

# Enterprise Continuous Monitoring and Threat Detection on AWS

Author: Adedayo

Specialization: Cloud Security & Threat Detection

Platform: Amazon Web Services (AWS)

## 1. Introduction

This document describes the design and implementation of a centralized monitoring and threat detection framework in an AWS environment.

I implemented this solution to improve visibility into API activity, network traffic, security misconfigurations, and threat indicators, while enabling rapid incident response.

## 2. Objectives

The primary objectives were to:

- Enable continuous security monitoring
- Detect suspicious IAM and API activity
- Identify network-based threats
- Centralize security findings
- Automate low-severity remediation
- Improve incident response readiness

## 3. Security Monitoring Architecture

The monitoring framework was built using the following AWS services:

- Amazon GuardDuty
- AWS CloudTrail
- AWS Security Hub
- VPC Flow Logs
- Amazon CloudWatch
- Amazon SNS
- AWS Lambda

All security events and findings were centralized for analysis and response.

## 4. GuardDuty Configuration

I enabled Amazon GuardDuty to monitor:

The screenshot shows the Amazon GuardDuty Summary page. On the left, there's a sidebar with options like 'Protection plans' (S3 Protection, EKS Protection, Extended Threat Detection), 'Accounts', and 'Usage'. The main area has a summary card with 'Attack sequences - new' (0), 'Total findings' (0), 'Resources with findings' (0), and 'Accounts with findings' (0). Below this is a 'Findings' section with counts for Critical (0), High (0), Medium (0), and Low (0) severity levels. A large watermark 'GUARD' is overlaid across the page.

- CloudTrail logs
- VPC Flow Logs
- DNS activity

GuardDuty was configured to generate findings for suspicious behavior and compromised credentials.

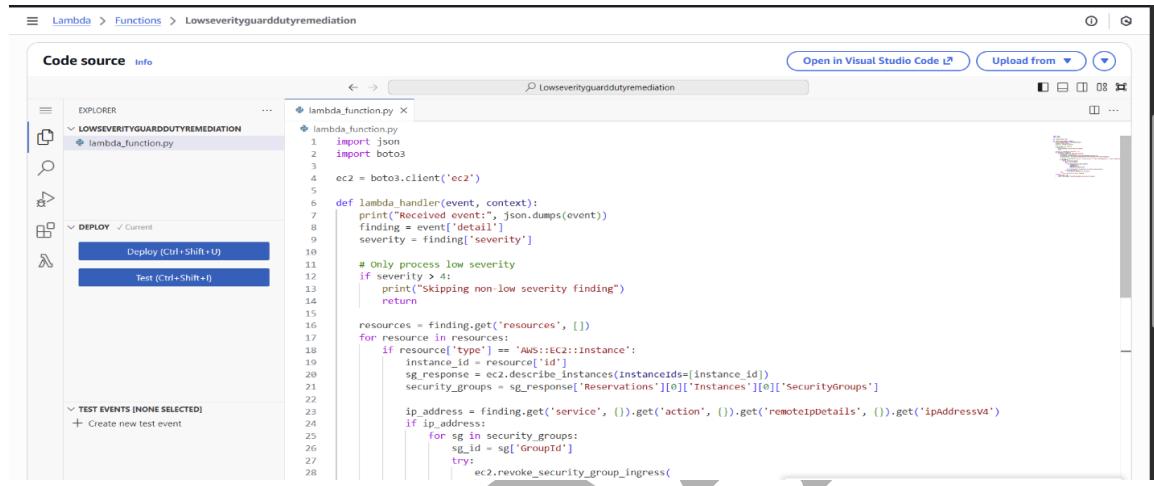
An SNS topic was created to distribute GuardDuty alerts to the security team.

The screenshot shows the Amazon SNS Topics page. The 'GuarddutyFindings' topic is selected. It shows details like Name (GuarddutyFindings), ARN (arn:aws:sns:us-east-1:879381257906:GuarddutyFindings), Display name (-), and Type (Standard). The 'Subscriptions' tab is active, showing one subscription with the ID 'arn:aws:sns:us-east-1:879381257906:GuarddutyFindings'. There are buttons for 'Edit', 'Delete', 'Publish message', and 'Create subscription'.

Sample findings were generated to validate alert delivery.

## 5. Automated Remediation for Low-Severity Findings

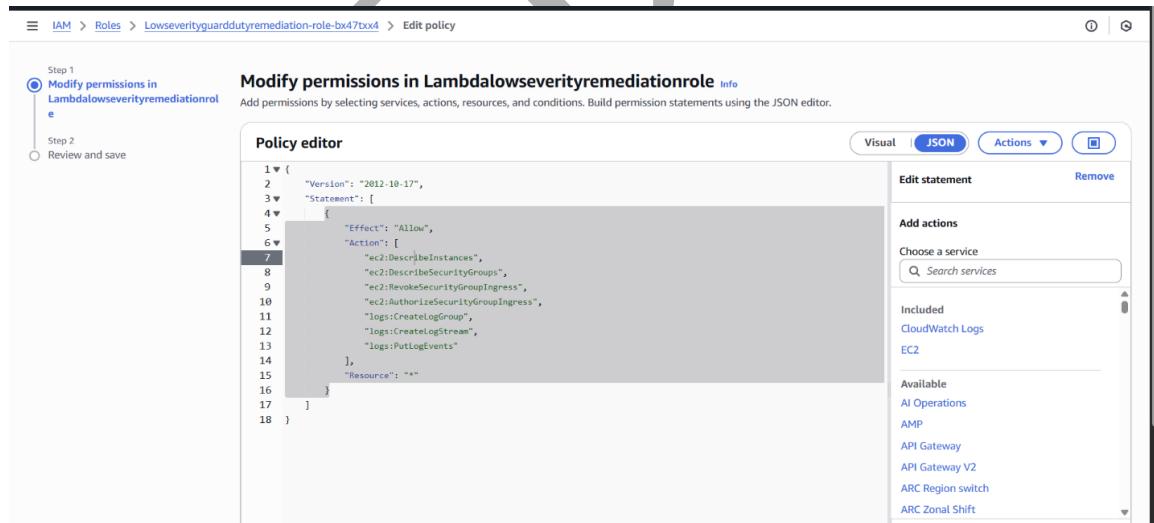
I implemented automated remediation for low-severity findings using AWS Lambda.



The screenshot shows the AWS Lambda function editor for a function named "Lowseverityguarddutyremediation". The code is written in Python and uses the AWS SDK (boto3) to interact with the EC2 service. It processes a JSON event, extracts a finding ID, and then checks if the severity is low. If so, it retrieves the EC2 instance details and iterates through security groups to revoke ingress rules from specific IP addresses. The Lambda function has a role associated with it, which is shown in the "DEPLOY" section of the sidebar.

```
lambda_function.py
1 import json
2 import boto3
3
4 ec2 = boto3.client('ec2')
5
6 def lambda_handler(event, context):
7     print("Received event: ", json.dumps(event))
8     finding = event['detail']
9     severity = finding['severity']
10
11     # Only process low severity
12     if severity > 4:
13         print("Skipping non-low severity finding")
14         return
15
16     resources = finding.get('resources', [])
17     for resource in resources:
18         if resource['type'] == 'AWS::EC2::Instance':
19             instance_id = resource['id']
20             sg_response = ec2.describe_instances(InstanceIds=[instance_id])
21             security_groups = sg_response['Reservations'][0]['Instances'][0]['SecurityGroups']
22
23             ip_address = finding.get('service', {}).get('action', {}).get('remoteIpDetails', {}).get('ipAddressV4')
24             if ip_address:
25                 for sg in security_groups:
26                     sg_id = sg['GroupId']
27                     try:
28                         ec2.revoke_security_group_ingress(
```

A dedicated IAM role was created with permissions to modify tagged security groups.



The screenshot shows the AWS IAM Role policy editor for a role named "LambdaLowseverityremediationrole". The policy is titled "Modify permissions in LambdaLowseverityremediationrole". The "Visual" tab is selected, showing a JSON-based policy document. The policy grants the "Allow" effect for several EC2 actions: DescribeInstances, DescribeSecurityGroups, RevokeSecurityGroupIngress, AuthorizeSecurityGroupIngress, CreateLogGroup, CreateLogStream, and PutLogEvents. The "Action" column lists these actions, and the "Resource" column is set to "\*". The "Add actions" section shows a search bar and a list of available services, with "EC2" currently selected.

```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "ec2:DescribeInstances",
8                 "ec2:DescribeSecurityGroups",
9                 "ec2:RevokeSecurityGroupIngress",
10                "ec2:AuthorizeSecurityGroupIngress",
11                "logs:CreateLogGroup",
12                "logs:CreateLogStream",
13                "logs:PutLogEvents"
14            ],
15            "Resource": "*"
16        }
17    ]
18 }
```

In production environments, permissions would be restricted to specific ARNs and tagged resources to enforce least privilege.

A Lambda function was deployed and triggered using EventBridge when low-severity findings occurred.

## 6. CloudTrail and API Monitoring

I configured a centralized CloudTrail trail to capture all AWS API activity.

Logs were forwarded to CloudWatch Logs for near real-time analysis.

The screenshot shows the AWS CloudTrail 'Trails' section. A specific trail named 'SecureSphereCloudtrail' is selected. The 'General details' section includes:

- Trail logging: Logging (green checkmark)
- Trail name: SecureSphereCloudtrail
- Multi-region trail: Yes
- Apply trail to my organization: Not enabled

The 'CloudWatch Logs' section shows:

- Log group: aws-cloudtrail-logs-securesphere

The 'IAM Role' section shows:

- arn:aws:iam::879381257906:role/service-role/securespherecloudtrailrole

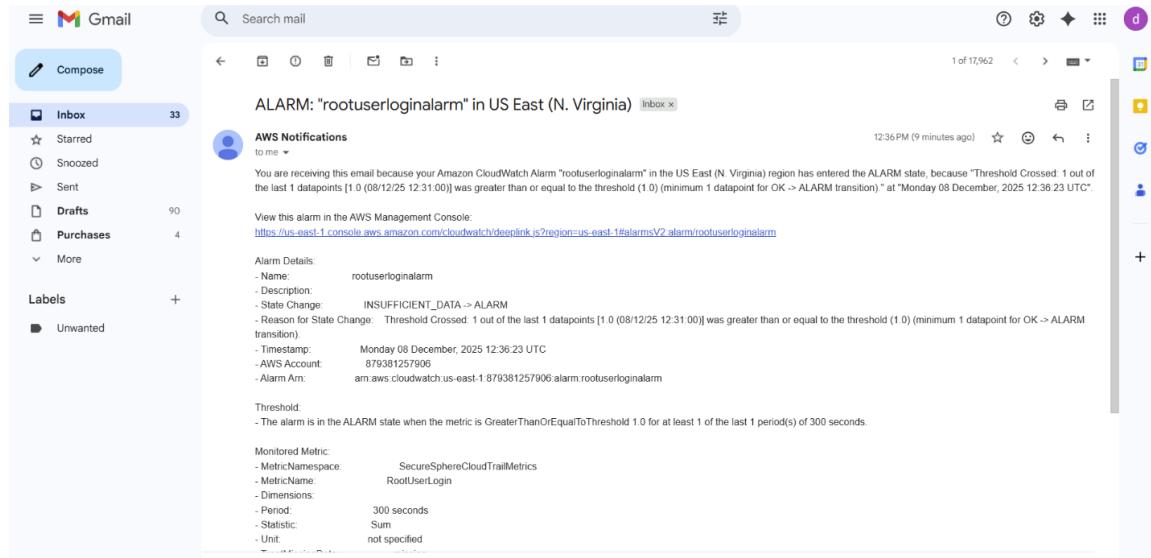
SNS notifications were configured to alert the security team when suspicious events were detected.

## 7. CloudWatch Metric Filters and Alarms

Metric filters were created for critical security events, including:

- Unauthorized API calls
- Console login without MFA
- Root account usage
- IAM policy changes
- CloudTrail configuration changes
- AWS Config changes
- S3 bucket policy changes

Each metric was linked to a CloudWatch alarm and SNS notification.



Manual test events were generated to validate metric activation and alert delivery.

## 8. Security Hub Integration

AWS Security Hub was enabled to aggregate findings from multiple services, including:

- AWS Config
- GuardDuty
- Inspector
- Macie

Security standards were reviewed and findings were prioritized based on risk level.

## 9. S3 Security Monitoring and Remediation

A test S3 bucket was created with public access enabled to validate detection capabilities.

GuardDuty and Security Hub detected the misconfiguration.

**Block public access settings are disabled for the S3 bucket**

High | New Last updated 3 hours ago.

All bucket-level block public access settings were disabled for the S3 bucket. Access to the bucket is controlled by ...

[Read more](#)

[View JSON](#) [Actions](#)

**Overview** **Resources (1)**

Type	PolicyIAMUser/S3BlockPublicAccessDisabled
Posture Management	...
Region	us-east-1
Account	879381257906
Age	3 hours
Created time	December 08, 2025, 16:43 (UTC+00:00)

[View more](#)

**Resources**

Public access was removed to remediate the high severity finding.

**Top 10 attributes**

Attribute	Value	Severity
Block public access settings are disabled for the S3 bucket	High	High
S3 general purpose buckets should block public access	High	High
VPC default security groups should not allow inbound or outbound traffic	High	High
VPC default security groups should not allow inbound or outbound traffic	High	High
A Kubernetes API commonly used in Impact tactics invoked from a Tor exit node IP address	High	High

**Overview**

Finding trends are an average count of findings for the last 90 days. Filters on the findings table do not affect the graph.

**Finding count**

**Filter by severity**

**Saved filter sets**

**Choose a filter set**

**Findings**

**Severity**

- Critical (13)
- High** (157)
- Medium (175)
- Low (76)
- Informational (103)

**Top 10 attributes**

**Account ID**

**Compliance**

Category	Status	Control	Config rule	Standards	Requirements
Software and Configuration Checks/Industry and Regulatory Standards	Pass	S3.8	securityhub-s3-bucket-level-public-access-prohibited-f5c4e845	standards/aws-foundational-security-best-practices/v/1.0.0	standards/cis-aws-foundations-benchmark/v/5.0.0
Posture Management	...	...	...	...	...
Region	us-east-1	...	...	...	...
Account	879381257906	...	...	...	...
Age	3 hours	...	...	...	...
Created time	December 08, 2025, 16:41 (UTC+00:00)	...	...	...	...

[View more](#)

**Remediation**

For information on how to correct this issue, consult the AWS Security Hub controls documentation.

Server access logging was enabled to improve audit visibility.

## 10. Network Security and VPC Flow Logs

VPC Flow Logs were enabled for all production VPCs.

Logs were sent to CloudWatch Log Groups for monitoring and analysis.

An EC2 instance was used to generate network traffic for validation.

Both ACCEPT and REJECT traffic was verified in log streams.

Timestamp	Message
2025-12-09T21:47:36.000Z	2 879381257906 eni-05397f70fd742f16 89.187.190.210 172.31.26.227 0 0 1 36 1765316866 1765316866 REJECT OK
2025-12-09T21:47:36.000Z	2 879381257906 eni-05397f70fd742f16 119.91.218.190 172.31.26.227 12155 28891 17 1 28 1765316856 1765316866 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 52.207.222.50 52080 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 13.218.199.213 46621 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 54.90.191.9 54655 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 3.87.127.143 34767 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 44.281.148.133 57989 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 52.207.222.50 59400 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 13.218.199.213 48023 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 54.90.191.9 34898 123 17 1 76 1765316864 1765316899 REJECT OK
2025-12-09T21:47:44.000Z	2 879381257906 eni-05397f70fd742f16 172.31.26.227 3.87.127.143 44469 123 17 1 76 1765316864 1765316899 REJECT OK

## 11. Network Monitoring Alerts

Metric filters were created for:

- Rejected traffic events
- Log delivery failures

Metric filter "LogdeliveryFailure" has been created.

**Metric filters (2)**

**RejectLogFilter**

- Filter pattern: "REJECT"
- Field selection criteria:
- Metric: SecureSphereVPCFlowLogsMonitoring / RejectedTraffic
- Metric value: 1
- Default value: -
- Applied on transformed logs
- Unit: -

**LogdeliveryFailure**

- Filter pattern: "NODATA" "SKIPDATA"
- Field selection criteria:
- Metric: SecureSphereVPCFlowLogs / logfailuredelivery
- Metric value: 1
- Default value: -
- Applied on transformed logs
- Unit: -

CloudWatch alarms were configured to notify the security team of network anomalies.

Compose

Inbox 44

Starred Snoozed Sent Drafts 90 Purchases 4 More

Labels + Unwanted

Search mail

ALARM: "FlowLogsalarm" in US East (N. Virginia) [Inbox](#)

AWS Notifications to me 3 of 17,974 10:38 PM (3 minutes ago)

You are receiving this email because your Amazon CloudWatch Alarm "FlowLogsalarm" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [50.0 (09/12/25 22:33:00)] was greater than or equal to the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition)" at "Tuesday 09 December, 2025 22:38:08 UTC".

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

Alarm Details:  
 - Name: FlowLogsalarm  
 - Description:  
 - Stat Change: INSUFFICIENT\_DATA->ALARM  
 - Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [50.0 (09/12/25 22:33:00)] was greater than or equal to the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition).  
 - Timestamp: Tuesday 09 December, 2025 22:38:08 UTC  
 - AWS Account: 879381257906  
 - Alarm Arn: arn:aws:cloudwatch:us-east-1:879381257906:alarm:FlowLogsalarm

Threshold:  
 - The alarm is in the ALARM state when the metric is GreaterThanOrEqualToThreshold 1.0 for at least 1 of the last 1 period(s) of 300 seconds.

Monitored Metric:  
 - MetricNamespace: SecureSphereVPCFlowLogsMonitoring  
 - MetricName: RejectedTraffic  
 - Dimensions:  
 - Period: 300 seconds  
 - Statistic: Sum

## 12. Documentation and Governance

All configurations, alerting workflows, and remediation procedures were documented.

This documentation supported audits, incident response, and knowledge transfer.

## 13. Outcomes and Impact

This implementation delivered the following improvements:

- Centralized threat detection
- Faster incident response
- Improved network visibility
- Automated remediation
- Reduced security misconfigurations
- Enhanced compliance monitoring

#### 14. Conclusion

This monitoring and threat detection framework provides comprehensive visibility into cloud security risks.

Through centralized logging, automated response, and continuous analysis, the environment supports secure and resilient operations.