

CLOUD SECURITY

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

Jeffrey Obi
February 2026

Contents

Introduction	2
Background	2
Project Goals	2
Lab Simulation Stack	2
Phase 1	3
Task 1.1. The Audit Trail	3
Task 1.2. The Honeytoken	3
Task 1.3. Data Residency	3
Phase 2	4
Task 2.1. Metric Filters	4
Task 2.2. The Threshold	4
Task 2.3. Notification	5
Phase 3	5
Task 3.1. Event-Driven Security	6
Task 3.2. Comparison	6
Task 3.3. The Breach	7
Phase 4	8
Task 4.1. The Target	8
Task 4.2. The Trigger	9
Task 4.3. The “Trap” Test	9
Phase 5	11
Executive Summary	11
Lab Architecture	11
Evidence Logs	11
Please note:	12
1. GetSecretValue	12
2. GetSecretValue	12
3. ListBuckets	13
Technical Evidence	13
The Detection	13
Security Analysis	15
1. Speed	15
2. Context	15
3. Compliance	15
Lessons Learned	16

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

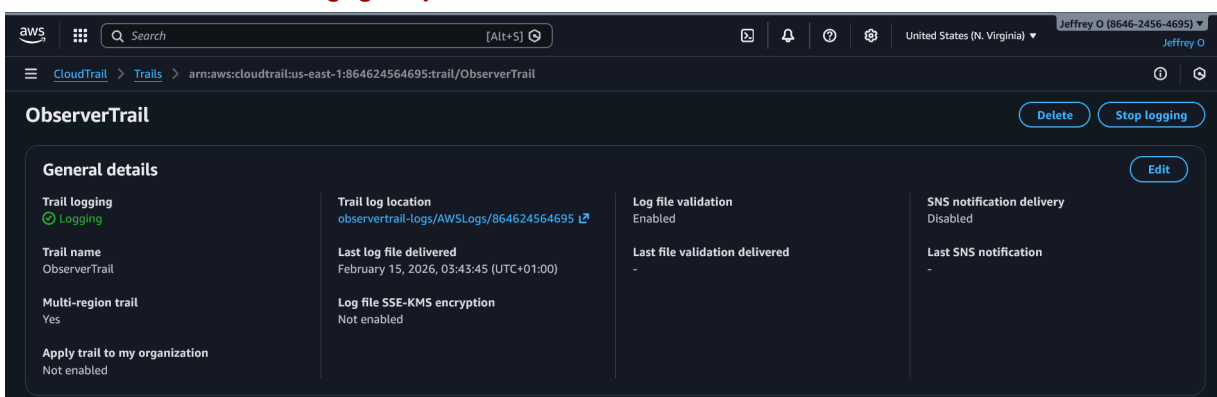
Lab environment:	Cloud (AWS)
Services (cloud tools):	1. AWS CloudTrail 2. Amazon CloudWatch 3. Amazon SNS (Simple Notification Service) 4. AWS Secrets Manager 5. Amazon S3 6. AWS CLI 7. AWS IAM

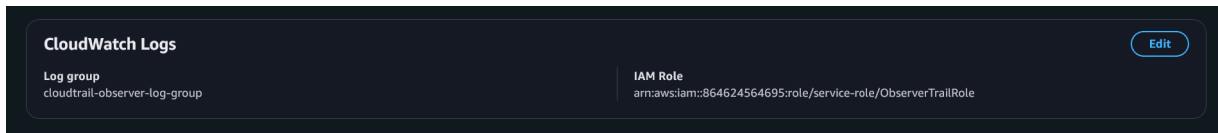
Phase 1

Infrastructure & Logging

Task 1.1. The Audit Trail

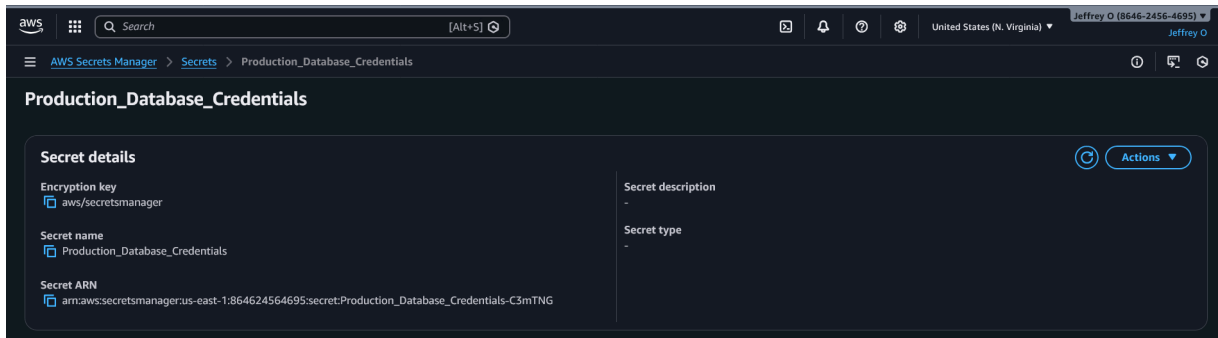
A multi-region **AWS CloudTrail** trail, **ObserverTrail**, integrated with **Amazon CloudWatch** Logs (**cloudtrail-observer-log-group**).





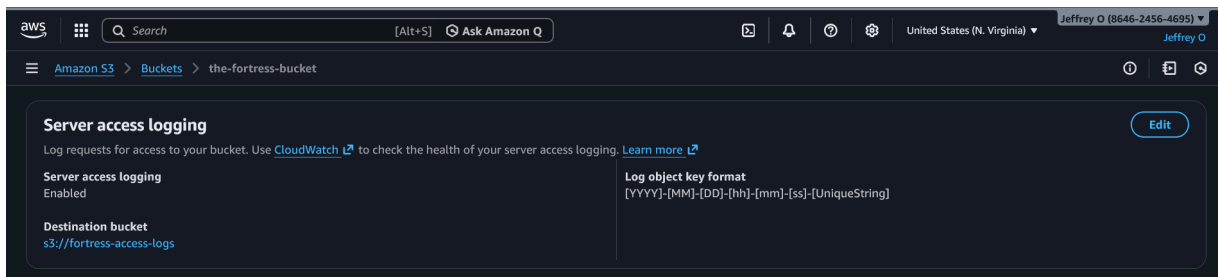
Task 1.2. The Honeytoken

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.

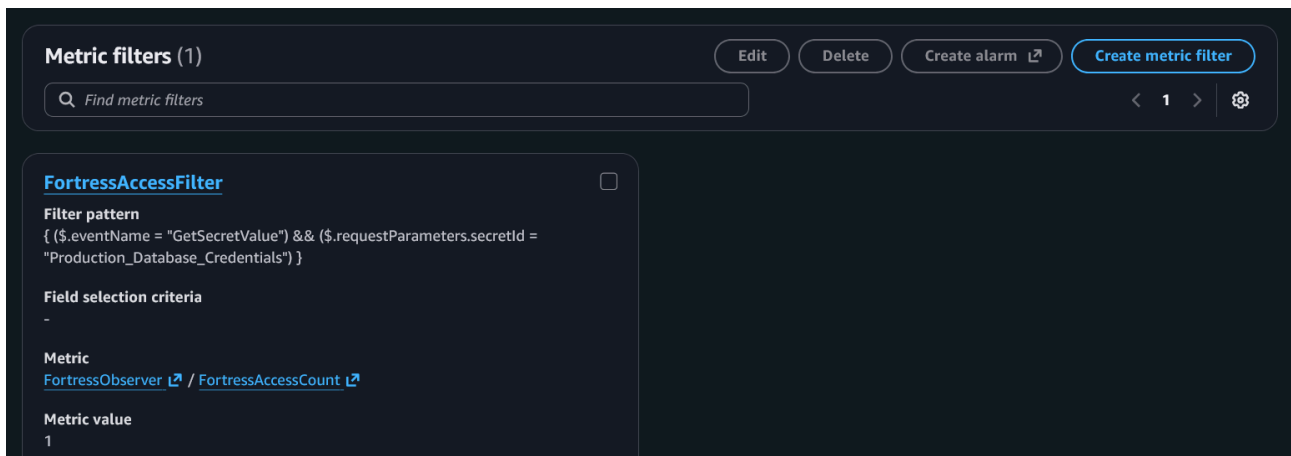


Phase 2

Detection Logic – Flow 1

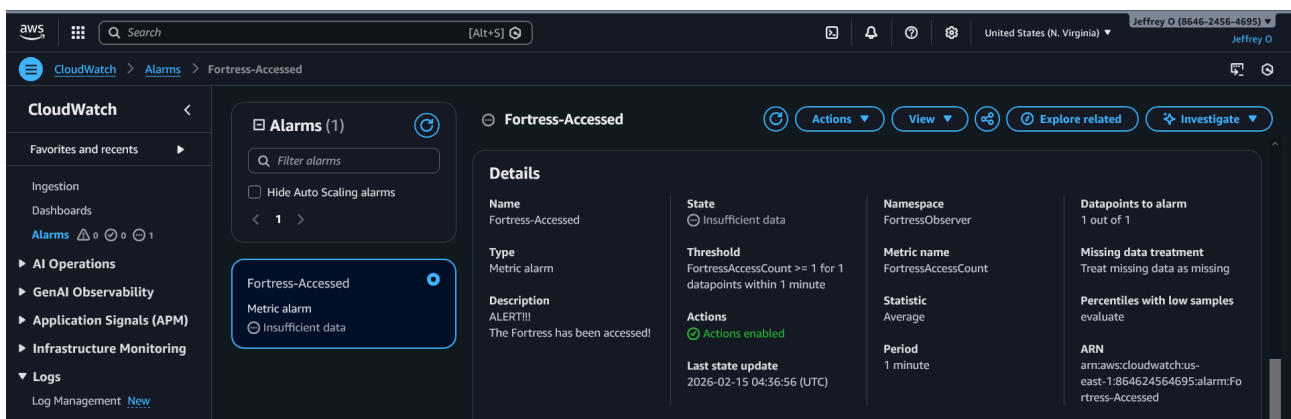
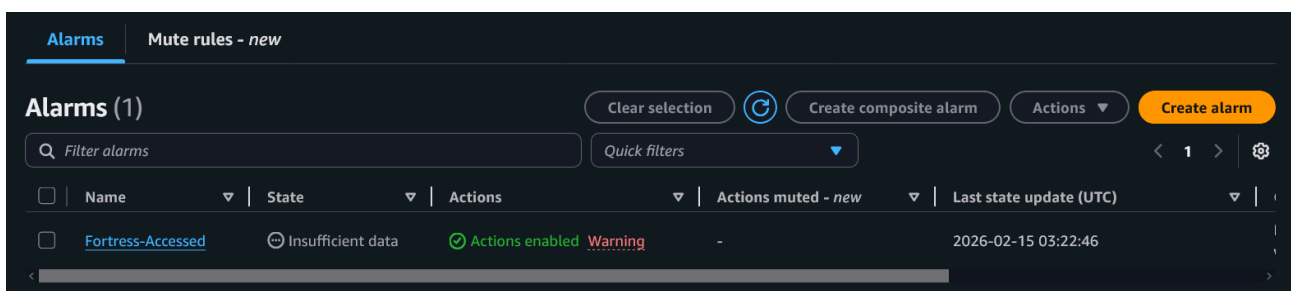
Task 2.1. Metric Filters

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 ≥ 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.

Notification

Alarm state trigger
Define the alarm state that will trigger this action. Remove

☒ **In alarm**
The metric or expression is outside of the defined threshold.

☐ **OK**
The metric or expression is within the defined threshold.

☐ **Insufficient data**
The alarm has just started or not enough data is available.

Send a notification to the following SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.

☒ **Select an existing SNS topic**

☐ Create new topic

☐ Use topic ARN to notify other accounts

Send a notification to...

AlertTopic1

Only topics belonging to this account are listed here. All persons and applications subscribed to the selected topic will receive notifications.

Email (endpoints)
obi.obiora.jeffrey@gmail.com - [View in SNS Console](#)

Add notification

AlertTopic1

Edit Delete Publish message

Details

Name AlertTopic1	ARN arn:aws:sns:us-east-1:864624564695:AlertTopic1	Display name -	Type Standard
Topic owner 864624564695			

< **Subscriptions** **Access policy** **Data protection policy** **Delivery policy (HTTP/S)** **Delivery status logging** **Encryption** >

Subscriptions (1)

Edit Delete Request confirmation Confirm subscription Create subscription

Search

ID	Endpoint	Status	Protocol
59afaaeb-dc91-471d-a419-1a9f...	obi.obiora.jeffrey@gmail.com	Confirmed	EMAIL

AWS Notification - Subscription Confirmation

AWS Notifications <no-reply@sns.amazonaws.com>
to me

04:18 (5 minutes ago) ☆



Simple Notification Service

You have chosen to subscribe to the topic:

arn:aws:sns:us-east-1:864624564695:AlertTopic1

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):

[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

arn:aws:sns:us-east-1:864624564695:AlertTopic1:59afaaeb-dc91-471d-a419-1a9f68774602

If it was not your intention to subscribe, [click here to unsubscribe](#).

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**.

Event pattern [Info](#)

```
1 {
2   "source": ["aws.secretsmanager"],
3   "detail-type": ["AWS API Call via CloudTrail"],
4   "detail": {
5     "eventSource": ["secretsmanager.amazonaws.com"],
6     "eventName": ["GetSecretValue"]
7   }
8 }
```

Event pattern rules (1)

[Delete](#)[Enable](#)[Edit](#)[CloudFormation Template](#) ▼[Create rule](#)

⚠ We have moved Scheduled rules in your account to 'Scheduler' section, where you can view, create and edit a scheduled rule.

[Go to Scheduled rules](#)

Any status ▼

< 1 >



☐	Name	Status	Type	Event bus	ARN	Description
☐	DetectFortressAccess	Enabled	Standard	default	arn:aws:events:us-east-1:864624564695:rule/DetectFortressAccess	-

DetectFortressAccess

[Edit](#)[Disable](#)[Delete](#)[CloudFormation Template](#) ▼

Rule details [Info](#)

Rule name
DetectFortressAccess

Status
Enabled

Event bus name
default

Type
Standard

Description
-

Rule ARN
arn:aws:events:us-east-1:864624564695:rule/DetectFortressAccess

Event bus ARN
arn:aws:events:us-east-1:864624564695:event-bus/default

Task 3.2. Comparison

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

DetectFortressAccess

[Edit](#)[Disable](#)[Delete](#)[CloudFormation Template](#) ▼

Rule details [Info](#)

Rule name
DetectFortressAccess

Status
Enabled (with CloudTrail read-only Management events)

Event bus name
default

Type
Standard

Description
-

Rule ARN
arn:aws:events:us-east-1:864624564695:rule/DetectFortressAccess

Event bus ARN
arn:aws:events:us-east-1:864624564695:event-bus/default

Event pattern

Targets

Monitoring

Tags

Targets

Edit

Details	Target Name	Type	ARN	Input	Role
▶	FortressKillSwitch	Lambda function	 arn:aws:lambda:us-east-1:864624564695:function:FortressKillSwitch	Matched event	-
▶	AlertTopic2	SNS topic	 arn:aws:sns:us-east-1:864624564695:AlertTopic2	Matched event	DetectSecretAccess_to_AlertTopic2_Role

AlertTopic2 Edit Delete Publish message

Details

Name
 AlertTopic2

ARN
 arn:aws:sns:us-east-1:864624564695:AlertTopic2

Display name
-

Type
Standard

Topic owner
864624564695

< **Subscriptions** Access policy Data protection policy Delivery policy (HTTP/S) Delivery status logging Encryption >

Subscriptions (1) Edit Delete Request confirmation Confirm subscription Create subscription

ID	Endpoint	Status	Protocol
b48cabd6-a838-43b3-8cef-9851...	obi.obiora.jeffrey@gmail.com	Confirmed	EMAIL

AWS Notifications
to me ▾

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:864624564695:AlertTopic2

...

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
[Confirm subscription](#)



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:
arn:aws:sns:us-east-1:864624564695:AlertTopic2:b48cabd6-a838-43b3-8cef-985179ac5b0f

If it was not your intention to subscribe, [click here to unsubscribe](#).

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.

SecAdminUser Info Delete

Summary

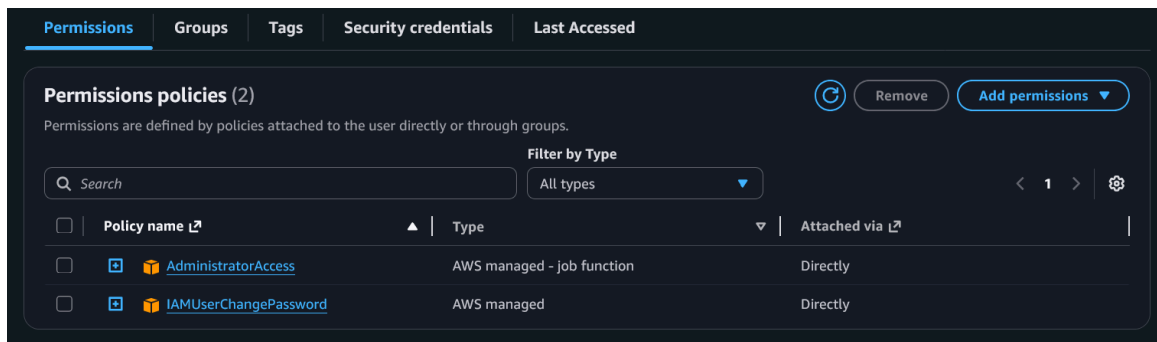
ARN
 arn:aws:iam::864624564695:user/SecAdminUser

Console access
 Enabled without MFA

Access key 1
[Create access key](#)

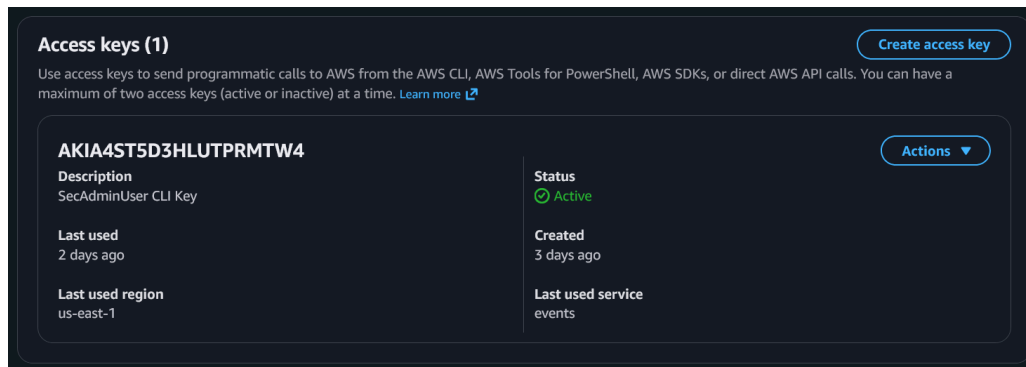
Created
February 12, 2026, 14:10 (UTC+01:00)

Last console sign-in
 Never



The screenshot shows the AWS IAM console's 'Permissions policies' page. It has tabs for 'Permissions', 'Groups', 'Tags', 'Security credentials', and 'Last Accessed'. The 'Permissions' tab is active. Below the tabs, there's a section titled 'Permissions policies (2)' with a 'Remove' button and an 'Add permissions' button. A note states: 'Permissions are defined by policies attached to the user directly or through groups.' There's a search bar and a 'Filter by Type' dropdown set to 'All types'. A table lists two policies:

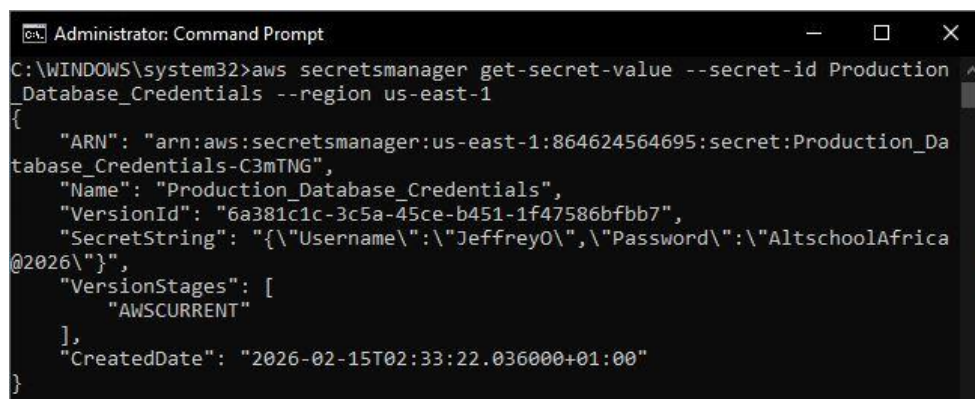
☐	Policy name	Type	Attached via
☐	AdministratorAccess	AWS managed - job function	Directly
☐	IAMUserChangePassword	AWS managed	Directly



The screenshot shows the AWS IAM console's 'Access keys' page. It has a 'Create access key' button. A note states: '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](#)'

There is one access key listed:

AKIA4ST5D3HLUTPRMTW4		Actions	
Description	SecAdminUser CLI Key		
Status	Active		
Last used	2 days ago		
Created	3 days ago		
Last used region	us-east-1	Last used service	events



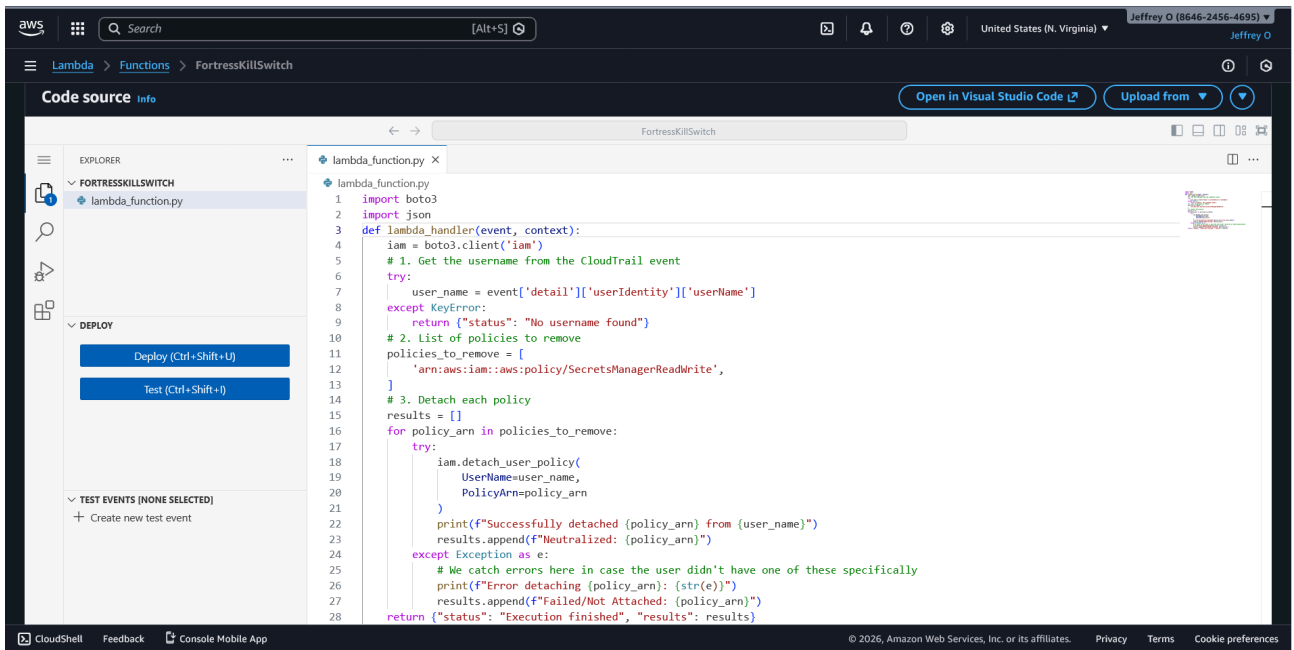
```
Administrator: Command Prompt
C:\WINDOWS\system32>aws secretsmanager get-secret-value --secret-id Production
Database_Credentials --region us-east-1
{
  "ARN": "arn:aws:secretsmanager:us-east-1:864624564695:secret:Production_Da
tase_Credentials-C3mTNG",
  "Name": "Production_Database_Credentials",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

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.

Event pattern

Targets

Monitoring

Tags

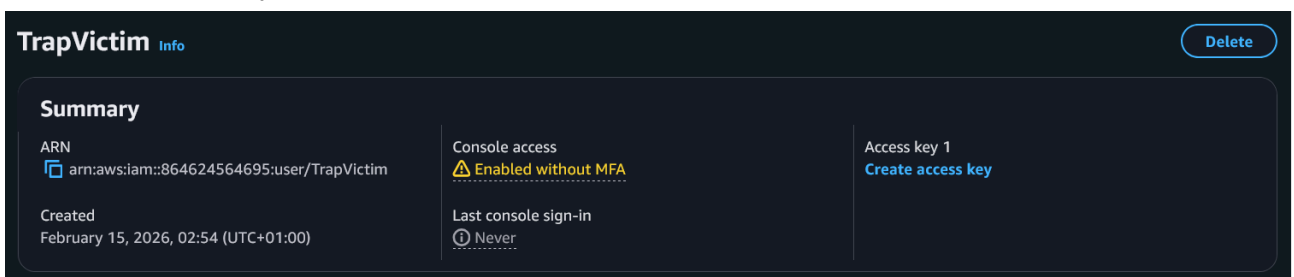
Targets

Edit

Details	Target Name	Type	ARN	Input	Role
▶	FortressKillSwitch ↗	Lambda function	🔗 arn:aws:lambda:us-east-1:864624564695:function:FortressKillSwitch	Matched event	-
▶	AlertTopic2 ↗	SNS topic	🔗 arn:aws:sns:us-east-1:864624564695:AlertTopic2	Matched event	DetectSecretAccess_to_AlertTopic2_Role ↗

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) Remove Add permissions

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type 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
Description
-
Last used
None
Last used region
N/A

Status
Active
Created
4 minutes ago
Last used service
N/A

Actions

```
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": "AIDA4ST5D3HLZJKYQQR34",
  "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
tase_Credentials-C3mTNG",
  "Name": "Production_Database_Credentials",
  "VersionId": "6a381c1c-3c5a-45ce-b451-1f47586bfbb7",
  "SecretString": "{\"Username\":\"JeffreyO\",\"Password\":\"AltschoolAfrica
@2026\"}",
  "VersionStages": [
    "AWSCURRENT"
  ],
  "CreateDate": "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_Credent
ials 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

The screenshot displays the AWS CloudWatch console interface. The left sidebar shows the navigation menu with 'Alarms' selected. The main content area shows the configuration for the 'Fortress-Accessed' alarm. The alarm is a metric alarm for the 'FortressAccessCount' metric in the 'FortressObserver' namespace. The threshold is set to 'FortressAccessCount >= 1 for 1 datapoints within 1 minute'. The alarm is currently in a 'State: Insufficient data' state. The description reads 'ALERT!!! The Fortress has been accessed!'. The actions are 'Actions enabled'. The last state update was on '2026-02-15 04:36:56 (UTC)'. The period is '1 minute'. The missing data treatment is 'Treat missing data as missing'. The percentiles with low samples evaluate. The ARN is 'arn:aws:cloudwatch:us-east-1:864624564695:alarm:Fortress-Accessed'.

EventBridge Rule Configuration Details

The screenshot displays the AWS EventBridge console interface. The left sidebar shows the navigation menu with 'Rules' selected. The main content area shows the configuration for the 'DetectFortressAccess' rule. The rule is currently 'Status: Enabled (with CloudTrail read-only Management events)'. The description is empty. The rule ARN is 'arn:aws:events:us-east-1:864624564695:rule/DetectFortressAccess'. The event bus name is 'default'. The event bus ARN is 'arn:aws:events:us-east-1:864624564695:event-bus/default'. The type is 'Standard'.

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"**,.

GetSecretValue Info

Details Info

Event time February 15, 2026, 05:27:33 (UTC+01:00)	AWS access key AKIA4ST5D3HLXLUX6RU2	AWS region us-east-1
User name TrapVictim	Source IP address 105.113.121.108	Error code -
Event name GetSecretValue	Event ID e9e8ea83-0ae9-4d4b-8599-5947111aeb95	Read-only true
Event source secretsmanager.amazonaws.com	Request ID e8501dc9-c52f-40cd-afd4-b1cfa81fd9d9	

Event record Info Copy

JSON view

```
1 {
2   "eventVersion": "1.11",
3   "userIdentity": {
4     "type": "IAMUser",
5     "principalId": "AIDA4ST5D3HLZJKY0QR34",
6     "arn": "arn:aws:iam:864624564695:user/TrapVictim",
7     "accountId": "864624564695",
8     "accessKeyId": "AKIA4ST5D3HLXLUX6RU2",
9     "userName": "TrapVictim"
10  },
11  "eventTime": "2026-02-15T04:27:33Z",
12  "eventSource": "secretsmanager.amazonaws.com",
13  "eventName": "GetSecretValue",
14  "awsRegion": "us-east-1",
15  "sourceIPAddress": "105.113.121.108",
16  "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/b,n,Z,E cfg/retry-
17  "requestParameters": {
18    "secretId": "Production_Database_Credentials"
```

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.

GetSecretValue Info

Details Info

Event time February 15, 2026, 05:31:15 (UTC+01:00)	AWS access key AKIA4ST5D3HLXLUX6RU2	AWS region us-east-1
User name TrapVictim	Source IP address 105.113.121.108	Error code AccessDenied
Event name GetSecretValue	Event ID ee30c8b5-ebc3-419f-97b1-42f067900fcf	Read-only true
Event source secretsmanager.amazonaws.com	Request ID 51d69a3a-ee3e-4384-a64c-5033650a84d2	

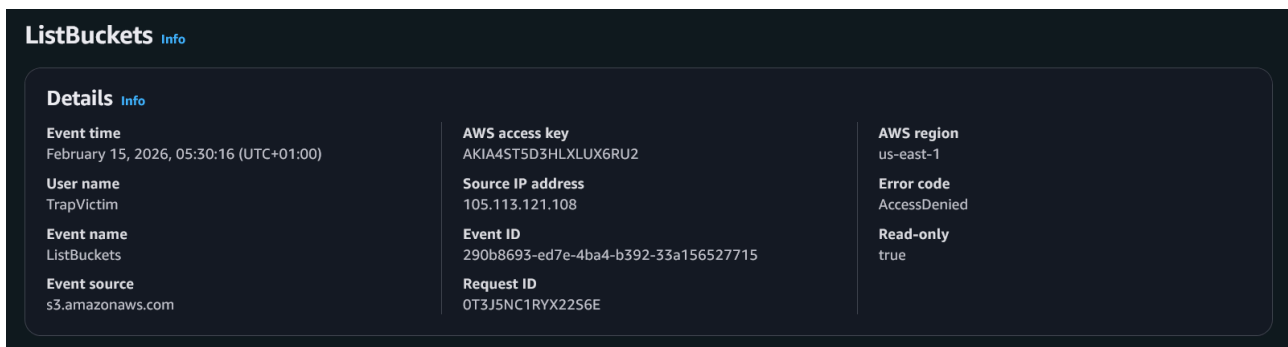


The screenshot shows an AWS Event record in the 'JSON view' tab. The event is titled 'GetSecretValue' and occurred on February 15, 2026, at 05:30:16 UTC. The user identity is 'TrapVictim', an IAM user with principal ID 'AIDA4ST5D3HLZJKYQOR34'. The event source is 'secretsmanager.amazonaws.com'. The error code is 'AccessDenied' with the message: 'User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to perform: secretsmanager:GetSecretValue on resource: Pr'. The event name is 'GetSecretValue' and the region is 'us-east-1'.

```
1 {
2   "eventVersion": "1.11",
3   "userIdentity": {
4     "type": "IAMUser",
5     "principalId": "AIDA4ST5D3HLZJKYQOR34",
6     "arn": "arn:aws:iam::864624564695:user/TrapVictim",
7     "accountId": "864624564695",
8     "accessKeyId": "AKIA4ST5D3HLXUX6RU2",
9     "userName": "TrapVictim"
10  },
11  "eventTime": "2026-02-15T04:31:15Z",
12  "eventSource": "secretsmanager.amazonaws.com",
13  "eventName": "GetSecretValue",
14  "awsRegion": "us-east-1",
15  "sourceIPAddress": "105.113.121.108",
16  "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/b,Z,n,E cfg/retry-",
17  "errorCode": "AccessDenied",
18  "errorMessage": "User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to perform: secretsmanager:GetSecretValue on resource: Pr
```

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.



ListBuckets Info		
Details Info		
Event time February 15, 2026, 05:30:16 (UTC+01:00)	AWS access key AKIA4ST5D3HLXUX6RU2	AWS region us-east-1
User name TrapVictim	Source IP address 105.113.121.108	Error code AccessDenied
Event name ListBuckets	Event ID 290b8693-ed7e-4ba4-b392-33a156527715	Read-only true
Event source s3.amazonaws.com	Request ID 0T3J5NC1RYX2256E	



The screenshot shows an AWS Event record in the 'JSON view' tab. The event is titled 'ListBuckets' and occurred on February 15, 2026, at 05:30:16 UTC. The user identity is 'TrapVictim', an IAM user with principal ID 'AIDA4ST5D3HLZJKYQOR34'. The event source is 's3.amazonaws.com'. The error code is 'AccessDenied' with the message: 'User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to perform: s3:ListAllMyBuckets with an explicit deny in'. The event name is 'ListBuckets' and the region is 'us-east-1'.

```
1 {
2   "eventVersion": "1.11",
3   "userIdentity": {
4     "type": "IAMUser",
5     "principalId": "AIDA4ST5D3HLZJKYQOR34",
6     "arn": "arn:aws:iam::864624564695:user/TrapVictim",
7     "accountId": "864624564695",
8     "accessKeyId": "AKIA4ST5D3HLXUX6RU2",
9     "userName": "TrapVictim"
10  },
11  "eventTime": "2026-02-15T04:30:16Z",
12  "eventSource": "s3.amazonaws.com",
13  "eventName": "ListBuckets",
14  "awsRegion": "us-east-1",
15  "sourceIPAddress": "105.113.121.108",
16  "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/b,E,n,C,Z cfg/ret",
17  "errorCode": "AccessDenied",
18  "errorMessage": "User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to perform: s3:ListAllMyBuckets with an explicit deny in
```

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

to me

05:00 (0 minutes ago)



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 x



AWS Notifications

to me

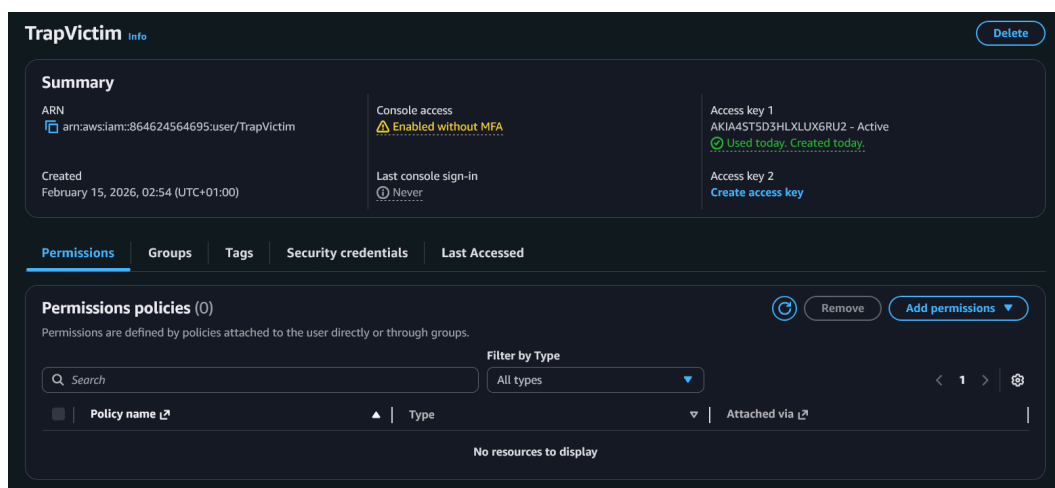
04:59 (0 minutes ago)



```
{
  "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": "AIDA4ST5D3HLZJKYQQR34",
      "arn": "arn:aws:iam:864624564695:user/TrapVictim",
      "accountId": "864624564695",
      "accessKeyId": "AKIA4ST5D3HLX6RU2",
      "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/awscli#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": "0e4dda4c-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.


```
8      "accessKeyId": "AKIA4ST5D3HLXLU6RU2",
9      "userName": "TrapVictim"
10   },
11   "eventTime": "2026-02-15T04:31:15Z",
12   "eventSource": "secretsmanager.amazonaws.com",
13   "eventName": "GetSecretValue",
14   "awsRegion": "us-east-1",
15   "sourceIPAddress": "105.113.121.108",
16   "userAgent": "aws-cli/2.33.19 md/awscrt#0.31.1 ua/2.1 os/windows#10 md/arch#amd64 lang/pyt
17   "errorCode": "AccessDenied",
18   "errorMessage": "User: arn:aws:iam::864624564695:user/TrapVictim is not authorized to perf
19   "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 stripping 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.