NAME: AMUDA RANTI

ROLE:CLOUD SECURITY ENGINEER

TASK: Tasked with Securing AWS S3 Bucket Configuration

I was tasked with securing an AWS S3 bucket to meet the organization’s security standards.
The goal was to ensure that sensitive data stored in the S3 bucket was secure, with proper access control, encryption, and logging in place. Here’s how I approached this project:

1. Setting Up the S3 Bucket

The first step I took was to create an S3 bucket that would hold all the files. Since we wanted to store sensitive data, I made sure to choose a region that complied with our data residency requirements.

```
aws s3 mb s3://Ranti23 --region us-east-1
```

Once the bucket was created, I checked the configuration to ensure everything was aligned with best practices, such as ensuring it was correctly named and configured for future security needs.

2. Configuring the Bucket's Security Settings

I started by focusing on the public access settings. One of the easiest and most common security mistakes is leaving a bucket open to the public, so I took extra care to block any public access to the bucket. I used the AWS Console to ensure no public access was allowed:

Go to the bucket settings → "Permissions" → "Block public access" → Enable all options to block public access.

I also made sure to configure bucket versioning. This was crucial because it helps track any changes made to objects in the bucket, making it easier to recover data if something goes wrong (e.g., if data is deleted or corrupted).

```
aws s3api put-bucket-versioning --bucket your-secure-bucket-name
--versioning-configuration Status=Enabled
```

3. Implementing IAM Roles and Policies
Next, I turned my attention to access control. I wanted to ensure that only authorized users and services could access the S3 bucket. To do this, I created an IAM policy and attached it to an IAM role.

For example, I created a read-only policy to allow only specific IAM roles or users to read from the bucket. This way, sensitive data was protected from unauthorized changes while still allowing access to the right people.

Here’s a quick look at the policy I used for read-only access:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your-secure-bucket-name/*"
    }
  ]
}
```

I attached this policy to an IAM role that was specifically created for this purpose. This role had strict access limits, ensuring that users or applications could only access the bucket in the ways that were required, following the principle of least privilege.

4. Setting Up Encryption for Data Security

One of the most critical aspects of this project was ensuring that all data stored in the S3 bucket was encrypted at rest. I enabled Server-Side Encryption (SSE) for the bucket to ensure that all objects uploaded were encrypted by default.

I used AES-256 encryption (which is AWS’s default encryption algorithm) for simplicity and compliance, but I also explored AWS KMS for more granular control over encryption keys (though AES-256 was sufficient for our needs at the time).

```
  aws s3api put-bucket-encryption --bucket your-secure-bucket-name
--server-side-encryption-configuration '
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "AES256"
      }
    }
  ]
}'
```

I also verified that every file uploaded to the bucket was being automatically encrypted, which ensured that our data was safe from unauthorized access, even if someone gained access to the bucket itself.

5. Enabling Logging and Monitoring

To keep track of all the actions performed on the bucket, I set up server access logging. This would allow us to audit who accessed the bucket and when, providing an essential layer of monitoring.

First, I created a separate log bucket to store the logs:

```
aws s3 mb s3://your-log-bucket-name
```

Then, I enabled logging on the original bucket and pointed the logs to the log bucket. This way, every time someone accessed the S3 bucket, an entry would be recorded in the log bucket.

```
aws s3api put-bucket-logging --bucket your-secure-bucket-name
--bucket-logging-status '{
  "LoggingEnabled": {
    "TargetBucket": "your-log-bucket-name",
    "TargetPrefix": "access-logs/"
  }
}'
```

I made sure the logs were being generated by verifying the log files were appearing in the log bucket.

6. Testing and Validation

Once all the configurations were in place, I conducted several tests to verify that everything was working as expected:

Public access test: I tried accessing the bucket's contents from a browser (without credentials) to confirm that the public access was successfully blocked.

Encryption test: I uploaded a test file and checked that it was encrypted by running the following command:

```
aws s3api head-object --bucket your-secure-bucket-name --key test-file.txt --query
"ServerSideEncryption"
```

This returned "AES256", confirming the file was encrypted.

Logging test: Finally, I checked the log bucket to ensure that the logs were being created correctly. I was able to see detailed logs of who accessed the bucket and when.

7. Compliance and Best Practices

To ensure that the project aligned with industry best practices, I also followed additional security guidelines:

I reviewed the security policies to ensure least privilege access.
I made sure multi-factor authentication (MFA) was enabled for sensitive actions.
I regularly reviewed the bucket configurations to confirm compliance with organizational and regulatory requirements.

Configuration Files: IAM policies, bucket policies, encryption settings, and logging configurations.

This experience helped me develop a deep understanding of AWS S3 security best practices and provided me with hands-on experience implementing them in a real-world scenario.
I’m proud of the outcome and the attention to detail I put into ensuring the S3 bucket was both secure and compliant. It not only gave me practical skills in cloud security but also showed me the importance of doing things the right way when it comes to securing cloud infrastructure.