SETU
Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Cert Alert

AWS Cert Alert
Final Report

Oisín Chelmiah
C00246756

Supervisor:
Dr Keara Barrett

Abstract

This report outlines the design and implementation of a certificate management system within Amazon Web Services (AWS) using a state machine with Lambda code. The system, aptly named AWS Cert Alert, automates the process of gathering information about Transport Layer Security (TLS) certificates stored in the account, logging expiring certificates in Security Hub, passing certificate details to a DynamoDB database, updating a dashboard, and notifying users of updates. Additionally, the system was expanded to include scanning Cloud Formation stacks, logging orphaned resources in Security Hub, logging all resources in another DynamoDB database, and updating a new page on the dashboard.

Acknowledgments

Before I get started, I’d like to thank a few people for helping me throughout the course of this project.

Firstly, I’d like to thank my supervisor Dr Keara Barrett for providing guidance throughout the project and keeping me on track.

I’d also like to thank my workplace mentors Shauna Bond and Rocia Fernandes from Unum Technology Centre Ireland for helping me come up with the idea for this project while I was on work placement.

Finally, I’d like to thank my family, friends, and my partner Evelyn Keane for helping me get through the year with my sanity intact.

Table of Contents

Table of Figures

Introduction

This report provides an overview of the Cert Alert system created within AWS. The system uses a state machine with Lambda code to automate the process of gathering information about certificates in use by AWS accounts, logging expiring certificates in Security Hub, passing certificate details to a DynamoDB database, updating a dashboard, and notifying users of updates.

In addition to the initial functionality, the system was expanded to include scanning Cloud Formation stacks, logging orphaned resources in Security Hub, logging all resources in another DynamoDB database, and updating a new page on the dashboard.

This report outlines the design and implementation of the system, as well as any issues faced throughout production and any changes that were made along the way. Additionally, it discusses potential areas for future development and improvement.

Overview

Why do this project?

Organizations are increasingly using cloud infrastructure to manage their IT infrastructure, and AWS is a popular cloud service provider. However, the management of these certificates and resources can become increasingly complex, especially when dealing with large-scale environments. AWS provides various tools for certificate management, such as Certificate Manager and Security Hub, but they still have limitations in terms of providing an overall view of certificate and resource status across multiple accounts. AWS's current tools for managing certificates and resources are limited and often require manual intervention, which can lead to errors and vulnerabilities. This can result in potential security issues and compliance violations, which can be costly for organizations in terms of money and reputation.

Organisations often face challenges with certificate management and resource management such as expired certificates, orphaned resources, and insecure signature algorithms. Best case scenario, these issues can cause operational downtime, security risks, and compliance issues, all of which can have a negative impact on the organisation's reputation and finances. In extreme cases, an expired certificate can cause the collapse of a whole organisation’s service.

A recent example of an expired certificate having a huge effect on an organisation can be viewed in a report by the Irish Times[1] about a major incident report that was submitted to the Department of Communications. According to the report, the 999 Emergency Call Answering Service as well as a fallback automated answering system lost all functionality after the certificate that the operators used to communicate with the computer system expired. Over 216 callers were affected, even though the system was only inoperable for an hour and twelve minutes, proving that certificate expiry not only affects the organisation but also the public that depend on the organisation.

AWS Cert Alert was developed to address these challenges by automating and streamlining the process of managing certificates, resources, and CloudFormation stacks within AWS accounts. The system utilizes a state machine with Lambda code to gather information about certificates in use, log expiring certificates in Security Hub, pass certificate details to a DynamoDB database, update a dashboard created with QuickSight Dashboard, and notify users of updates via email using Simple Notification Service.

With AWS Cert Alert, organisations can have a better overall view of the status of their certificates and resources, reducing the risk of operational downtime, security risks, and compliance issues. The system provides a central point for certificate and resource management, allowing organisations to quickly identify and address any issues that may arise. This saves time and resources, allowing organisations to focus on their core business operations.

[1] https://www.irishexaminer.com/news/arid-40967141.html

Project Description

Figure 1 – Cert Alert Logo

AWS Cert Alert is a comprehensive certificate management system designed to automate and streamline the process of managing certificates, resources, and Cloud Formation stacks within AWS accounts. The system utilizes a state machine with Lambda code to gather information about certificates in use, log expiring certificates in Security Hub, pass certificate details to a DynamoDB database, update a dashboard created with Quick Sight Dashboard, and notify users of updates via email using Simple Notification Service (SNS).

In addition to the initial functionality, AWS Cert Alert was expanded to include scanning Cloud Formation stacks, logging orphaned resources in Security Hub, logging all resources in another DynamoDB database, and updating a new page on the dashboard. This section of the system is called Stack Tracker to differentiate it from the Cert Alert part of the system.

To add to the certificate management aspect of the system, the system was further enhanced to check whether certificates are in use and evaluate the security of the signature algorithm being used. Based on the evaluation, the system makes a suggestion as to whether the certificate should be renewed, removed, or if the signature algorithm should be updated. This functionality provides users with valuable insights into the security of their certificates and enables them to take appropriate action to maintain a secure environment.

These additions provide enhanced security and organisation for the AWS accounts associated with the system. With the ability to detect expiring certificates, evaluate the security of the signature algorithm being used, and take automated actions to maintain a secure environment, AWS Cert Alert is a valuable tool for organisations looking to maintain secure and organized AWS environments.

Project Outline

Processing – Lambdas & State Machine

The main processing part of the system is performed by an AWS Step Functions state machine, depicted below. Each block represents one of the functions used by the state machine. These functions are written in Python 3.11 and are stored in AWS Lambda. The state machine is configured to run on a schedule by Amazon Event Bridge.

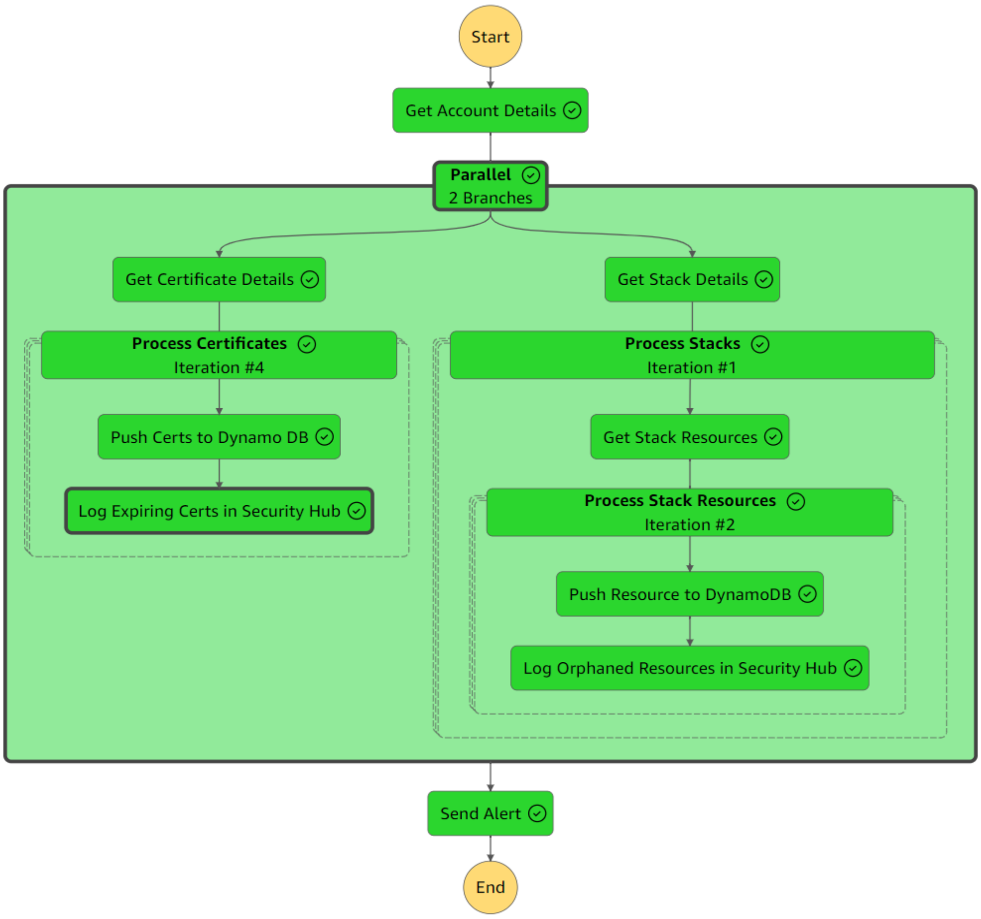


Figure 2 - Cert Alert State Machine

The state machine functions as follows:

- First, the state machine will check what AWS account it is running in.
- The state machine will then split into two parallel streams. One stream will scan all the TLS certificates stored in the account, the other stream will get the details of all resources currently allocated to stacks stored in Cloud Formation.
- For each certificate, the state machine will store all the details of the certificate in the Cert Alert database. Any certificates that have expired or have an upcoming expiry date are then logged as vulnerabilities in Security Hub.

- For each stack resource, the state machine will store the details of the resource in the Stack Tracker database. Any resources that have become orphaned are logged as vulnerabilities in Security Hub.
- Once all the processing has been complete, the Cert Alert dashboard will automatically update itself based on the data stored in the databases. An email notification is then sent to a user specified during the creation of the system to notify them that the dashboard has been updated.

The way the state machine is designed allows it to be updated with ease. Any new functionalities can be added in parallel to the already existing functionalities without altering them. This ensures that the system is adaptable and scalable.

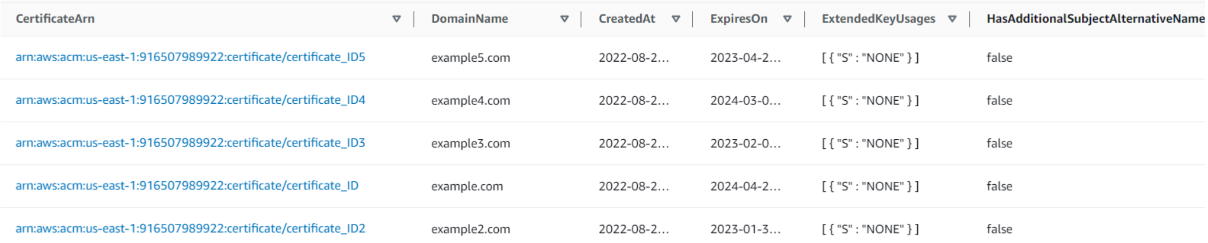
Logging – Dynamo Databases & Security Hub

The logging functionality of the system is split into two parts:

- Logging all certificate and resource details in databases
- Logging certificate expiry and orphaned resource vulnerabilities in Security Hub

Certificate details are stored in the Cert Alert database and stack resource details are stored in the Stack Tracker database. The benefit of storing these details in databases is that they can now be accessed by the dashboard part of the Cert Alert system.

Cert Alert Database

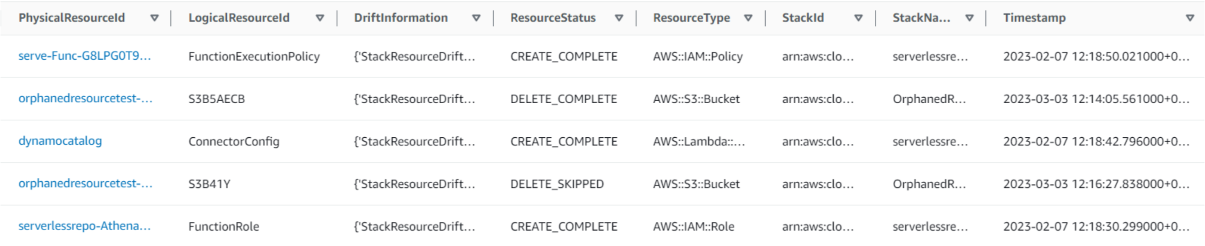


CertificateArn	DomainName	CreatedAt	ExpiresOn	ExtendedKeyUsages	HasAdditionalSubjectAlternativeName
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	example5.com	2022-08-2...	2023-04-2...	[{ "S" : "NONE" }]	false
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	example4.com	2022-08-2...	2024-03-0...	[{ "S" : "NONE" }]	false
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	example3.com	2022-08-2...	2023-02-0...	[{ "S" : "NONE" }]	false
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	example.com	2022-08-2...	2024-04-2...	[{ "S" : "NONE" }]	false
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	example2.com	2022-08-2...	2023-01-3...	[{ "S" : "NONE" }]	false

Figure 3 – Cert Alert Database

The data stored in the Cert Alert databases is taken directly from whatever is stored in Amazon Certificate Manager (ACM). Certificate details include certificate Amazon Resource Number (ARN), domain name, creation and expiration dates, key and signature algorithms, and contains a lot of extra data that can be hard to read, especially for anyone new to certificate management. The dashboard section of the system takes in this data and displays it in an easy-to-read format as well as highlighting the most important details of each certificate.

Stack Tracker Database Description

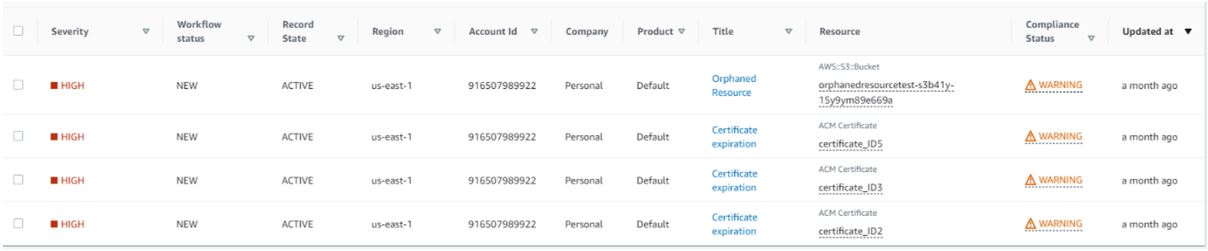


PhysicalResourceId	LogicalResourceId	DriftInformation	ResourceStatus	ResourceType	StackId	StackNa...	Timestamp
serve-Func-G8LPG0T9...	FunctionExecutionPolicy	{'StackResourceDrift...	CREATE_COMPLETE	AWS::IAM::Policy	arn:aws:clo...	serverlessre...	2023-02-07 12:18:50.021000+0...
orphanedresourcetest-...	S3B5AECB	{'StackResourceDrift...	DELETE_COMPLETE	AWS::S3::Bucket	arn:aws:clo...	OrphanedR...	2023-03-03 12:14:05.561000+0...
dynamocatalog	ConnectorConfig	{'StackResourceDrift...	CREATE_COMPLETE	AWS::Lambda::...	arn:aws:clo...	serverlessre...	2023-02-07 12:18:42.796000+0...
orphanedresourcetest-...	S3B41Y	{'StackResourceDrift...	DELETE_SKIPPED	AWS::S3::Bucket	arn:aws:clo...	OrphanedR...	2023-03-03 12:16:27.838000+0...
serverlessrepo-Athena...	FunctionRole	{'StackResourceDrift...	CREATE_COMPLETE	AWS::IAM::Role	arn:aws:clo...	serverlessre...	2023-02-07 12:18:30.299000+0...

Figure 4 – Stack Tracker Database

The Stack Tracker database contains details of resources stored in Cloud Formation stacks. A CloudFormation stack is a collection of AWS resources that you can manage as a single unit.

Security Hub



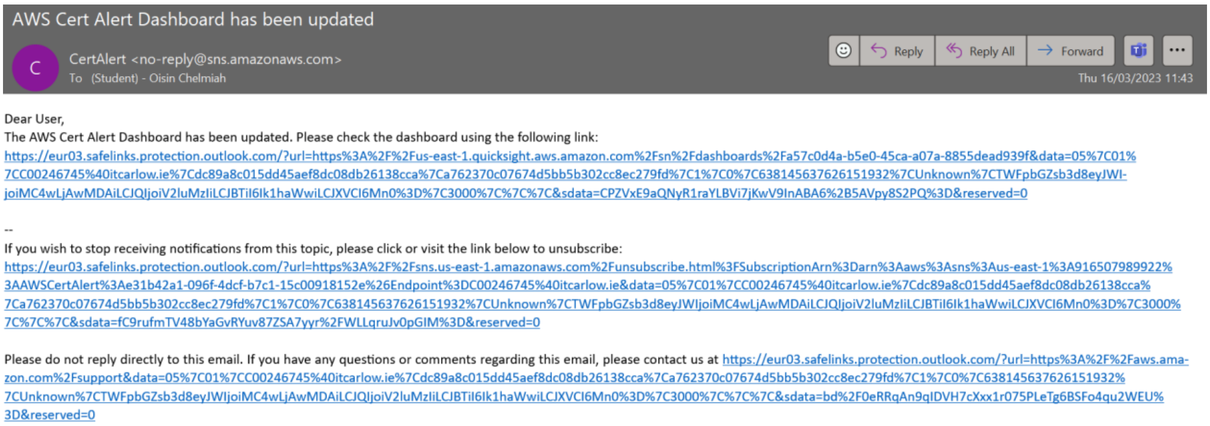
□	Severity	Workflow status	Record State	Region	Account Id	Company	Product	Title	Resource	Compliance Status	Updated at
□	HIGH	NEW	ACTIVE	us-east-1	916507989922	Personal	Default	Orphaned Resource	AWS::S3::Bucket orphanedresourcetest-s3b41y-15y9ym89e669a	⚠ WARNING	a month ago
□	HIGH	NEW	ACTIVE	us-east-1	916507989922	Personal	Default	Certificate expiration	ACM Certificate certificate_ID5	⚠ WARNING	a month ago
□	HIGH	NEW	ACTIVE	us-east-1	916507989922	Personal	Default	Certificate expiration	ACM Certificate certificate_ID3	⚠ WARNING	a month ago
□	HIGH	NEW	ACTIVE	us-east-1	916507989922	Personal	Default	Certificate expiration	ACM Certificate certificate_ID2	⚠ WARNING	a month ago

Figure 5 – Security Hub

Security Hub is an AWS service that scans AWS accounts for vulnerabilities and stores these vulnerabilities in a log. My system adds to this service by creating three logs for vulnerabilities that Security Hub does not currently scan for:

- Certificate Expired – A certificate has passed its expiration date and needs to be renewed.
- Certificate with Upcoming Expiry – A certificate has not expired yet, but its expiry date is within 45 days.
- Orphaned Resource – A resource that was part of a Cloud Formation stack where the stack was deleted but the resource itself still exists.

Notification – Email



AWS Cert Alert Dashboard has been updated

CertAlert <no-reply@sns.amazonaws.com>
To (Student) - Oisin Chelmiah
Thu 16/03/2023 11:43

Dear User,
The AWS Cert Alert Dashboard has been updated. Please check the dashboard using the following link:
https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fus-east-1.quicksight.aws.amazon.com%2Fsn%2Fdashboards%2Fa57c0d4a-b5e0-45ca-a07a-8855dead939f&data=05%7C01%7CC00246745%40itcarlow.ie%7Cdc89a8c015dd45aef8dc08db26138cca%7Ca762370c07674d5bb5b302cc8ec279fd%7C1%7C0%7C638145637626151932%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=CPZVxE9aQNyR1raYLBVi7jKwV9InABA6%2B5AVpy8S2PQ%3D&reserved=0

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsns.us-east-1.amazonaws.com%2Funsubscribe.html%3FSubscriptionArn%3Darn%3Aaws%3Asns%3Aus-east-1%3A916507989922%3AAWSCertAlert%3Ae31b42a1-096f-4dcf-b7c1-15c00918152e%26Endpoint%3DC00246745%40itcarlow.ie&data=05%7C01%7CC00246745%40itcarlow.ie%7Cdc89a8c015dd45aef8dc08db26138cca%7Ca762370c07674d5bb5b302cc8ec279fd%7C1%7C0%7C638145637626151932%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=fC9rufmTV48bYaGvRYuv87ZSA7yyr%2FWLLqruJv0pGIM%3D&reserved=0

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Faws.amazon.com%2Fsupport&data=05%7C01%7CC00246745%40itcarlow.ie%7Cdc89a8c015dd45aef8dc08db26138cca%7Ca762370c07674d5bb5b302cc8ec279fd%7C1%7C0%7C638145637626151932%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=bd%2F0eRRqAn9qIDVH7cXxx1r075PLeTg6BSFo4qu2WEU%3D&reserved=0

Figure 6 – SNS Email Notification

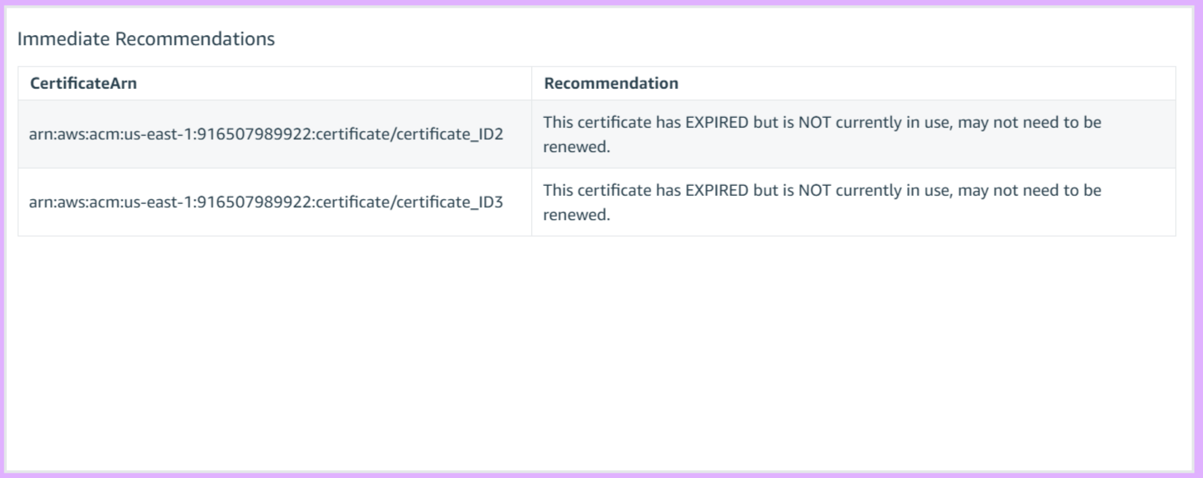
The email notification is sent using Amazon SNS. The email is sent as the last function of the state machine and can be configured to be sent to multiple users or an email list if needed. The email contains a link to the dashboard that can be viewed once the recipient logs into their AWS account.

Dashboard – Quick Sight Dashboard

The dashboard is split into two pages, the Cert Alert page and the Stack Tracker page. The pages in full can be viewed at *Appendix 1* and *Appendix 2*. The section below breaks down each page into its individual elements. The dashboard is fully customisable the owner of the AWS account that the Cert Alert system is deployed in. The read only version of the dashboard is the version that is sharable, however a user must have an AWS account to view the dashboard. The dashboard can also be exported to a PDF file for easy sharing.

Cert Alert Page

Immediate Recommendations



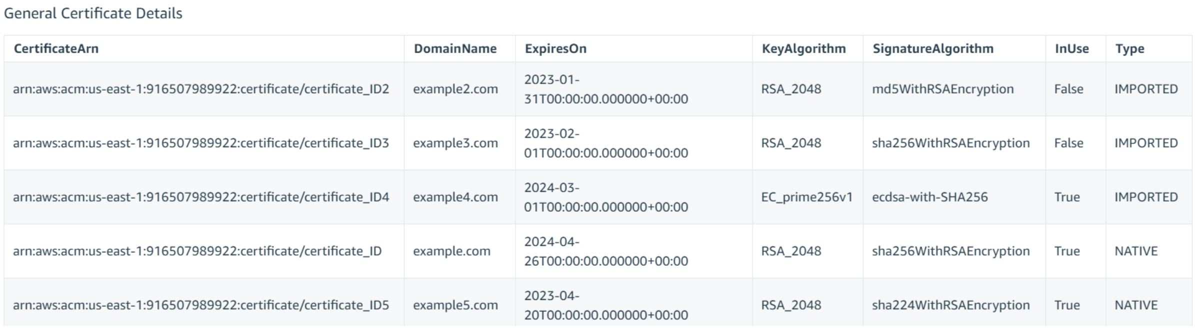
Immediate Recommendations

CertificateArn	Recommendation
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	This certificate has EXPIRED but is NOT currently in use, may not need to be renewed.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	This certificate has EXPIRED but is NOT currently in use, may not need to be renewed.

Figure 7 – Cert Alert: Immediate Recommendations

The Immediate Recommendations table is the very first thing the user sees when they open the dashboard. It gives recommendations based on all of the data supplied on each certificate and then shows any certificates that require immediate action. These certificates are certificates that need to be renewed, certificates that are not in use. and certificates using insecure signature algorithms.

General Certificate Details Table



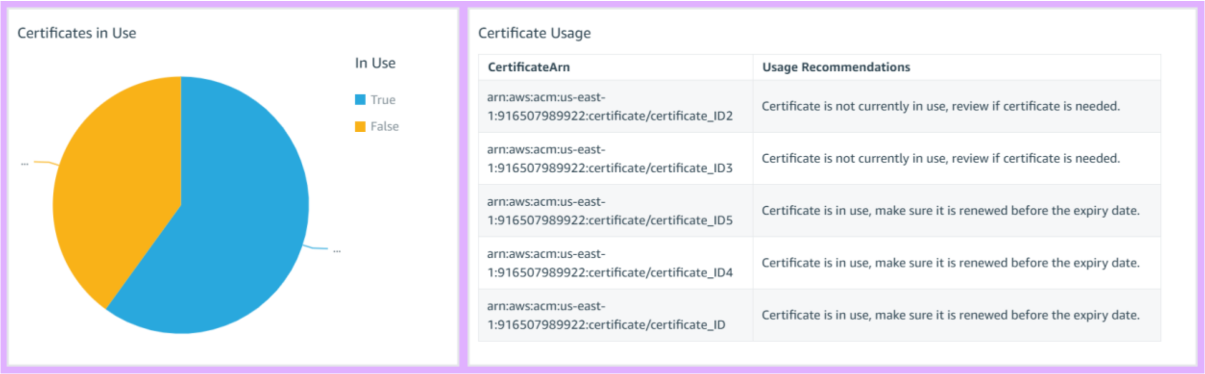
General Certificate Details

CertificateArn	DomainName	ExpiresOn	KeyAlgorithm	SignatureAlgorithm	InUse	Type
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	example2.com	2023-01-31T00:00:00.000000+00:00	RSA_2048	md5WithRSAEncryption	False	IMPORTED
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	example3.com	2023-02-01T00:00:00.000000+00:00	RSA_2048	sha256WithRSAEncryption	False	IMPORTED
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	example4.com	2024-03-01T00:00:00.000000+00:00	EC_prime256v1	ecdsa-with-SHA256	True	IMPORTED
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	example.com	2024-04-26T00:00:00.000000+00:00	RSA_2048	sha256WithRSAEncryption	True	NATIVE
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	example5.com	2023-04-20T00:00:00.000000+00:00	RSA_2048	sha224WithRSAEncryption	True	NATIVE

Figure 8 – Cert Alert: General Certificate Details

This table displays key information on all of the certificates. Details include: certificate ARN, domain name, expiry date, key algorithm, signature algorithm, whether the certificate is in use, and whether the certificate is imported or a native AWS certificate.

Certificates in Use



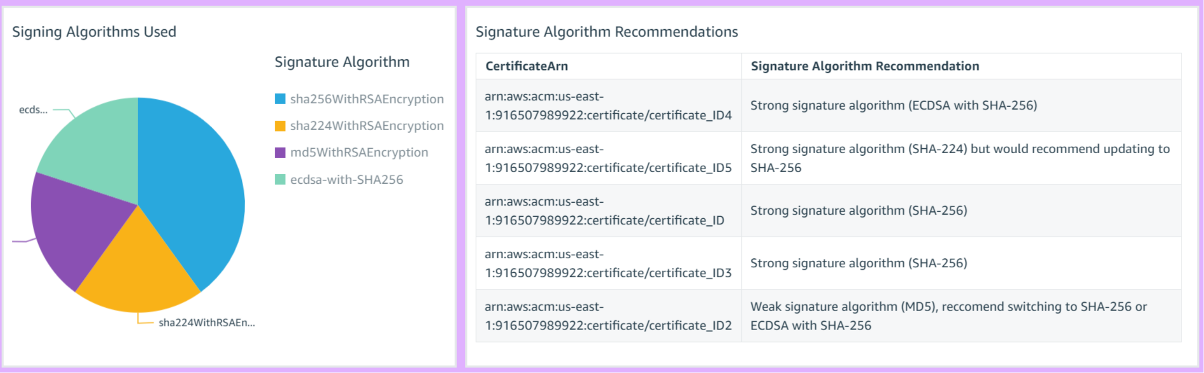
Certificate Usage

CertificateArn	Usage Recommendations
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	Certificate is not currently in use, review if certificate is needed.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	Certificate is not currently in use, review if certificate is needed.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	Certificate is in use, make sure it is renewed before the expiry date.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	Certificate is in use, make sure it is renewed before the expiry date.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	Certificate is in use, make sure it is renewed before the expiry date.

Figure 9 – Cert Alert: Certificates in Use

The certificates in use table has a pie chart associated with it to show whether certificates are in use in the AWS account.

Signature Algorithm



Signature Algorithm Recommendations

CertificateArn	Signature Algorithm Recommendation
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	Strong signature algorithm (ECDSA with SHA-256)
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	Strong signature algorithm (SHA-224) but would recommend updating to SHA-256
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	Strong signature algorithm (SHA-256)
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	Strong signature algorithm (SHA-256)
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	Weak signature algorithm (MD5), reccomend switching to SHA-256 or ECDSA with SHA-256

Figure 10 – Cert Alert: Signature Algorithm

The signature algorithm section gives details about what signature algorithms each certificate is using and gives recommendations based on these algorithms. If a certificate is using an old or insecure signature algorithm, the system will recommend to update or change the algorithm.

Other Details

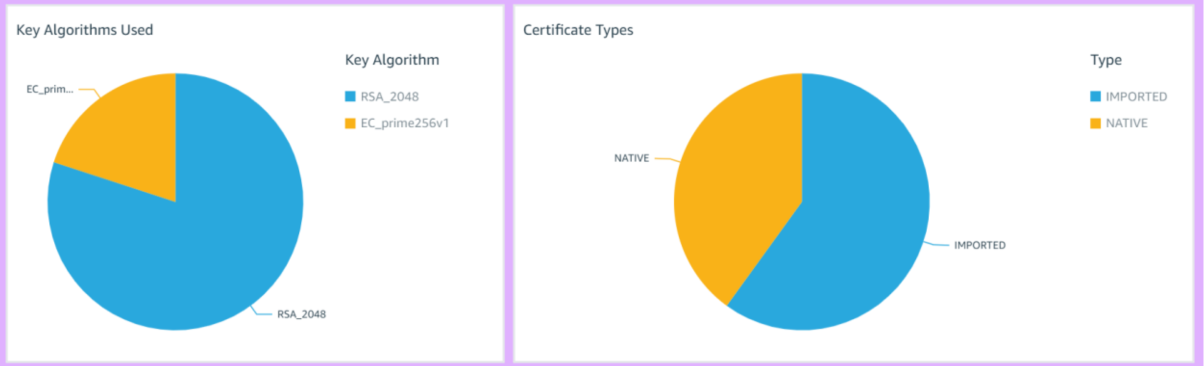
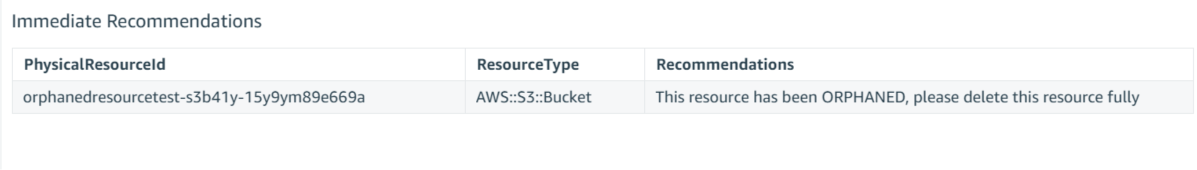


Figure 11 – Cert Alert: Other Details

The dashboard also has charts depicting what key algorithms are in use and whether certificates are native to AWS or imported from another certificate authority.

Stack Tracker Page

Immediate Recommendations



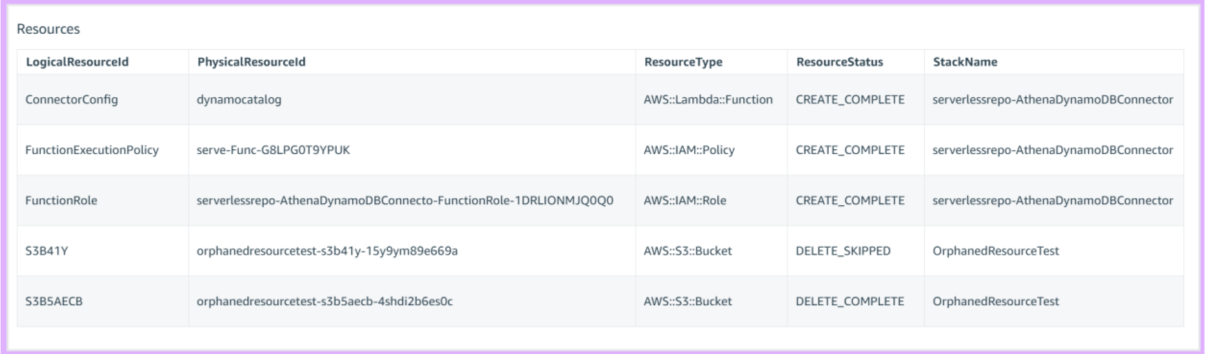
Immediate Recommendations

PhysicalResourceId	ResourceType	Recommendations
orphanedresourcetest-s3b41y-15y9ym89e669a	AWS::S3::Bucket	This resource has been ORPHANED, please delete this resource fully

Figure 12 – Stack Tracker: Immediate Recommendations

Like the Cert Alert page, this table is the first thing that a user will see when accessing the dashboard. It details resources any orphaned resources that should be reviewed and deleted if necessary.

Resource Details



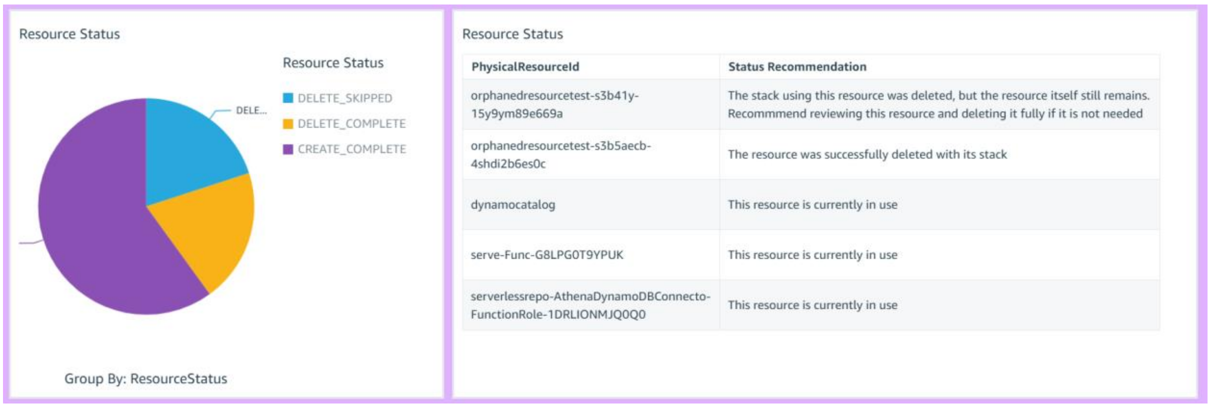
Resources

LogicalResourceId	PhysicalResourceId	ResourceType	ResourceStatus	StackName
ConnectorConfig	dynamocatalog	AWS::Lambda::Function	CREATE_COMPLETE	serverlessrepo-AthenaDynamoDBConnector
FunctionExecutionPolicy	serve-Func-G8LPG0T9YPUK	AWS::IAM::Policy	CREATE_COMPLETE	serverlessrepo-AthenaDynamoDBConnector
FunctionRole	serverlessrepo-AthenaDynamoDBConnecto-FunctionRole-1DRLIONMJQ0Q0	AWS::IAM::Role	CREATE_COMPLETE	serverlessrepo-AthenaDynamoDBConnector
S3B41Y	orphanedresourcetest-s3b41y-15y9ym89e669a	AWS::S3::Bucket	DELETE_SKIPPED	OrphanedResourceTest
S3B5AECB	orphanedresourcetest-s3b5aecb-4shdi2b6es0c	AWS::S3::Bucket	DELETE_COMPLETE	OrphanedResourceTest

Figure 13 – Stack Tracker: Resource Details

This table contains all the details of resources stored in Cloud Formation stacks. Details include: logical resource ID, physical resource ID, resource type, resource status, and the name of the stack the resource is stored in.

Resource Status



PhysicalResourceId	Status Recommendation
orphanedresourcetest-s3b41y-15y9ym89e669a	The stack using this resource was deleted, but the resource itself still remains. Recommmend reviewing this resource and deleting it fully if it is not needed
orphanedresourcetest-s3b5aecb-4shdi2b6es0c	The resource was successfully deleted with its stack
dynamocatalog	This resource is currently in use
serve-Func-G8LPG0T9YPUK	This resource is currently in use
serverlessrepo-AthenaDynamoDBConnecto-FunctionRole-1DRLIONMJQ0Q0	This resource is currently in use

Figure 14 – Stack Tracker: Resource Status

This section details the status of each resource. The resource status field has three possible values:

- CREATE_COMPLETE – The resource and its stack are currently in use.
- DELETE_COMPLETE – The resource and its stack have been deleted.
- DELETED_SKIPPED – The stack the resource was a part of was deleted but the resource still exists, meaning it has become orphaned.

Other Details

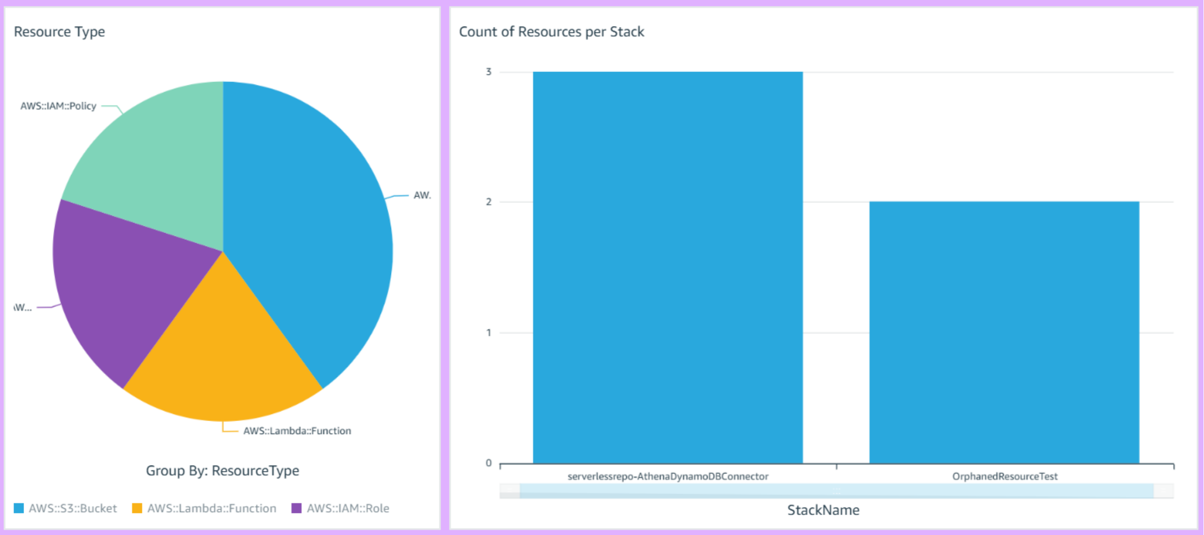


Figure 15 – Stack Tracker: Other Details

The dashboard also gives a breakdown of how many of each type of resource is being used in the AWS account as well as how many resources each stack contains.

Project Review

Specification Achievements

Specification	Achieved
Logging to Database – Core	✓
Logging to Security Hub – Non-Core	✓
Email – Core	✓
Text Message – Non-Core	✗
Displaying Cert Details – Core	✓
Analyse Signing Algorithms Used – Non-Core	✓
Displaying Resources – Core	✓
Filtering – Non-Core	✓
Exporting – Non-Core	✓
Sharing – Non-Core	✓

Table 1 – Specification Achievements

Possible Future Developments

Development	Details
Integrations with third-party tools	AWS Cert Alert could be integrated with third-party security tools such as Splunk for extra logging or ServiceNow to create tickets for the vulnerabilities.
Automated certificate remediation	In addition to suggesting remediation actions, the system could be enhanced to automatically take corrective actions when possible, such as renewing or updating certificates.
Integration with AWS KMS	The system could be integrated with AWS Key Management Service (KMS) to enable users to centrally manage keys used for certificate signing and encryption. This would help ensure that keys are secure and managed in compliance with industry standards.

Table 2 – Possible Future Developments

General Issues

Issue	Details	Mitigation
Getting Certificates	Had trouble getting certificates initially as I did not have a website to assign certificates to.	Created a mockCerts Lambda function which contained an array of mock certificates to use when testing the system. A sample mock certificate can be viewed at *Appendix 3*.
Analysing the Data	The dashboard provides several recommendations for the certificates and the stack resources. When I initially tried to generate these recommendations I had trouble figuring out when to analyse the data and create these recommendations.	After some research, I learned about Quick Sight language, which is similar to the programming language implemented by Excel. Using Quick Sight language allowed me to perform some code-like functions on the dashboard itself and hence generate the recommendations I needed.
Displaying the Data	The certificate data stored in ACM was tough to read and understand. There was a lot of unnecessary data that would be hard for an inexperienced user to read.	To make it easier to read, I decided on breaking down the data into separate segments and presenting each of these segments separately.

Table 3 – General Issues

Learning Outcomes

Learning Outcome	Description
Proficient with New Languages	Before starting the project I had very little experience with Python, but after a few months of development I feel very confident in my ability in the language. I also got to learn a new language – Quick Sight language – which I was able to pick up quickly due to my prior programming knowledge.
Certificate Management	Throughout the development of this project I learned a lot about certificates and certificate management which built upon my knowledge about TLS certificate, encryption algorithms, signature algorithms, certificate authorities,
AWS Proficiency	By developing this system completely in AWS I exponentially improved my knowledge and skills within the platform.

Table 4 – Learning Outcomes

Testing

Test Number	Details	Result
1	Created an array with one mock certificate.	**PASS**
2	Added multiple certificates to the array.	**PASS**
3	Set some certificates in the with an expired expiration date.	**PASS**
4	Set some certificates in the with an expiration date within 45 days.	**PASS**
5	Changed signature algorithms of some certificates.	**PASS**
6	Changed key algorithms of some certificates.	**PASS**
7	Changed some certificates to be native instead of imported.	**PASS**
8	Changed some of the certificates to show as not in use.	**PASS**
9	Created a Cloud Formation stack with resources.	**PASS**
10	Successfully deleted the stack and all its resources	**PASS**
11	Created a new stack, deleted the stack without deleting the resources.	**PASS**
12	Combined the Cert Alert and Stack Tracker functionalities into one system.	**PASS**

Table 5 – Testing

Summary & Conclusion

In conclusion, the AWS Cert Alert project has been a valuable learning experience in the design, implementation, and deployment of a comprehensive certificate management system within AWS. Through our use of AWS serverless technologies, including Lambda functions, Step Functions, DynamoDB, and QuickSight Dashboard, we have created a highly scalable, reliable, and cost-effective system that provides valuable insights into certificate expiration, resource management, and potential security risks.

Looking forward, there are several possible avenues for future development and improvement of the system. These include expanding the system to automated remediations, integrating with third-party services for enhanced functionality, and incorporating Amazon KMSs. I believe that the project has tremendous potential for further innovation and look forward to seeing its continued success in the future.

Overall, the AWS Cert Alert project has been an exciting and rewarding experience, and I am grateful for the opportunity to have worked on such a challenging and impactful project. I are confident that the skills and knowledge I have gained through this project will serve me well in my future endeavours, and I look forward to applying them to new challenges and opportunities.

Plagiarism Declaration

I declare that all material in this submission is entirely my own work except where duly acknowledged. I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

I have provided a complete bibliography of all works and sources used in the preparation of this submission. I understand that failure to comply with the Institute's regulations governing plagiarism constitute a serious offence.

Student Name: Oisín Chelmiah

Student Number: C00246745

Date: 17/04/2023

Signed: Oisín Chelmiah

Glossary

AWS – Amazon Web Services

TLS – Transport Layer Security

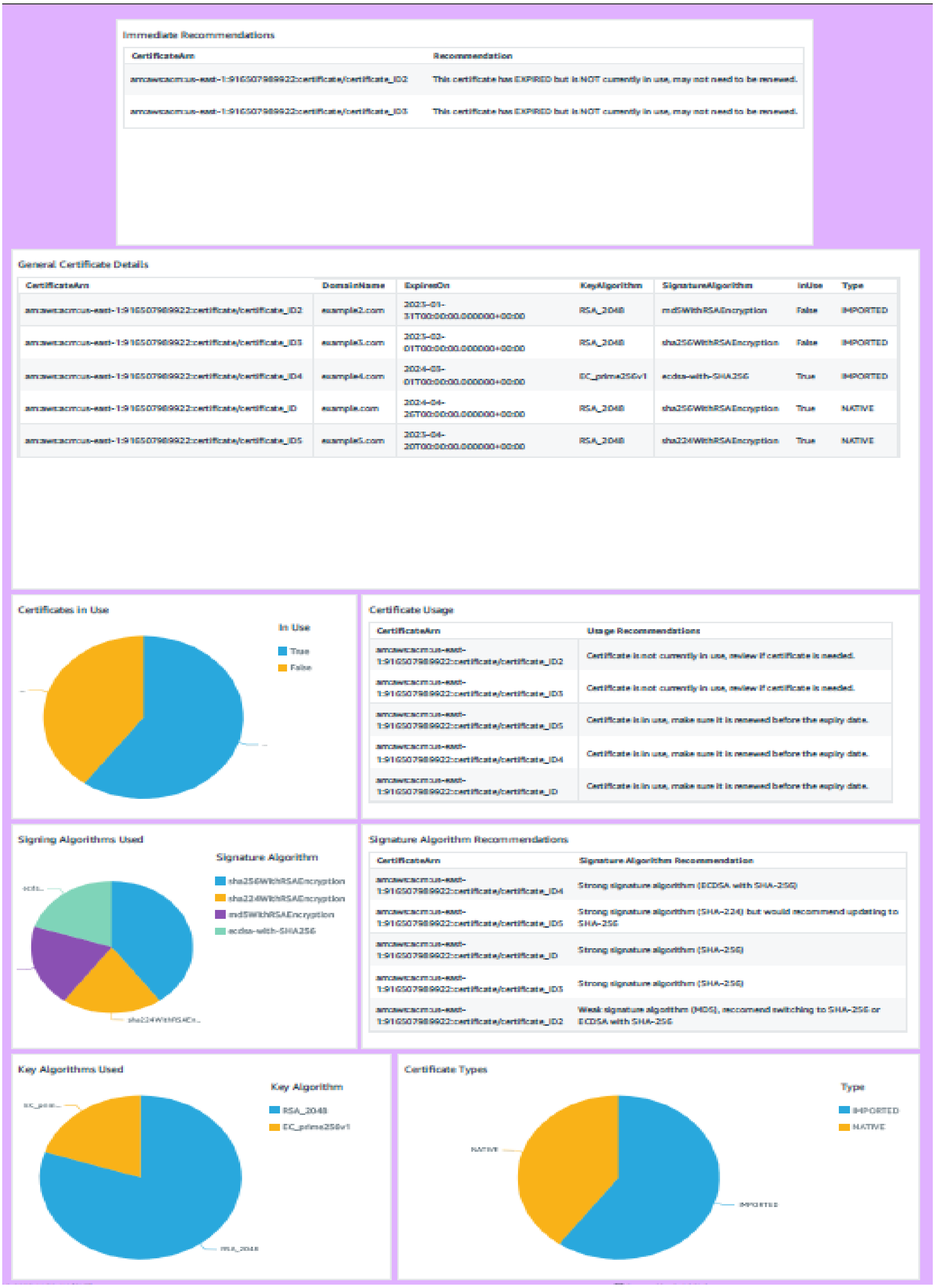
SNS – Simple Notification Service

ACM – Amazon Certificate Manager

ARN – Amazon Resource Number

KMS – Key Management System

Appendices



Immediate Recommendations

CertificateArn	Recommendation
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	This certificate has EXPIRED but is NOT currently in use, may not need to be renewed.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	This certificate has EXPIRED but is NOT currently in use, may not need to be renewed.

General Certificate Details

CertificateArn	DomainName	ExpiresOn	KeyAlgorithm	SignatureAlgorithm	InUse	Type
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	example2.com	2023-01-31T00:00:00.000000+00:00	RSA_2048	md5WithRSAEncryption	False	IMPORTED
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	example3.com	2023-03-01T00:00:00.000000+00:00	RSA_2048	sha256WithRSAEncryption	False	IMPORTED
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	example4.com	2024-03-01T00:00:00.000000+00:00	EC_prime256v1	ecdsa-with-SHA256	True	IMPORTED
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	example.com	2024-04-26T00:00:00.000000+00:00	RSA_2048	sha256WithRSAEncryption	True	NATIVE
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	example5.com	2023-04-20T00:00:00.000000+00:00	RSA_2048	sha224WithRSAEncryption	True	NATIVE

Certificates in Use

Certificate Usage

CertificateArn	Usage Recommendations
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	Certificate is not currently in use, review if certificate is needed.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	Certificate is not currently in use, review if certificate is needed.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	Certificate is in use, make sure it is renewed before the expiry date.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	Certificate is in use, make sure it is renewed before the expiry date.
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	Certificate is in use, make sure it is renewed before the expiry date.

Signing Algorithms Used

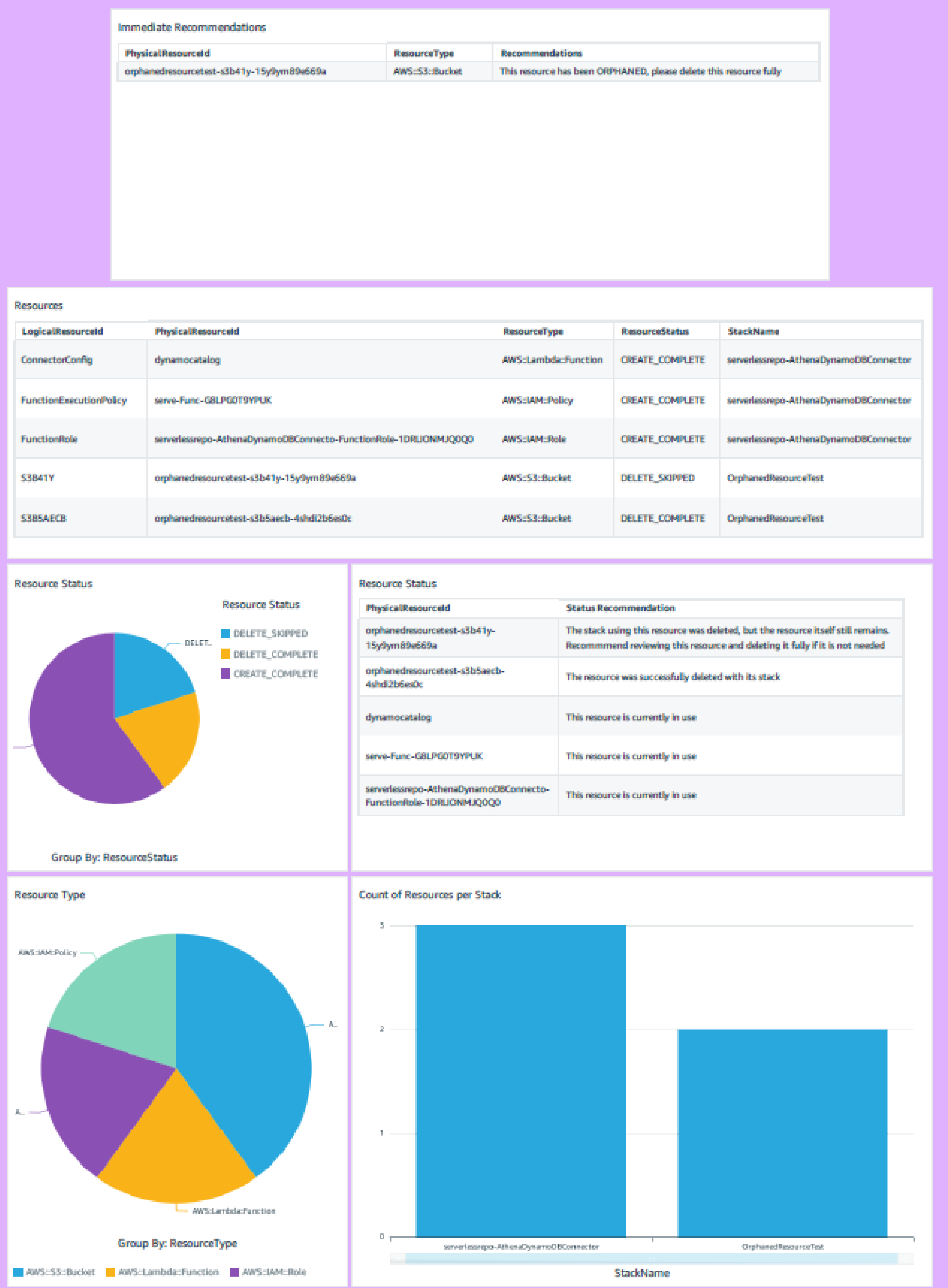
Signature Algorithm Recommendations

CertificateArn	Signature Algorithm Recommendation
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4	Strong signature algorithm (ECDSA with SHA-256)
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5	Strong signature algorithm (SHA-224) but would recommend updating to SHA-256
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID	Strong signature algorithm (SHA-256)
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3	Strong signature algorithm (SHA-256)
arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2	Weak signature algorithm (MD5), reccomend switching to SHA-256 or ECDSA with SHA-256

Key Algorithms Used

Certificate Types

Appendix 1 – Cert Alert Page



Immediate Recommendations

PhysicalResourceId	ResourceType	Recommendations
orphanedresourcetest-s3b41y-15y9ym89e669a	AWS::S3::Bucket	This resource has been ORPHANED, please delete this resource fully

Resources

LogicalResourceId	PhysicalResourceId	ResourceType	ResourceStatus	StackName
ConnectorConfig	dynamocatalog	AWS::Lambda::Function	CREATE_COMPLETE	serverlessrepo-AthenaDynamoDBConnector
FunctionExecutionPolicy	serve-Func-G8LPG0T9YPUK	AWS::IAM::Policy	CREATE_COMPLETE	serverlessrepo-AthenaDynamoDBConnector
FunctionRole	serverlessrepo-AthenaDynamoDBConnecto-FunctionRole-1DRUONMJQ0Q0	AWS::IAM::Role	CREATE_COMPLETE	serverlessrepo-AthenaDynamoDBConnector
S3B41Y	orphanedresourcetest-s3b41y-15y9ym89e669a	AWS::S3::Bucket	DELETE_SKIPPED	OrphanedResourceTest
S3B5AECB	orphanedresourcetest-s3b5aecb-4shdi2b6es0c	AWS::S3::Bucket	DELETE_COMPLETE	OrphanedResourceTest

Resource Status

Resource Status

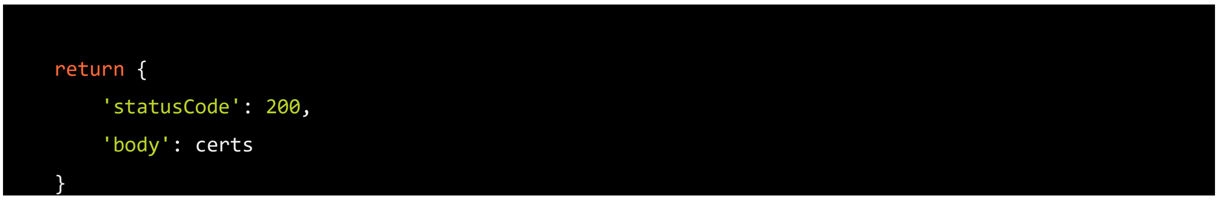
PhysicalResourceId	Status Recommendation
orphanedresourcetest-s3b41y-15y9ym89e669a	The stack using this resource was deleted, but the resource itself still remains. Recommmend reviewing this resource and deleting it fully if it is not needed
orphanedresourcetest-s3b5aecb-4shdi2b6es0c	The resource was successfully deleted with its stack
dynamocatalog	This resource is currently in use
serve-Func-G8LPG0T9YPUK	This resource is currently in use
serverlessrepo-AthenaDynamoDBConnecto-FunctionRole-1DRUONMJQ0Q0	This resource is currently in use

Resource Type

Count of Resources per Stack

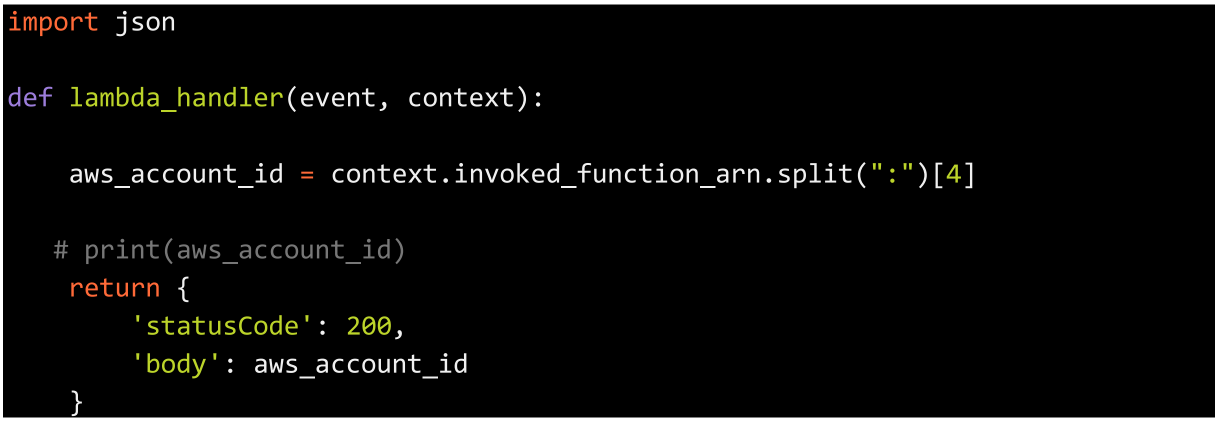
Appendix 2 – Stack Tracker Page

```
import json
import boto3

def lambda_handler(event, context):

    account = event

    certs = [
        {
            "CertificateArn": "arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID",
            "DomainName": "example.com",
            "SubjectAlternativeNameSummaries": [
                "example.com",
                "other.example.com"
            ],
            "HasAdditionalSubjectAlternativeNames": "false",
            "Status": "ISSUED",
            "Type": "NATIVE",
            "KeyAlgorithm": "RSA_2048",
            "SignatureAlgorithm": "sha256WithRSAEncryption",
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
            ],
            "InUse": "True",
            "RenewalEligibility": "INELIGIBLE",
            "NotBefore": "2022-06-14T23:42:49+00:00",
            "NotAfter": "2032-06-11T23:42:49+00:00",
            "CreatedAt": "2022-08-25T19:28:05.531000+00:00",
            "ImportedAt": "2022-08-25T19:28:05.544000+00:00",
            "ExpiresOn": "2024-04-26T00:00:00.000000+00:00"
        },
        {
            "CertificateArn": "arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID2",
            "DomainName": "example2.com",
            "SubjectAlternativeNameSummaries": [
                "example2.com",
                "other2.example.com"
            ],
            "HasAdditionalSubjectAlternativeNames": "false",
            "Status": "ISSUED",
            "Type": "IMPORTED",
            "KeyAlgorithm": "RSA_2048",
            "SignatureAlgorithm": "md5WithRSAEncryption",
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
            ],
            "InUse": "False",
            "RenewalEligibility": "INELIGIBLE",
            "NotBefore": "2022-06-14T23:42:49+00:00",
            "NotAfter": "2023-01-31T23:42:49+00:00",
            "CreatedAt": "2022-08-25T19:28:05.531000+00:00",
            "ImportedAt": "2022-08-25T19:28:05.544000+00:00",
            "ExpiresOn": "2023-01-31T00:00:00.000000+00:00"
        },
        {
            "CertificateArn": "arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID3",
            "DomainName": "example3.com",
            "SubjectAlternativeNameSummaries": [
                "example3.com",
                "other.example3.com"
            ],
            "HasAdditionalSubjectAlternativeNames": "false",
            "Status": "ISSUED",
            "Type": "IMPORTED",
            "KeyAlgorithm": "RSA_2048",
            "SignatureAlgorithm": "sha256WithRSAEncryption",
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
            ],
            "InUse": "False",
            "RenewalEligibility": "INELIGIBLE",
            "NotBefore": "2022-06-14T23:42:49+00:00",
            "NotAfter": "2032-06-11T23:42:49+00:00",
            "CreatedAt": "2022-08-25T19:28:05.531000+00:00",
            "ImportedAt": "2022-08-25T19:28:05.544000+00:00",
            "ExpiresOn": "2023-02-01T00:00:00.000000+00:00"
        },
        {
            "CertificateArn": "arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID4",
            "DomainName": "example4.com",
            "SubjectAlternativeNameSummaries": [
                "example4.com",
                "other.example4.com"
            ],
            "HasAdditionalSubjectAlternativeNames": "false",
            "Status": "ISSUED",
            "Type": "IMPORTED",
            "KeyAlgorithm": "EC_prime256v1",
            "SignatureAlgorithm": "ecdsa-with-SHA256",
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
            ],
            "InUse": "True",
            "RenewalEligibility": "INELIGIBLE",
            "NotBefore": "2022-06-14T23:42:49+00:00",
            "NotAfter": "2032-06-11T23:42:49+00:00",
            "CreatedAt": "2022-08-25T19:28:05.531000+00:00",
            "ImportedAt": "2022-08-25T19:28:05.544000+00:00",
            "ExpiresOn": "2024-03-01T00:00:00.000000+00:00"
        },
        {
            "CertificateArn": "arn:aws:acm:us-east-1:916507989922:certificate/certificate_ID5",
            "DomainName": "example5.com",
            "SubjectAlternativeNameSummaries": [
                "example5.com",
                "other.example5.com"
            ],
            "HasAdditionalSubjectAlternativeNames": "false",
            "Status": "ISSUED",
            "Type": "NATIVE",
            "KeyAlgorithm": "RSA_2048",
            "SignatureAlgorithm": "sha224WithRSAEncryption",
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
            ],
            "InUse": "True",
            "RenewalEligibility": "INELIGIBLE",
            "NotBefore": "2022-06-14T23:42:49+00:00",
            "NotAfter": "2032-06-11T23:42:49+00:00",
            "CreatedAt": "2022-08-25T19:28:05.531000+00:00",
            "ImportedAt": "2022-08-25T19:28:05.544000+00:00",
            "ExpiresOn": "2023-04-20T00:00:00.000000+00:00"
        }
    ]
```



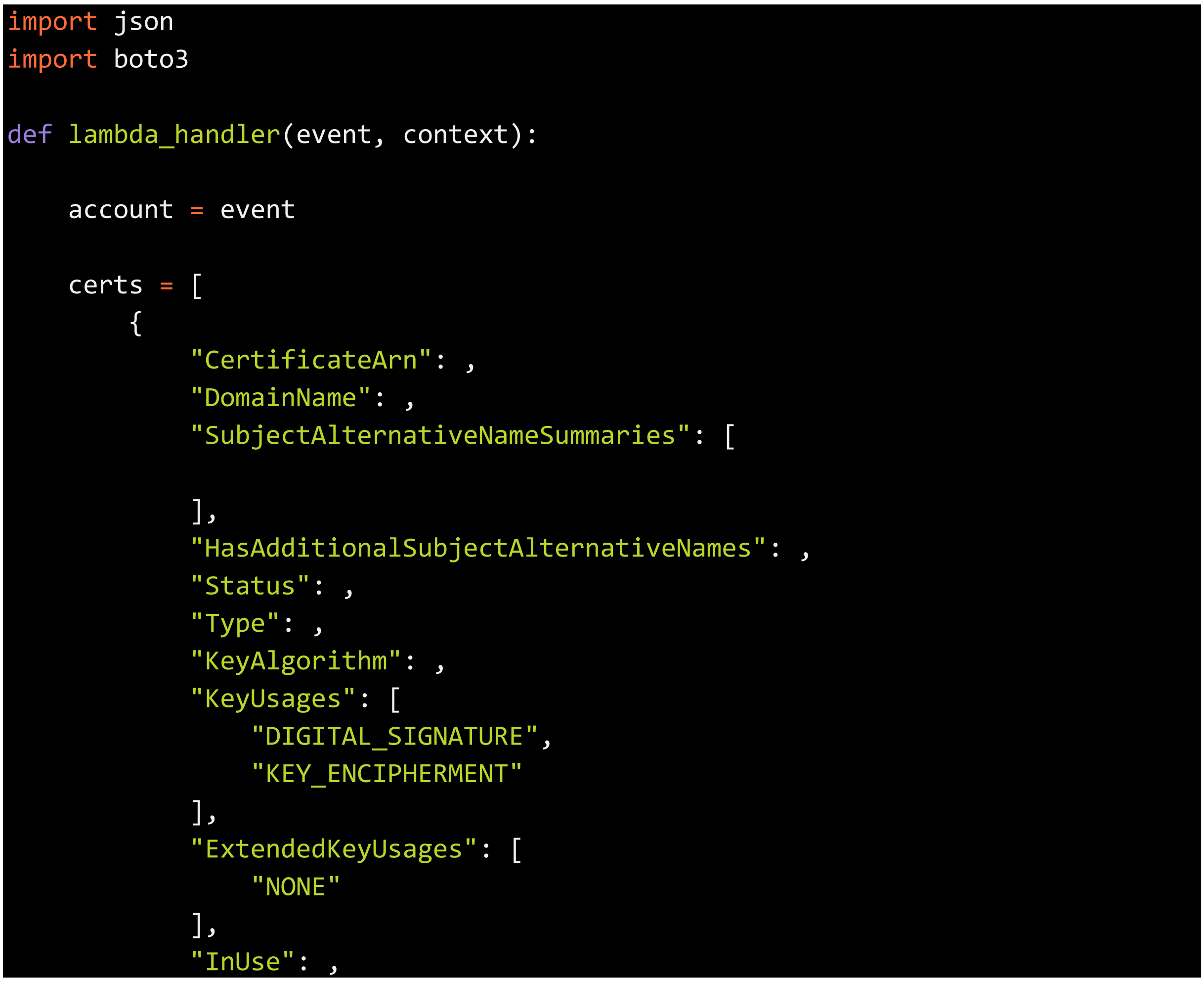
```
    return {
        'statusCode': 200,
        'body': certs
    }
```

Appendix 3 – mockCerts.py



```
import json

def lambda_handler(event, context):

    aws_account_id = context.invoked_function_arn.split(":")[4]

   # print(aws_account_id)
    return {
        'statusCode': 200,
        'body': aws_account_id
    }
```

Appendix 4 – readAccount.py



```
import json
import boto3

def lambda_handler(event, context):

    account = event

    certs = [
        {
            "CertificateArn": ,
            "DomainName": ,
            "SubjectAlternativeNameSummaries": [

            ],
            "HasAdditionalSubjectAlternativeNames": ,
            "Status": ,
            "Type": ,
            "KeyAlgorithm": ,
            "KeyUsages": [
                "DIGITAL_SIGNATURE",
                "KEY_ENCIPHERMENT"
            ],
            "ExtendedKeyUsages": [
                "NONE"
            ],
            "InUse": ,
            "RenewalEligibility": ,
            "NotBefore": ,
            "NotAfter": ,
            "CreatedAt": ,
            "ImportedAt":
        }
    ]

    return {
        'statusCode': 200,
        'body': certs
    }
```

Appendix 5 – readCerts.py

```
import json
import boto3
from datetime import date, datetime

def lambda_handler(event, context):
    result = get_stacks(event, context)
    return {
        'statusCode': 200,
        'body': result
        #'body': json.dumps(result, default=str)
    }

def get_stacks(event, context):
    cf_client = boto3.client('cloudformation')
    response = cf_client.list_stacks()
    stacks = response['StackSummaries']
    for stack in stacks:
        for resource in stack:
            stack[resource] = str(stack[resource])
    return stacks
```

Appendix 6 – listStacks.py

```
import json
import boto3

def lambda_handler(event, context):
    result = scan_stack(event, context)
    return {
        'statusCode': 200,
        'body': result
    }
```

```
def scan_stack(event, context):
    cf_client = boto3.client('cloudformation')
    response = cf_client.describe_stack_resources(
        StackName=event['StackId'],
    )
    stack = response['StackResources']
    for resource in stack:
        for item in resource:
            resource[item] = str(resource[item])
    return stack
```

Appendix 7 – findOrphanedResources.py

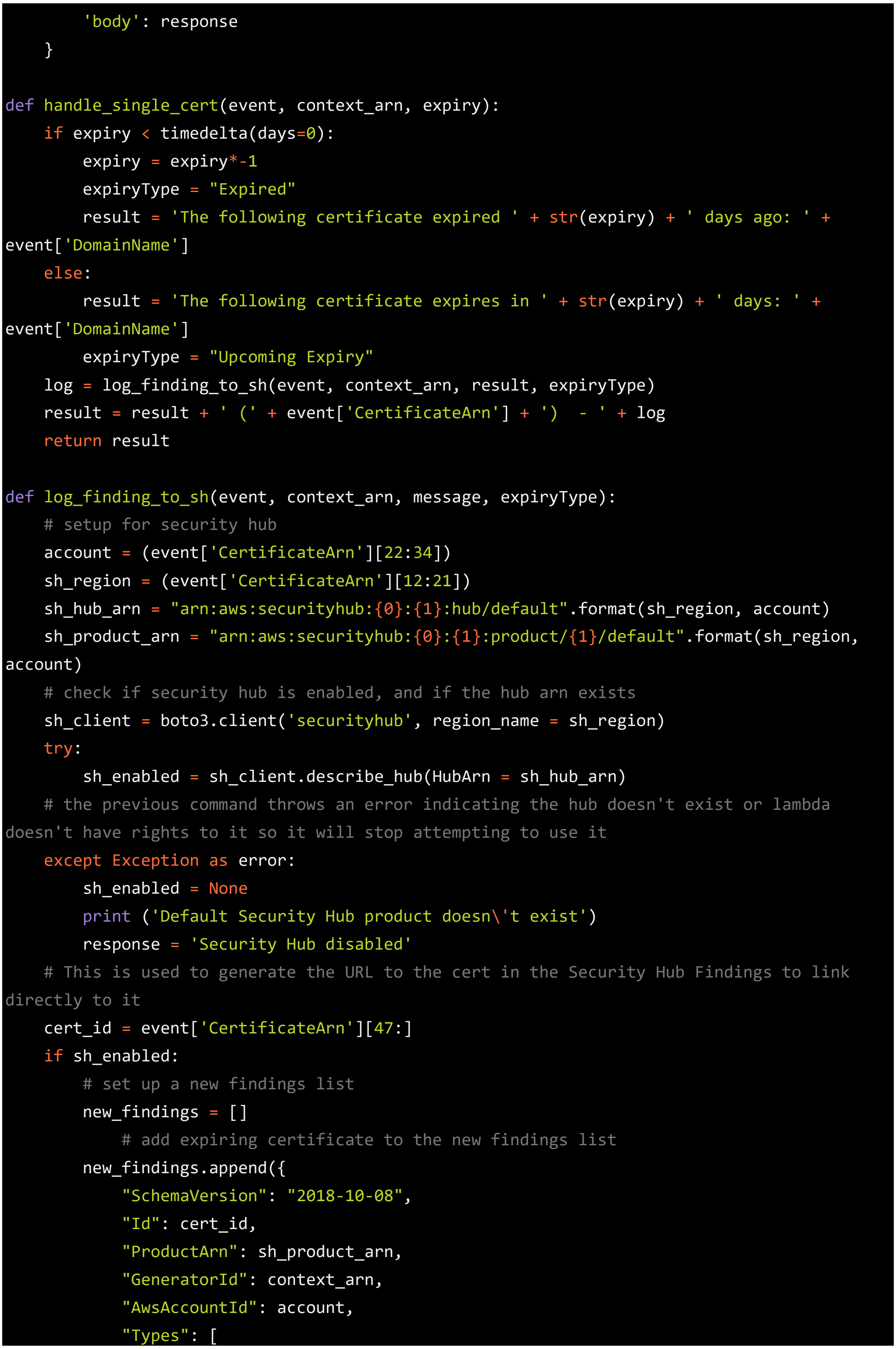
```
import json
import boto3
from datetime import datetime, timedelta, timezone

def lambda_handler(event,context):
    dynamodb = boto3.resource('dynamodb')
    cert = event

    #table name
    table = dynamodb.Table('CertAlert')
    #inserting values into table
    cert["Renewal"] = check_expiry_date(cert)
    response = table.put_item(
       Item=cert
    )
    return {
        'statusCode': 200,
        'body': response
    }

def check_expiry_date(cert):
    utc = timezone.utc
    # make today timezone aware
    today = datetime.now().replace(tzinfo=utc)
    time = datetime.now().isoformat(timespec='seconds')
    now = datetime.strptime(time[0:10], '%Y-%m-%d')
    expiry_days = timedelta(days=45)
    expiryDate = datetime.strptime(cert['ExpiresOn'][0:10], '%Y-%m-%d')
    expiry = expiryDate - now
    if expiry < timedelta(days=0):
        expiry = expiry*-1
        expiryType = "This certificate has expired, please renew if 
certificate is needed"
    elif expiry < expiry_days:
        expiryType = "Expiration date in the next 45 days, please renew"
    else:
        expiryType = "No renewal needed"
    return expiryType
```

Appendix 8 – writeToDynamoDB.py

```
import json
import boto3

def lambda_handler(event,context):
    dynamodb = boto3.resource('dynamodb')
    resources = event
    #table name
    table = dynamodb.Table('OrphanedResources')
    #inserting values into table
    response = table.put_item(
       Item=resources
    )
    return {
        'statusCode': 200,
        'body': response
    }
```

Appendix 9 – pushToDynamoDBOrphanedResources.py

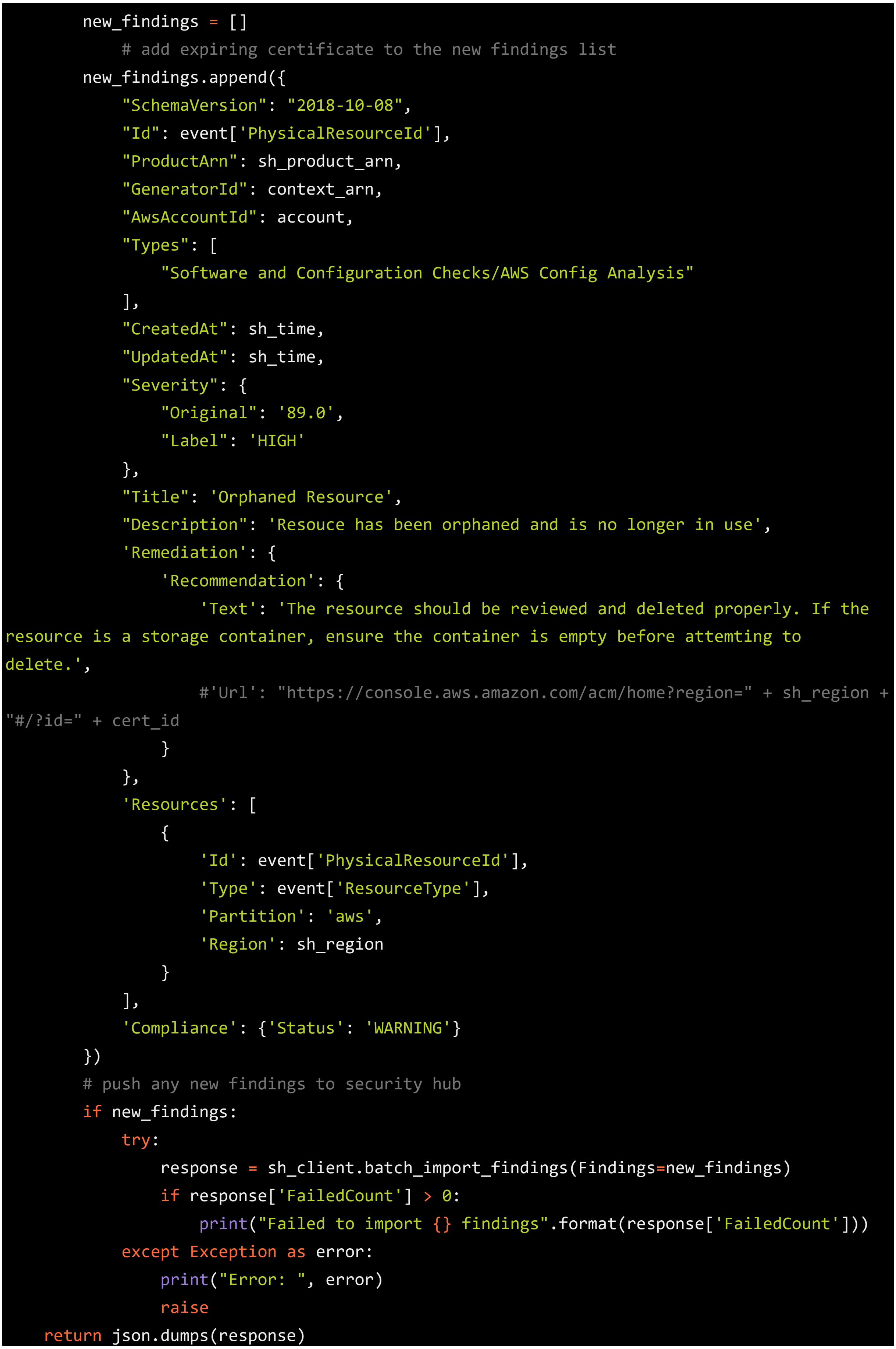
```
import json
import boto3
import os
from datetime import datetime, timedelta, timezone

utc = timezone.utc
# make today timezone aware
today = datetime.now().replace(tzinfo=utc)
time = datetime.now().isoformat(timespec='seconds')
now = datetime.strptime(time[0:10], '%Y-%m-%d')
sh_time = today.strftime("%Y-%m-%dT%H:%M:%S.000Z")
expiry_days = timedelta(days=45)

def lambda_handler(event, context):
    # check the expiry window before logging to Security Hub
    expiryDate = datetime.strptime(event['ExpiresOn'][0:10], '%Y-%m-%d')
    expiry = expiryDate - now
    if expiry < expiry_days:
        response = handle_single_cert(event, context.invoked_function_arn, expiry)
    else:
        response = "The current certificate does not have an upcoming expiration date"
    return {
        'statusCode': 200,
```



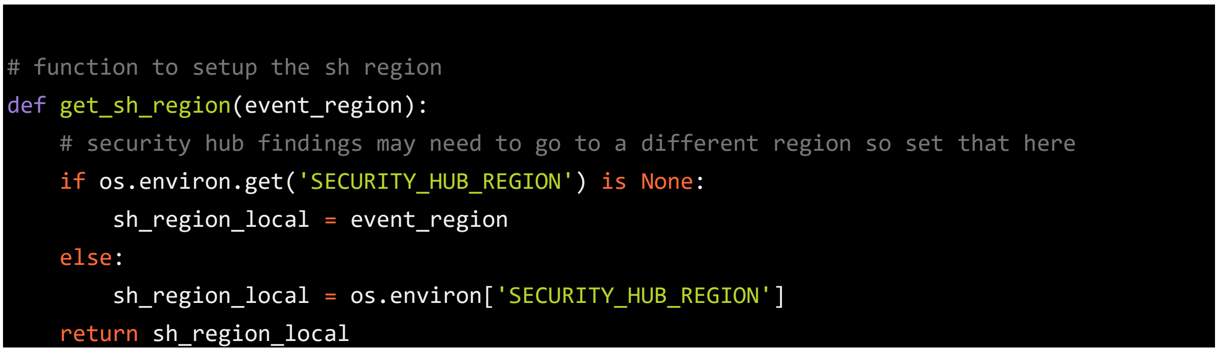
```
        'body': response
    }

def handle_single_cert(event, context_arn, expiry):
    if expiry < timedelta(days=0):
        expiry = expiry*-1
        expiryType = "Expired"
        result = 'The following certificate expired ' + str(expiry) + ' days ago: ' +
event['DomainName']
    else:
        result = 'The following certificate expires in ' + str(expiry) + ' days: ' +
event['DomainName']
        expiryType = "Upcoming Expiry"
    log = log_finding_to_sh(event, context_arn, result, expiryType)
    result = result + ' (' + event['CertificateArn'] + ')  - ' + log
    return result

def log_finding_to_sh(event, context_arn, message, expiryType):
    # setup for security hub
    account = (event['CertificateArn'][22:34])
    sh_region = (event['CertificateArn'][12:21])
    sh_hub_arn = "arn:aws:securityhub:{0}:{1}:hub/default".format(sh_region, account)
    sh_product_arn = "arn:aws:securityhub:{0}:{1}:product/{1}/default".format(sh_region,
account)
    # check if security hub is enabled, and if the hub arn exists
    sh_client = boto3.client('securityhub', region_name = sh_region)
    try:
        sh_enabled = sh_client.describe_hub(HubArn = sh_hub_arn)
    # the previous command throws an error indicating the hub doesn't exist or lambda
doesn't have rights to it so it will stop attempting to use it
    except Exception as error:
        sh_enabled = None
        print ('Default Security Hub product doesn\'t exist')
        response = 'Security Hub disabled'
    # This is used to generate the URL to the cert in the Security Hub Findings to link
directly to it
    cert_id = event['CertificateArn'][47:]
    if sh_enabled:
        # set up a new findings list
        new_findings = []
            # add expiring certificate to the new findings list
        new_findings.append({
            "SchemaVersion": "2018-10-08",
            "Id": cert_id,
            "ProductArn": sh_product_arn,
            "GeneratorId": context_arn,
            "AwsAccountId": account,
            "Types": [
                "Software and Configuration Checks/AWS Config Analysis"
            ],
            "CreatedAt": sh_time,
            "UpdatedAt": sh_time,
            "Severity": {
                "Original": '89.0',
                "Label": 'HIGH'
            },
            "Title": 'Certificate expiration',
            "Description": expiryType,
            'Remediation': {
                'Recommendation': {
                    'Text': message + '. A new certificate for ' + event['DomainName'] + '
should be imported to replace the existing imported certificate before expiration',
                    'Url': "https://console.aws.amazon.com/acm/home?region=" + sh_region +
"#/?id=" + cert_id
                }
            },
            'Resources': [
                {
                    'Id': cert_id,
                    'Type': 'ACM Certificate',
                    'Partition': 'aws',
                    'Region': sh_region
                }
            ],
            'Compliance': {'Status': 'WARNING'}
        })
        # push any new findings to security hub
        if new_findings:
            try:
                response = sh_client.batch_import_findings(Findings=new_findings)
                if response['FailedCount'] > 0:
                    print("Failed to import {} findings".format(response['FailedCount']))
            except Exception as error:
                print("Error: ", error)
                raise
    return json.dumps(response)

# function to setup the sh region
def get_sh_region(event_region):
    # security hub findings may need to go to a different region so set that here
    if os.environ.get('SECURITY_HUB_REGION') is None:
        sh_region_local = event_region
    else:
        sh_region_local = os.environ['SECURITY_HUB_REGION']
    return sh_region_local
```

Appendix 10 – writeToSecurityHub.py

```
import json
import boto3
import os
from datetime import datetime, timedelta, timezone

utc = timezone.utc
# make today timezone aware
today = datetime.now().replace(tzinfo=utc)
time = datetime.now().isoformat(timespec='seconds')
now = datetime.strptime(time[0:10], '%Y-%m-%d')
sh_time = today.strftime("%Y-%m-%dT%H:%M:%S.000Z")
expiry_days = timedelta(days=45)

def lambda_handler(event, context):
    response = check_for_orphans(event, context)
    return {
        'statusCode': 200,
        'body': response
    }

def check_for_orphans(event, context):
    if event['ResourceStatus'] == "DELETE_SKIPPED":
        response = log_finding_to_sh(event, context.invoked_function_arn)
    else:
        response = "Resource successfully deleted"
    return response

def log_finding_to_sh(event, context_arn):
    # setup for security hub
    account = (event['StackId'][33:45])
    sh_region = (event['StackId'][23:32])
    sh_hub_arn = "arn:aws:securityhub:{0}:{1}:hub/default".format(sh_region, account)
    sh_product_arn = "arn:aws:securityhub:{0}:{1}:product/{1}/default".format(sh_region,
account)
    # check if security hub is enabled, and if the hub arn exists
    sh_client = boto3.client('securityhub', region_name = sh_region)
    try:
        sh_enabled = sh_client.describe_hub(HubArn = sh_hub_arn)
    # the previous command throws an error indicating the hub doesn't exist or lambda
doesn't have rights to it so it will stop attempting to use it
    except Exception as error:
        sh_enabled = None
        print ('Default Security Hub product doesn\'t exist')
        response = 'Security Hub disabled'

    if sh_enabled:
        # set up a new findings list
```

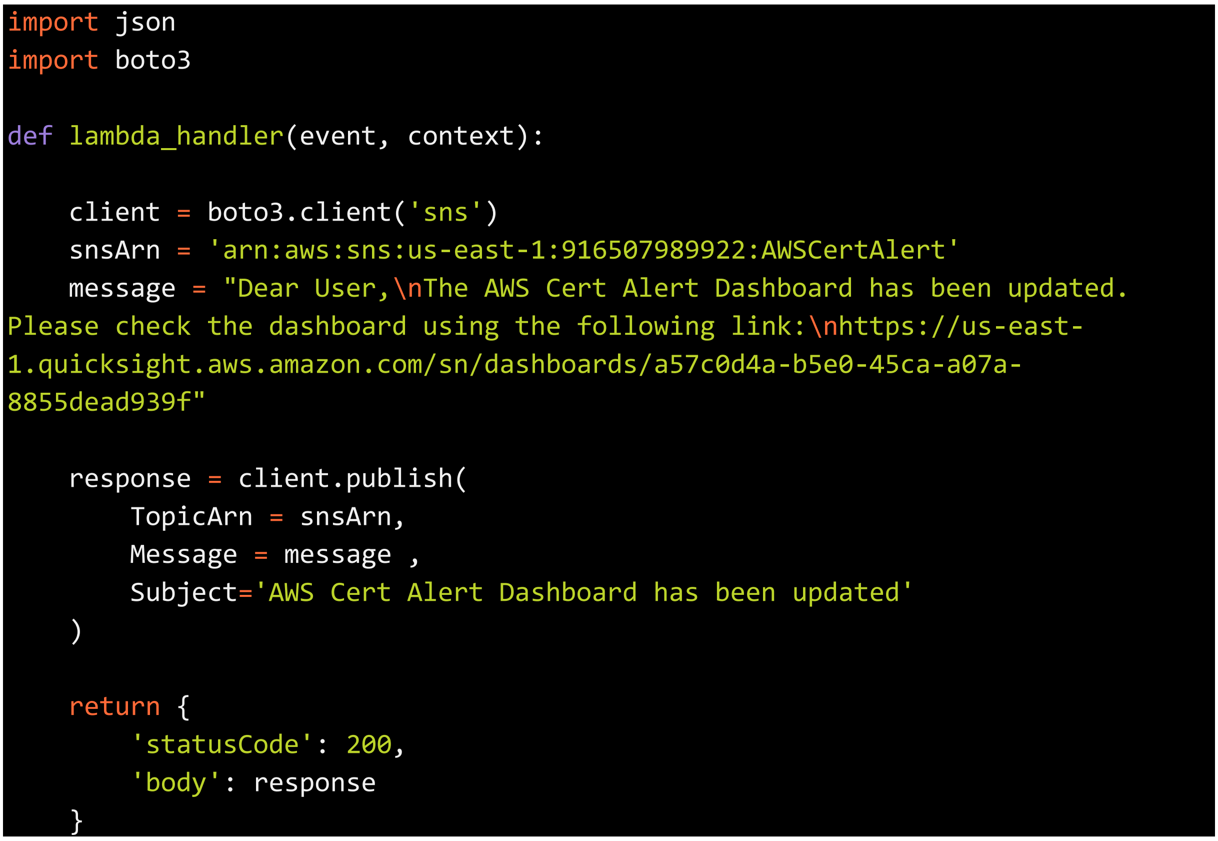


```
        new_findings = []
            # add expiring certificate to the new findings list
        new_findings.append({
            "SchemaVersion": "2018-10-08",
            "Id": event['PhysicalResourceId'],
            "ProductArn": sh_product_arn,
            "GeneratorId": context_arn,
            "AwsAccountId": account,
            "Types": [
                "Software and Configuration Checks/AWS Config Analysis"
            ],
            "CreatedAt": sh_time,
            "UpdatedAt": sh_time,
            "Severity": {
                "Original": '89.0',
                "Label": 'HIGH'
            },
            "Title": 'Orphaned Resource',
            "Description": 'Resouce has been orphaned and is no longer in use',
            'Remediation': {
                'Recommendation': {
                    'Text': 'The resource should be reviewed and deleted properly. If the
resource is a storage container, ensure the container is empty before attemting to
delete.',
                    #'Url': "https://console.aws.amazon.com/acm/home?region=" + sh_region +
"#/?id=" + cert_id
                }
            },
            'Resources': [
                {
                    'Id': event['PhysicalResourceId'],
                    'Type': event['ResourceType'],
                    'Partition': 'aws',
                    'Region': sh_region
                }
            ],
            'Compliance': {'Status': 'WARNING'}
        })
        # push any new findings to security hub
        if new_findings:
            try:
                response = sh_client.batch_import_findings(Findings=new_findings)
                if response['FailedCount'] > 0:
                    print("Failed to import {} findings".format(response['FailedCount']))
            except Exception as error:
                print("Error: ", error)
                raise
    return json.dumps(response)
```



```python
# function to setup the sh region
def get_sh_region(event_region):
    # security hub findings may need to go to a different region so set that here
    if os.environ.get('SECURITY_HUB_REGION') is None:
        sh_region_local = event_region
    else:
        sh_region_local = os.environ['SECURITY_HUB_REGION']
    return sh_region_local
```

Appendix 11 – writeOrphanedResourcesToSecurityHub.py



```python
import json
import boto3

def lambda_handler(event, context):

    client = boto3.client('sns')
    snsArn = 'arn:aws:sns:us-east-1:916507989922:AWSCertAlert'
    message = "Dear User,\nThe AWS Cert Alert Dashboard has been updated. Please check the dashboard using the following link:\nhttps://us-east-1.quicksight.aws.amazon.com/sn/dashboards/a57c0d4a-b5e0-45ca-a07a-8855dead939f"

    response = client.publish(
        TopicArn = snsArn,
        Message = message ,
        Subject='AWS Cert Alert Dashboard has been updated'
    )

    return {
        'statusCode': 200,
        'body': response
    }
```

Appendix 12 – sendAlert.py