1. Grant access to all principals in your organization to your resource

Use case

There are times where you want a resource, such as an object in an S3 bucket, to be accessible to all principals in your organization. An example is a read-only configuration file that platform-level code running in each AWS account needs to configure the account or an SNS topic that lives in a central account that each AWS account in your organization needs to publish messages to.

Policy type - Resource-based policy

Values to use in policy:

- AWS Organizations organization ID: o-123456
- Bucket name: MY-BUCKET
- Account: 111111111111
- Action: s3:GetObject

The Approach:

1. **Identifying the Right Condition Key:**

I started by choosing aws:PrincipalOrgID as the condition key. This key is crucial because it helps verify if the IAM entities making the requests belong to our organization. It compares the organization ID in the request context against the one specified in our policy.

2. **Understanding Policy Elements:**

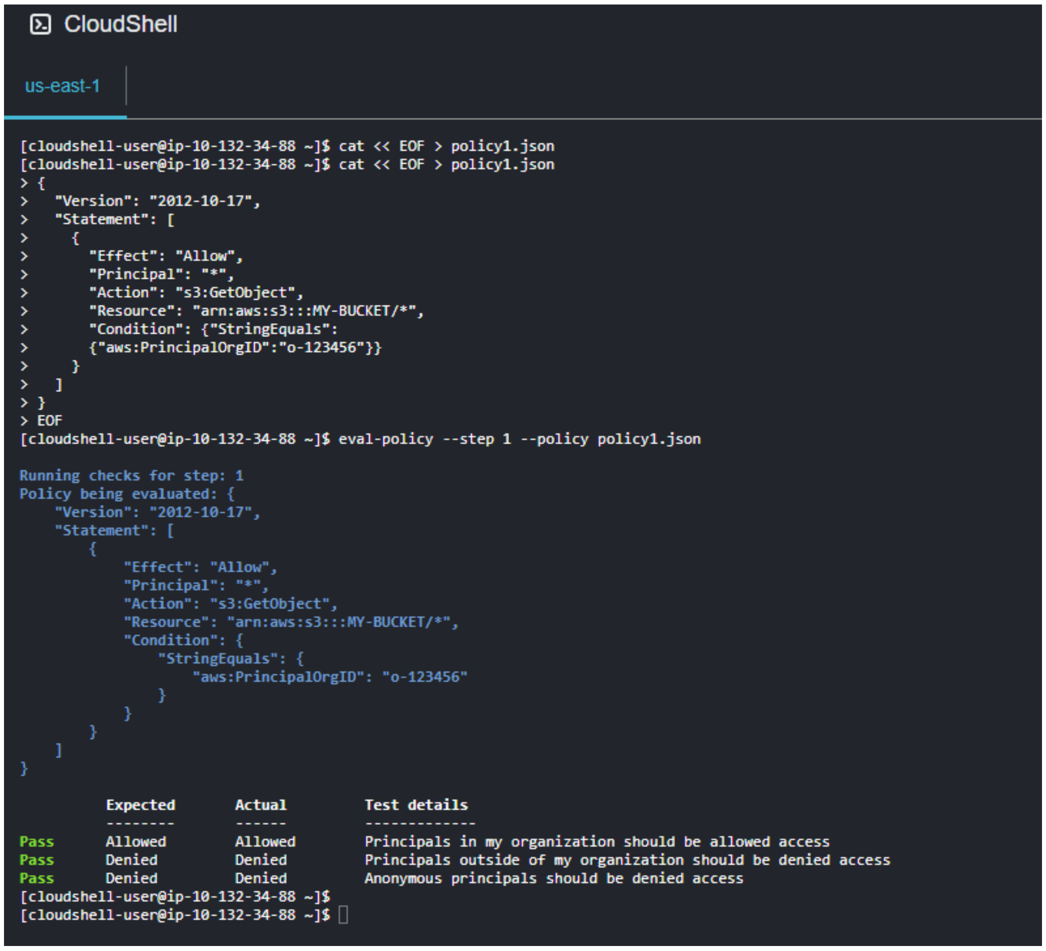
Principal: Set to * to imply all IAM roles/users in our AWS account.
Action: s3:GetObject to specify the type of allowed interaction with the S3 bucket.
Resource: arn:aws:s3:::MY-BUCKET/* ensures the policy covers all objects within the bucket.
Condition: The StringEquals condition ensures that only requests aligned with our organization ID (o-123456) are granted access.

3. **Policy Creation Command:**

I used the cat << EOF > policy1.json command to create a new JSON policy file. This command effectively starts writing the specified content into policy1.json until it encounters EOF (End of File), marking the end of input:

```
cat << EOF > policy1.json
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:GetObject",
   "Resource": "arn:aws:s3:::MY-BUCKET/*",
   "Condition": {"StringEquals":
   {"aws:PrincipalOrgID":"o-123456"}}
  }
 ]
}
EOF
```

4. The Execution:



```
[cloudshell-user@ip-10-132-34-88 ~]$ cat << EOF > policy1.json
[cloudshell-user@ip-10-132-34-88 ~]$ cat << EOF > policy1.json
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Effect": "Allow",
>       "Principal": "*",
>       "Action": "s3:GetObject",
>       "Resource": "arn:aws:s3:::MY-BUCKET/*",
>       "Condition": {"StringEquals":
>       {"aws:PrincipalOrgID":"o-123456"}}
>     }
>   ]
> }
> EOF
[cloudshell-user@ip-10-132-34-88 ~]$ eval-policy --step 1 --policy policy1.json

Running checks for step: 1
Policy being evaluated: {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::MY-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-123456"
                }
            }
        }
    ]
}

          Expected       Actual         Test details
          --------       ------         ------------
Pass      Allowed        Allowed        Principals in my organization should be allowed access
Pass      Denied         Denied         Principals outside of my organization should be denied access
Pass      Denied         Denied         Anonymous principals should be denied access
[cloudshell-user@ip-10-132-34-88 ~]$
[cloudshell-user@ip-10-132-34-88 ~]$
```

Testing and Validation

After crafting the policy, I ran it through CloudShell with the command `eval-policy --step 1 --policy policy1.json`. This step is critical as it ensures the policy behaves as expected before it's applied live, preventing any unintended access issues.

The Outcome

The policy effectively restricted access to MY-BUCKET, allowing only those within our organization who meet the organization ID condition. It simplified our access management, ensuring security without the need to manually list and update individual AWS account IDs.

This approach not only secured our resources but also streamlined the management of access permissions across our AWS Organization, demonstrating the importance of precise and well-thought-out IAM policies in maintaining operational efficiency and security in the cloud.