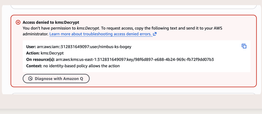
ENCRYPT DATA WITH AWS KMS



Introducing Today's Project!

In this project, I will demonstrate. how to encrypt a database with AWS KMS. The goal is to learn more about encryption, its nuances, and how it can be applied to everyday world functions.

Tools and concepts

Services I used include AWS KMS(Key Management Service), Dynamo DB, AWS IAM. Key concepts I learned include encryption, database tables, KMS using permissions to actions rather than just access to the key itself; creating a user to test access.

Project reflection

This project took me approximately. a little over an hour. The most challenging part was understanding how encryption and the different types of encryption works. It was most rewarding to see the test user get access again to the database!

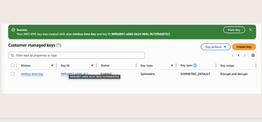
I chose to do this project today because I want to sharpen my cloud AWS skills. Something that would make learning with NextWork even better is nothing, y'all are the best right now!

Encryption and KMS

Encryption is the process of turning plain text data into a secure format. Companies and developers do this to secure their data from unauthorized users. Encryption keys are the secure code that informs an algorithm on how it should encrypt.

AWS KMS is a vault for the encryption keys. Key management systems are important because they help users secure and manage the keys used to encrypt data. Unauthorized access to the key equals exposing the encrypted data which puts the security at risk.

Encryption keys are broadly categorized as symmetric and asymmetric. I setup symmetric keys because I will bei using the exact same key to encrypt and decrypt data! Asymmetric keys would be a good choice if we need different keys for encryption and decryption.



Encrypting Data

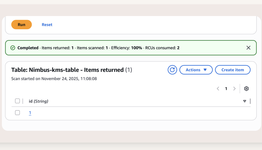
My encryption key will safeguard data in DynamoDB, which is a fast and flexible AWS database service. DynamoDB is great for applications that need fast access to large amount of data (i.e. gaming)

The different encryption options in DynamoDB include AWS owned, AWS managed, key managed by users (i.e. self) Their differences are based on who creates and manages the key and wheher we have visibility. I selected the key managed by users to get full access and control over the key.

Data Visibility

Rather than controlling who has access to the key, KMS manages user permissions by controlling the actions that people can do with that key. In my case, even if i gave the test user the permission to see the key, it would need the premission to decrypt.

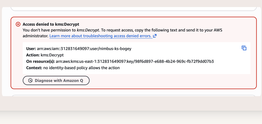
Despite encrypting my DynamoDB table, I could still see the table's items because I am users of the key. DynamoDB uses transparent data encryption, which means it does the encryption/decryption process for me because it know I am authorized.



Denying Access

I configured a new IAM user to. validate whether unahorized users can access encrypted data. The permission policies I granted this user are DynamoDB full access but NOT encryption, decryption permission with AWS KMS.

After accessing the DynamoDB table as the test user, I encountered an access denied message because the test user has no access to decryption with the key. This confirmed that encryption keys can be used to secure data!



EXTRA: Granting Access

To let my test user use the encryption key, I added the test user into the KMS console! My key's policy was updated to allow the test to encrypt, decrypt, re-encrypt, etc., using the key!

Using the test user, I retried accessing the DynamoDB table. I observed that the test use can see the data inside which confirmed that making the test user a key user is effective to authorized someone to see encrypted data.

Encryption secures data instead of an entire resource or an entire service. I could combine encryption with other access control tools like security groups, and permission policies to have multiple layers of security-- the resource layer, and then the data layer.