Task 4 Report – Implementing IAM Policies, Secure Storage, and Data Encryption on AWS

Objective:

The objective of this task was to implement a secure cloud storage solution using Amazon S3, focusing on fine-grained access control using IAM policies and data encryption at rest. This task applied key cloud security principles such as permission isolation, encryption, and IAM-based access management.

Tools and Technologies Used:

- **Amazon S3 (Simple Storage Service)** – For storing and managing cloud data.
- **IAM (Identity and Access Management)** – To control access to AWS resources.
- **AWS Root Account** – Used for administrative setup.
- **IAM User (nandiniemp)** – Used for secure, restricted access operations.
- **SSE-S3 (AES-256)** – For automatic server-side encryption of stored data.
- **Bucket Policies (JSON)** – To manage access permissions at the bucket level.
- **AWS Management Console and Cloud Console** – For performing and verifying all configurations.
-

Step-by-Step Implementation:

1. Creating the IAM User

1. Logged into the **AWS Management Console** using the root account.
2. Navigated to **IAM > Users > Add User**.
3. Created a new IAM user named nandiniemp with the following configuration:
   - **Access Type**: Enabled **Programmatic Access** (for access key and secret key generation).

   - **Console Access**: Disabled (as testing was done using the AWS CLI/SDK).
4. No permissions were attached at creation time to follow the principle of least privilege.
5. Saved the access key ID and secret access key securely for use in testing and scripting.

2. Creating the S3 Bucket from the Root Account

1. Logged in using the AWS root account.
2. Navigated to the Amazon S3 service.
3. Created a new S3 bucket named: secure-storage-codetech.
4. Configured the bucket with the following settings:
   - **Public Access Settings**: All public access blocked.
   - **Versioning**: Enabled to maintain object versions.
   - **Default Encryption**: Enabled **Server-Side Encryption with Amazon S3-managed keys (SSE-S3)** using AES-256.
   - **Bucket Key**: Disabled (kept default settings to avoid unnecessary overrides).

3. Creating and Configuring the IAM Policy

1. Navigated to the IAM service and selected **Policies > Create Policy**.
2. Created a custom policy named: SecureS3-USER.
3. The policy granted:
   - s3:ListBucket on the bucket ARN: arn:aws:s3:::secure-storage-codetech
   - s3:GetObject and s3:PutObject on all objects within the bucket: arn:aws:s3:::secure-storage-codetech/*
4. The policy was saved and reviewed in **JSON format** as secure-s3-access-policy.json.

3. Attaching Policy to IAM User

1. Navigated to **IAM > Users** and selected the IAM user: nandiniemp.
2. Attached the SecureS3-USER policy to this user.
3. Confirmed the user only had this policy and no unnecessary permissions.

4. Access Testing via IAM User Account

1. Logged into the **AWS Management Console** using IAM user credentials (nandiniemp).
2. Verified the following:
   - **Successfully listed objects** in the bucket using s3:ListBucket.
   - **Uploaded new objects** to the bucket using s3:PutObject.
   - **Downloaded objects** using s3:GetObject.
3. Attempted to open an object via public URL:
   - Initially received Access Denied error.
   - Updated the bucket policy to allow s3:GetObject only on that specific object for testing.
   - Retried the operation and confirmed object access worked as intended.

Problems Faced and Resolutions:

Issue	**Resolution**
Access Denied when listing bucket contents	Added s3:ListBucket permission to the IAM policy
Object not accessible via URL	Modified bucket policy to explicitly allow s3:GetObject for that object path
Over-restrictive access due to combined IAM and bucket policies	Carefully aligned IAM policy and bucket policy to allow secure object access
Attempted root credentials usage in test	Switched to using IAM credentials as per security best practices

Security Best Practices Followed:

- **Principle of Least Privilege**: IAM user granted only necessary permissions.
- **Encryption at Rest**: Enabled server-side encryption with SSE-S3 to protect stored data.
- **IAM over Root Usage**: Root account was used only for setup; all access tests were performed using a dedicated IAM user.

- **No Public Access**: Public access was blocked at the bucket level. Object-level access was granted selectively only for testing.

Conclusion:

This task demonstrated the implementation of a secure and controlled storage system on AWS using Amazon S3. Key configurations included IAM policy management, encrypted data storage, versioning, and access restriction using best practices. Through IAM user testing and policy adjustments, secure operational access was validated and enforced.