SOFE 4840 Winter 2022 - Software & Computer Security

Project Phase 2:Development & Design

Project Topic: Cloud Security

Project Group No: 16

Group Members:

Ashwin Shanmugam - 100700236
Tegveer Singh - 100730432
Mihirkumar Patel - 100702168
Esam Uddin - 100711116

Project Github Repo:
https://github.com/ashwin1609/Software-and-Computer-Security-Project

Project Introduction

For this phase of the project we decided to create a simple login website and made use of AWS services like Lambda, AWS CLI, API Gateway and DynamoDB. AWS or Amazon Web Services is one of the most popular cloud computing platforms currently available. AWS Lambda is used to provide computing resources on the cloud to manage serverless applications. DynamoDB, on the other hand, is a cloud-based key-value type NoSQL database.

The objective of our project is to prevent security threats and vulnerabilities to cloud applications through attacks on REST APIs. The project implements this by encrypting the routes for a web application using BCrypt. We use AWS Key Management System or KMS to generate securely encrypted API keys.

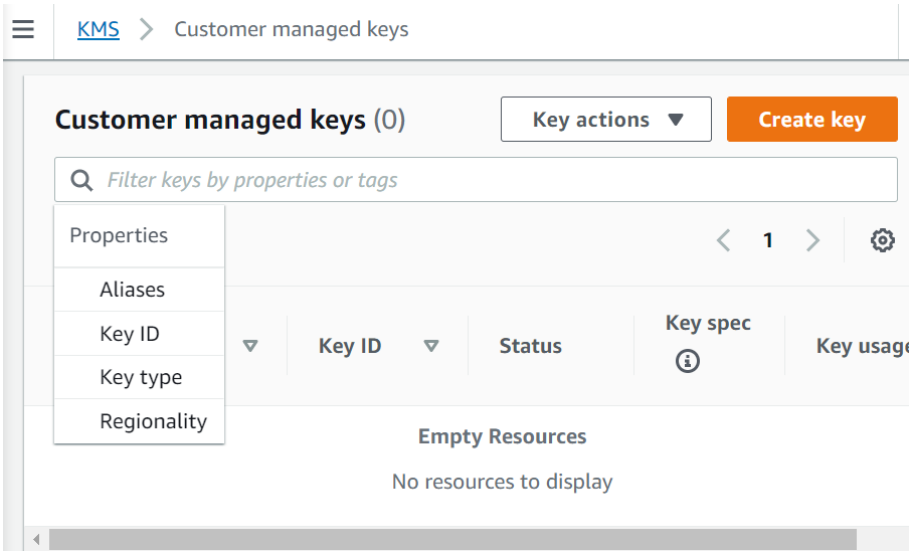
The project also involves encrypting and decrypting the data that is sent to the backend of an application. This is currently implemented by encrypting and decrypting simple text files. As an add-on to the project, the group will also implement Identity Access Management(IAM) and propose proper authorization strategies to enhance the security in cloud-based applications. IAM assigns roles to its users and these roles can be allotted by an administrator. The project will focus on defining some of these roles through customized JSON files.

All proposed policies and services are implemented on common web application features like login, registration, authentication, etc.

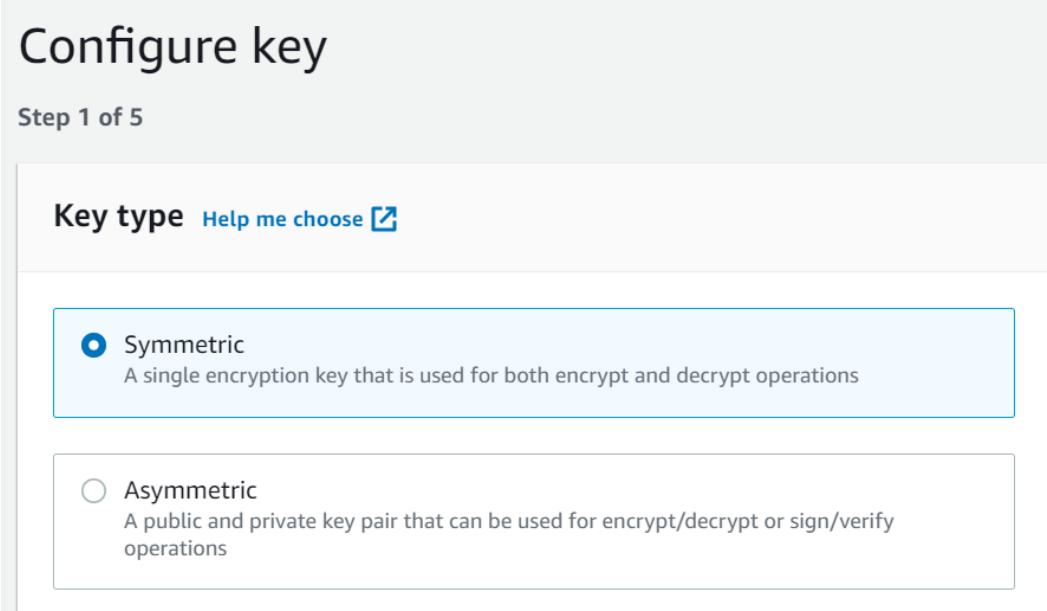
Data Encryption using AWS KMS and OpenSSL

AWS KMS is used to create and manage cryptographic keys. For this part of our project, we simply created a txt file that contains data on user information that is retrieved from the login website that was created earlier (hosted using AWS Amplify). KMS helped us control access to the encrypted data as we were able to define permissions to use keys. A customer managed key was created using this service and assigned an IAM user from which the key can be accessed.

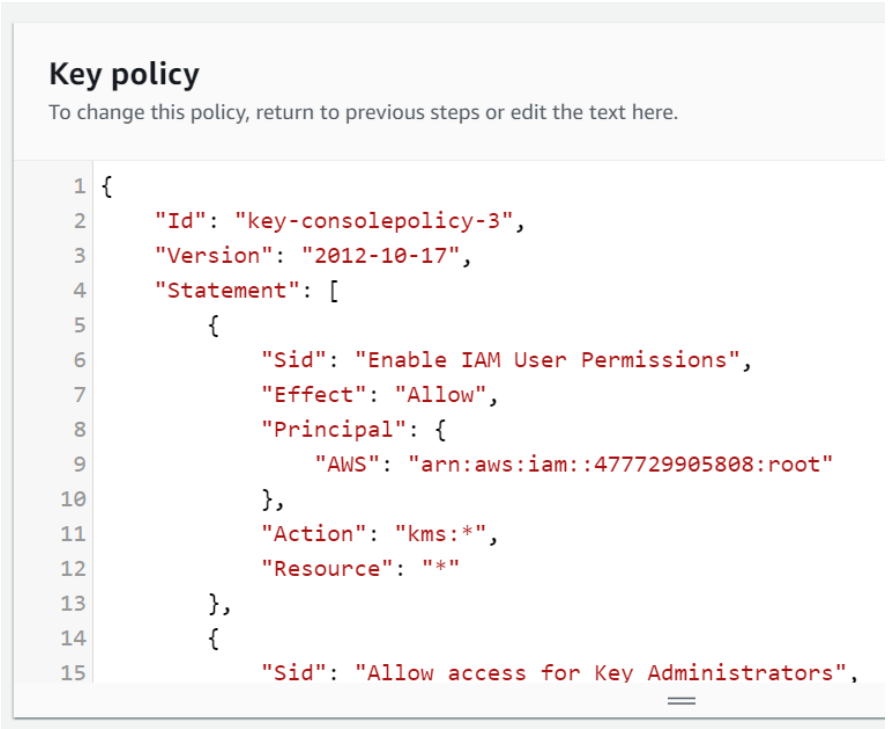
Creating a Customer managed key:



As part of the key configuration, a symmetric key was selected to be used for both encrypt and decrypt operations. Permission here is quite important, so the sender and recipient of the encrypted data will need to have valid AWS credentials to call AWS KMS. The symmetric encryption algorithm is both fast and efficient, but most importantly, it has the ability to provide confidentiality and authenticity of data which is the main focus of this portion of the project.



The following is the default key configuration created. The key policy is the main way to control access to KMS keys. As shown below, it includes the IAM user permissions. The complete policy is included in the Github repo of the project.



Key Policy

Created customer managed key

Aliases	Key ID	Status	Key spec	Key usage
keyOne	7562e23e-57b5-4ea1-96dc-6e17378ba271	Enabled	SYMMETRIC_DEFAULT	Encrypt and decrypt

Once the key is created, its cryptographic configuration like the key spec and key usage are also established (unmodifiable).

```
C:\Users\esam1\Security Project>aws configure
AWS Access Key ID [None]: AKIAW6OXL3SIHIN56PWK
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json

C:\Users\esam1\Security Project>aws kms generate-data-key --key-id alias/keyOne --key-spec AES_256 --region us-east-1
{
    "CiphertextBlob": "AQIDAHh1Ca6UcYVcCFp9U6m7/5s2l9mDrEX/4hp8y6p5vdWMtAEL6nCDB9XyRClRANbqA9x3AAAAfjB8BgkqhkiG9w0BBwagbzB
tAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMcMqheWQjOArFT1n3AgEQgDsZLtOvI+edAjVzeUxt8ZhGfGuAH+WftF0M3pW2yBnC71FIJjcwlYOuz
0Rb4TcVSyROX+ThdRxqrAYHtQ==",
    "Plaintext": "4so7Kw8/jhUPnnuX2jD25Z8cjAihwUtUY24vwfzAey8=",
    "KeyId": "arn:aws:kms:us-east-1:477729905808:key/7562e23e-57b5-4ea1-96dc-6e17378ba271"
}
```

AWS CLI was configured in order to generate the required data key by AWS KMS which can be used to perform the encrypt and decrypt operations.

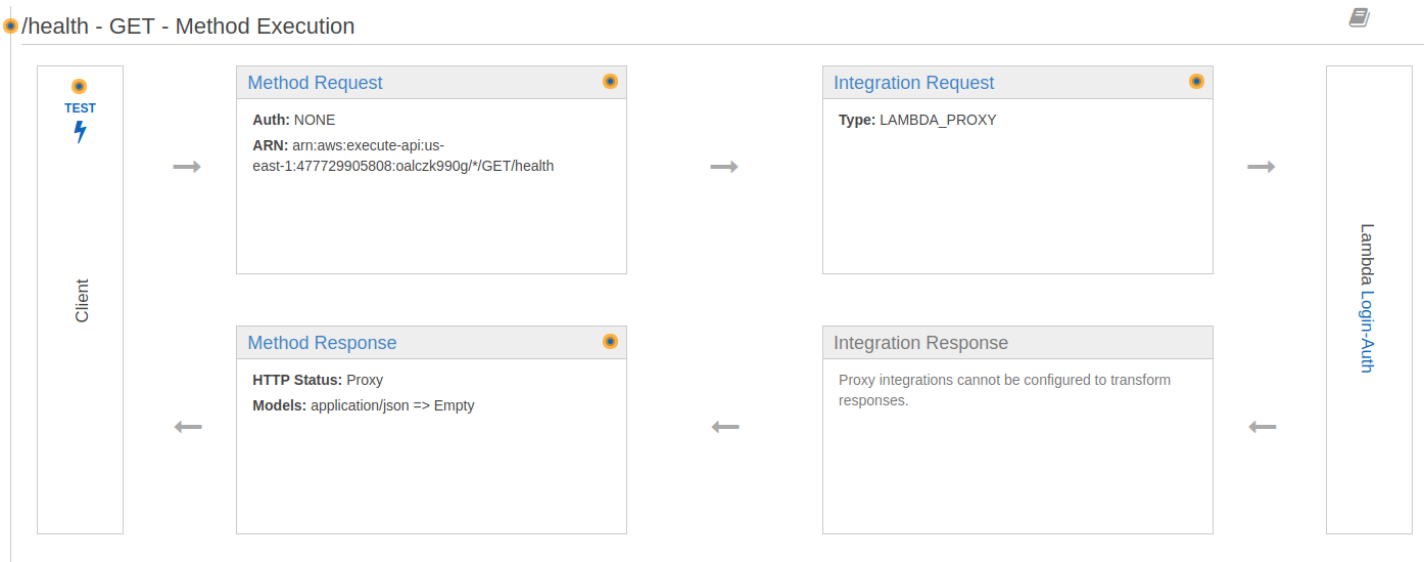
Sample keys created:

{
"CiphertextBlob":
"AQIDAHh1Ca6UcYVcCFp9U6m7/5s2l9mDrEX/4hp8y6p5vdWMtAEB7cbmsFkXT6O3GXY
cJWkUAAAAfjB8BgkqhkiG9w0BBwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQ
MEAS4wEQQMeNzziHA/Hy4/ia9pAgEQgDuXELDuQ+iVW061IzkNVj44eQ8O8KRLqYgnJc
heyVG+hPfSRdSj/vXC5hFnmCAVm8XPo8qyQ1X+qEQWAw==",
"Plaintext": "apbNORAzPt2VxJu5M7ie4wx/0NH88mN/YpcwF8B3IB0=",
"KeyId":
"arn:aws:kms:us-east-1:477729905808:key/7562e23e-57b5-4ea1-96dc-6e17378ba271"
}

Security Protocols Implemented:

- Data encryption using AWS KMS and OpenSSL
- Unique API Keys using API Gateway
- Multi-factor authorization using AWS
- Passwords encrypted and stored using BCrypt

Table Name : UserInfo
Role : Login-Admin



Architecture Diagram

Following is the architecture of the proposed implementation for our project. As previously mentioned, The API gateway secures access to AWS Lambda which is connected to the DynamoDB as backend. The lambda services have access controlled through AWS IAM.