From keyless to careless

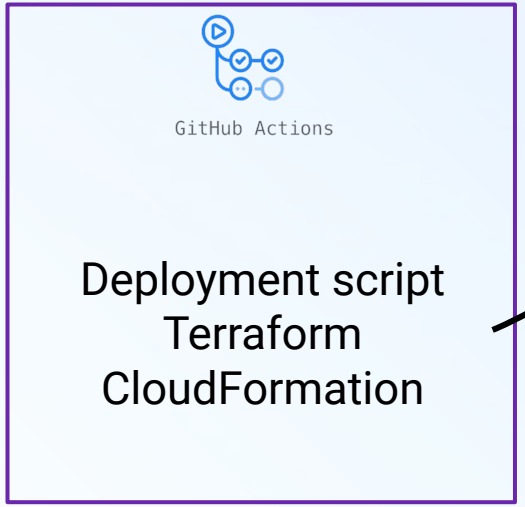
Abusing misconfigured OIDC authentication in cloud environments

Christophe Tafani-Dereeper
April 26th, 2024

SWISS CYBERSECURITY CONFERENCE
INSOMNI'HACK

CI/CD

Cloud environment

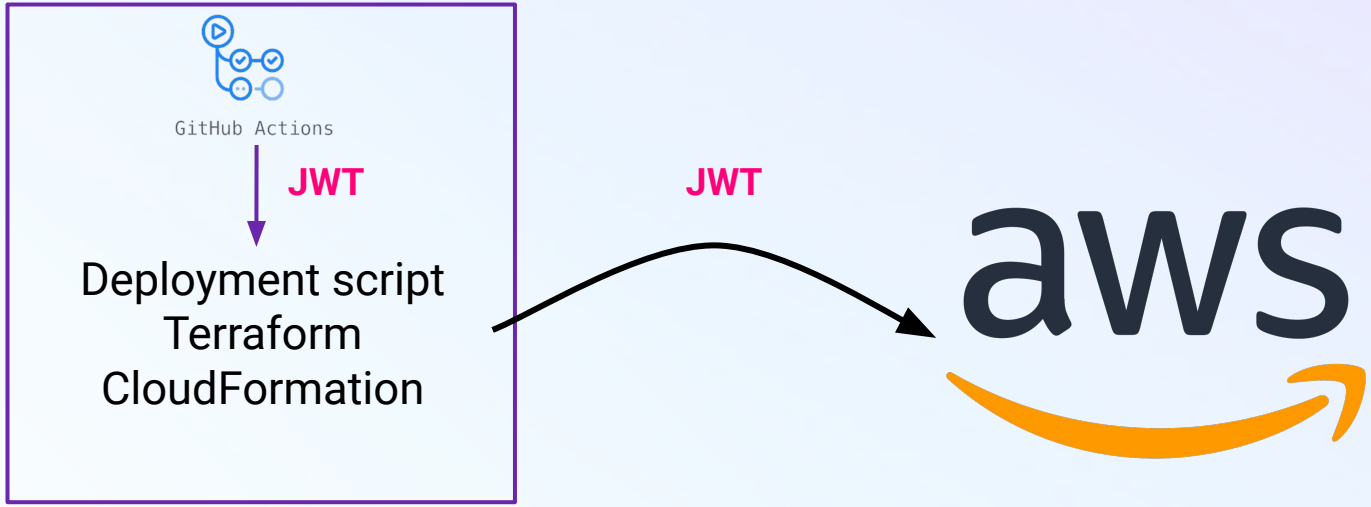


Hardcoded credentials


YOU GET BREACHED!
YOU GET BREACHED!
EVERYBODY GETS BREACHED!

CI/CD

Cloud environment




BY APPOINTMENT TO
H.M. QUEEN ELIZABETH II
MEDALLISTS
WORCESTERSHIRE MEDAL SERVICE LTD
BROMSGROVE
National Cyber
Security Centre
Thank you for your support.
NATIONAL CYBER SECURITY CENTRE
UK NCSC

Christophe Tafani-Dereeper

🇫🇷 🇨🇭

DATADOG

Previously: Hacknowledge nexthink

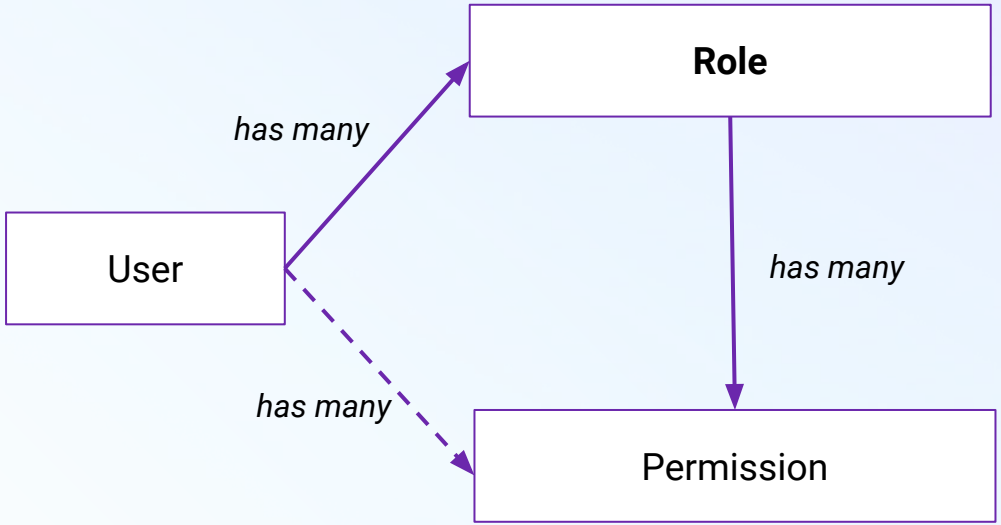
Today's agenda

- AWS IAM fundamentals
- OIDC in CI/CD environments
- Misconfigurations
- Hacking vulnerable roles at scale
- Practical attack and defense tips

AWS IAM fundamentals

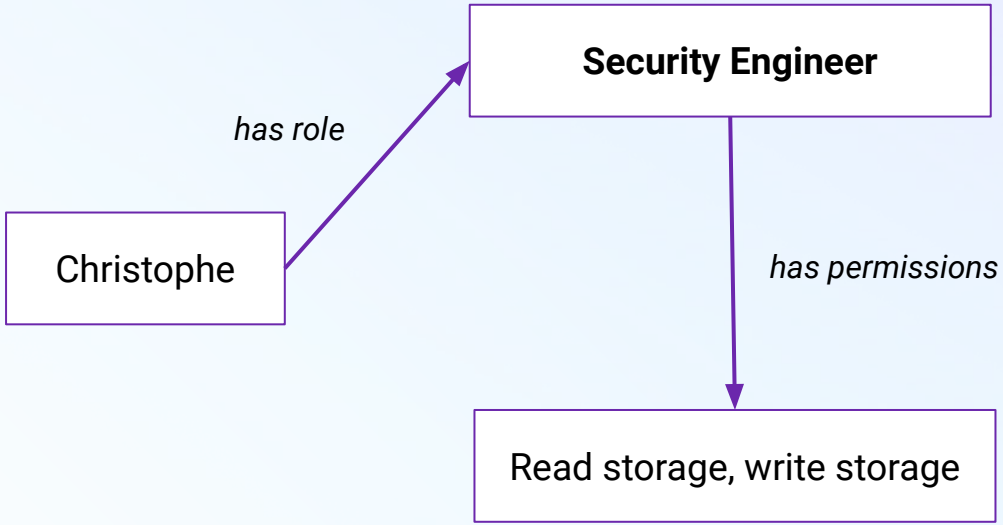
Identity types (outside AWS)

Role := Set of permissions



Identity types (outside AWS)

Role := Set of permissions



Identity types in AWS

Identity := What you can authenticate/sign-in as

IAM users

IAM roles

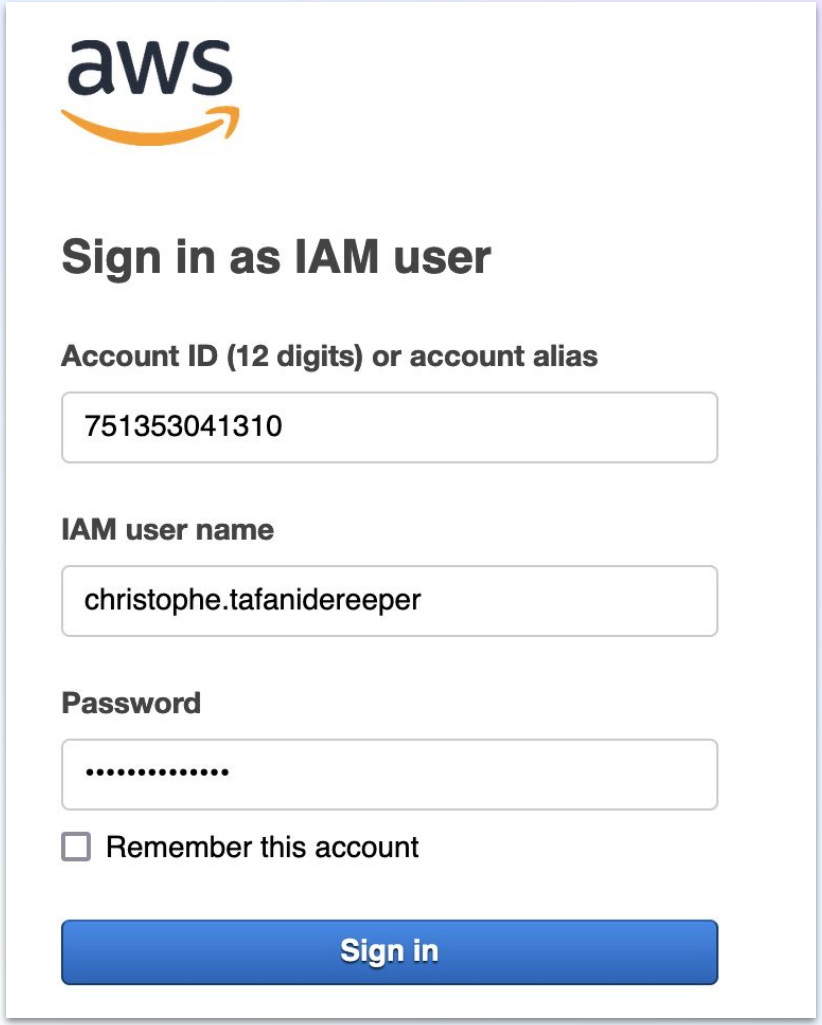
IAM users

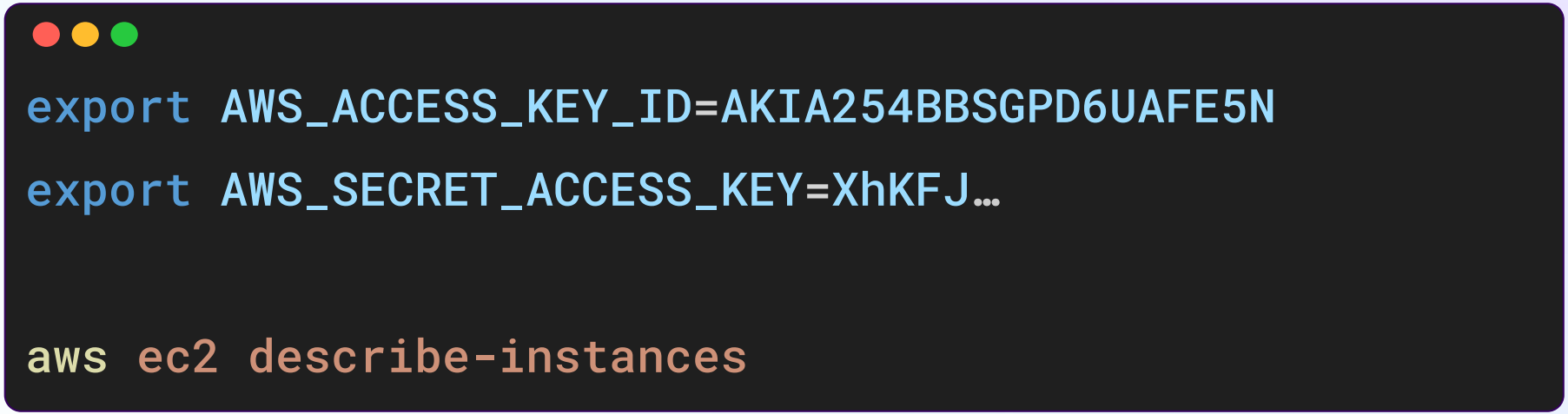
```
christophe.tafanidereeper
I_L0ve_Raclette
```

AWS Console (UI) access

```
AKIA254BBSGPD6UAFE5N
XhKFJc639Z0c…
```

Programmatic access
(AWS CLI)


aws
Sign in as IAM user
Account ID (12 digits) or account alias
751353041310
IAM user name
christophe.tafanidereeper
Password
Remember this account
Sign in


export AWS_ACCESS_KEY_ID=AKIA254BBSGPD6UAFE5N
export AWS_SECRET_ACCESS_KEY=XhKFJ…
aws ec2 describe-instances

The problem with IAM users

The problem with IAM users

Commits on Nov 4, 2020

Update enter.py ···

Update with AWS keys

th3g1itch committed 5 minutes ago

```
186     <string name="all_scoresBtnTxt">All Scores</string>
187     <string name="amezon_app_id">AKIA[redacted]JA</string>
188     <string name="amezon_app_key">Yxwz[redacted]rLg5W</string>
```

[redacted] updated .gitignore

1 contributor

2 lines (2 sloc) | 105 Bytes

```
1  aws_access_key_id = AK[redacted]DA
2  aws_secret_access_key = Cu[redacted]j1
```

Extra {"aws_access_key_id":"AK[redacted]", "aws_secret_access_key":"[redacted]"}

Save | Save and Add Another | Save and Continue Editing | Cancel

The problem with IAM users

> “For every 10k commits,
> we found on average
> 84 AWS IAM credentials leaked.”

- GitGuardian
The State of Secrets Sprawl 2022

https://arxiv.org/pdf/2307.03958.pdf (July 2023)

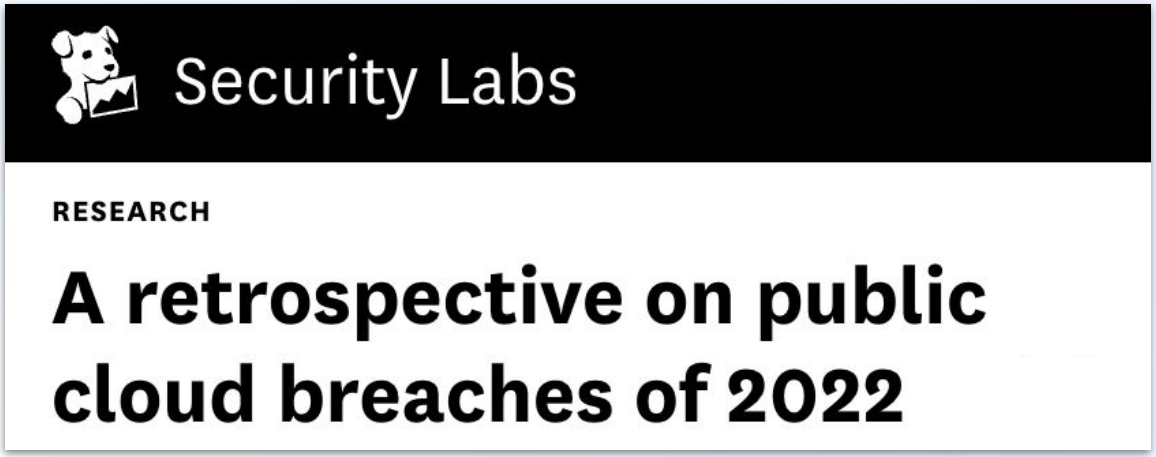
Secrets Revealed in Container Images: An Internet-wide Study on Occurrence and Impact

> “ In this paper, we analyze 337,171 images from Docker Hub and 8,076 other private registries unveiling that 8.5 % of images indeed include secrets. ”

	Regular Expressions *(Section 5.1.1 / Appendix C)*		(Distinct) Matches *(Sec. 5.1.2)*		Valid Secrets *(Section 5.1.3)*		
Domain	**Potential Threat / *(Service)* Type**		**Images**	**Variables**	**Images**	**Variables**	**Total**
	[redacted]		[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Cloud	Manage services, create new API keys, reconfigure DNS, access emails / SMS, control voice calls, read / alter private repositories, … *Alibaba*[76], *Amazon AWS*[76], *Azure*[76], *DigitalOcean*[76], *Github*[76], *Gitlab (v1, v2)*[76], *Google Cloud*[76], *Google Services*[58], *Heroku*[76], *IBM Cloud Identity Service*[76], *Login Radius*[76], *MailChimp*[58], *MailGun*[58], *Microsoft Teams*[76], *Netlify*[76], *Twilio*[58]		6,208,995 (74,460)	416 (84)	2,880	67	2,920
API	[redacted]		[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

The problem with IAM users

Misplaced IAM user keys are the most common cause for cloud data breaches!

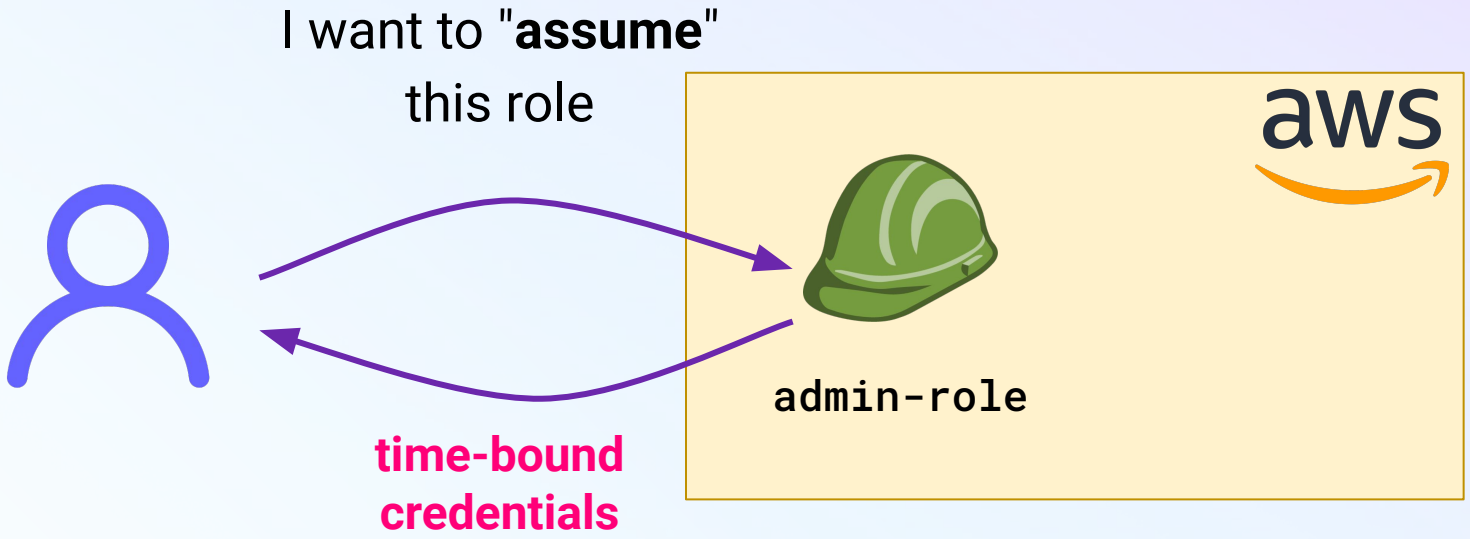


https://github.com/ramimac/aws-customer-security-incidents
https://breaches.cloud
https://blog.christophetd.fr/cloud-security-breaches-and-vulnerabilities-2021-in-review/
https://securitylabs.datadoghq.com/articles/public-cloud-breaches-2022-mccarthy-hopkins/

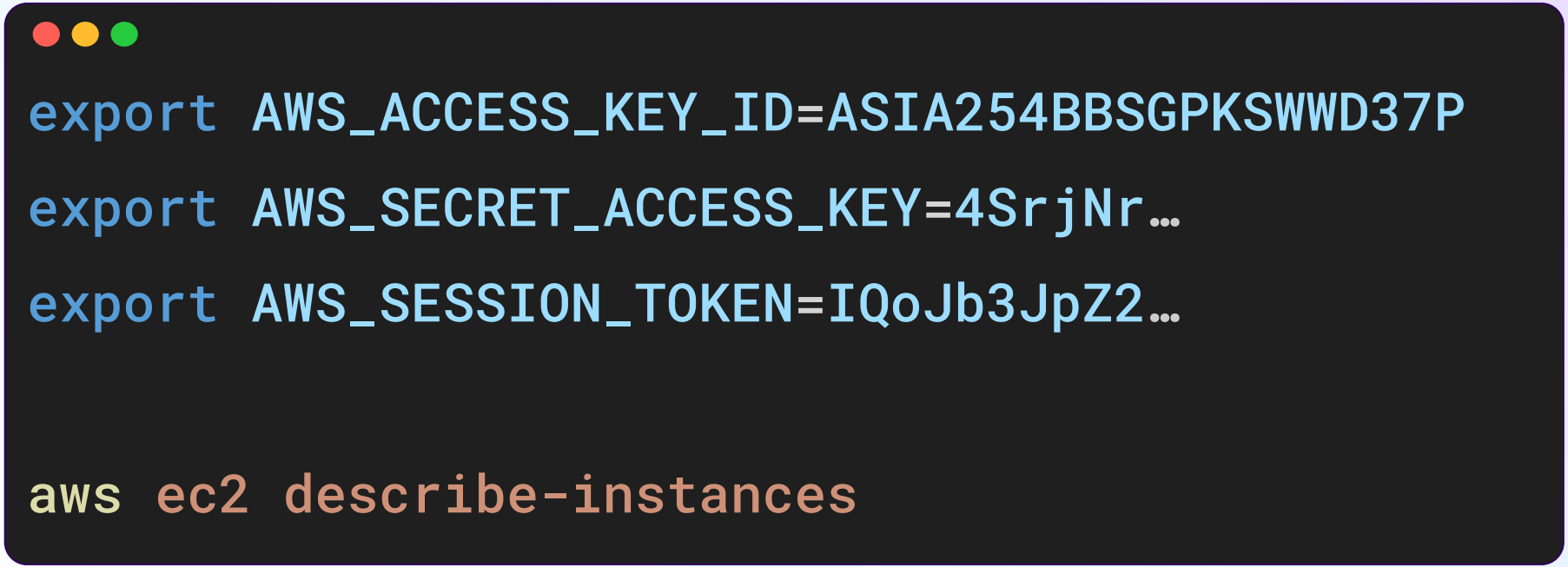
The solution: IAM Roles (STS)

- "Credentials vending machine"
- Returns time-bound credentials (max 12 hours)
- IAM roles have permissions and a trust policy

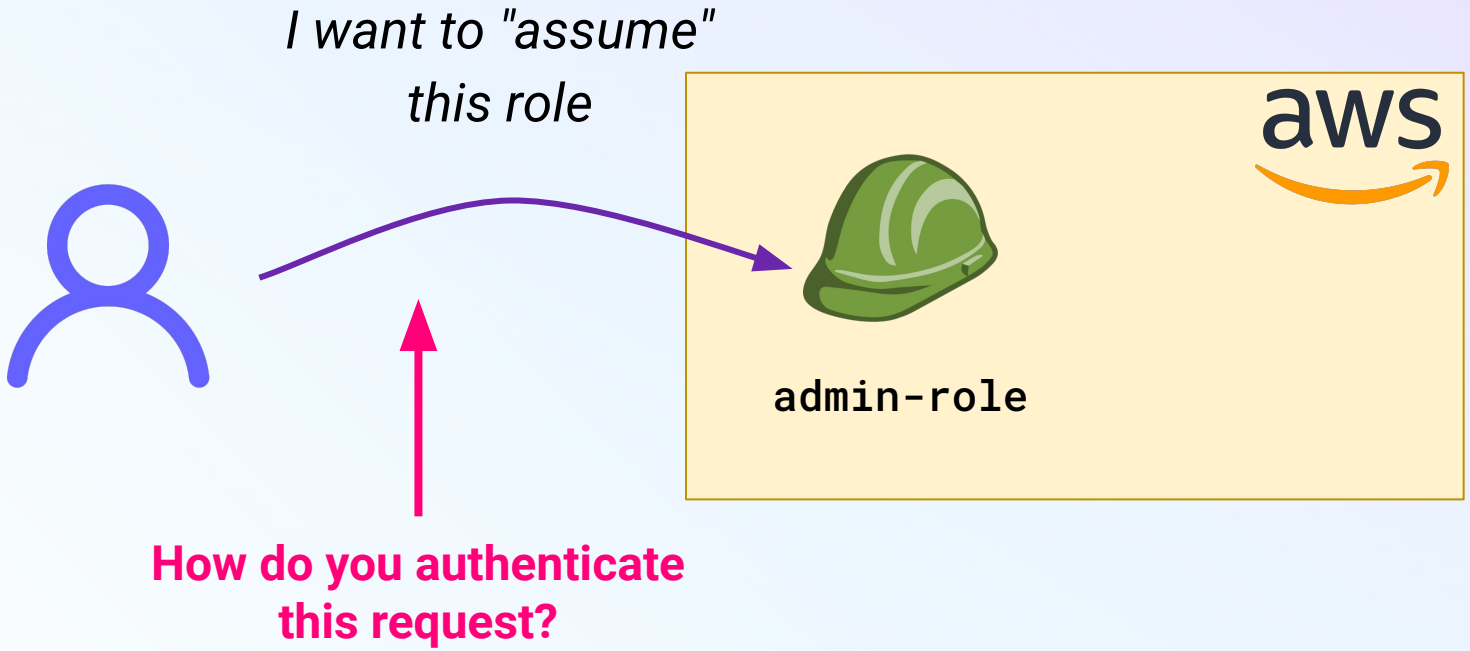
IAM Roles


I want to "**assume**"
this role
aws
admin-role
time-bound
credentials

```
$ aws sts assume role \
    --role-arn aws:iam::751353041310:role/my-role \
    --role-session-name christophe
```


export AWS_ACCESS_KEY_ID=ASIA254BBSGPKSWWD37P
export AWS_SECRET_ACCESS_KEY=4SrjNr…
export AWS_SESSION_TOKEN=IQoJb3JpZ2…
aws ec2 describe-instances

Bootstrapping identity to assume IAM roles



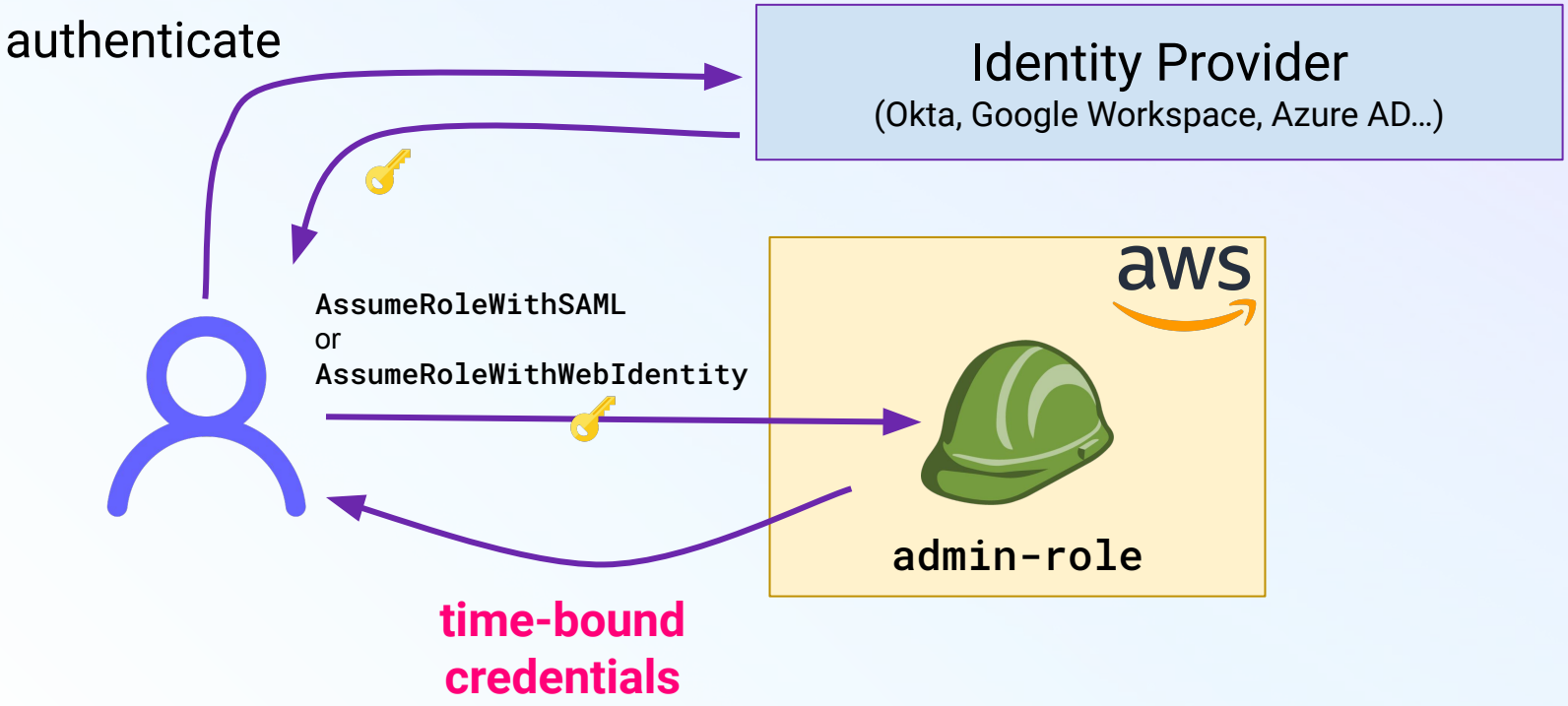
Bootstrapping identity to assume IAM roles

- Platform-managed identities for workloads
  - EC2 instance roles
  - Lambda execution roles
- IAM users / roles (humans)

When you're already inside AWS

- **SAML IdP (humans)**
- **OpenID Connect IdP (mostly machines)**

Typical identity bootstrapping for humans



Cloud authentication in CI/CD pipelines

(don't do this)

CI/CD

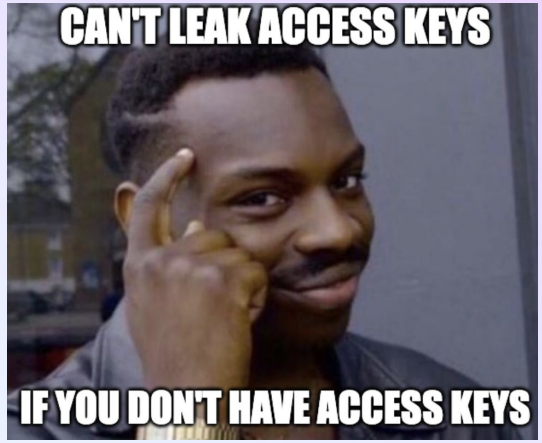
GitHub Actions

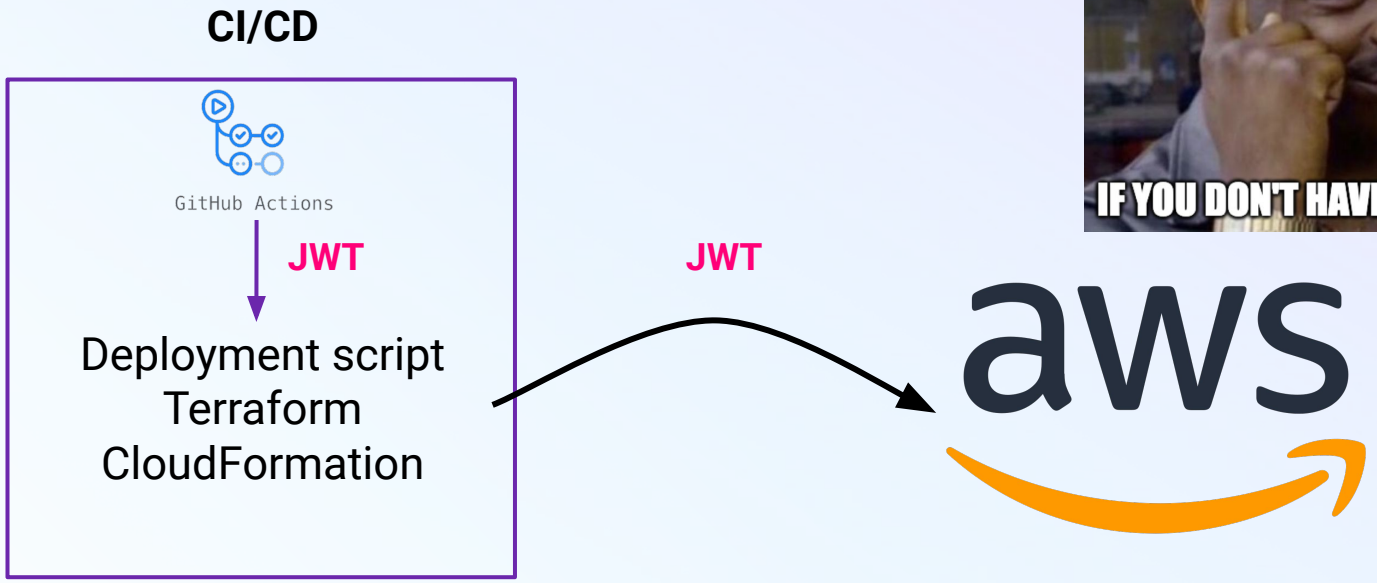
Deployment script
Terraform
CloudFormation

IAM user access keys

aws

Do this instead





Role trust policy

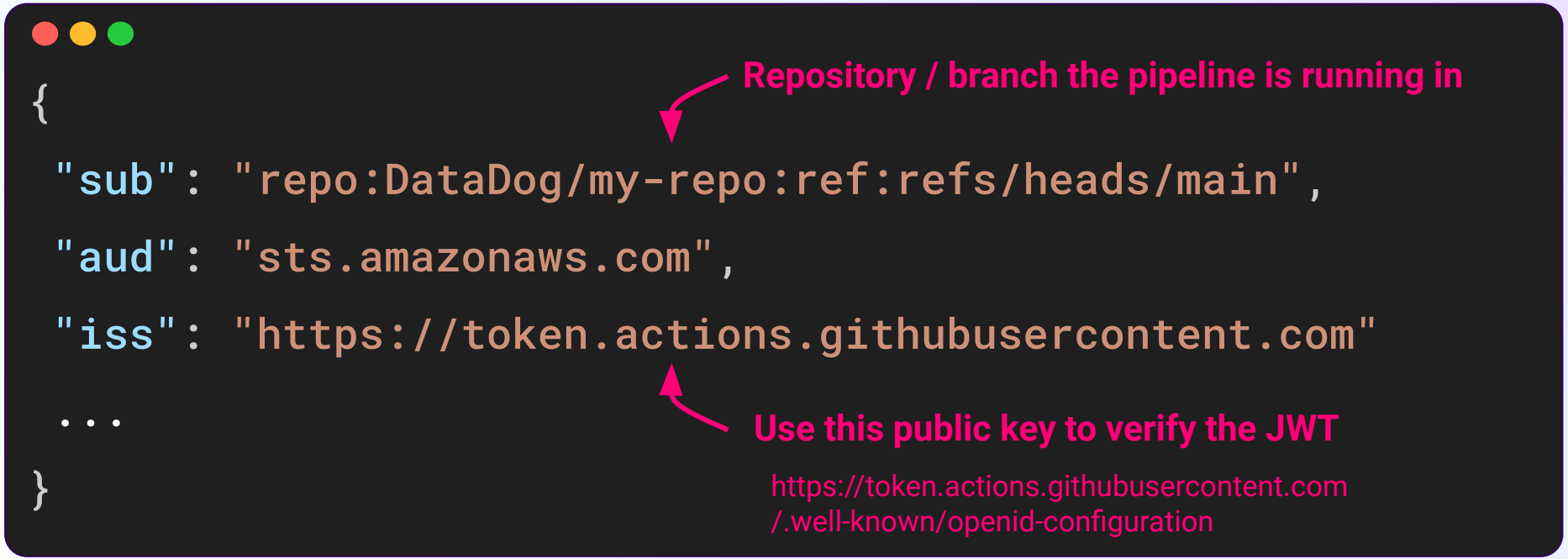
```
{
 "Effect": "Allow",
 "Principal": {
   "Federated": "arn:aws:iam::751353041310:oidc-provider/token.actions.githubusercontent.com"
 },
 "Action": "sts:AssumeRoleWithWebIdentity",
 "Condition": {
   "StringEquals": {
     "token.actions.githubusercontent.com:sub": "repo:DataDog/my-repo:ref:refs/heads/main",
     "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
   }
 }
}
```

JWT subject

Get that JWT

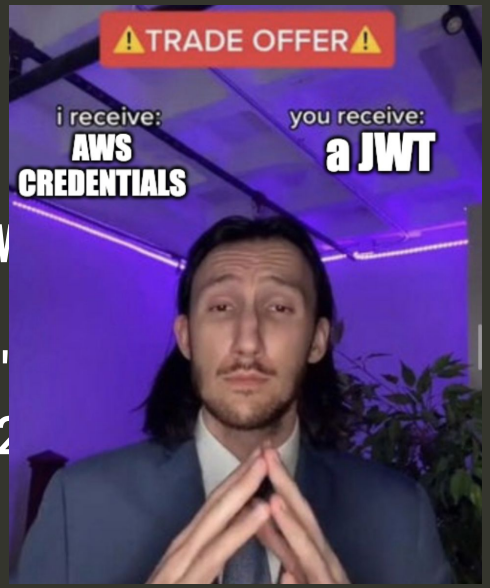
```
$ curl -H "Authorization: Bearer $ACTIONS_ID_TOKEN_REQUEST_TOKEN" \
   "$ACTIONS_ID_TOKEN_REQUEST_URL&audience=sts.amazonaws.com"


eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ikh5cTROQVRBanNuc...
```


{
"sub": "repo:DataDog/my-repo:ref:refs/heads/main",
"aud": "sts.amazonaws.com",
"iss": "https://token.actions.githubusercontent.com"
...
}
Repository / branch the pipeline is running in
Use this public key to verify the JWT
https://token.actions.githubusercontent.com
/.well-known/openid-configuration

```
$ aws sts assume-role-with-web-identity
    --role-arn aws:iam::751353041310:role/my-role \
    --role-session-name christophe \
    --web-identity-token jwt

{
    "Credentials": {
        "AccessKeyId": "ASIA254BBSGPKSW
        "SecretAccessKey": "4SrjNr..",
        "SessionToken": "IQoJb3JpZ2...
        "Expiration": "2024-04-23T17:42
    }
}
```



It's simpler *and* more secure!

```yaml
- name: Configure AWS credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    role-to-assume: arn:aws:iam::751353041310:role/deploy-role
    role-duration-seconds: 3600
    role-session-name: GitHubActions
```

What could go wrong?™

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
    "token.actions.githubusercontent.com:sub": "repo:DataDog/my-repo:ref:refs/heads/main",
  }
}
```

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  }
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:DataDog/my-repo:*",
  }
}
```

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  }
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:DataDog/*",
  }
}
```

Lax condition on JWT subject

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  }
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:DataDog*",
  }
}
```

DatadogHello/my-malicious-repo

Lax condition on JWT subject

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  }
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "*",
  }
}
```

No condition on JWT subject

```
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  }
}
```

Impact of a misconfigured trust policy

- Any GitHub Action can assume the role!
  - Only need to know/guess its ARN (AWS account ID + role name)
- Typically privileged deployment roles
- Access to the AWS environment

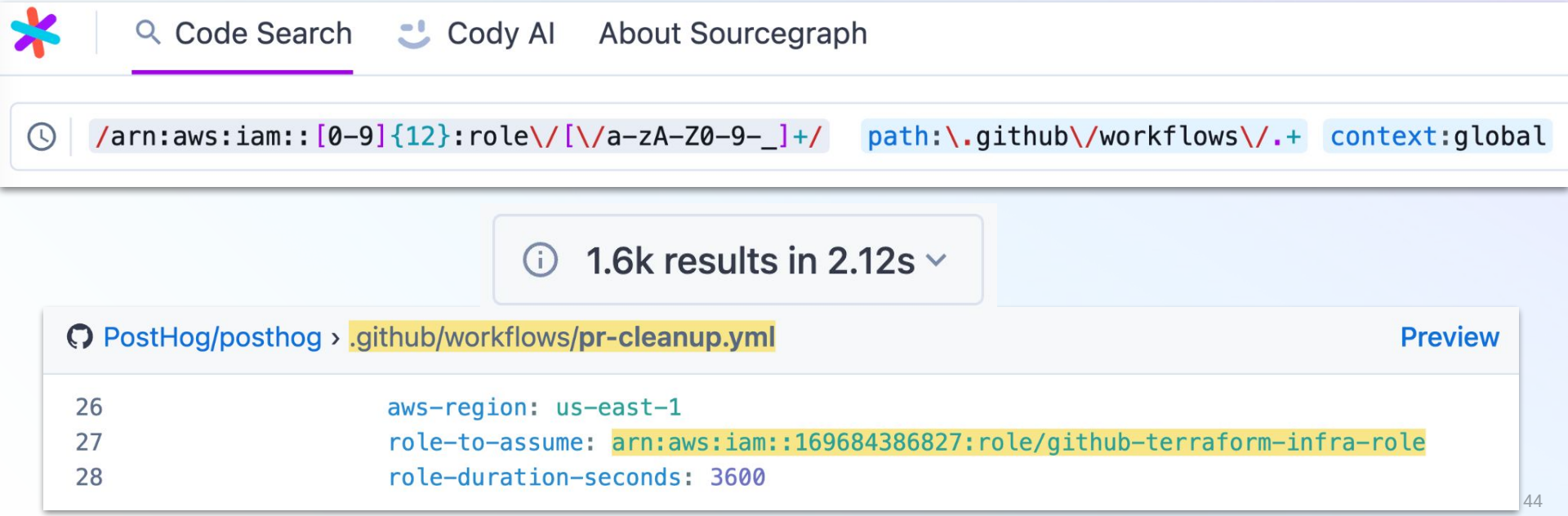
Hunting for vulnerable IAM roles

in the wild

DATADOG

Step 1: Find roles!

- Search for role ARNs in publicly-available data sources
  - GitHub, SourceGraph, build logs…



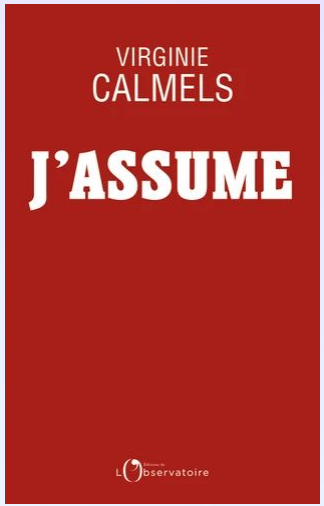
Step 2: Create your GitHub Action and retrieve a JWT

```yaml
name: Retrieve GitHub Actions token
on: workflow_dispatch
permissions:
  id-token: write # This is required for requesting the JWT
  contents: read  # This is required for actions/checkout
```

```
$ curl -H "Authorization: Bearer $ACTIONS_ID_TOKEN_REQUEST_TOKEN" \
  "$ACTIONS_ID_TOKEN_REQUEST_URL&audience=sts.amazonaws.com"
```

Step 3: Try to assume the roles

- Using `sts:AssumeRoleWithWebIdentity`
- See if you get credentials back ☺️



Our results

- Automated role collection and scanning
- Found **dozens of vulnerable roles**, granting us access to an AWS account
- Reported them responsibly
  - sometimes challenging

Case study - Catching a big fish in the sea

```
$ aws sts get-caller-identity
arn:aws:iam::900804735337:role/github_action_mirror_repos_role
```

Catching a big fish in the sea 🇬🇧 👑

```
44          echo "updating"
45          git="git --git-dir ${repo}.git"
46
47          aws_remote="https://git-codecommit.${AWS_REGION}.amazonaws.com/v1/repos/${repo}"
48
```

govuk-infrastructure / README.md ↑ Top

Preview | Code | Blame 49 lines (30 loc) · 1.77 KB Raw

Some AWS services for GOV.UK are still configured using the legacy alphagov/govuk-aws (public) and alphagov/govuk-aws-data (private) repos.

```
BASE=https://git.codecommit.us-east-1.amazonaws.com/v1/repos/
aws_remote=$BASE/govuk-aws-data
git clone $aws_remote
```

Accessing private GitHub repositories

```
root@reproduction:~# git clone $aws_remote --depth 1
Cloning into 'govuk-aws-data'...
remote: Counting objects: 560, done.
Receiving objects: 100% (560/560), 157.87 KiB | 1.20 MiB/s, done.
Resolving deltas: 100% (49/49), done.
```

```
root@reproduction:~# cd govuk-aws-data
root@reproduction:~/govuk-aws-data# ls -l
total 8
-rw-r--r--  1 root root 1759 May 10 09:35 README.md
drwxr-xr-x 82 root root 4096 May 10 09:35 data
drwxr-xr-x  2 root root   98 May 10 09:35 docs
drwxr-xr-x  2 root root   74 May 10 09:35 tools
root@reproduction:~/govuk-aws-data# ls -l data/
total 0
drwxr-xr-x 5 root root 58 May 10 09:35 ap
drwxr-xr-x 5 root root 58 May 10 09:35 ap
drwxr-xr-x 5 root root 58 May 10 09:35 ap
drwxr-xr-x 5 root root 58 May 10 09:35 ap
drwxr-xr-x 5 root root 58 May 10 09:35 ap
drwxr-xr-x 5 root root 58 May 10 09:35 ap
drwxr-xr-x 5 root root 58 May 10 09:35 ap
```

govuk-infrastructure / terraform / deployments / github / **mirror.tf**

```
Action = [
  "codecommit:GitPull",
  "codecommit:GitPush"
]
```

Public disclosure to the UK government

#1979003 Misconfigured AWS IAM role in AWS account 900804735337 allows any GitHub Action to retrieve credentials — November 27, 2023

To: **Cabinet Office** • Critical

- Vulnerability disclosure program in place
- Issue fixed 26 hours after initial report!
- Excellent contact with the GOV UK team

So… what happened? 👀

```
"Condition" : {




  },
  "StringEquals" : {
    "token.actions.githubusercontent.com:aud" : "${one(aws_iam_openid_connect_provider.github_provider.client_id_list)}"
  }
}
```

Permalink to (old) vulnerable code

Terraform handling of duplicate object keys

Bulletin ID: HCSEC-2023-26

Affected Products / Versions: All Terraform releases to date.

Publication Date: August 24, 2023

Remediation

Due to potential impact on Terraform core and HCL dependencies, this behavior is not something that can be immediately modified. While the security implications identified to date are restricted to a specific situation, the implications of a behavior change are potentially much wider.

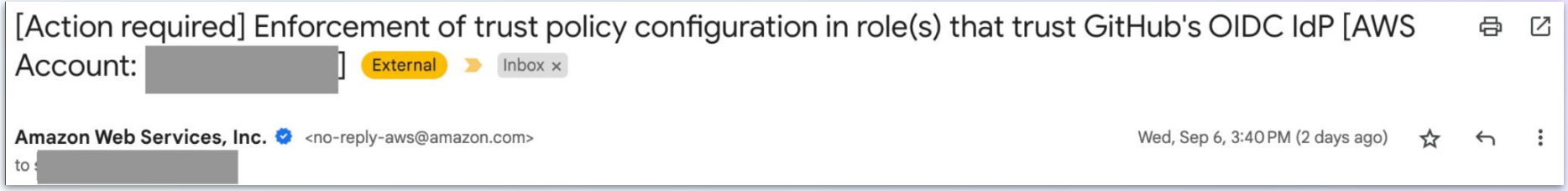
Improvements in the Terraform AWS provider

Released in 5.19.0

```
Error: "policy" contains duplicate JSON keys: duplicate key "Statement.0.Condition.StringEquals

  with aws_iam_policy.test,
  on main.tf line 54, in resource "aws_iam_policy" "test":
  54:   policy = <<EOF
  55: {
  56:   "Version": "2012-10-17",
  57:   "Statement": [
  58:     {
  59:       "Effect": "Allow",
  60:       "Action": "s3:PutObject",
  61:       "Resource": "*",
  62:       "Condition": {
  63:         "StringEquals": {
  64:           "s3:prefix": ["one/", "two/"]
  65:         },
  66:         "StringEquals": {
  67:           "s3:versionid": "abc123"
  68:         }
  69:       }
```

Improvements on the AWS side


[Action required] Enforcement of trust policy configuration in role(s) that trust GitHub's OIDC IdP [AWS Account: [redacted]] External Inbox ×

Amazon Web Services, Inc. <no-reply-aws@amazon.com> Wed, Sep 6, 3:40 PM (2 days ago)

to [redacted]

```
 Error: creating IAM Role: MalformedPolicyDocument: Trust policy with
trusted principal (...) must evaluate, using StringEquals, StringLike
or StringEqualsIgnoreCase, token.actions.githubusercontent.com:sub
which is not scoped to all.
```

Are we secure yet?

- Much better for new roles!
- Some misconfigurations still fall through the cracks

```
"StringLike": {
  "token.actions.githubusercontent.com:sub" : "repo:*"
}
```

- Previously-created roles can still be vulnerable

Practical tips


For defenders 🛡️

"Keyless" OIDC authentications is still highly worth it!

- ➢ Easier to configure
- ➢ More secure

For defenders 🛡️

- Check if you have roles used by GitHub actions

```
$ aws iam list-roles --output json | jq -r '
 .Roles[]
 | select((.AssumeRolePolicyDocument.Statement[]?.Principal.Federated? // empty) |
endswith("githubusercontent.com"))
 | .RoleName'
```

- Check your trust policies
  - https://github.com/Rezonate-io/github-oidc-checker/

https://securitylabs.datadoghq.com/cloud-security-atlas/vulnerabilities/iam-role-can-be-assumed-by-any-github-action/

For defenders 🛡️

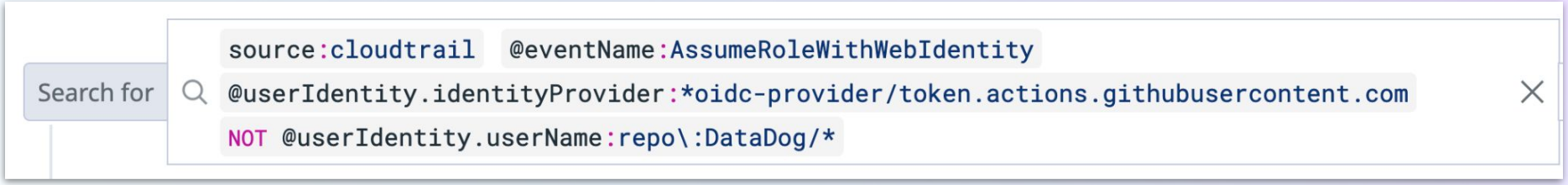
- Use an org-custom GitHub OIDC provider

```
{
  "iss": "https://token.actions.githubusercontent.com/octocat-inc",
  "jti": "6f4762ed-0758-4ccb-808d-ee3af5d723a8",
  "sub": "repo:octocat-inc/private-server:ref:refs/heads/main",
  "aud": "http://octocat-inc.example/octocat-inc",
```

- GitHub Enterprise Cloud only ☹️

https://awsteele.com/blog/2023/01/11/improve-github-actions-oidc-security-posture-with-custom-issuer.html
https://docs.github.com/en/enterprise-cloud@latest/actions/deployment/security-hardening-your-deployments/about-security-hardening-with-openid-connect#customizing-the-issuer-value-for-an-enterprise

For defenders - detection 🛡️




```
{
 "userIdentity": {
   "type": "WebIdentityUser",
   "userName": "repo:SOURCE-ORG/SOURCE-REPO:ref:refs/heads/BRANCH",
   "identityProvider": "arn:aws:iam::012345678901:oidc-provider/token.actions.githubusercontent.com"
 },
 "eventName": "AssumeRoleWithWebIdentity",
 "requestParameters": {
   "roleArn": "arn:aws:iam::012345678901:role/github-actions-role",
 }
}
```

For defenders - detection 🛡️

```
SELECT *
FROM <event-data-store-id>
WHERE eventSource ='sts.amazonaws.com' AND eventName = 'AssumeRoleWithWebIdentity'
AND userIdentity.identityprovider LIKE
'%:oidc-provider/token.actions.githubusercontent.com'
AND userIdentity.username NOT LIKE 'repo:YOUR-GITHUB-ORG/%'
ORDER BY eventTime DESC
```

CloudTrail SQL Lake query

For pentesters 😈

- Look for GitHub Actions usage
- Abuse vectors through pull requests too

For pentesters 😈

- You can find the AWS account ID of a target from multiple ways
  - "How to find the AWS Account ID of any S3 Bucket"
  - S3 bucket names are everywhere (DNS CNAME, JS code…)

- You can perform *unauthenticated* IAM roles enumeration
  - "Unauthenticated Enumeration of IAM Users and Roles"
  - Build your own wordlists

Wrapping up

What about other platforms?

Same concepts in other clouds and CI platforms!

- Azure AD app registrations "Federated credentials"
- Google Cloud Workload Identity Federation

Takeaways

- Use OIDC keyless authentication in your CI/CD pipelines
- Proactively watch out for overly-permissive trust policies
- Many roles remain vulnerable - go find them!

Literature / related research

- "No keys attached: Exploring GitHub-to-AWS keyless authentication flaws" (Datadog)
- "Identifying vulnerabilities in GitHub Actions & AWS OIDC Configurations" (Tinder)
- "Misconfigured Actions Place GCP & AWS Accounts At Risk" (Rezonate)
- "Hacking Github AWS integrations again" (Daniel Grzelak)
- "A security community success story of mitigating a misconfiguration" (Scott Piper)

Thank you!

@christophetd

christophetd@infosec.exchange

christophe@tafani-dereeper.me

christophetd.fr

slides & feedback

bit.ly/aws-oidc