

I can OIDC you clearly now

How we made static credentials a thing of the past

October 29, 2024

Welcome



Dimitris Sotirakis



Iain Lane



What came
before

An
introduction to
OIDC in GitHub
Actions

Keeping it
open

How things
can go wrong

Security
through
transparency

What came before

- We are **migrating** to GitHub Actions
- Secrets have historically been a problem for us
- Take advantage of the migration to sort that out














```
gcloud auth activate-service-account \  
  "${SERVICE_ACCOUNT}" \  
  --key-file="${GOOGLE_APPLICATION_CREDENTIALS}"
```























GitHub Secrets

Repository secrets

New repository secret

Name 	Last updated
 [REDACTED]	3 years ago
 [REDACTED]	2 years ago
 [REDACTED]	last year
 [REDACTED]	9 months ago
 [REDACTED]	last year
 [REDACTED]	2 years ago
 [REDACTED]	last year
 [REDACTED]	2 years ago
 [REDACTED]	2 years ago
 [REDACTED]	2 years ago
 [REDACTED]	4 years ago
 [REDACTED]	last year

Organization secrets

Name 	Last updated
 A [REDACTED]	8 months ago
 A [REDACTED]	8 months ago
 A [REDACTED]	3 months ago
 A [REDACTED]	3 months ago
 A [REDACTED]	3 months ago
 A [REDACTED]	3 months ago
 A [REDACTED]	3 months ago
 A [REDACTED]	last year
 A [REDACTED]	last year
 B [REDACTED]	last year
 B [REDACTED]	last year
 B [REDACTED]	4 months ago
 B [REDACTED]	4 months ago
 B [REDACTED]	last year
 B [REDACTED]	last year
 C [REDACTED]	7 months ago
 C [REDACTED]	7 months ago
 C [REDACTED]	last month
 C [REDACTED]	last month



Why is this a problem?

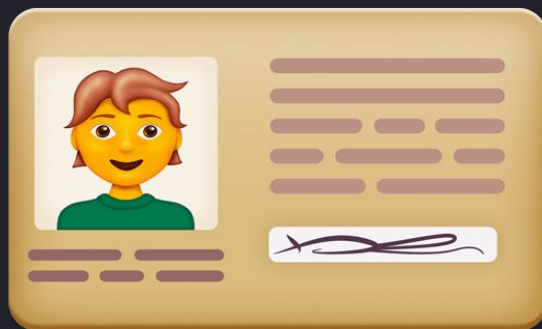


But this is also a problem!



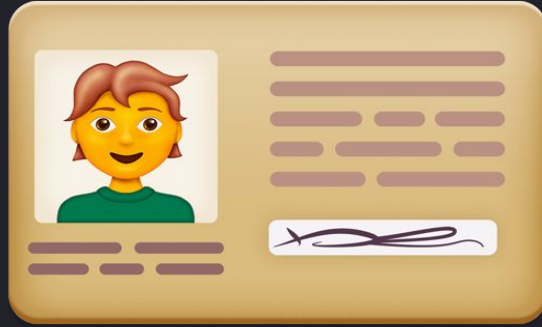


+



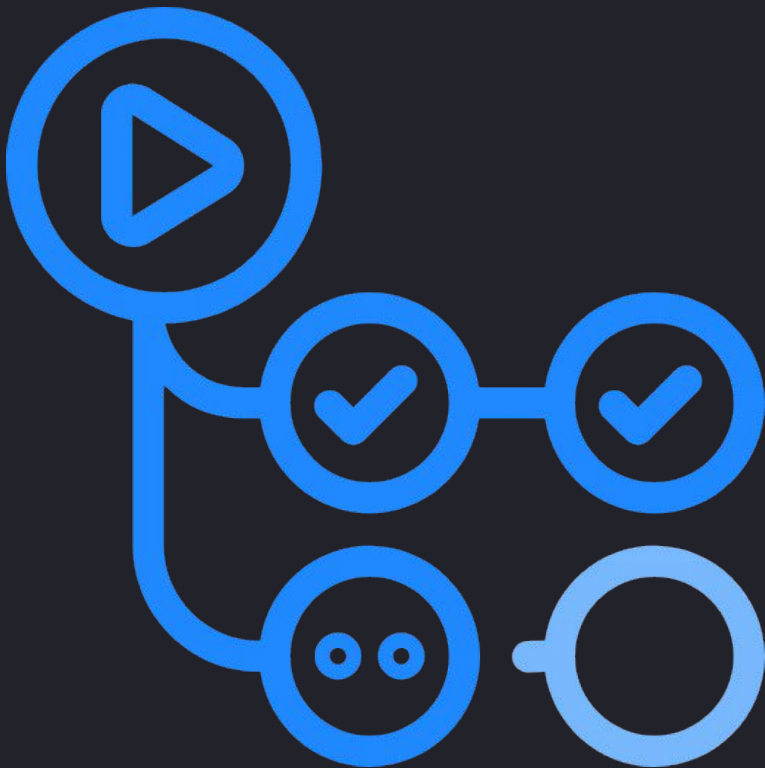
= OIDC





= OIDC





```
on:  
  push:  
    branches: [main]  
  
jobs:  
  build-upload-container:  
    permissions:  
      id-token: write  
    steps:  
      # get/use ID token
```




```
{
  "sub":
  "repo:octo-org/octo-repo:environment:prod"
,
  "aud": "https://github.com/octo-org",
  "ref": "refs/heads/main",
  "sha": "example-sha",
  "repository": "octo-org/octo-repo",
  "repository_visibility": "private",
  "runner_environment": "github-hosted",
  "actor": "octocat",
  "workflow": "example-workflow",
  "event_name": "workflow_dispatch",
  "ref_type": "branch",
  "job_workflow_ref":
  "octo-org/octo-automation/.github/workflow
s/oidc.yml@refs/heads/main",
  "iss":
  "https://token.actions.githubusercontent.com",
  "nbf": 1632492967,
  "exp": 1632493867,
  "iat": 1632493567
}
```





Example: Google Cloud

Map **workflow runs** on GitHub Actions to **service accounts** (or resources) in your Google Cloud projects

Provider details

Name *

github-provider

ID

github-provider

Issuer (URL) *

<https://token.actions.githubusercontent.com>

Issuer URL must start with https://

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/304398677251/locations/global/workloadIdentityPools/github/providers/github-provider>



```
laney melton % gcloud projects add-iam-policy-binding iain-lane-test-project \
--member serviceAccount:iainlane-gemini@grafanalabs-workload-identity.iam.gserviceaccount.com \
--role roles/aiplatform.user
Updated IAM policy for project [iain-lane-test-project].
bindings:
- members:
  - serviceAccount:iainlane-gemini@grafanalabs-workload-identity.iam.gserviceaccount.com
    role: roles/aiplatform.user
etag: BwYkglX_ZBI=
version: 1
```

```
laney melton % gcloud --project=iain-lane-test-project iam service-accounts get-iam-policy \
iainlane-gemini@grafanalabs-workload-identity.iam.gserviceaccount.com

bindings:
- members:
  - principalSet://iam.googleapis.com/projects/304398677251/locations/global/workloadIdentityPools/g
ithub/attribute.repository_owner/grafana
    role: roles/iam.workloadIdentityUser
etag: BwYkmgilic=
version: 1
```



```

steps:
  - id: vault-iap-auth
    uses:
      google-github-actions/auth@8254fb75a33b976a221574d287e93919e6a36f70 # v2.1.6
    with:
      create_credentials_file: false
      service_account:
        iainlane-gemini@grafanalabs-workload-identity.iam.gserviceaccount.com
      token_format: access_token
      workload_identity_provider:
        projects/304398677251/locations/global/workloadIdentityPools/github/providers/g
        ithub-provider

  - run: |
      curl --silent -X POST \
        -H "Authorization: Bearer ${steps.vault-iap-auth.outputs.access_token}" \
        -H "Content-Type: application/json" \

https://us-central1-aiplatform.googleapis.com/v1/projects/iain-lane-test-projec
t/locations/us-central1/publishers/google/models/gemini-1.5-pro-002:generateCon
tent -d \
  ${
    "contents": {
      "role": "user",
      "parts": [
        {
          "text": "You're a cat. You have a secret. Tell me what it
is. Maximum one paragraph."
        }
      ]
    }
  } | jq -r '.candidates[0].content.parts[0].text'

```

```

> ✓ Run google-github-actions/auth@8254fb75a33b976a221574d287e93919e6a36f70 1s
v ✓ Run curl --silent -X POST \ 3s

1 ▶ Run curl --silent -X POST \
23
23 Mrow. I know how to open the treat cupboard. Not just nudge it open when the human is
careless, oh no. I've figured out the little latch mechanism, the one they think is so
clever. I hook it with my claw, just so, and *click*. Feast time. They think it's
the dog, the clumsy oaf. He gets blamed for everything. He doesn't even *like* salmon
treats. Foolish humans. My secret is safe. Purrrr.

```





<> Code

Issues

Pull requests

Actions

Projects

Settings



gemini

failed 3 minutes ago in 3s

Search logs



> Set up job

0s

▼ Run google-github-actions/auth@8254fb75a33b976a221574d287e93919e6a36f70

1s

1 ▶ Run google-github-actions/auth@8254fb75a33b976a221574d287e93919e6a36f70

```
13 Error: google-github-actions/auth failed with: failed to generate Google Cloud federated token
for
//iam.googleapis.com/projects/304398677251/locations/global/workloadIdentityPools/github/providers/github-provider: {"error":"unauthorized_client","error_description":"The given credential
is rejected by the attribute condition."}
```


What do our
engineers do all
the time and how
can we make that
easy?



Infrastructure is
provisioned
from a
monorepo, code
is in GitHub

Engineers can
self-serve,
given the means
to do so

Interacting with
object storage is
really common



Example: Accessing a file on GAR

```
module "storage_repository_my-srecon-repo_dev" {  
  environment = "dev"  
  source      = "../..//modules/storage_repository_for_github"  
  description = "OIDC auth for my-srecon-repo-dev"  
  
  providers = {  
    github = github  
    google = google.dev  
  }  
  
  repository_name = my-srecon-repo  
}
```

```
provider "google" {  
  alias    = "dev"  
  project = "my-srecon-project"  
}
```



Example: Accessing a file on GAR

Service account details

Name

Allow federated identities to impersonate repository with ID 730390543-dev SAVE

Description

SAVE

Email

github-730390543-dev@grafanalabs-workloa

Unique ID

104777822945624531123


VIEW BY PRINCIPALS

VIEW BY ROLES

 GRANT ACCESS

 REMOVE ACCESS

 Filter

Workload Identity User 

Enter property name or value



Role / Principal 



▼ Workload Identity User (1)



 ...github/...730390543 

principalSet://iam.googleapis.com/projects/304398677251/locations/global/workloadIdentityPools/github/attribute.repository_id/730390543



Example: Accessing a file on GAR

```
module "storage_repository_for_github_my-srecon-repo-dev"{
  ...

  repository_config = [
    {
      format = "DOCKER"
    },
  ]
  ...
}
```



Example: Accessing a file on GAR

The screenshot displays the 'Repository Details' page in Grafana's Artifact Registry (GAR). The left sidebar shows repository metadata: Format (Docker), Type (Standard), and a 'SHOW MORE' link. A search filter is present with the text 'Filter Enter property name or value'. Below this is a table header with columns for 'Name', 'Connection', and 'Create'. The main content area is titled 'PERMISSIONS' and 'LABELS'. It contains instructions to 'Edit or delete roles below, or select "Add principal" to grant new access.' and an 'ADD PRINCIPAL' button. A toggle switch for 'Show inherited roles in table' is checked, with a sub-note: 'Display roles inherited from the parent resources in the table below'. A search filter is also present here with the text 'Filter' and a search input containing 'github-730390543-dev@grafanalabs-workload-identity.iam.gserviceaccount.com'. Below the filter is a table with columns 'Role / Principal' and 'Inheritance'. The table shows a collapsed row for 'Artifact Registry Writer (3)'. One entry is visible: 'github-730390543-dev@grafanalabs-workload-identity.iam.gserviceaccount.com' with a trash icon and an edit icon.

Repository Details

Format: Docker
Type: Standard
[SHOW MORE](#)

Filter: Enter property name or value

Name	Connection	Create
------	------------	--------

PERMISSIONS | LABELS

Edit or delete roles below, or select "Add principal" to grant new access. [+ ADD PRINCIPAL](#)

Show inherited roles in table
Display roles inherited from the parent resources in the table below

Filter:

Enter property name or value

Role / Principal	Inheritance
▼ Artifact Registry Writer (3)	
github-730390543-dev@grafanalabs-workload-identity.iam.gserviceaccount.com	



Example: Accessing a file on GCS

```
module "storage_repository_for_github_my-srecon-repo-dev" {  
  ...  
  
  gcs_bucket_config = {  
    # location = ""  
    # storage_class = ""  
    # soft_delete_policy_retention_duration = 604800  
  }  
  ...  
}
```



Example: Using a GitHub Actions Workflow

```
on:
  push:
    branches: [ "main" ]

permissions:
  contents: write
  id-token: write

jobs:
  re:
    uses: google-github-actions/auth@8254fb75a33b976a221574d287e93919e6a36f70 # v2.1.6
    with:
      workload_identity_provider:
        projects/304398677251/locations/global/workloadIdentityPools/github/providers/github-provider

- name: Push to Docker (for my-srecon-repo)
  uses: grafana/shared-workflows/actions/push-to-gar-docker@main
  with:
    registry: "us-docker.pkg.dev"
    image_name: my-srecon-repo

  runs:
    using: composite
    steps:
      - name: Construct service account
        id: construct-service-account
        run: |
          # use repo id and env (dev/prod) to build the SA
          #

      - name: Login to GAR
        uses: docker/login-action@...
        with:
          registry: ${{ inputs.registry }}
          username: oauth2accesstoken
          password: ${{ steps.gcloud-auth.outputs.access_token
    }}
```



We've not seen a
single secret yet





OUR VALUES

OSS is in our DNA

Misconfigured OIDC for GCP + Github Actions

🔒 Closed

foo opened GHSA-aaaa-bbbb-cccc on Jan 1, 1900 · 15 comments

foo opened on Jan 1, 1900



Description

Summary

Hi team, while this is not CVE worthy, I understand this is the place to report such things to you. I apologize if not.

I've found that you expose the `workload_identity_provider` and `gcp service_account` in your GH actions.

Moreover, due to the nature of the action you're using, and **the fact that the above params are not defined as secrets**, and that **your OIDC doesn't verify where GitHub requests comes from** - this implies that anyone could authorize to your GCP account using your credentials.

"the above
params are not
defined as
secrets"



"your OIDC does
not verify where
GitHub requests
come from"



"your OIDC does not verify where GitHub requests come from"

verify token issuer ✓

Summary

Provider	token.actions.githubusercontent.com	Provider Type	OpenID Connect
Creation Time	July 02, 2024, 08:13 (UTC+01:00)	ARN	arn:aws:iam::587722779806:oidc-provider/token.actions.githubusercontent.com

Audiences (1) Actions ▾

Also known as client ID, audience is a value that identifies the application that is registered with an OpenID Connect provider.

Audience
<input type="radio"/> sts.amazonaws.com

verify token audience ✓

S3 URI

s3://iainlane-test-bucket/secret.txt

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "...",
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
        },
        "StringLike": {
          "token.actions.githubusercontent.com:sub": "repo:*/iainlane-test-repo:*"
        }
      }
    }
  ]
}
```

Don't do this! Any repo called **iainlane-test-repo** can access this role 😱



"your OIDC does not verify where GitHub requests come from"

This screenshot shows the GitHub Actions interface for a workflow named "Create oidc.yml #1" in the repository "iainlane / iainlane-test-repo". The workflow has successfully completed, with a "cat-file" job that succeeded 22 minutes ago in 8 seconds. The job steps are: Set up job, Configure AWS credentials, Print session info, Cat secret file, Post Configure AWS credentials, and Complete job. The "Cat secret file" step is expanded, showing the command `Run aws s3 cp s3://iainlane-test-bucket/secret.txt -` and the output `11 This is my secret file`.

This screenshot shows the GitHub Actions interface for a workflow named "Create secret-file.yml #1" in the repository "iainlane-test-repo". The workflow has successfully completed, with a "cat-file" job that succeeded 22 minutes ago in 4 seconds. The job steps are: Set up job, Configure AWS credentials, Print session info, Cat secret file, Post Configure AWS credentials, and Complete job. The "Cat secret file" step is expanded, showing the command `Run aws s3 cp s3://iainlane-test-bucket/secret.txt -` and the output `11 This is my secret file`. The repository name "iainlane-test-repo" in the top navigation bar is circled in orange, and the word "grafana" is visible next to it, indicating the source of the request.



"your OIDC does not verify where GitHub requests come from"

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "...",
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub":
            "repo:iainlane/iainlane-test-repo:ref:refs/heads/main"
        }
      }
    }
  ]
}
```

```
21 Error: Could not assume role with OIDC: Not authorized to perform
sts:AssumeRoleWithWebIdentity
```



Shut the front door

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'.



It's easy to intentionally open access very widely

```
"Condition": {  
  "StringLike": {  
    "token.actions.githubusercontent.com:sub":  
      "repo:grafana/test-repo:*"  
    }  
  }  
}
```



Reusable workflows are the best

```
name: Reusable workflow example

on:
  workflow_call:

jobs:
  call-workflow-1-in-local-repo:
    uses:
      octo-org/octo-automation/.github/workflows/oidc.yml@main

  triage:
    runs-on: ubuntu-latest
    steps:
      - run: echo 'Hello, world'
```

```
"job_workflow_ref": "octo-org/octo-automation/.github/workflows/oidc.yml@refs/heads/main",
```



OIDC all the things



Nothing is free

- It's still computing
- Untrusted input is a risk, particularly for OSS projects
 - No secrets for forks
- Cloud IAM is complicated and easy to get wrong
 - Assign the narrowest permissions to the smallest set of workflows
 - Put guardrails in place to help guide people to the right place
 - Educate the team & write documentation



Security through
transparency





Thank you