace from code; consider history rewrite if necessary.
3. **Record** outcome: add notes, link to the rotation ticket, and resolve/dismiss appropriately.

Dismissal Reasons (standard policy menu)

- False positive / test fixture
- Secret revoked / rotated
- Accepted risk (non-prod/short-lived) — with justification
- Not a secret (format match only)

8 Visibility & Permissions

- Minimum: users need permission to *read security alerts* on the repo.
- Organization security admins can view org-wide alerts depending on role settings.

9 Where Alerts Live

- Repository: `Security` → `Secret scanning`.
- Related features live nearby (e.g., dependency graph, code scanning) but are distinct.

10 Exam Quick Hits

- Config file name and path: `.github/secret_scanning.yml`.
- **Push protection** blocks pushes (pre-merge), not only PR checks.
- Differentiate *provider* vs *non-provider* patterns; know that many providers support validity checks.
- Favor *org-level custom patterns* for consistency and reuse.
- Document your dismissal reasons and route alerts (e.g., to AppSec/Platform) for audits.

11 Sample Regex (Reasoning Practice)

```
^sk-[A-Za-z0-9]{48}$
```

This stands in for a typical provider token shape. For the exam, recognize patterns and consider: is there a prefix, exact length, and allowed alphabet? Add keywords/anchors where appropriate to reduce false positives.

Exam Tip

Compile tip: This document uses `minted` for syntax highlighting. Compile with `-shell-escape` (e.g., `pdflatex -shell-escape GHAS_Secret_Scanning_Cheat_Sheet.tex`). If your toolchain can't use `minted`, switch to `verbatim` or `listings`.

Prepared for quick recall during reviews and exams. Tailor the allowlists, patterns, and routing to your organization's policies.