GitGuardian [GitHub Actions Security cheat sheet — 2022]

PLAINTEXT SECRETS

Never store any API key, token or password in plain text!

GitHub Secrets allows you to store and use your secrets in a safe way for your workflows. Running a CI workflow is also a good place to **implement secrets detection and remediation.**
You can leverage the `ggshield-action` for that.

GitGuardian [gitguardian-shield]

MINIMALLY SCOPED CREDENTIALS

Every credential used in the workflow should have the minimum required permissions to execute the job.

In particular, use the **'permissions' key** to make sure the **GITHUB_TOKEN** is configured with the least privileges for each job.

To limit their scope, **environnement variables** should be declared at the step level when possible.

SELF-HOSTED RUNNERS

It is strongly recommended to not use self-hosted runners for open-source repositories!

```
runs-on: self-hosted
```

With self-hosted runners, you are responsible for hardening your virtual machines:
- configure a dedicated low-privileged user
- preferably use isolated and ephemeral workloads to execute the jobs
- use logging and security monitoring to ensure visibility

For public repositories, you should also configure GitHub to require approval for any workflow runs.

PREFER USING OPENID CONNECT

As a best practice, use **OpenID Connect (OIDC) instead of long-lived secrets** to allow your workflows to interact with your cloud provider.

```yaml
# .github/workflows/fake-build.yaml

name: Create an issue before building

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  create_issue:
    name: Create an issue
    runs-on: ubuntu-latest

    permissions:
      issues: write

    steps:

      - name: Create issue using REST API
        uses: someperson/post-issue@f054a8b539a109f9f41c372932f1ae047eff08c9
        with:
          token: ${{ secrets.GITHUB_TOKEN }}


  build:
    name: Install dependencies
    runs-on: ubuntu-latest
    steps:

    - uses: actions/checkout@v3
      with:
        ref: ${{ github.event.pull_request.head.sha }}


    - name: Setup Python 3.10
      uses: actions/setup-python@v3
      with:
        python-version: '3.10'

    - name: Install dependencies
    - run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

####                          Rest of the workflow ...
```

```yaml
# .github/workflows/fake-deployment.yaml

on: deployment

jobs:
  deploy:
    name: Build and test
    runs-on: ubuntu-latest
    steps:

      - name: Deploy cloud resource service
      - uses: cloudprovider/deploy@v13d241b293754004c80624b5567555c4a39ffbe3
        with:
          token: ${{ secrets.CLOUD_PROVIDER_SECRET }}
```

SPECIFIC VERSION TAGS

When using third-party actions, **pin the version with a commit hash** rather than a tag to shield your workflow from potential supply-chain compromise.

PULL_REQUEST_TARGET

Don't check out external PRs when using the `pull_request_target` event:

```
on: pull_request_target
...
 - uses: actions/checkout@v3
   with:
    ref: ${{ github.event.pull_request.head.sha }}
```

You are giving write permission and secrets access to untrusted code. Any building step, script execution, or even action call could be used to compromise the entire repository.

UNTRUSTED INPUT

Don't directly reference values you don't control, like:

```
echo "${{github.event.pull_request.title}}"
```

It's all too easy for a malicious PRs to be executed in your workflow. Instead:

- use an action with arguments (recommended):

```
uses: fakeaction/printtitle@v3
with:
  title: ${{ github.event.pull_request.title }}
```

- or bind the value to an intermediate environment variable:

```
- name: Print title
  env:
    PR_TITLE: ${{ github.event.pull_request.title
  run: |
  echo "$PR_TITLE"
```