Configuring Amazon S3 security settings and access controls

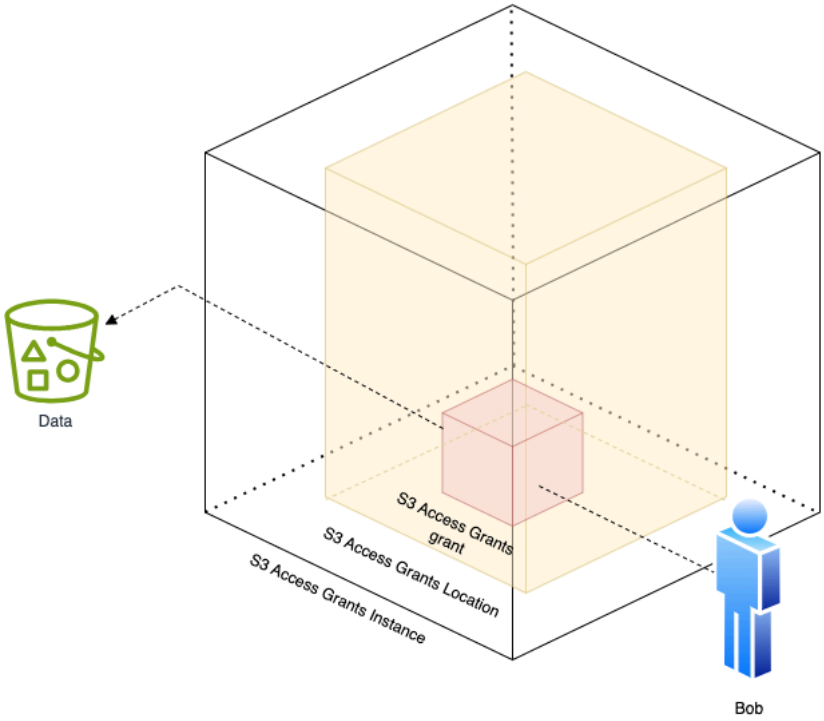
- Getting started at an AWS hosted workshop
- S3 Security Best Practices
  - Prepare Your Lab
    - Attach IAM Role to EC2 Instance
    - Connect to the EC2 Instance
    - Bucket Name
  - Lab 1 - S3 Security Exercises
    - Require HTTPS
    - Require SSE-KMS Encryption
    - Restrict Access to an S3 VPC Endpoint
    - Use AWS Config Rules to Detect a Public Bucket
    - Use Amazon Access Analyzer for S3
  - Lab 2 - S3 Access Grants
    - S3 Access Grants Lab - Initial Setup
    - **Configure S3 Access Grants for IAM user**
  - Lab 3 - Enabling Malware Protection for S3 by using GuardDuty
  - Lab 4 - S3 Access Control Lists
  - Lab Summary

Configure S3 Access Grants for IAM user

IAM User

In this module, we will configure S3 Access Grants to give access to IAM users/roles.

The scenario is user `Bob` want to access a report in a S3 bucket



We will simulate the user `Bob`, in this case your current session in Cloud Shell can be used to assumeRole with `Bob` credentials. Then we can invoke the `s3:GetDataAccess` to get the temporary credentials from S3 Access Grants and then have access to the report stored on S3.

First, lets create the trust policy that will allow your current identity to AssumeRole to `Bob` role

```
source workshop.env
cat << EOF > bob-trust-policy.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BobTrust",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$AWS_ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

Next, we will create the `Bob` IAM role

```
aws iam create-role \
    --role-name Bob-Role \
    --assume-role-policy-document file://bob-trust-policy.json
```

Next, we will create an IAM policy to `Bob` with the minimum actions required to have the user be able to request credentials.

```
source workshop.env
cat << EOF > bob-iam-policy.json
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "UseS3AccessGrants",
         "Effect":"Allow",
         "Action": "s3:GetDataAccess",
         "Resource":"$S3AG_INSTANCE_ARN"
       }
   ]
}
EOF
```

This policy only allows the user `Bob` to use temporary credentials to access data stored on S3. It is important to note this paradigm change. Instead of giving direct permissions to Bob via IAM policies, we will only allow Bob to invoke S3 Access Grants. Is now under S3 Access Grants governance to allow this user or group to which S3 bucket, prefix and objects that Bob can access.

Now, you will attach the IAM policy to the `Bob` role

```
aws iam put-role-policy \
    --role-name Bob-Role \
    --policy-name S3AGPolicy \
    --policy-document file://bob-iam-policy.json
```

Next, we will create a profile in your Cloud Shell to use the IAM role Bob.

```
source workshop.env
cd ~
mkdir .aws
cat << EOF > ~/.aws/config
[profile bob]
role_arn = arn:aws:iam::$AWS_ACCOUNT_ID:role/Bob-Role
credential_source = EcsContainer
EOF
cd ~
```

You can test now the profile running the following command:

```
aws s3 ls --profile bob
```

Step review
The output should look like:

```
An error occurred (AccessDenied) when calling the ListBuckets operation: User:
arn:aws:sts::092526071050:assumed-role/Bob-Role/botocore-session-1732775731 is not
authorized to perform: s3:ListAllMyBuckets because no identity-based policy allows the
s3:ListAllMyBuckets action
```

This is the expected result. User Bob don't have permissions to access any S3 service. If you see something different from this. review your configuration settings.

Creating your first S3 Access Grants grant

Let's create a grant to user Bob. This grant will use the default location that we created previously. The permission will be READ and WRITE to the bucket `s3-<account-id>-access-grants-workshop`

```
source workshop.env
aws s3 mb "s3://s3-$AWS_ACCOUNT_ID-access-grants-workshop"
```

```
echo "This is Bob report" > bob-report.txt
aws s3 cp bob-report.txt s3://s3-$AWS_ACCOUNT_ID-access-grants-workshop/bob/bob-report.t
aws s3control create-access-grant \
--account-id $AWS_ACCOUNT_ID \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=s3-$AWS_ACCOUNT_ID-access-grants-work
--permission READWRITE \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::$AWS_ACCOUNT_ID:role/Bob-Role
```

Now that you have a grant to allow user Bob to access data in a S3 bucket, let's get the credentials and use it to retrieve the report.

```
source workshop.env
S3AG_CREDS=$(aws s3control get-data-access \
--account-id $AWS_ACCOUNT_ID \
--target s3://s3-$AWS_ACCOUNT_ID-access-grants-workshop/bob/* \
--permission READWRITE \
--privilege Default \
--region $AWS_DEFAULT_REGION \
--profile bob)

cat << EOF > ~/.aws/credentials
[S3AG]
aws_access_key_id=$(echo $S3AG_CREDS | jq -r '.Credentials.AccessKeyId')
aws_secret_access_key=$(echo $S3AG_CREDS | jq -r '.Credentials.SecretAccessKey')
aws_session_token=$(echo $S3AG_CREDS | jq -r '.Credentials.SessionToken')
EOF
```

The commands above will request temporary credentials through the GetDataAccess API using Bob credentials. After we receive a JSON response with the credentials, we write those credentials in a new profile called `S3AG`

Now, let's test if the credentials are working.

```
aws s3 ls s3://s3-$AWS_ACCOUNT_ID-access-grants-workshop/bob/  --profile S3AG
```

You should be able to see the file `bob-report.txt` with the command above. Next can try running the command using different prefixes and the response should be access denied.

```
aws s3 ls s3://s3-$AWS_ACCOUNT_ID-access-grants-workshop/  --profile S3AG
```

Automating the process

Let's review the basic concepts of S3 Access Grants:

- Instance: Is the account and region level resource. Can exist only one, and it is used to enable the service.
- Location: Exists inside an instance and can have more than one. The first location use s3:// as the resource scope and must have an IAM policy that allow the location to vend credentials to users.
- Grant: Give permissions to IAM users and roles to access S3 resources. Resources can be buckets or prefixes under a particular location.

Some key points about S3 Access Grants that you should know:

- Buckets must exist before you create a Location (non-default) or a Grant
- Prefixes do not need to exist to create a grant
- S3 Access Grants can give permissions directly to federated users using Identity Center. We wil this concept later on.

Answer

AWS CLI provide a method that you can invoke an external process to return the credentials.
We will create a simple python script that will generate those credentials via GetDataAccess and return

to AWS CLI

First, let's create our credential processor. Open the file editor `nano` in your Cloud shell to create a file called `get_credentials.py`

```
nano get_credentials.py
```

Then, copy and paste the code below in your editor and save with `Ctrl-x`

```
#!/bin/python
import argparse
import json
import boto3
from botocore.exceptions import ClientError

__version__ = '0.1.0'  # Version control


def main(*local_args):
    parser = argparse.ArgumentParser()
    parser.add_argument('--account', help='AWS Account Id')
    parser.add_argument('--target', help='S3 URI')
    parser.add_argument('--permission', choices=['READ', 'WRITE', 'READWRITE'],
                        help='Permission level for this credential')
    parser.add_argument('--duration', type=int, help='The session duration in seconds. D
    parser.add_argument('--privilege', choices=['Default', 'Minimum'], help='Privilege ty
                        default='Default')
    parser.add_argument('--profile', default='default', help='AWS Profile to request cre
    parser.add_argument('--version', action='version', version=f'%(prog)s {__version__}'

    if local_args:
        args = parser.parse_args(local_args)
    else:
        args = parser.parse_args()

    boto3.setup_default_session(profile_name=args.profile)
    client = boto3.client('s3control')
    try:
        resp = client.get_data_access(AccountId=args.account,
                                      Target=args.target,
                                      Permission=args.permission,
                                      DurationSeconds=args.duration,
                                      Privilege=args.privilege)
    except ClientError as e:
        print(e)
        return
    # We need to add Version to the response to be compatible with the AWS CLI credential
    # https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-sourcing-external.ht
    resp['Credentials']['Version'] = 1
    return resp['Credentials']


if __name__ == '__main__':
    resp = main()
    print(json.dumps(resp, default=str))
```

Next we will delete the file `~/.aws/credentials` as we will get the credentials dynamically.

```
rm ~/.aws/credentials
```

Next we will make the file `get_credentials.py` executable

```
chmod +x get_credentials.py
```

And to finish the configuration, we will add a new profile section in the ~/.aws/config to instruct AWS CLI to use our `get_credentials.py` to generate the credentials.

```
source workshop.env
cat <<EOF >> ~/.aws/config
[profile s3ag]
credential_process = "/home/cloudshell-user/get_credentials.py" --account $AWS_ACCOUNT_ID
EOF
```

Now, you can test your profile `s3ag` running any AWS CLI command to read or write in the S3 prefix that your user Bob is allowed.

```
source workshop.env
aws s3 ls s3://s3-$AWS_ACCOUNT_ID-access-grants-workshop/bob/ --profile  s3ag
```

Lab complete
Congratulations. You finished the S3 Access Grants lab.
In this lab, you learned how to configure S3 Access Grants with IAM users or roles and how to setup AWS CLI to use S3 Access Grants credentials.

▶ **Bonus information**

Previous Next