

OIDC and CICD: Why Your CI Pipeline Is Your Greatest Security Threat



Ted Hahn,



TCB TECHNOLOGIES

thahn@tcbtech.com

tcbtech.com/oidc-cicd

github.com/tcbtechnologies/oidc-cicd



Mark Hahn,



Qualys®

mhahn@qualys.com

gitlab.com/tcbtech/oidc-talk/

Configuring your CI correctly is vital



































- History of credentials
 - Why using long lived tokens is insecure
 - Examples of what can go wrong
- How to use OIDC
 - Create the roles in your infra (AWS, Kube Clusters)
 - Setup pipelines to use OIDC
 - Section off privileges into roles attached to branches
- Examples using OIDC:
 - Create OIDC Providers permissions in AWS (also GCP and Azure, if there is time)
 - Configure GitHub (And CircleCI, Gitlab)
 - Run a pipeline and see identity (and changing by pipeline stage)

History of Credentials



- Manually entered for builds that ran by hand
- Automated builds
- Access tokens
- SAML / SAML-Like
- OAuth2
- OIDC

Straw Poll: How many of your credentials look like this?

Repository secrets New repository secret

Name 	Last updated
 ALIBABA_CLOUD_ACCESS_KEY_ID	1 minute ago  
 ALIBABA_CLOUD_ACCESS_KEY_SECRET	1 minute ago  
 AWS_ACCESS_KEY_ID	4 months ago  
 AWS_SECRET_ACCESS_KEY <small>This secret overrides an organization secret</small>	4 months ago  
 GCP_ACCESS_KEY	4 months ago  
 GF_AUTH_GOOGLE_CLIENT_ID	3 minutes ago  
 GF_AUTH_GOOGLE_CLIENT_SECRET	3 minutes ago  
 OPENAI_API_KEY	8 minutes ago  
 REACT_APP_API_KEY	7 minutes ago  
 REACT_APP_AUTH_DOMAIN	7 minutes ago  
 STRIPE_TOKEN	10 minutes ago  

Organization secrets Manage organization secrets

Name 	Last updated
 AWS_SECRET_ACCESS_KEY <small>This secret is overridden by a repository secret</small>	2 months ago

Long lived credentials like this are obviously insecure

- No rotation
- Coarse grained access
- Poor attribution

What can go wrong

- Secrets can leak easily!

```
echo $CLOUD_SECRET | base64
```

- Vendors can leak secrets - CircleCI had a leak in 2023
"We recommended that all customers rotate their secrets, including OAuth tokens, Project API Tokens, SSH keys, and more"
- Vendors can get hacked via CICD - CloudFlare Thanksgiving Day 2023 Incident

CD - Confused Deputy

- Developer Branches are untrusted code
- Your CI configuration lives in the repo itself now
- There are no guarantees here

How to use OIDC

Three Simple Steps:

- Create Roles in your Cloud
- Setup pipelines to use OIDC
- Section off privileges into roles attached to branches

Create the roles in your infra (AWS, Kube Clusters)

Create roles that provide the access you need

Setup the policies/permissions that to the least privileges necessary

For Example

```
Statement = [  
  {  
    Action = "sts:AssumeRoleWithWebIdentity"  
    Effect = "Allow"  
    # Or Gitlab, or CircleCI, or...  
    Sid     = "Github"  
    Principal = {  
      Federated = aws_iam_openid_connect_provider.github.arn  
    }  
    Condition = {  
      "StringEquals" : {  
        "token.actions.githubusercontent.com:aud" : "sts.amazonaws.com",  
      },  
      . . .  
    }  
  }  
]
```

Setup pipelines to use OIDC

Create the OIDC token in your pipeline

Using the pipeline syntax for your provider

(they just create them)

```
- name: Configure AWS credentials from Test account
  uses: aws-actions/configure-aws-credentials@v4
  with:
    role-to-assume: arn:aws:iam::783153433147:role/github-actions
    aws-region: us-east-1
```

OIDC Token Example

```
{
  "aud": "sts.amazonaws.com"
  "sub": "project_path:tcbtech/oidc-talk:ref_type:branch:ref:mark",
  "iss": "https://gitlab.com",
  "iat": 1705018870,
  "nbf": 1705018865,
  "exp": 1705022470,

  "namespace_id": "8163212",
  "namespace_path": "tcbtech",
  "project_id": "53428581",
  "project_path": "tcbtech/oidc-talk",
  . . .
}
```

Section off privileges into roles attached to branches

Setup the roles or service accounts

Attach them to the proper policies and permissions

Attach them to branches

Section off privileges into roles attached to branches : Example

```
...  
  "StringLike" : {  
    "gitlab.com:sub" : "project_path:${var.gitlab_org}/${var.gitlab_repo}:"  
  }  
...
```

Separate the protected branch from the user branches

```
# arn:aws:iam::905418421134:role/github-actions-master

Trust Relationship:
  . . .
  "StringLike" : {
    "gitlab.com:sub" : "project_path:${var.gitlab_org}/${var.gitlab_repo}:master"
  }
  . . .

Policy:
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr: . . .",
        "ecr:UploadLayerPart",
        "ecr:CompleteLayerUpload",
        "ecr:PutImage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

```
# arn:aws:iam::905418421134:role/github-actions

Trust Relationship:
  . . .
  "StringLike" : {
    "gitlab.com:sub" : "project_path:${var.gitlab_org}/${var.gitlab_repo}:*"
  }
  . . .

Policy:
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr: . . .",
        "ecr:UploadLayerPart",
        "ecr:CompleteLayerUpload",
        "ecr:PutImage"
      ],
      "Effect": "Allow",
      "Resource": "*stage*"
    }
  ]
}
```

Separate S3 permissions for the protected branch

```
# arn:aws:iam::905418421134:role/github-actions-master

Trust Relationship:
  . . .
  "StringLike" : {
    "gitlab.com:sub" : "project_path:${var.gitlab_org}/${var.gitlab_repo}:master"
  }
  . . .

Policy:
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3: . . .",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:PutObject",
      ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:::yourorg_prod_web",
                   "arn:aws:s3:::yourorg_prod_web/*" ]
    }
  ]
}
```

```
# arn:aws:iam::905418421134:role/github-actions

Trust Relationship:
  . . .
  "StringLike" : {
    "gitlab.com:sub" : "project_path:${var.gitlab_org}/${var.gitlab_repo}:*"
  }
  . . .

Policy:
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3: . . .",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:PutObject",
      ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:::yourorg_stage_web",
                   "arn:aws:s3:::yourorg_stage_web/*" ]
    }
  ]
}
```

Demo

Show of hands, how many people use [Gitlab](#) vs [Github](#)?

GitHub action jobs for AWS and GCP

```
git commit --allow-empty -m "Demo."  
git push -f origin head:force-ci
```

GitLab action jobs for AWS and GCP

```
git push -f gitlab head:force-ci
```


Takeaways

- Prevent Developers from abusing CI's access by tying roles to protected branches
- Understand that Unit Tests in CI run as CI - And Developers run as that CI, too
- Splitting roles by pipeline makes it simple - Simple is good

References

GitHub

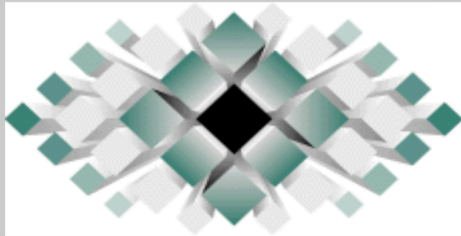
- <https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services>
- <https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/configuring-openid-connect-in-google-cloud-platform>

GitLab

- https://docs.gitlab.com/ee/ci/cloud_services/aws/index.html
- https://docs.gitlab.com/ee/ci/cloud_services/google_cloud/

MY
CICD

♥️ OIDC



TCBTECH.COM



Qualys®

<https://tcbtech.com/oidc-cicd>

```
https://github.com/tcbtechnologies/oidc-cicd
```

```
https://gitlab.com/tcbtech/oidc-talk/
```

Thank you!

<https://tcbtech.com/oidc-cicd>