External Scanning → SARIF Upload & Categories (GitHub)

October 30, 2025

Build note. This document uses the `minted` package (Pygments). Compile with:

```
# Example (latexmk)
latexmk -pdf -shell-escape external-scanning-sarif-upload.tex

# Or (pdflatex, run twice if needed)
pdflatex -shell-escape external-scanning-sarif-upload.tex
```

1 Why SARIF & Categories

GitHub Code Scanning ingests SARIF 2.1.0 files so that third-party scanning results (for example Bandit, Sonar, etc.) appear alongside CodeQL alerts. When uploading multiple analyses for the *same commit* you should assign a unique *category* per analysis to prevent uploads from overwriting one another and to enable filtering by category in the Code scanning UI.

2 Generate SARIF From Your Tool

2.1 Bandit (Python)

If your Bandit build supports SARIF directly, you can emit SARIF 2.1.0 like this:

```
pip install bandit
bandit -r . -f sarif -o bandit.sarif
```

If you need an external formatter, install the SARIF formatter and pipe Bandit results through it (common in some setups):

```
pip install bandit-sarif-formatter
bandit -r . -f sarif -o bandit.sarif
```

2.2 CodeQL CLI (exporting SARIF)

If you analyze with the CodeQL CLI, export SARIF and set a distinct category at the same time:

```
codeql database analyze my-db my-suite.qls \
  --format=sarifv2.1.0 \
  --output results.sarif \
  --sarif-category codeql-python
```

3 Preferred Upload: GitHub Actions

Add a minimal workflow that uploads your SARIF (from any tool). Use the `category` input to keep analyses separate.

```yaml
name: Upload third-party SARIF

on:
  push:
  workflow_dispatch:

permissions:
  security-events: write
  actions: read
  contents: read

jobs:
  upload:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5

      # (Optional) Run your scanner here to produce results.sarif

      - name: Upload SARIF
        uses: github/codeql-action/upload-sarif@v4
        with:
          sarif_file: results.sarif
          category: bandit-sast   # separates this analysis
```

3.1 Multiple SARIF Files in One Run

When uploading multiple SARIF files for a single commit, give each a distinct category.

```yaml
- name: Upload Bandit SARIF
  uses: github/codeql-action/upload-sarif@v4
  with:
    sarif_file: bandit.sarif
    category: bandit-sast

- name: Upload Trivy SARIF
  uses: github/codeql-action/upload-sarif@v4
  with:
    sarif_file: trivy.sarif
    category: trivy-container-scan
```

4 API Upload (outside Actions)

If your scanner runs off-platform, you can POST SARIF directly to GitHub. The payload must be *gzip* compressed and Base64 encoded.

4.1 Prepare the SARIF payload

```
# Linux: write a single-line Base64 string
gzip -c results.sarif | base64 -w 0 > results.sarif.gz.b64

# macOS: base64 strips newlines by default
# gzip -c results.sarif | base64 > results.sarif.gz.b64
```

4.2 POST to the SARIF ingestion endpoint

Replace placeholders like `OWNER`, `REPO`, and `COMMIT_SHA_HERE`. The token needs `security_events:write` on private repos.

```
export GH_TOKEN=ghp_your_token_here

curl -sS -X POST \
  -H "Accept: application/vnd.github+json" \
  -H "Authorization: Bearer $GH_TOKEN" \
  https://api.github.com/repos/OWNER/REPO/code-scanning/sarifs \
  -d @- <<'JSON'
{
  "commit_sha":  "COMMIT_SHA_HERE",
  "ref":         "refs/heads/main",
  "sarif":       "$(cat results.sarif.gz.b64)",
  "checkout_uri":"file:///home/runner/work/REPO/REPO",
  "tool_name":   "Bandit"
}
JSON
```

5 Making Categories Stick

You have three reliable ways to ensure each upload has a unique category:

1. **CodeQL CLI**: pass `-sarif-category my-category`.
2. **GitHub Action `upload-sarif`**: set `with: category: my-category`.
3. **Embed in SARIF**: add a unique `runAutomationDetails.id` to the SARIF file *before* upload.

5.1 Embedding `runAutomationDetails.id` in SARIF

A minimal SARIF snippet showing where to add the identifier:

```
{
  "version": "2.1.0",
  "$schema": "https://json.schemastore.org/sarif-2.1.0.json",
  "runs": [
    {
      "tool": { "driver": { "name": "Bandit" } },
      "runAutomationDetails": { "id": "bandit-sast" },
      "results": [ /* ... */ ]
    }
  ]
}
```

Important. If you reuse the same tool *and* category for another upload to the same commit, the newer upload replaces the earlier one. Keep categories unique per tool, per slice (language, directory, service), or per pipeline stage.

6 After Upload: Where Results Appear & Gotchas

- Navigate to **Security** → **Code scanning alerts** to see findings. Filter by *Tool* (e.g., Bandit, CodeQL) and by *Category*.
- **Fingerprints**: include `partialFingerprints` in results or let `upload-sarif` compute them with access to the source tree; this reduces duplicate alerts across runs.
- **Size/limits**: gzip'ed SARIF should be reasonably small; extremely large runs may be truncated in the UI.
- **Spec**: target SARIF 2.1.0.

7 Ready-to-Use Workflow Templates

7.1 Bandit → Code Scanning (simple)

```yaml
name: Bandit (SAST) → Code Scanning

on:
  push:
  pull_request:

permissions:
  security-events: write
  actions: read
  contents: read

jobs:
  bandit:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5
      - run: pipx install bandit
      - run: bandit -r . -f sarif -o bandit.sarif || true
      - name: Upload SARIF
        uses: github/codeql-action/upload-sarif@v4
        with:
          sarif_file: bandit.sarif
          category: bandit-sast
```

7.2 Multi-scan Monorepo Slice (two uploads, distinct categories)

```
name: Monorepo multi-scan

on:
  push:
  pull_request:

permissions:
  security-events: write
  actions: read
  contents: read

jobs:
  scans:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5

      - name: Python SAST (Bandit)
        run: |
          pipx install bandit
          bandit -r services/api -f sarif -o bandit-api.sarif || true

      - name: Container scan (Trivy)
        run: |
          curl -sfL
          ↪   https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh
          ↪   | sh -s -- -b /usr/local/bin
          trivy image --format sarif --output trivy.sarif ghcr.io/org/app:latest
          ↪   || true

      - name: Upload Bandit SARIF
        uses: github/codeql-action/upload-sarif@v4
        with:
          sarif_file: bandit-api.sarif
          category: bandit-sast-api

      - name: Upload Trivy SARIF
        uses: github/codeql-action/upload-sarif@v4
        with:
          sarif_file: trivy.sarif
          category: trivy-container-scan
```

Appendix: Quick Checklist

- Produce valid **SARIF 2.1.0**.
- **One category per analysis** per commit to avoid overwrite and enable filtering.
- Prefer **GitHub Actions** with `upload-sarif`; use the API when off-platform.
- Keep SARIF files reasonably small; split huge scans across slices with distinct categories.
- For repeatability, pin tool versions and document the exact command lines in the repo.