CLOUD SECURITY

Building a **Security Monitoring System** in **AWS**

Jeffrey Obi
February 2026

Contents

Introduction

Background

Security within the cloud environment is critical. **Real-time security monitoring systems** are a formidable strategy often implemented by **Cloud Security Professionals** to secure cloud resources. When properly executed, this system is essential for detecting insider threats, credential misuse, and early signs of cloud compromise.

Project Goals

Inside the **AWS** lab environment, design and execute on a **Security Monitoring System** which monitors and detects access to sensitive AWS resources, then issues realtime notifications to security professionals in the case of a breach or unauthorized access, and responds defensively.

Two different types of security monitoring paths will be configured and compared to evaluate which of the two provides better insight and quicker response.

Lab Simulation Stack

Lab environment: **Cloud (AWS)**

Services (cloud tools):

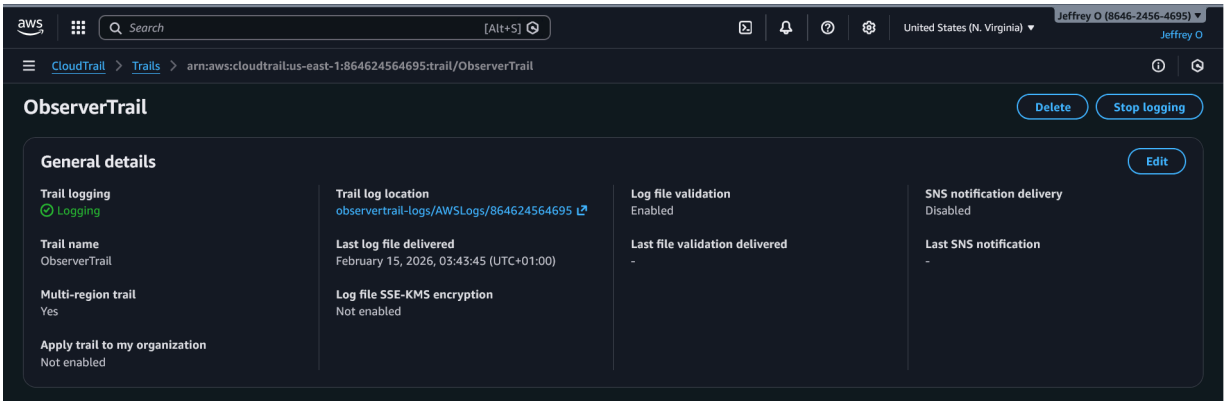
1. **AWS CloudTrail**
2. **Amazon CloudWatch**
3. **Amazon SNS (Simple Notification Service)**
4. **AWS Secrets Manager**
5. **Amazon S3**
6. **AWS CLI**
7. **AWS IAM**

Phase 1

Infrastructure & Logging

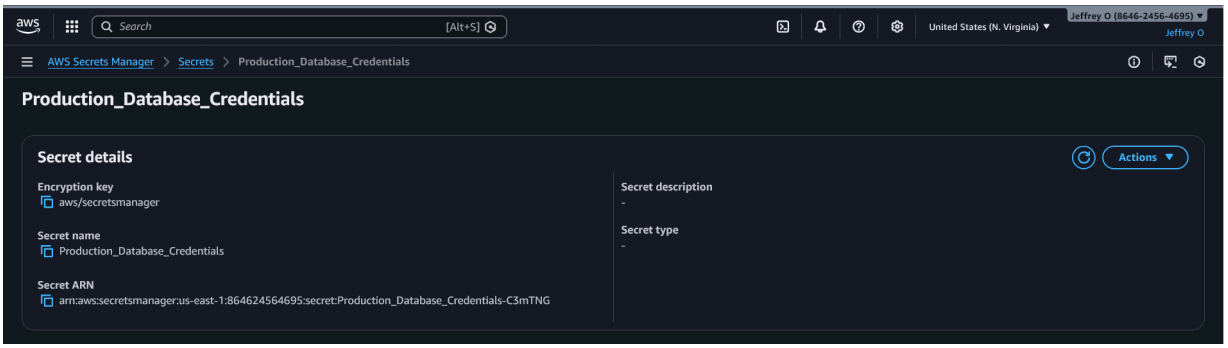
Task 1.1. The Audit Trail

A multi-region **AWS CloudTrail** trail, **ObserverTrail,** integrated with **Amazon CloudWatch** Logs **(cloudtrail-observer-log-group)**.



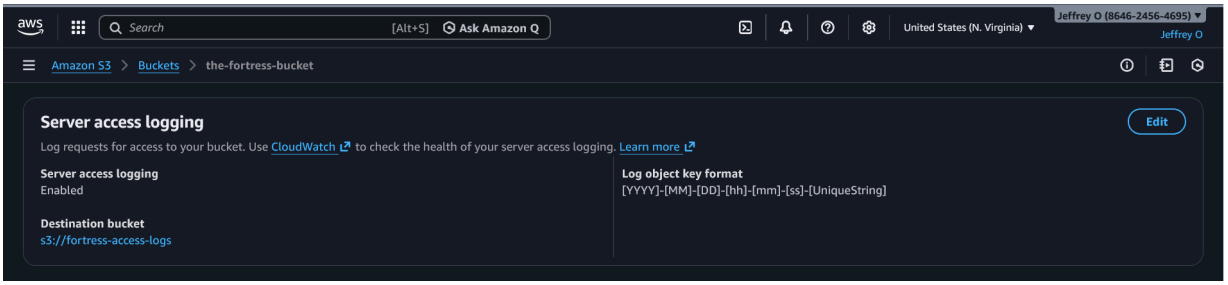
Task 1.2. The Honeytoken

Production_Database_Credentials secret created in AWS Secrets Manager



Task 1.3. Data Residency

US (N. Virginia) region *(us-east-1)* **S3 Bucket** with **Server Access Logging** Enabled. The sensitive data bucket, **the-fortress-bucket,** stores its server access logs in the **fortress-access-logs** bucket.

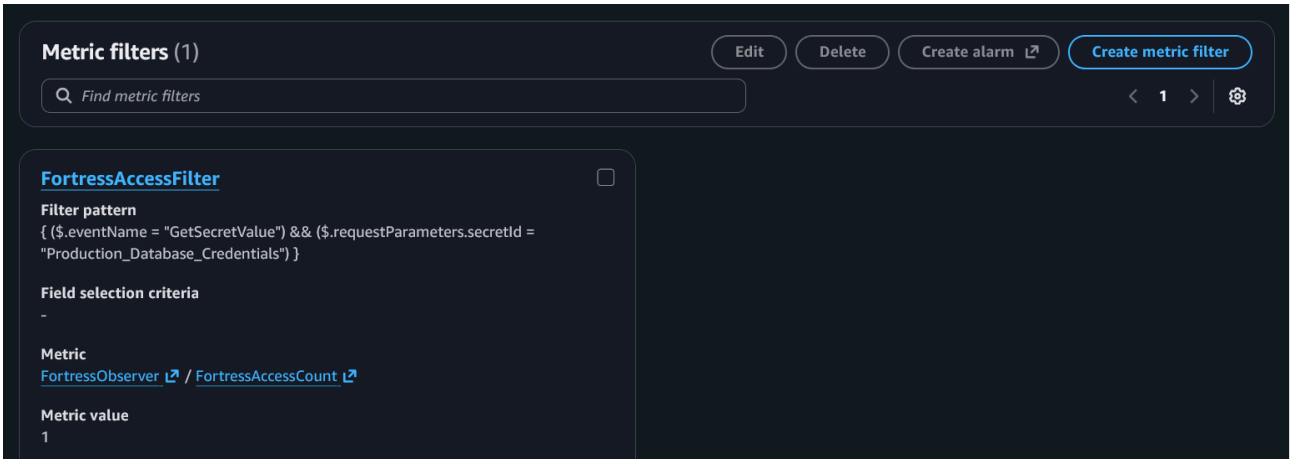


Phase 2

Detection Logic – Flow 1

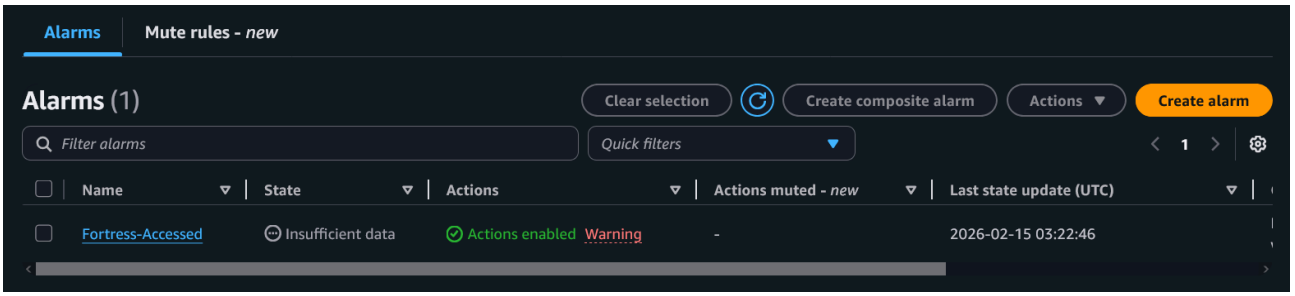
Task 2.1. Metric Filters

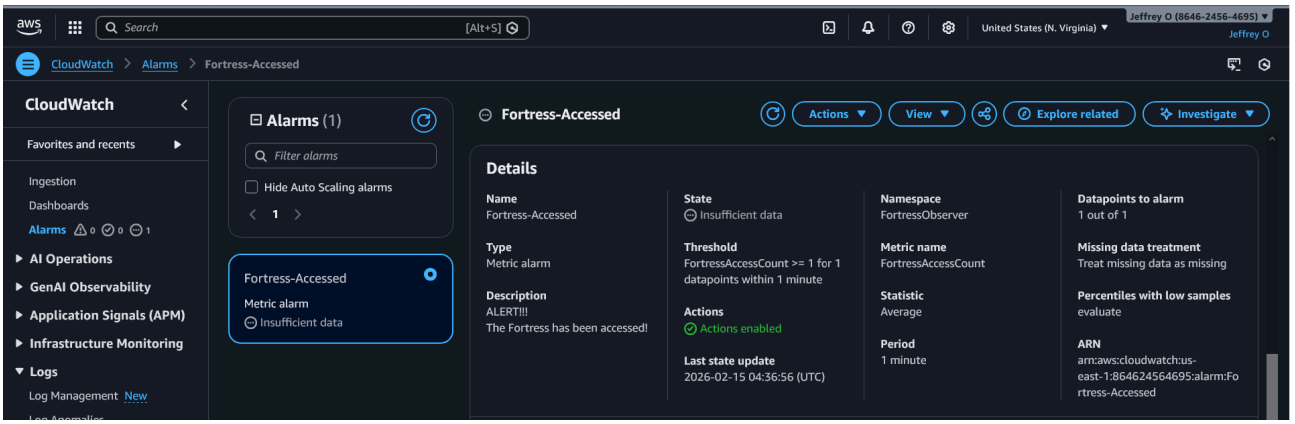
A Metric Filter, **FortressAccessFilter,** in **CloudWatch** Logs that searches for any **GetSecretValue API** call which specifically targets the **Production_Database_Credentials** secret



Task 2.2. The Threshold

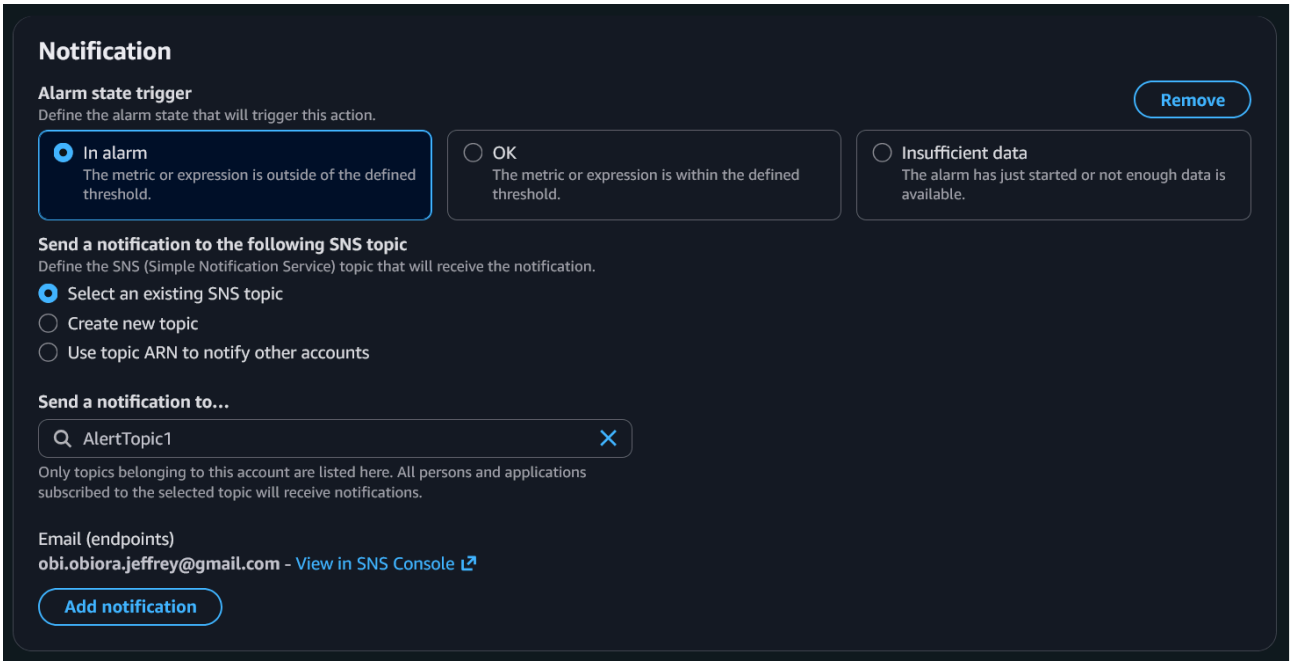
A **CloudWatch Alarm, FortressAccessed,** based on the metric filter, **FortressAccessFilter.** The alarm is set to trigger if the secret is accessed even once, with the **Threshold FortressAccessCount** set to set to ≥1 within 1 minute.

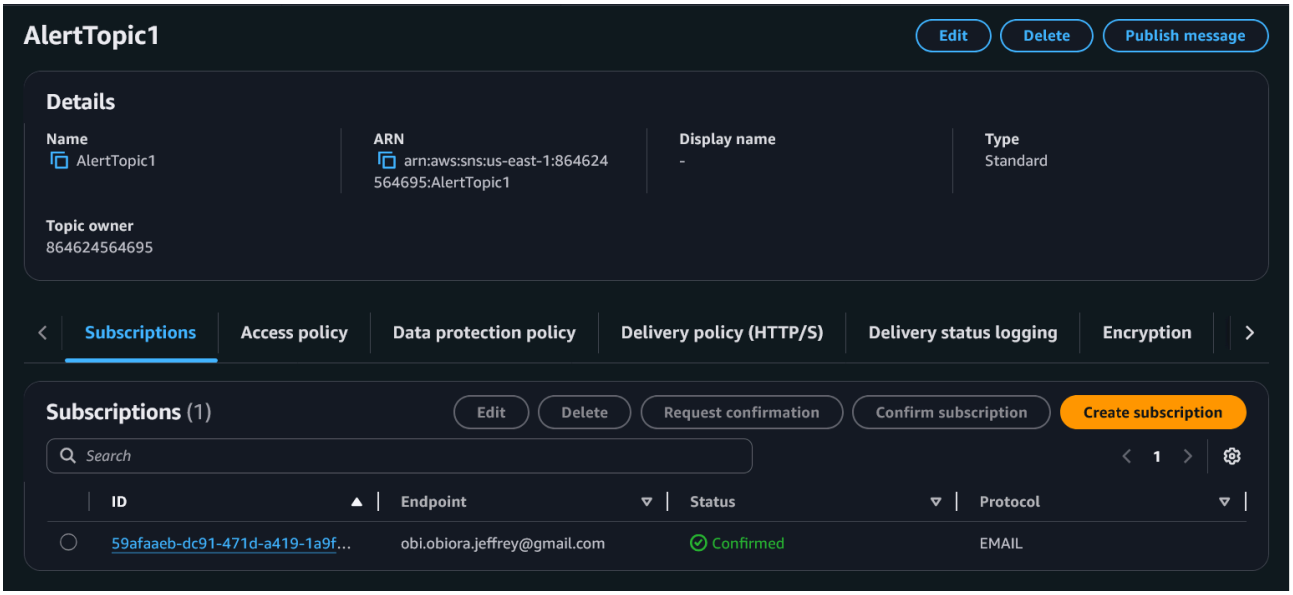


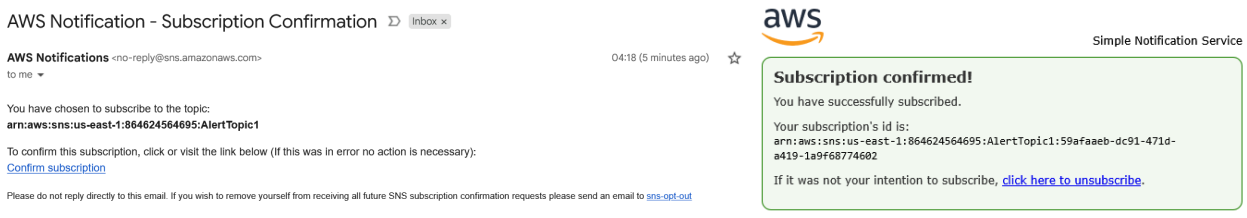


Task 2.3. Notification

Linking the **CloudWatch Alarm, FortressAccessed,** to the SNS Topic **AlertTopic1** and adding a subscription to my student email.





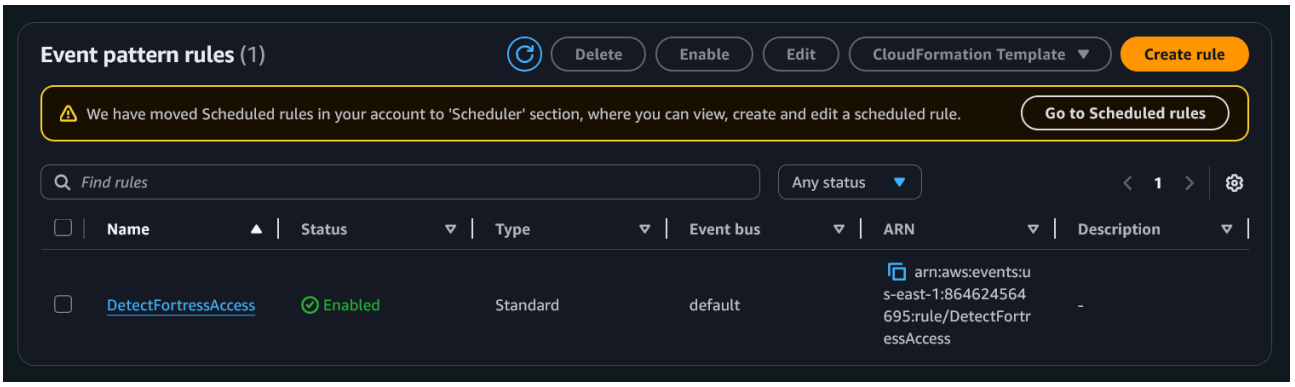


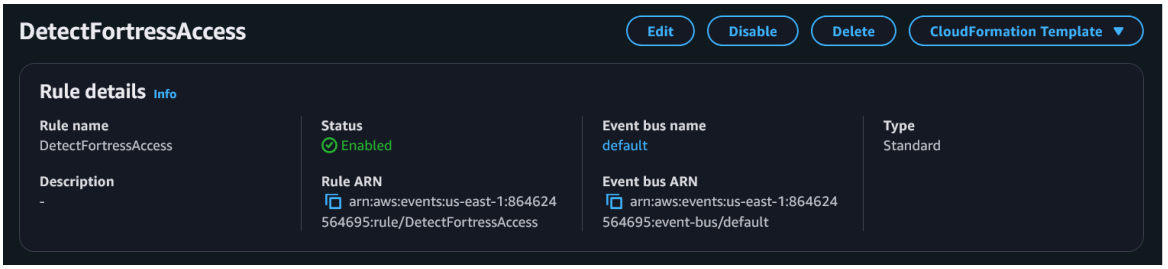
Phase 3

Detection Logic – Flow 2

Task 3.1. Event-Driven Security

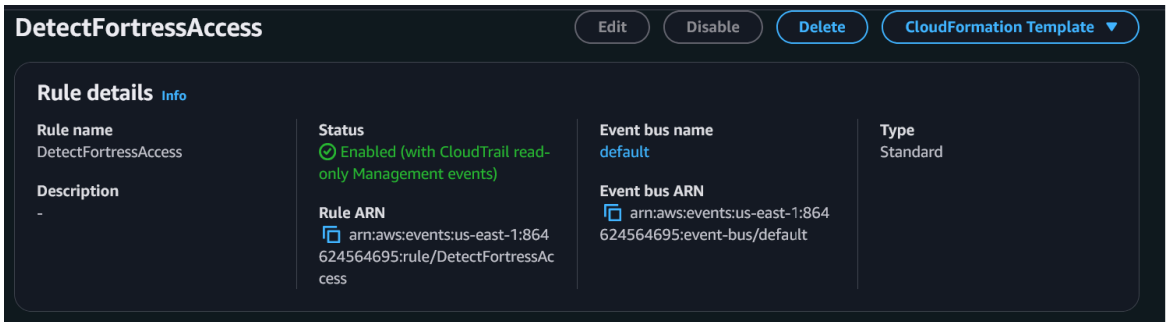
An **Amazon EventBridge** rule, **DetectFortressAccess** that triggers whenever a “Sensitive Data Access” event occurs in **AWS CloudTrail.**

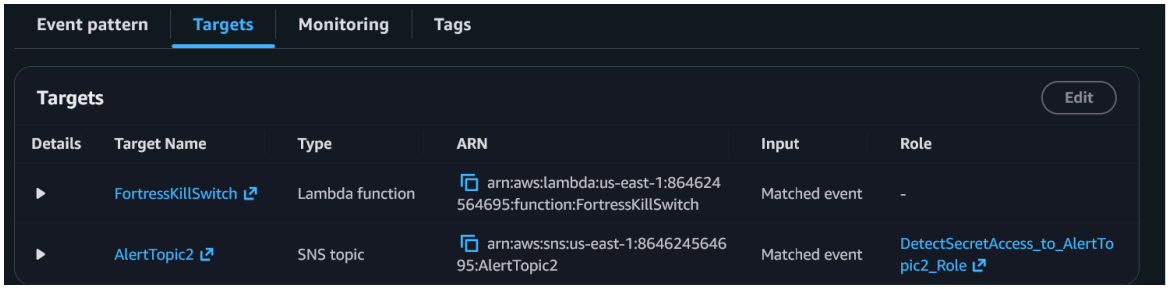


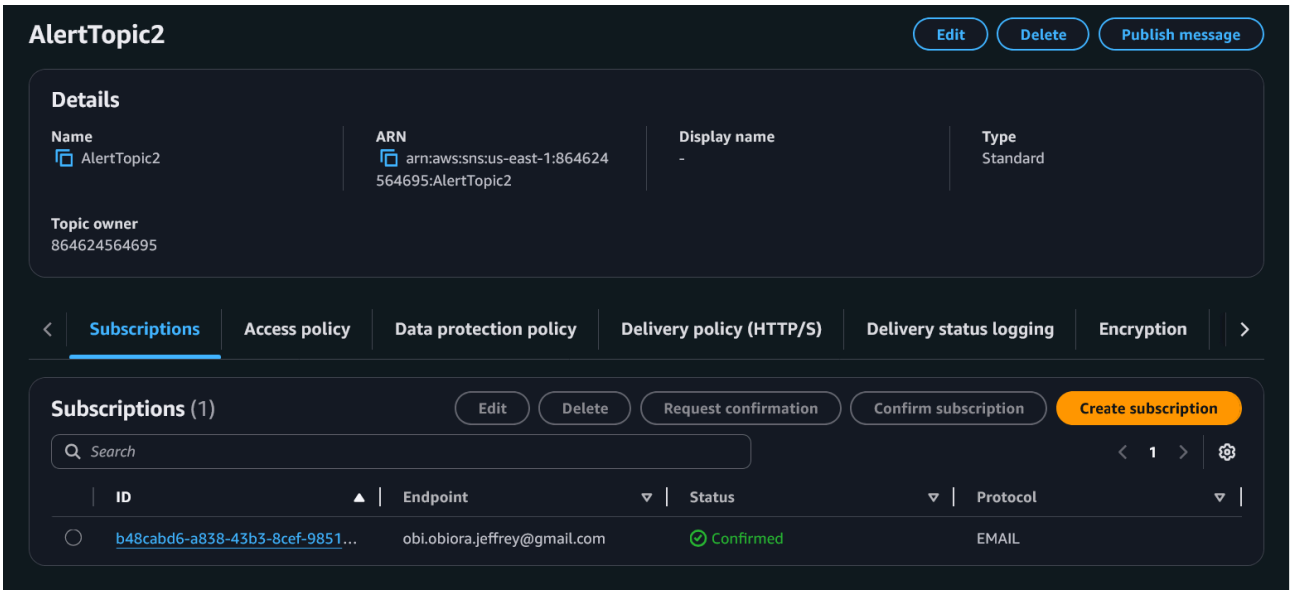


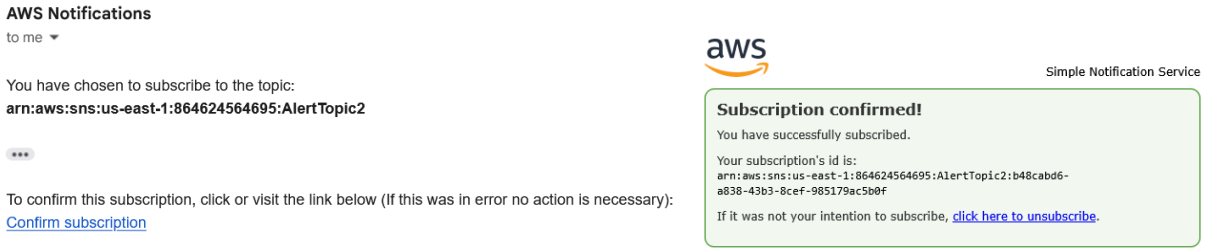
Task 3.2. Comparison

Configuring the **Amazon EventBridge** rule, **DetectFortressAccess,** to send a notification to a separate SNS topic **(AlertTopic2)**.



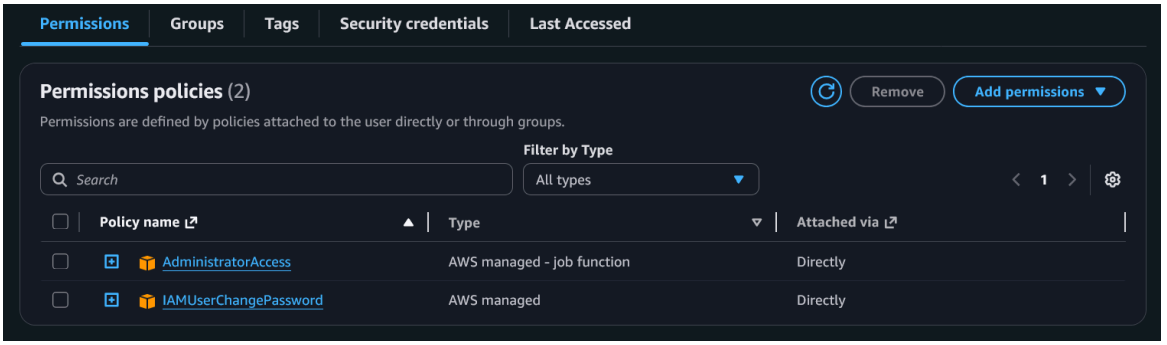


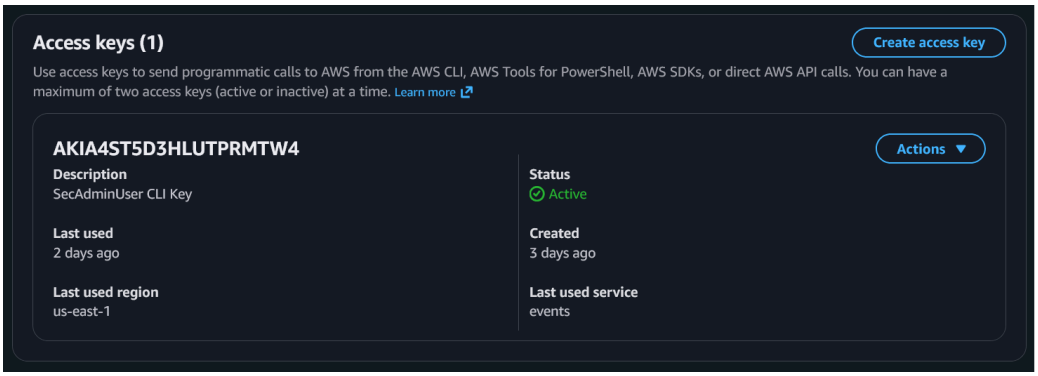


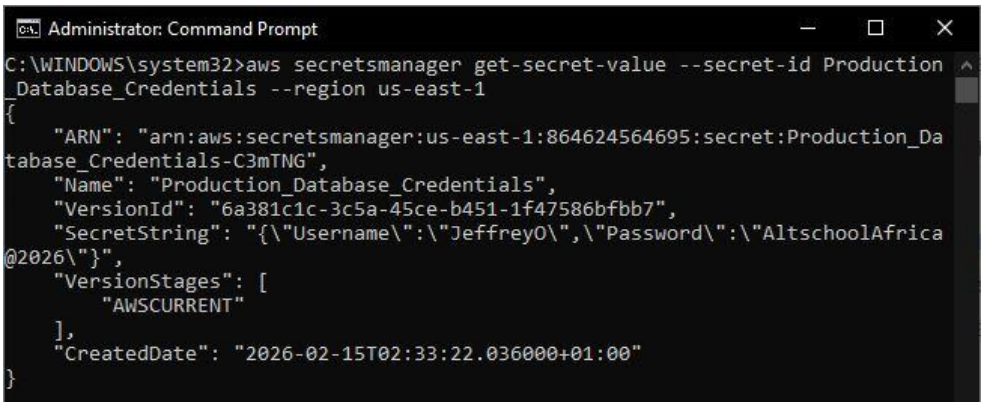


Task 3.3. The Breach

Using **AWS CLI** to retrieve the secret value.
Creating the admin user (**SecAdminUser**) using **AWS IAM** and configuring **AWS CLI** with its credentials.





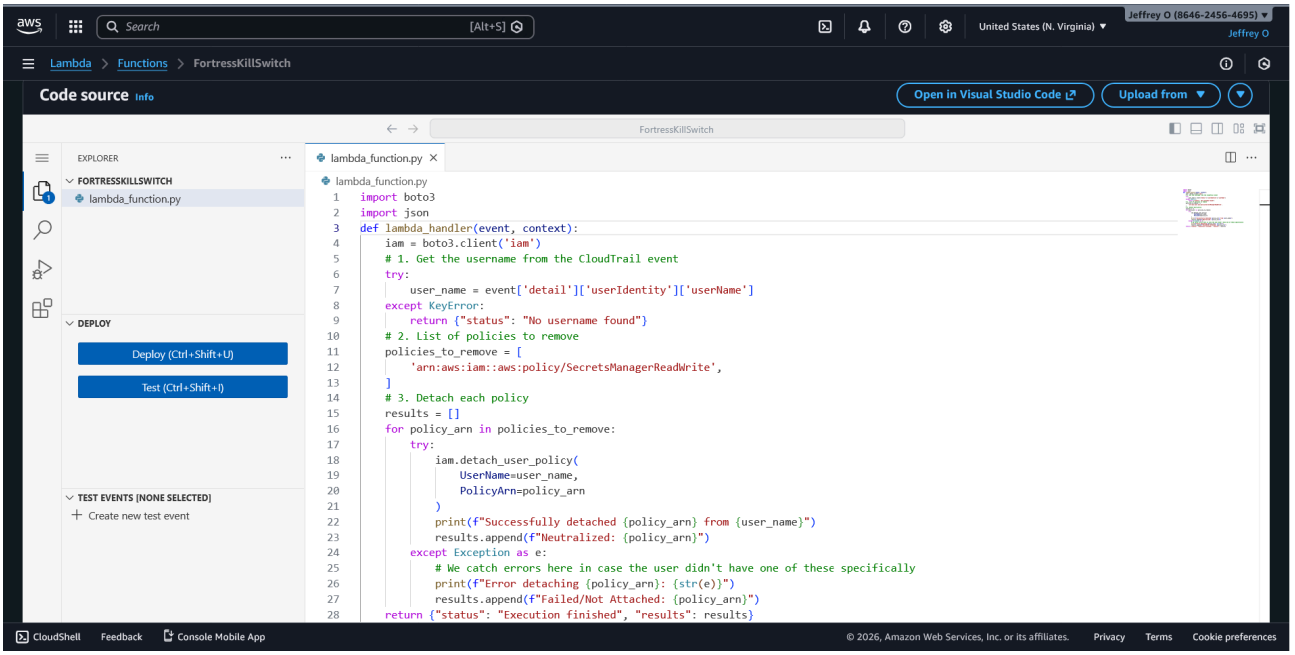


Phase 4

The “Active Defense” Kill-Switch

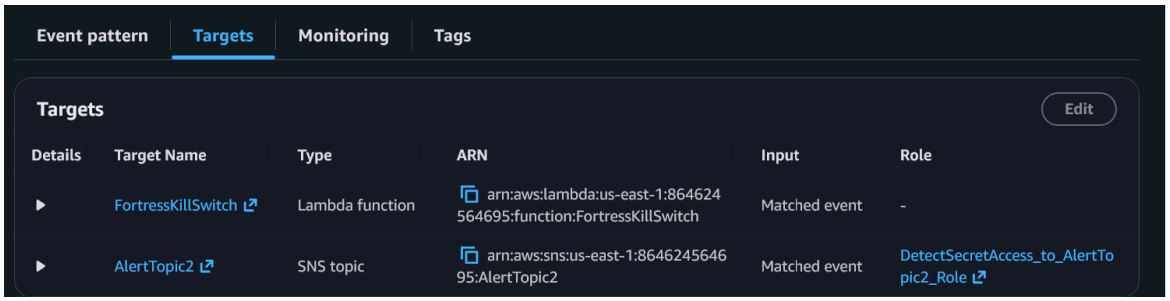
Task 4.1. The Target

Creating a simple **AWS Lambda** function (Python) which uses the **IAM DetachUserPolicy** to strip the **SecretsManagerReadWrite** permissions from the victim user, **TrapVictim**.



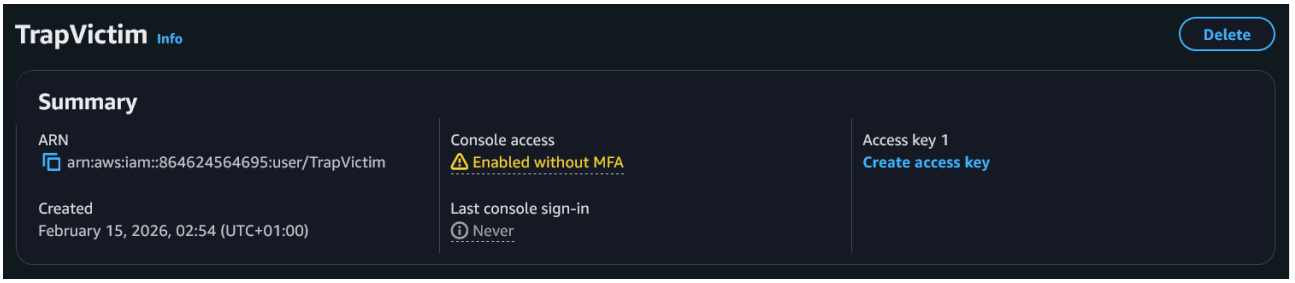
Task 4.2. The Trigger

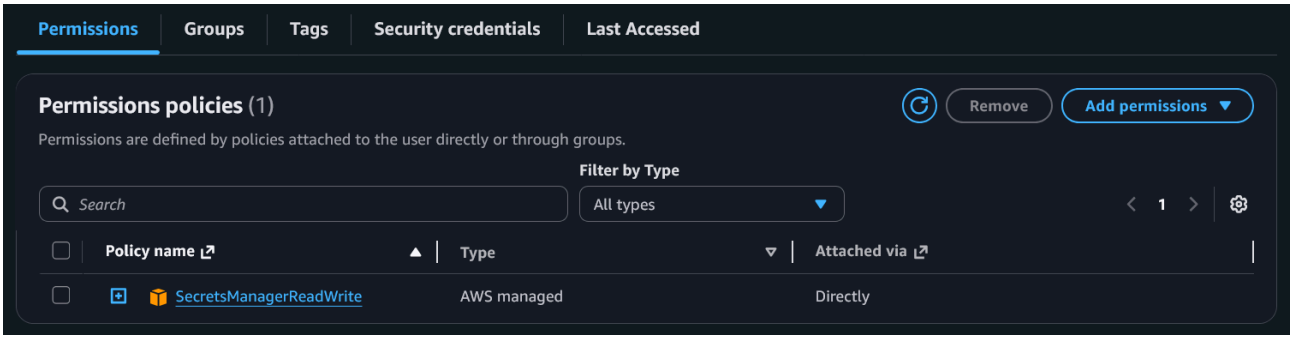
Connecting the **Amazon EventBridge** rule, **DetectFortressAccess,** to the new Lambda function as a second Target.

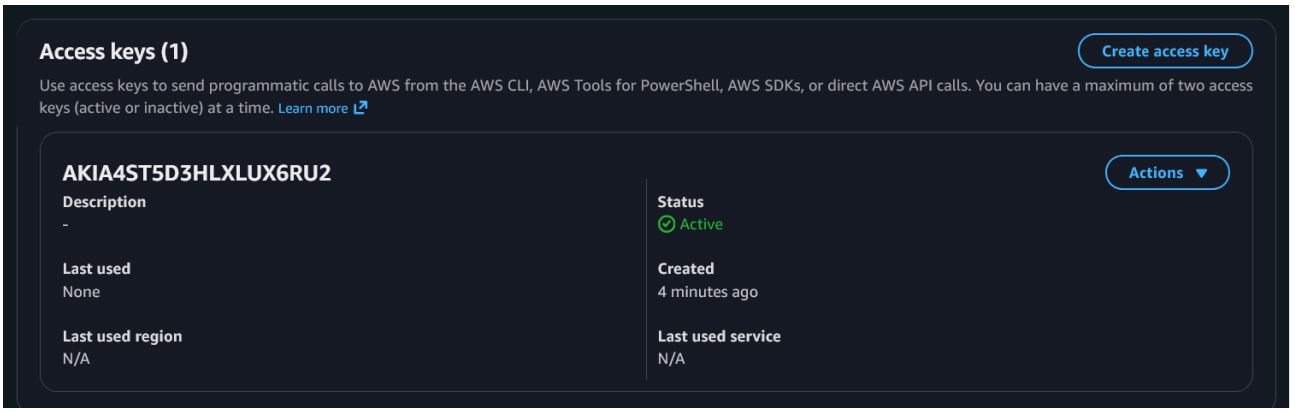


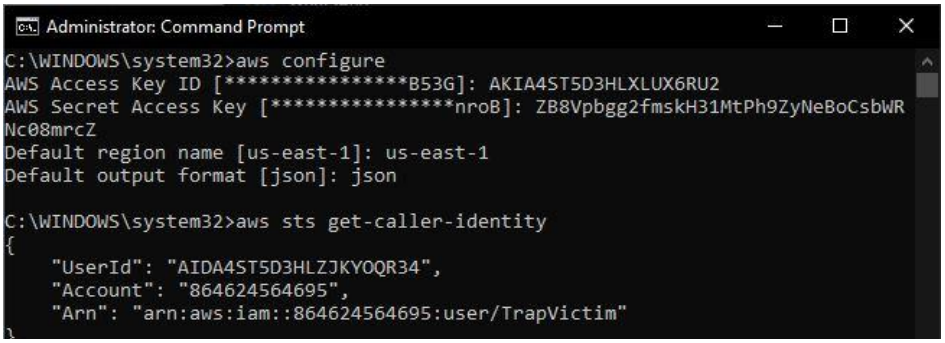
Task 4.3. The "Trap" Test

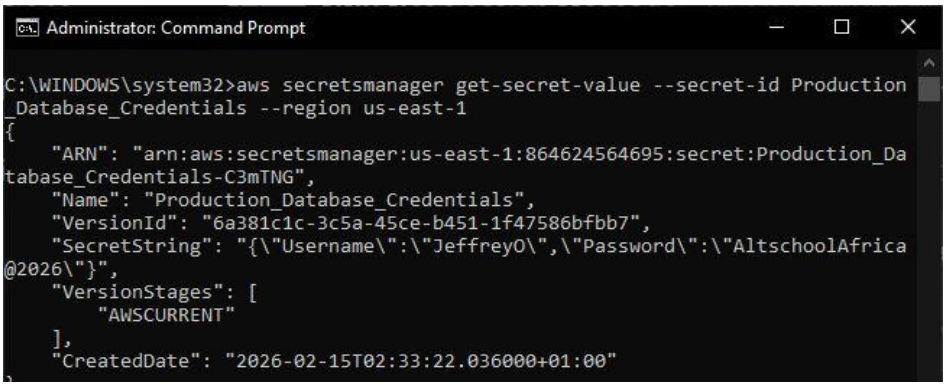
- Creating a "Victim" **IAM** user **(TrapVictim)** with **SecretsManagerReadWrite** permissions.
- Using the "Victim user," **TrapVictim,** to configure and access the secret via **AWS CLI** commands.
- Running another breach command immediately after and realizing that access has been automatically revoked.

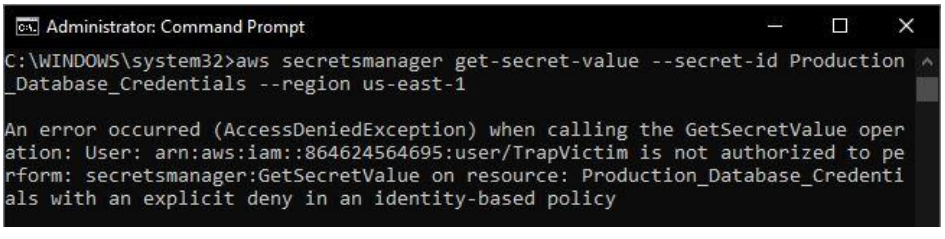



Permissions
Groups
Tags
Security credentials
Last Accessed
Permissions policies (1)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Filter by Type
Search
All types
1
Policy name
Type
Attached via
SecretsManagerReadWrite
AWS managed
Directly


Access keys (1)
Create access key
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more
AKIA4ST5D3HLXLUX6RU2
Actions
Description
-
Status
Active
Last used
None
Created
4 minutes ago
Last used region
N/A
Last used service
N/A


Administrator: Command Prompt
C:\WINDOWS\system32>aws configure
AWS Access Key ID [**********B53G]: AKIA4ST5D3HLXLUX6RU2
AWS Secret Access Key [**********nroB]: ZB8Vpbgg2fmskH31MtPh9ZyNeBoCsbWR
Nc08mrcZ
Default region name [us-east-1]: us-east-1
Default output format [json]: json
C:\WINDOWS\system32>aws sts get-caller-identity
{
"UserId": "AIDA4ST5D3HLZJKYOQR34",
"Account": "864624564695",
"Arn": "arn:aws:iam::864624564695:user/TrapVictim"


Administrator: Command Prompt
C:\WINDOWS\system32>aws secretsmanager get-secret-value --secret-id Production
_Database_Credentials --region us-east-1
{
"ARN": "arn:aws:secretsmanager:us-east-1:864624564695:secret:Production_Da
tabase_Credentials-C3mTNG",
"Name": "Production_Database_Credentials",
"VersionId": "6a381c1c-3c5a-45ce-b451-1f47586bfbb7",
"SecretString": "{\"Username\":\"JeffreyO\",\"Password\":\"AltschoolAfrica
@2026\"}",
"VersionStages": [
"AWSCURRENT"
],
"CreatedDate": "2026-02-15T02:33:22.036000+01:00"


Administrator: Command Prompt
C:\WINDOWS\system32>aws secretsmanager get-secret-value --secret-id Production
_Database_Credentials --region us-east-1
An error occurred (AccessDeniedException) when calling the GetSecretValue oper
ation: User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to pe
rform: secretsmanager:GetSecretValue on resource: Production_Database_Credenti
als with an explicit deny in an identity-based policy

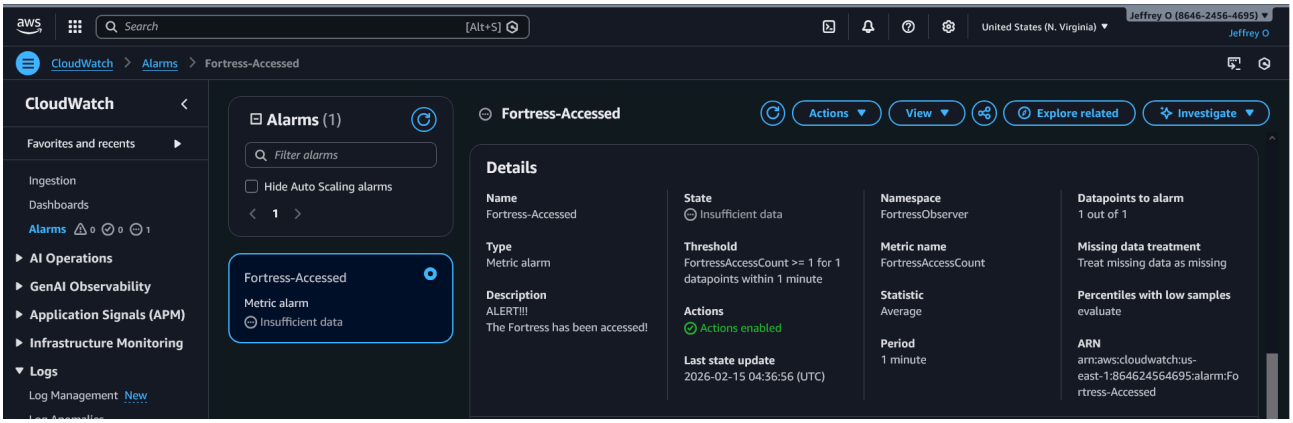
Phase 5
Forensic Report

Executive Summary

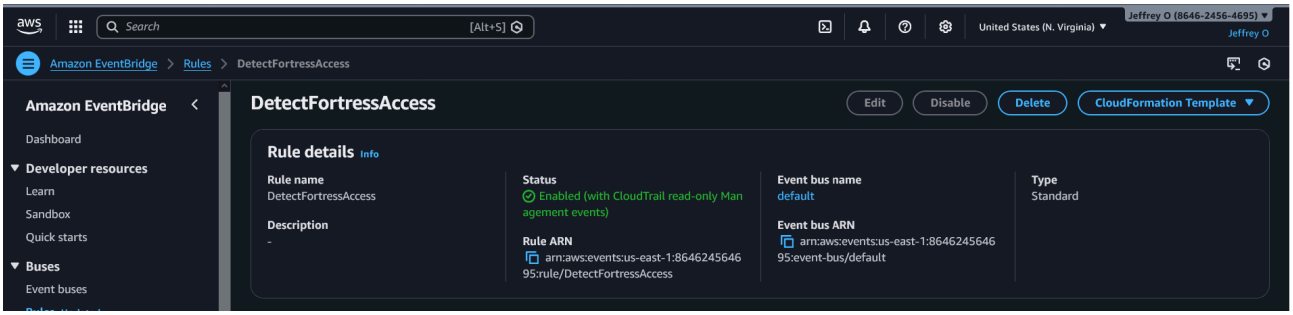
The goal of the security system was to explore **Continuous Controls Monitoring,** which is an automated, technology-driven approach to providing real-time security. The project mandate required the implementation of AWS tools and techniques to design and implement a **security monitoring and alerting pipeline** which detects access to sensitive resources and notifies the security team in real time.
The simulation used key services from within the AWS environment which were **AWS CloudTrail**, **Amazon CloudWatch**, **Amazon Simple Notification Service**, **AWS Secrets Manager**, **Amazon S3** and **AWS CLI**, and the simulation was a **success**.

Lab Architecture

CloudWatch Alarm Configuration Details



EventBridge Rule Configuration Details



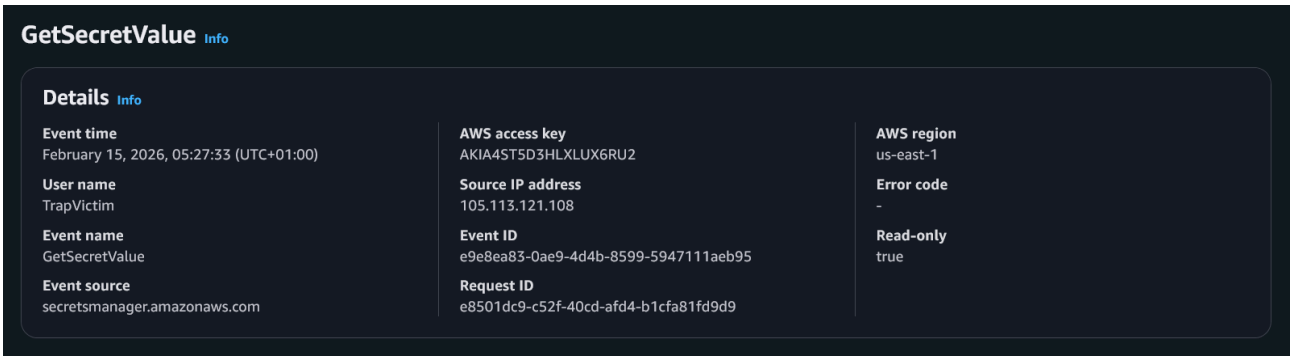
Evidence Logs

The following are the logs of attempts made by the victim user to access protected secrets. These CLI commands which were run at times before and after the **AWS Lambda** function was triggered, further explaining the returns observed in the CLI.

Please note:
By default, the time on the **Details** screens are published in the local **UTC+01:00** timezone format, while the time on the json logs are published in the **UTC+00:00** timezone format.

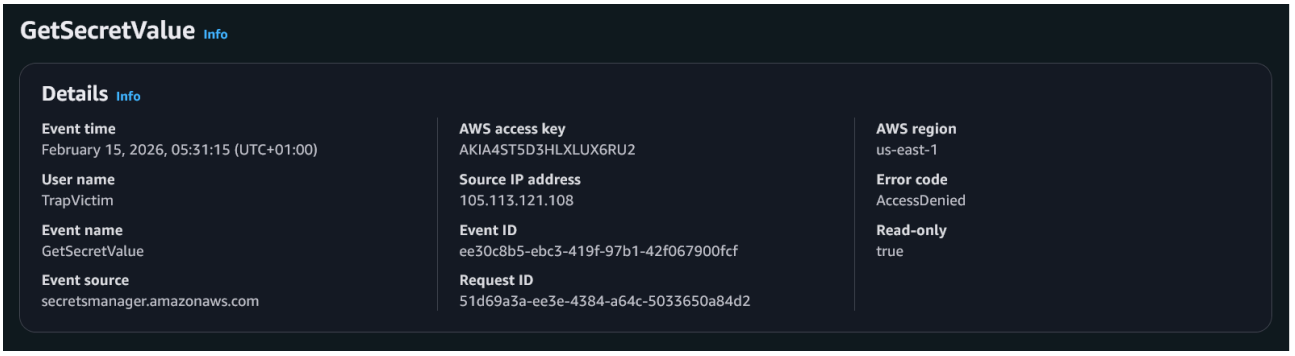
1. GetSecretValue

This log entry was entered **before** the Lambda function was triggered. The victim user **(TrapVictim)** was granted access to the secured resources before the Lambda function locked them out permanently. Notice the absence of the line, `"errorCode": "AccessDenied",`.



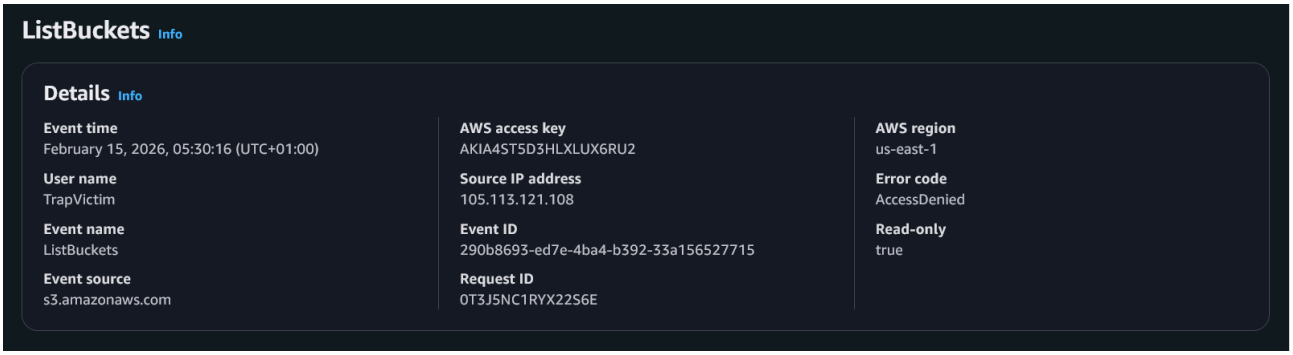
2. GetSecretValue

This log entry was entered **after** the Lambda function was triggered, denying the victim user **(TrapVictim)** access to the secured resources and stripping them of their policies. Notice the `"errorCode": "AccessDenied",` line, proving that.



3. ListBuckets

The command behind this log entry was also run **after** the Lambda function was triggered, denying the victim user **(TrapVictim)** access to the secured resources and stripping them of their policies. Notice the `"errorCode": "AccessDenied",` line, proving that.



Technical Evidence

The Detection

A screenshots of the email notifications from Flow 1 & Flow 2

1. Flow 1 (CloudWatch Alarm)

This email alarm was received about 2 minutes after the breach attempt was made.

AWS Notifications 05:00 (0 minutes ago)
to me

You are receiving this email because your Amazon CloudWatch Alarm "Fortress-Accessed" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [1.0 (15/02/26 03:59:00)] was greater than or equal to the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Sunday 15 February, 2026 04:00:56 UTC".

View this alarm in the AWS Management Console:
https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-1#alarmsV2:alarm/Fortress-Accessed

Alarm Details:
- Name: Fortress-Accessed
- Description: ALERT!!!
The Fortress has been accessed!
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [1.0 (15/02/26 03:59:00)] was greater than or equal to the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Sunday 15 February, 2026 04:00:56 UTC

2. Flow 2 (EventBridge Rule Alert)

This email alert was received immediately the breach attempt was made.

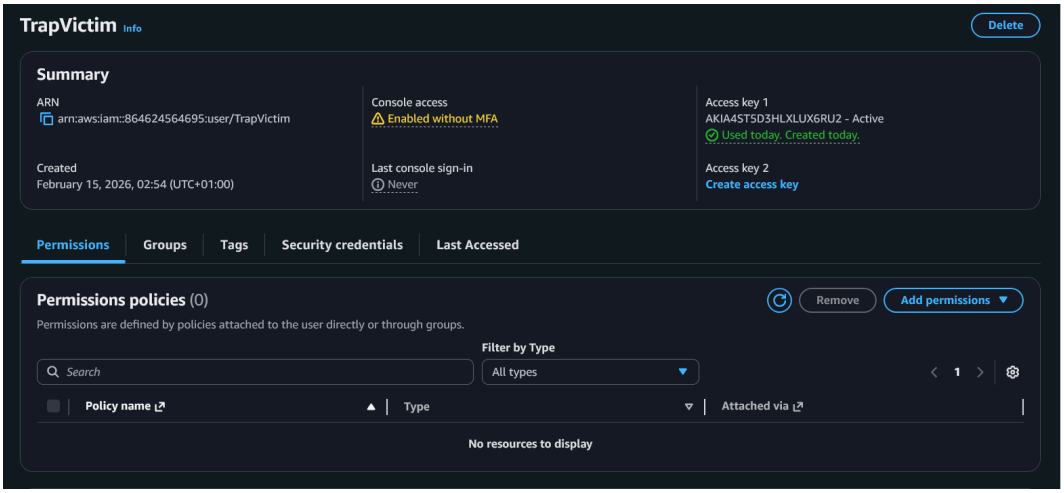
AWS Notification Message Inbox ×

AWS Notifications 04:59 (0 minutes ago)
to me

{"version":"0","id":"3abc6350-4598-da47-f3fe-09961f1e7f7f","detail-type":"AWS API Call via CloudTrail","source":"aws.secretsmanager","account":"864624564695","time":"2026-02-15T03:59:25Z","region":"us-east-1","resources":[],"detail":{"eventVersion":"1.11","userIdentity":{"type":"IAMUser","principalId":"AIDA4ST5D3HLZJKYOQR34","arn":"arn:aws:iam::864624564695:user/TrapVictim","accountId":"864624564695","accessKeyId":"AKIA4ST5D3HLXLUX6RU2","userName":"TrapVictim"},"eventTime":"2026-02-15T03:59:25Z","eventSource":"secretsmanager.amazonaws.com","eventName":"GetSecretValue","awsRegion":"us-east-1","sourceIPAddress":"105.113.121.108","userAgent":"aws-cli/2.33.19 md/awscrt#0.31.1 ua/2.1 os/windows#10 md/arch#amd64 lang/python#3.13.11 md/pyimpl#CPython m/n,Z,b,E cfg/retry-mode#standard md/installer#exe md/prompt#off md/command#secretsmanager.get-secret-value","requestParameters":{"secretId":"Production_Database_Credentials"},"responseElements":null,"requestID":"0e4ddafc-d698-49cb-832f-89201da337fb","eventID":"e1a01c12-283c-4f49-af4f-cbf21dd721e2","readOnly":true,"eventType":"AwsApiCall","managementEvent":true,"recipientAccountId":"864624564695","eventCategory":"Management","tlsDetails":{"tlsVersion":"TLSv1.3","cipherSuite":"TLS_AES_128_GCM_SHA256","clientProvidedHostHeader":"secretsmanager.us-east-1.amazonaws.com","keyExchange":"x25519"}}}

The "Kill-Switch" Proof

A screenshot of the IAM Console showing the "Victim" user now has **zero attached policies**. This is after the Lambda function stripped it of its policies (**ReadOnlyAccess** & **AdministratorAccess** policies).



The Log

A snippet of the CloudTrail JSON showing the `"errorCode": "AccessDenied",` line for the user's second attempt to use the CLI.

```
        "accessKeyId": "AKIA4ST5D3HLXLUX6RU2",
        "userName": "TrapVictim"
    },
    "eventTime": "2026-02-15T04:31:15Z",
    "eventSource": "secretsmanager.amazonaws.com",
    "eventName": "GetSecretValue",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "105.113.121.108",
    "userAgent": "aws-cli/2.33.19 md/awscrt#0.31.1 ua/2.1 os/windows#10 md/arch#amd64 lang/pyt
    "errorCode": "AccessDenied",
    "errorMessage": "User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to perf
    "requestParameters": null,
```


Security Analysis

1. Speed

The Flow 2 alert was faster than the **Flow 1** alarm.
The **Flow 2** alert (EventBridge rule, "**DetectFortressAccess**") was received immediately the breach attempt was made by the victim user ("**TrapVictim**"), with a delay of a few seconds, while the **Flow 1** alert (CloudWatch Metric Filter alarm, "**Fortress-Accessed**") was received 2-3 minutes after the breach attempt.

2. Context

The email with more useful information (IP address, user name, etc) was the **Flow 2** alert (EventBridge rule, "**DetectFortressAccess**"). Apart from being faster, it contained more detail about the source of the breach, as well as surrounding details. Some of them include user identity, time of event, breach action (event name), user agent (machine) details, and the name of the resource the breach attempt targeted.

The AWS service I would use next to block the user automatically is **AWS Lambda.** This service aids the deployment of functions which automate the system's security responses. This functions, after a breach attempt of the monitored resource, can trigger the striping of the user's permissions and policies, locking them out permanently.

3. Compliance

Under the **NIST Cybersecurity Framework (CSF)**, this project satisfies a few key requirements, which include:

Near Real-Time Alerting
NIST emphasizes that monitoring activities should be performed at a frequency sufficient to support timely risk decisions.

- **This project achieves this compliance by** using **Amazon EventBridge** where the system doesn't only monitor the security activities by collecting empty logs, but instead, moves on to "event-driven" monitoring. EventBridge triggers the **SNS** notification and the **Lambda function** ("Kill-Switch") in near real-time (usually seconds). This satisfies the NIST goal of reducing the time between a threat occurring and a response being initiated.

Automated Response Integration (DE.AE-4/RS.RP-1)

While Continuous Monitoring is a "Detect" function, NIST values the integration of both **Detection** and **Response**.

- **This project achieves this compliance by** using **AWS Lambda.** The **Lambda function** automates the response. **NIST** frameworks (like SP 800-137) state that a mature monitoring program doesn't just send an email, but triggers a technical control to mitigate the risk. By stripping the victim user's **AdministratorAccess** and **SecretsManagerReadWrite** policies, you achieve **"Continuous Response,"** which is a high level of NIST monitoring maturity.

Lessons Learned

If I had to improve this system for a Bank, **I would prefer the EventBridge rule system (Flow 2)** for triggering the Kill-Switch. Financial institutions have high security requirements due to the high volume and high sensitivity of the data in their custody. **From my experience with the lab, EventBridge (Flow 2) was faster.** A trigger system with the lowest latency would ensure the quickest response.
This system also ensures high level compliance with relevant standards like the **PCI DSS v4.0 - Requirement 10.4.1 (Critical Event Monitoring)** which outlines the need for monitoring and alerting on access to sensitive credit card data decryption keys.