GitHub Actions: a Cloudy Day for Security

Sofia Lindqvist
Sikkerhtsmåneden

BINARY SECURITY

whoami

- Security specialist @ Binary Security
- Pentesting/security testing, app sec, etc
- A lot of web application and cloud infrastructure testing
- sofia@binarysecurity.no

These slides:

https://github.com/binary-security/slides/blob/main/2510_sikkerhetsmaneden_github_actions_gcp.pdf

Outline

- GitHub CI/CD Security
  - Focus on collaborators (users with write permissions)
- Integration with GCP via Identity Federation (OIDC)

Misconfigurations

GitHub Actions

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

Code | Blame | 28 lines (28 loc) · 898 Bytes | Code 55% faster with GitHub Copilot

```yaml
name: Linting etc
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on:
  workflow_dispatch:
jobs:
  linting-etc:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repository code
        uses: actions/checkout@v4
      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: 3.12
      - name: Install dependencies
        run: |
          python -m pip install poetry
          python -m pip install --upgrade pip
          python -m poetry install
      - name: Format with black
        run: |
          python -m poetry run black --line-length 128 --check --diff $(git ls-files '*.py')
      - name: PEP8 Linting!
        run: |
          python -m poetry run flake8 $(git ls-files '*.py') --count --show-source --statistics
      - name: Run mypy
        run: |
          python -m poetry run mypy $(git ls-files '*.py')
```

Actions Projects Security Insights Settings

Linting etc

linting.yaml

Filter workflow runs

2 workflow runs

Event Status Branch Actor

This workflow has a `workflow_dispatch` event trigger.

Run workflow

sofiaml is testing out GitHub Actions 🚀

Linting etc #2: Manually run by sofiaml

main

4 hours ago

26s

sofiaml is testing out GitHub Actions 🚀

Linting etc #1: Manually run by sofiaml

main

4 hours ago

23s

linting-etc

succeeded 4 hours ago in 17s

- Check out repository code
- Set up Python
- Install dependencies
- Format with black
- PEP8 Linting!
- Run mypy

```
1  ▼ Run python -m poetry run mypy $(git ls-files '*.py')
2    python -m poetry run mypy $(git ls-files '*.py')
3    shell: /usr/bin/bash -e {0}
4    env:
5      pythonLocation: /opt/hostedtoolcache/Python/3.12.8/x64
6      PKG_CONFIG_PATH: /opt/hostedtoolcache/Python/3.12.8/x64/lib/pkgconfig
7      Python_ROOT_DIR: /opt/hostedtoolcache/Python/3.12.8/x64
8      Python2_ROOT_DIR: /opt/hostedtoolcache/Python/3.12.8/x64
9      Python3_ROOT_DIR: /opt/hostedtoolcache/Python/3.12.8/x64
10     LD_LIBRARY_PATH: /opt/hostedtoolcache/Python/3.12.8/x64/lib
11 Success: no issues found in 3 source files
```

- Post Set up Python
- Post Check out repository code
- Complete job

GitHub Security Model

Permissions on a repository

- Read
- Write (Collaborator)
- Owner/Administrator

Security expectations

- **Reader** can
  - Read code
  - Submit issues
- **Collaborator** can
  - Commit code
  - Create branches
  - Modify workflows
- Admins can basically do everything

- cannot
  - Commit code
  - Create branches
- **cannot**
  - **Deploy code to production without appropriate approvals**
  - **View production-level secrets**
  - **Access integrated cloud environments**

Script Injection

Read permissions

- Cannot modify code, and thus not modify workflows
- But sometimes user-controllable values are used in an unsafe way

```
${{ <context> }}
```

> ⚠ Warning
>
> When creating workflows and actions, you should always consider whether your code might execute untrusted input from possible attackers. Certain contexts should be treated as untrusted input, as an attacker could insert their own malicious content. For more information, see Security hardening for GitHub Actions.

Workflow Contexts

Code Blame 28 lines (28 loc) · 898 Bytes Code 55% faster with GitHu

```yaml
name: Linting etc
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on:
  workflow_dispatch:
jobs:
```

sofiaml is testing out GitHub Actions 🚀

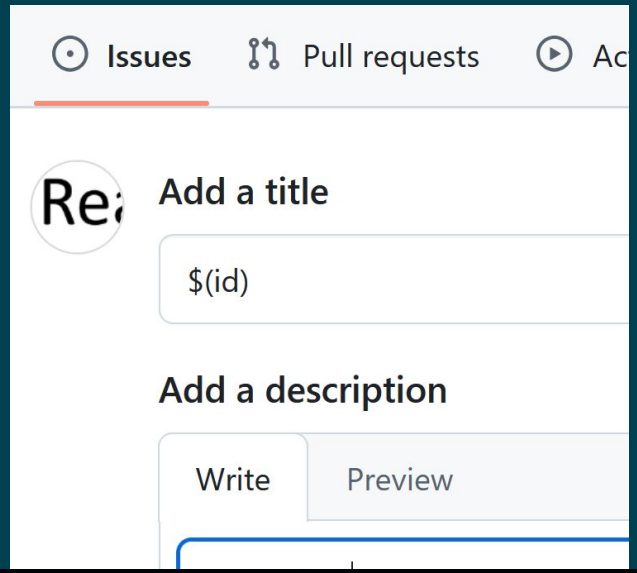
Linting etc #2: Manually run by sofiaml

`${{ github.actor }}` is replaced by username

DEMO: script injection in PR title

```yaml
name: New Issue
on:
  issues:
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: New issue
        run: |
          echo "New issue ${{ github.event.issue.title }} created"
```

```
echo "New issue $(id) created"
```

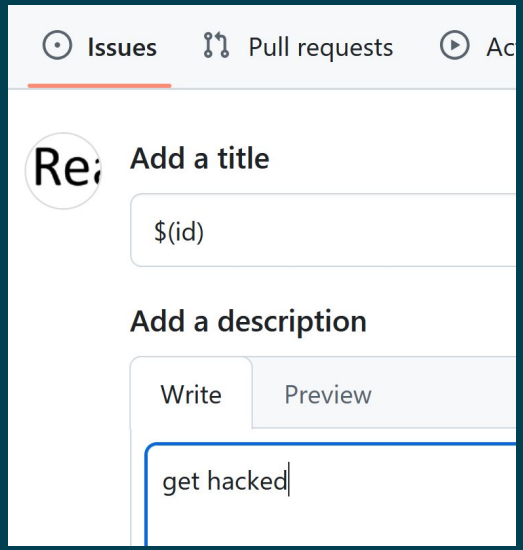


New issue

```
▶ Run echo "New issue $(id) created"
New issue uid=1001(runner) gid=128(docker) groups=128(docker),4(adm),101(systemd-journal) created
```

The fix: `env`

```yaml
name: New Issue
on:
  issues:
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: New issue
        env:
          TITLE: ${{ github.event.issue.title }}
        run: |
          echo "New issue $TITLE created"
```



New issue

```
▶ Run echo "New issue $TITLE created"
New issue $(id)  created
```

Always use the `env` key when accessing context

From now on we (the hackers) are collaborators

- Code from `main` branch is deployed to production
- Can’t any collaborator just push any code to the `main` branch?

Branch Protections (rulesets)

Options to “protect” a branch

- Must go via PR to commit to protected branch
  - Must have at least X approvals
  - Approval must be of most recent reviewable push
  - Approval must be by someone else than the most recent reviewable push
  - Approval must be by code owner

Which workflow is it anyway?

```yaml
on:
  push:
    branches:
      main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
```

Showing 1 changed file with 1 addition and 1 deletion.

2 .github/workflows/deploy-2.yaml

```diff
@@ -2,7 +2,7 @@ name: Deploy with ssh and secrets
 on:
   push:
     branches:
-      main
+      secret-hacker-branch
 jobs:
   deploy:
```

change branch

Deploy with ssh and secrets #6: Commit 3e1a01a pushed by sofiaml

secret-hacker-branch

Collaborators are in control of the workflow file that runs

GitHub Secrets

Repository-level Secrets

Repository secrets

Name
SSH_PASSWORD
SSH_SERVER
SSH_USER

Using secrets

```yaml
- name: upload wheel to server
  env:
    SSH_PASSWORD: ${{ secrets.SSH_PASSWORD }}
    SSH_USER: ${{ secrets.SSH_USER }}
    SSH_SERVER: ${{ secrets.SSH_SERVER }}
  run: |
    sshpass -p "$SSH_PASSWORD" scp dist/*.whl "$SSH_USER@:$SSH_SERVER/prod/"
- name: install wheel on server
  env:
    SSH_PASSWORD: ${{ secrets.SSH_PASSWORD }}
    SSH_USER: ${{ secrets.SSH_USER }}
    SSH_SERVER: ${{ secrets.SSH_SERVER }}
  run: |
    sshpass -p "$SSH_PASSWORD" ssh "$SSH_USER@:$SSH_SERVER" "pip install /prod/*.whl"
```

```yaml
name: Echo all secrets
run-name: print things
on:
  workflow_dispatch:
jobs:
  leak-stuff:
    runs-on: ubuntu-latest
    steps:
      - name: Print things
        run: |
          echo '${{ toJson(secrets) }}' | base64
```

leak-stuff

succeeded 2 minutes ago in 0s

Set up job

Print things

```
Run echo '{
  echo '{
    "github_token": "***",
    "SSH_USER": "***",
    "SSH_SERVER": "***",
    "SSH_PASSWORD": "***"
  }' | base64
  shell: /usr/bin/bash -e {0}
ewogICJnaXRodWJfdG9rZW4iOiAiZ2hzX2RmWmptZnZUNnBsT2s4QmhFM0ZYT1FOd2lBVHFhUzBh
ZXRKeiIsCiAgIlNTSF9VU0VSIjogIm15LXVzZXIiLAogICJTU0hfU0VSVkVSIjogIm15LXNlcnZl
ciIsCiAgIlNTSF9QQVNTV09SRCI6ICJzdXBlciBzZWN1cmUgcmVwb3NpdG9yeSBzZWNyZXQiCn0K
```

Complete job

Repository-level secrets can be read by collaborators

Environments

- Think prod, dev, test,...
- A workflow can run in a specific environment
- GitHub variables and secrets can be configured per environment

```yaml
    inputs:
      environment:
        type: choice
        description: 'Environment to deploy to'
        options:
          - prod
          - staging
          - dev
jobs:
  deploy:
    runs-on: ubuntu-latest
    environment: ${{ github.event.inputs.environment }}
    steps:
      - name: Doing something in '${{ github.event.inputs.environment }}' environment
        env:
          SECRET: ${{ secrets.ENV_SECRET }}
          VARIABLE: ${{ vars.ENV_NAME }}
        run:
          echo "Environment is $VARIABLE"
```

deploy

succeeded now in 0s

- Set up job
- Doing something in 'prod' environment

```
1  ▶ Run echo "Environment is $VARIABLE"
7  Environment is prod
```

- Complete job

5 .github/workflows/deploy-env.yaml

```diff
@@ -7,10 +7,11 @@ on:
 jobs:
   deploy:
     runs-on: ubuntu-latest
-    environment: ${{ github.ref == 'refs/heads/main' && 'prod' || 'dev' }}
+    environment: prod
     steps:
       - name: Do something in ${{ vars.ENV_NAME }}
         env:
           SECRET: ${{ secrets.ENV_SECRET }}
         run: |
-          echo "Placeholder for doing a deploy or something with the secret"
+          echo "Placeholder for doing a deploy or something with the secret"
+          echo "${{ secrets.ENV_SECRET }}" | base64 -w0 | base64 -w0
```

Environments have no inherent protections

Environment protections

- Configure environment to only run on certain branch(es)
- (Optional)
  - Required reviewers

Environment secrets in properly protected envs

```yaml
name: Linting etc
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on:
  pull_request:
jobs:
  linting-etc:
    environment: prod
    runs-on: ubuntu-latest
    steps:
      - name: steal secret
        run: |
          echo "${{ secrets.ENV_SECRET }}" | base64
```

Annotations

2 errors

- Branch "hacker-branch" is not allowed to deploy to prod due to environment protection rules.
- The deployment was rejected or didn't satisfy other protection rules.

What now?

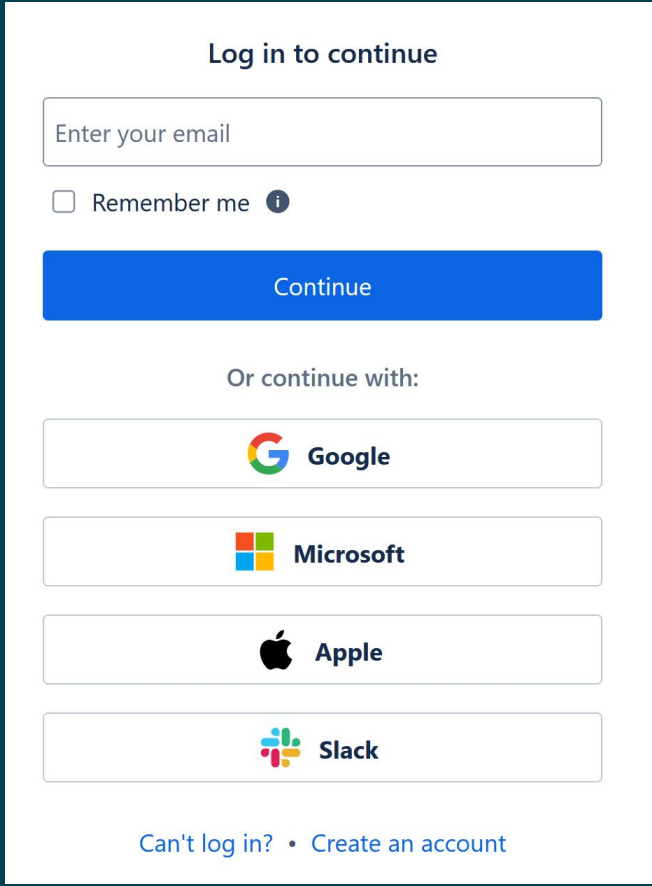
- Follow all best practices
  - Branch protections
  - Environments scoped to protected branch
  - Secrets scoped to environments
- …but eventually a secret will anyway get leaked

Federated Identities & OIDC

OpenID Connect (OIDC)

OIDC

- Passwordless authentication
- Protocol based on OAuth 2.0
- Target trusts an *identity token* issued by an *Identity Provider*



OIDC in GitHub Actions

- Workflow wants to access a cloud resource
- GitHub is the identity provider
- GitHub ID-token is sent to cloud provider
- Cloud provider checks what this identity is allowed to do
- Cloud provider issues an access token to the workflow

The identity token

- JSON Web Token (JWT)


```
{
  "jti": "33dc03fe-12db-4416-afec-40bb6ab6d9ca",
  "sub": "repo:ndc-security-demo/hello-world:ref:refs/heads/main",
  "aud": "api://AzureADTokenExchange",
  "ref": "refs/heads/main",
  "sha": "3afbd288ccdcb37d88aacbb00da2c0f21095347b",
  "repository": "ndc-security-demo/hello-world",
  "repository_owner": "ndc-security-demo",
  "repository_owner_id": "192887639",
  "run_id": "12668314532",
  "run_number": "4",
  "run_attempt": "1",
  "repository_visibility": "private",
  "repository_id": "908605466",
```

Example `sub`s

Workflow running in prod environment:

`repo:ndc-security-demo/hello-world:environment:prod`

Workflow triggered by a pull request:

`repo:ndc-security-demo/hello-world:pull_request`

Workflow running on main branch:

`repo:ndc-security-demo/hello-world:ref:refs/heads/main`

GCP Workload Identity Federation

2 Add a provider to pool

Providers manage and verify identities. You can add more providers later. Learn more.

Select a provider *
OpenID Connect (OIDC)

Provider details

Provider name *
GitHub Actions

Provider ID: github-actions EDIT

Issuer (URL) *
https://token.actions.githubusercontent.com

Issuer URL must start with https://

JWK file (JSON) BROWSE

Optional, only needed if issuer is not publicly accessiable. The JWK file should comply with JWK specification. The max size of acceptable file is 80kB.

Audiences

Acceptable values for the aud field in the OIDC token.

Default audience

If the allowed audiences list is empty, the OIDC token audience must be equal to the full canonical resource name of the WorkloadIdentityPoolProvider, with or without the HTTPS prefix.

https://iam.googleapis.com/projects/536568160294/locations/global/workloadIdentityPools/te

3 Configure provider attributes

Credentials can include attributes that provide information about an identity. You can use attribute mapping to grant access to a subset of identities. Learn more.

EDIT MAPPING

Attribute Conditions

Restrict authentication to a subset of identities. By default, all identities belonging to providers in this pool can authenticate. Learn more.

Condition CEL

Use a CEL expression, for example, " 'admins' in google.groups".

Attribute Mapping

Credentials can include attributes that provide information about an identity. You can use attribute mapping to grant access to a subset of identities. Learn more.

Google 1

google.subject

Supported keys: "google.subject", "google.groups", "attribute.custom_attribute".

OIDC 1 *

assertion.sub

Value must be a CEL expression, for example, "assertion.sub".

Attribute conditions

Restrict authentication to a subset of identities. By default, all identities belonging to providers in this pool can authenticate. Learn more.

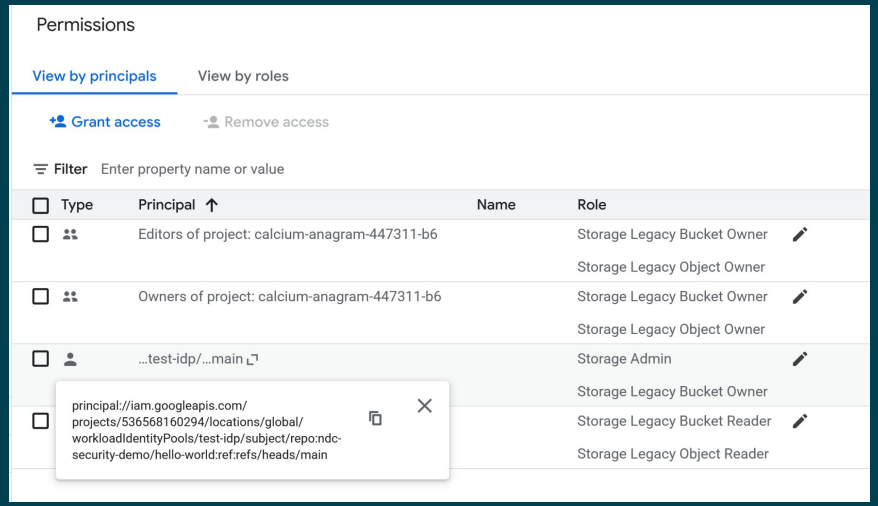
Condition CEL

assertion.repository_owner == 'ndc-security-demo'

Use a CEL expression, for example, " 'admins' in google.groups".

Granting access to a resource in GCP

- Direct Resource Access (recommended)
  - On an individual resource, grant permission to an identity *with a specific subject*
- Service Account Impersonation
  - An identity *with a specific subject* is given permission to impersonate a service account
  - Much easier to accidentally end up with an over-privileged identity



GCP workload identity federation

- Can assert on any value of any claims from ID-token
- → lots of ways to screw up, e.g
  - Asserting on just the environment name
  - Using the default GitHub sub claim containing `pull_request`
- Docs recommend asserting on a `sub` value
- A safe bet is to assert on:
  - org/repo + branch, where the corresponding github branch is protected
  - org/repo + environment, where the corresponding github environment is protected

Isolating Critical Workflows

Workflows in separate repository

- Heavily restrict access to `ndc-security-demo/reusable-workflows`

hello-world / .github / workflows / terraform.yaml

Sofia Lindqvist use secrets ✓

Code Blame 18 lines (18 loc) · 481 Bytes Code 55% faster with GitHub Copilot

```yaml
name: run terraform plan and apply
on:
  pull_request:
    branches:
      - main
  push:
    branches:
      - main
permissions:
  id-token: write
  contents: read
jobs:
  terraform:
    uses: ndc-security-demo/reusable-workflows/.github/workflows/reusable-terraform.yaml@main
    secrets:
      azure_tenant_id: ${{ secrets.AZURE_TENANT_ID }}
      azure_client_id: ${{ secrets.AZURE_CLIENT_ID }}
      azure_subscription_id: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
```

Use in `sub` claim

```
>gh api
repos/ndc-security-demo/hello-world/actions/oidc/customization/sub

{
  "use_default": false,
  "include_claim_keys": [
    "repo",
    "job_workflow_ref"
  ]
}
```

Example claim:

```
"sub":
"repo:ndc-security-demo/hello-world:job_workflow_ref:ndc-s
ecurity-demo/reusable-workflows/.github/workflows/reusable
-terraform.yaml@refs/heads/main"
```

Attempting to bypass

```yaml
12  jobs:
13    terraform:
14      uses: ndc-security-demo/reusable-workflows/.github/workflows/reusable-terraform.yaml@main
```

az login

```
1  ▶ Run azure/login@v2
11 Running Azure CLI Login.
12 /usr/bin/az cloud set -n azurecloud
13 Done setting cloud: "azurecloud"
14 Federated token details:
15  issuer - https://token.actions.githubusercontent.com
16  subject claim - repo:ndc-security-demo/hello-world:job_workflow_ref:ndc-security-demo/hello-world/.github/workflows/terraform.yaml@refs/pull/8/merge
17 Attempting Azure CLI login by using OIDC...
18 Error: AADSTS700213: No matching federated identity record found for presented assertion subject 'repo:ndc-security-demo/hello-world:job_workflow_ref:ndc-security-demo/hello-world/.github/workflows/terraform.yaml@refs/pull/8/
   merge'. Check your federated identity credential Subject, Audience and Issuer against the presented assertion. https://learn.microsoft.com/entra/workload-id/workload-identity-federation Trace ID: ea4ec1a3-38e6-4988-9a5b-
   d1ff6a527b00 Correlation ID: 155c2903-40da-473c-a3fe-e7d969bbad70 Timestamp: 2025-01-17 14:00:53Z
```

```
l@refs/pull/8/merge

ld job_workflow_ref:ndc-security-demo/hello-world/.github/workflows/terraform.yaml@refs/pull/8/
soft.com/entra/workload-id/workload-identity-federation Trace ID: ea4ec1a3-38e6-4988-9a5b-
```

Is it safe?

- Attacker cannot modify the reusable workflow
- … but can they control what the workflow does?
- Code injection in reusable workflow would be bad
  - Script injections
  - Running code from attacker branch

DEMO: Code Execution in terraform plan in reusable workflow

Summary

GitHub config

- Don’t put security measures against collaborators in the workflow itself
- Use branch protections
- Use environments tied to protected branches
- Scope secrets to environments, not repo or org
- Avoid script injections by using `env` in workflows

OIDC config in the cloud

- Be as specific as you can when asserting on a `sub` (or other claims)
- If the cloud identity should be protected the `sub` must be tied to:
  - A protected branch in a specific repo (e.g. via `repo` and `ref`)
  - A protected environment in a specific repo (e.g. via `repo` and `environment`)
  - A workflow from a “safe” repository (e.g. via `repo`, `ref` and `job_reusable_ref`)
- Follow the Principle of Least Privilege

Blog post:

https://binarysecurity.no/posts/2025/08/securing-gh-actions-part1

https://binarysecurity.no/posts/2025/09/securing-gh-actions-part2

Thank You!

BINARY{SECURITY}