Getting Started with Ingesting GitHub GHAS Alerts
How to use this guide

This deck is meant as a starting point for ingesting GitHub Advanced Security (GHAS) Alerts. These alerts can be fed into 3rd party solutions

- Logging
- Observability
- Security Information and Event Management (SIEM)
- Business Intelligence (BI)

It contains links to documentation and sample code. The code samples leverage octokit.js.
Topics

👋 GitHub Advanced Security platform overview
🤖 Polling with GitHub REST API
💬 Webhooks
🚀 Summary
GitHub platform overview
3 Types of GitHub GHAS Alerts

- Code Scanning Alerts
- Secret Scanning Alerts
- Dependabot Alerts
How to try GitHub Advanced Security

The matrix below illustrates which features are available for free during your trial, depending on whether you're using a private or public repository.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Public repo</th>
<th>Private repo without GHAS</th>
<th>Private repo with GHAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code scanning</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Secret scanning</td>
<td>✔️ *</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td>Dependency review</td>
<td>✔️</td>
<td>❌</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Ingestion Strategy Comparison

Polling

Pros
● Returns rich data set
● Has historical alerts

Cons
● Subject to rate limits
● Requires a dedicated host
● May require data sanitation (secret scanning alerts contain secrets)

Webhooks

Pros
● Push based
● Real time, event-driven

Cons
● Requires HTTP write endpoint
● No retry mechanism
● Returns only summaries
● No history, only new events
Polling with GitHub API’s
Code Scanning REST API docs example

Code Snippet

```javascript
// Octokit.js
// https://github.com/octokit/core.js#readme
const octokit = new Octokit({
    auth: 'personal-access-token123'
})

await octokit.request('GET /orgs/{org}/code-scanning/alerts', {
    org: 'ORG'
})
```

Example 200 Response

```json
[
    {
        "number": 4,
        "created_at": "2020-02-13T12:29:18Z",
        "state": "open",
        "dismissed_by": null,
        "dismissed_at": null,
        "dismissed_reason": null,
        "rule": {
            "id": "js/zipslip",
            "severity": "error",
            "tags": [
                "security",
                "external/cwe/cwe-022"
            ],
            "description": "Arbitrary file write during zip extraction",
            "name": "js/zipslip"
        }
    }
    ...
]```
Secret Scanning REST API docs example

Code Snippet

```javascript
// Octokit.js
// https://github.com/octokit/core.js#readme
const octokit = new Octokit({
  auth: 'personal-access-token123'
})

await octokit.request('GET /repos/{owner}/{repo}/secret-scanning/alerts', {
  owner: 'OWNER',
  repo: 'REPO'
})
```

Example 200 Response

```json
[
  {
    "number": 2,
    "created_at": "2020-11-06T18:48:51Z",
    "state": "resolved",
    "resolution": "false_positive",
    "resolved_at": "2020-11-07T02:47:13Z",
    "resolved_by": {
      "login": "monalisa",
      "id": 2,
      "node_id": "MDQ6VXNlcjI=",
      "avatar_url": "https://alambic.github.com/avatars/u/2?",
      "gravatar_id": "",
      "url": "https://api.github.com/users/monalisa",
      ...
    }
  }
]...
```javascript
const { lastIssues } = await octokit.graphql(  
  `query fetchRepoAlerts ($org: String!, $repo: String!) {  
    repository(owner: $org, name: $repo) {  
      vulnerabilityAlerts(first: 100) {  
        nodes {  
          createdAt  
          dismissReason  
          dismissedAt  
          dismisser {  
            login  
          }  
          securityAdvisory {  
            description  
            ghsaId  
            cvss {  
              score  
            }  
            severity  
            summary  
          }  
          vulnerableManifestPath  
          vulnerableManifestFilename  
        }  
        pageInfo {  
          hasNextPage  
          endCursor  
        }  
      }  
    }  
  }  
);```

**Sample Query**

```json```

```json```

**Example 200 Response**

```json```

```json```

---

**Dependabot GraphQL API docs example**

**Sample Query**

```javascript```

```javascript```

**Example 200 Response**

```json```

```json```
Webhook Events & Payloads
code_scanning_alert docs

Webhook payload example

```json
{
  "action": "reopened",
  "alert": {
    "number": 10,
    "created_at": "2020-07-22T14:06:31Z",
    "updated_at": "2020-07-22T14:06:31Z",
    "instances": [
      {
        "ref": "refs/heads/main",
        "analysis_key": ".github/workflows/workflow.yml:upload",
        "environment": "{"",
        "state": "open"
      }
    ],
    "state": "open",
    "fixed_at": null,
    "dismissed_by": null,
    "dismissed_at": null,
    "dismissed_reason": null,
    "rule": {
      "id": "Style/FrozenStringLiteralComment",
      "severity": "note",
      "description": "Add the frozen_string_literal comment to the top of files to hel"}
```
repository_vulnerability_alert docs

Webhook payload example

```json
{
   "action": "create",
   "alert": {
      "id": 91005700,
      "affected_range": ">= 2.0.4, < 2.0.6",
      "affected_package_name": "rack",
      "fixed_in": "2.0.6",
      "external_identifier": "CVE-2018-16470",
      "severity": "moderate",
      "ghsa_id": "GHSA-hg78-4f6x-99wq",
      "created_at": "2021-03-01T01:23:45Z"
   },
   "repository": {
      "id": 186633002,
      "node_id": "MDExOlJlcG9zaXRvcnkxODY4NTMwMDI",
      "name": "Hello-World",
      "full_name": "Codertocat/Hello-World",
      "private": false,
      "owner": {
         "login": "Codertocat",
         "id": 21031067,
         "node_id": "MDo6VXNlcjIwMDMwMDY3",
         "gravatar_id": ""
      }
   }
}
```
Webhook payload example

```json
{
    "action": "reopened",
    "alert": {
        "number": 191,
        "secret_type": "adafruit_io_key",
        "resolution": null,
        "resolved_by": null,
        "resolved_at": null
    },
    "repository": {
        "id": 257423561,
        "node_id": "MDEwOlJlcG9zaXVzc29yc2NyNjE=",
        "name": "Hello-World",
        "full_name": "Codertocat/Hello-World",
        "private": true,
        "owner": {
            "login": "Codertocat",
            "id": 30846345,
            "node_id": "MDEyOlJlcG9zaXVzc29yc2NyNjE=",
            "gravatar_id": "",
            "url": "https://api.github.com/users/Codertocat",
            "html_url": "https://github.com/Codertocat",
            "followers_url": "https://api.github.com/users/Codertocat/followers",
            "following_url": "https://api.github.com/users/Codertocat/following(/other_user)
        }
    }
}```
Testing Webhooks

GitHub keeps a log of each webhook delivery for 30 days.
Best Practices

- Use Webhooks in conjunction with the REST API’s to get the full picture
- Create a GitHub App for higher rate limits.
Appendix: Authentication
<table>
<thead>
<tr>
<th>Authentication Scheme</th>
<th>Also Known As</th>
<th>Description</th>
<th>How to Get It</th>
<th>Available Endpoints</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON Web Token</td>
<td>JWT (pronounced “jot”)</td>
<td>Authenticates as the GitHub App</td>
<td>GitHub docs, Octokit</td>
<td>List</td>
<td>Fetching application installation details or exchanging the JWT for an installation access token.</td>
</tr>
<tr>
<td>Installation access token</td>
<td>Server-to-server requests</td>
<td>Authenticates as a specific installation of the GitHub App</td>
<td>GitHub docs, Octokit</td>
<td>List</td>
<td>Opening an issue or providing feedback on a pull request</td>
</tr>
<tr>
<td>OAuth access token</td>
<td>User-to-server requests</td>
<td>Authenticates as a user of the GitHub App</td>
<td>GitHub docs</td>
<td>List</td>
<td>Authenticating as a user when a GitHub App needs to verify a user’s identity or act on a user’s behalf</td>
</tr>
<tr>
<td>Personal Access Token</td>
<td>PAT</td>
<td>Authenticates as a user</td>
<td>GitHub docs</td>
<td></td>
<td>PATs are an alternative to using passwords for authentication to GitHub</td>
</tr>
</tbody>
</table>
Authentication at a glance

Deciding which authentication type to use comes down to:

● What resource do I need to access?
● Who do I need to access it as?
Server-to-server requests are those made from the perspective of an installation and are authenticated by installation access tokens.

Using your JWT, generate an installation access token via:

```bash
curl -i -X POST \
  -H "Authorization: Bearer YOUR_JWT" \
  -H "Accept: application/vnd.github.machine-man-preview+json" \
  https://api.github.com/app/installations/:installation_id/access_tokens
```

As a security measure, these tokens expire after 1 hour. They can be used like:

```bash
curl -i \
  -H "Authorization: token YOUR_INSTALLATION_ACCESS_TOKEN" \
  -H "Accept: application/vnd.github.machine-man-preview+json" \
  https://api.github.com/installation/repositories
```
User-to-server requests

User-to-server requests act as a *user who has authorized your GitHub App* and are authenticated using an **OAuth access token**.

First, users authorize your GitHub App via **OAuth** and receive a **code**:

Then, your GitHub App trades the **code**, **client_id** and **client_secret** for an **OAuth access token** to be used like:

```
```

Unlike typical OAuth, the scope is determined by the GitHub App.
Additional Resources

- [Developer Documentation](#)
- GitHub [REST](#) and [GraphQL](#) APIs
- [GitHub Webhooks](#)
- [Octokit](#) and 3rd party libraries
- [smee.io](#) Tool for testing Webhooks
- [Platform Samples](#) repo
- [GitHub Advanced Security Workshop](#)
- Webhook handler samples
  - [github-webhook-handler](#) node.js
  - [python-github-webhooks](#) python
  - [github_webhook](#) ruby
  - [hookserv](#) go
  - [afterparty](#) rust
  - [Github-webhook-lambda](#) (AWS lambda)
  - [GitHub-Webhook-Function](#) (Azure Function)
  - [github-webhook-cloud-function](#) (Google Cloud Functions)
Summary

- Polling the API is great for getting rich data sets
- Webhooks are great for getting alerts as soon as they happen