IAM

Amazon Web Services

IAM Benefits & Usages

- IAM provides access and access permissions to AWS resources
- IAM is global to all AWS regions, creating a user account will apply to all the regions
- IAM policies allow for granular API level permissions for granting users and groups access to specific AWS resources

- Benefits:
  - Central control of AWS resources
  - Consolidated AWS bill for your users
  - Ensure users access only from specified networks
  - Easily manage security credentials
  - Provides temporary user access when needed
  - Federate with SAML providers such as active directory for temporary and single sign on access
  - Provide roles that other AWS resources can assume

IAM BENEFITS & USAGE (CONTD...)

- Allows you to manage users and groups within the AWS account
- Can specify password policy as well as MFA requirements on a per user basis
- Provides pre-built policy templates to assign to users and groups
    - Administrator access
    - Power user access – Does not allow user/group management
    - Read only access- Only view AWS resources (accounting)

S3 Bucket Policy

Elements of an access policy

- **Resources**
  - Used to identify resources (like a Bucket or Object) with Amazon Resource Names (ARNs)
- **Actions**
  - Actions we want to allow or deny
  - *Important:* an explicit deny always overrides an explicit allow
- **Effect**
  - Defines whether to allow or deny the above action
- **Principal**
  - An account or user that this policy applies to
  - Specific to S3 bucket policies, not user policies

S3 Bucket POLICY

Bucket policy examples

```
{
   "Version":"2012-10-17",
   "Statement": [
    {
      "Sid":"PutObjectAcl",
      "Effect":"Allow",
      "Principal": {
        "AWS": [
           "arn:aws:iam::111122223333:tom",
           "arn:aws:iam::444455556666:chris" ]
      },
      "Action":["s3:PutObject","s3:PutObjectAcl"],
      "Resource":["arn:aws:s3:::examplebucket/*"]
    } ]
}
```

Bucket Policy

```
{
   "Version":"2012-10-17",
   "Statement": [
    {
      "Sid":"GetObject",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::examplebucket/*"]
    }
   ]
}
```

Security Token Service

AWS Security Token Service

Amazon Security Token Service (STS) allows you to grant a trusted user temporary and controlled access to AWS resources.

- Grant temporary access
  - To existing IAM users
  - To web-based identity providers: Facebook/Amazon/Google
  - To your organization's existing identity system

- Credentials are associated with an IAM access control policy that limits what the user can do

- Amazon STS API
  - AWS SDKs
  - AWS CLI
  - AWS Tools for Windows Powershell

AWS Security Token Service

- STS returns temporary security credentials
  - These consist of an **access key** and a **session token**
- Access Key
  - Consists of an access key ID and a secret key
- Session Token
  - Used to validate our user's temporary security credentials
- Credentials expire after a certain amount of time

AWS Security Token Service: Key Terms

- Federation
  - Creating a trust relationship between an identity provider and AWS
  - Users can sign into an identity provider like Amazon, Facebook, Google, or any other recognized provider

- Identity broker
  - The broker is in charge of mapping the user to the right set of credentials

- Identity Store
  - An identity store is something like Facebook, Google, Amazon, or Active Directory

- Identities
  - A user or “identity” within an identity store

Temporary Credentials with Amazon EC2

- Assign an IAM role to the EC2 instance
- Get automatic temporary security credentials from the instance metadata using the AWS SDKs/CLI
- You don’t have to explicitly get credentials

Temporary Credentials with AWS SDKs

- Call the AWS STS API (*AssumeRole*) with your code
- Extract the credentials and session token, and use those values for future calls to AWS
- Make sure you renew credentials before the old ones expire (some SDKs do this for you)

Temporary Credentials with APIs

- Sign requests with your temporary security credentials that you get from AWS STS
- Use the access key ID and secret access key, and add your session token to the API request
  - Add the session token to an HTTP header
  - OR add it to a query string parameter named X-AMZ-Security-Token

Example Scenario

A corporate web application is deployed within an Amazon VPC, and is connected to the corporate data center via IPSec VPN. The application must authenticate against the on-premises LDAP server. Once authenticated, logged-in users can only access an S3 keyspace specific to the user.

Solution:

- Develop an identity broker to authenticate against LDAP
- Identity broker calls the STS API to receive temporary credentials
- Application can then access the temporary AWS permissions