Enterprise Automated Security Remediation and Server Hardening Framework on AWS

1. Problem Statement.

An online commerce platform experienced frequent configuration drift and delayed remediation of security findings. Manual intervention resulted in slow response times and increased exposure to misconfigurations. The organization required automated remediation workflows to reduce risk and improve security posture.

I implemented this framework to reduce manual security operations, prevent misconfigurations, and improve incident response time through automation and centralized monitoring.

The solution integrates detection, remediation, patching, and compliance enforcement into a unified security operations model.

2. Objectives

The primary objectives were to:

- Reduce manual security remediation efforts
- Automate response to common security incidents
- Establish hardened server baselines
- Improve patch management consistency
- Enforce baseline compliance
- Enhance monitoring and alerting
- Improve overall security maturity

3. Business and Security Context

The environment supported a rapidly scaling e-commerce platform relying heavily on EC2 and managed AWS services.

Rapid growth led to:

- Overly permissive security groups

- Outdated operating systems
- Delayed remediation
- Inconsistent server hardening

These risks increased exposure to compromise and operational disruption.

Automation was introduced to address these challenges.

4. Solution Architecture

The security automation platform was built using:

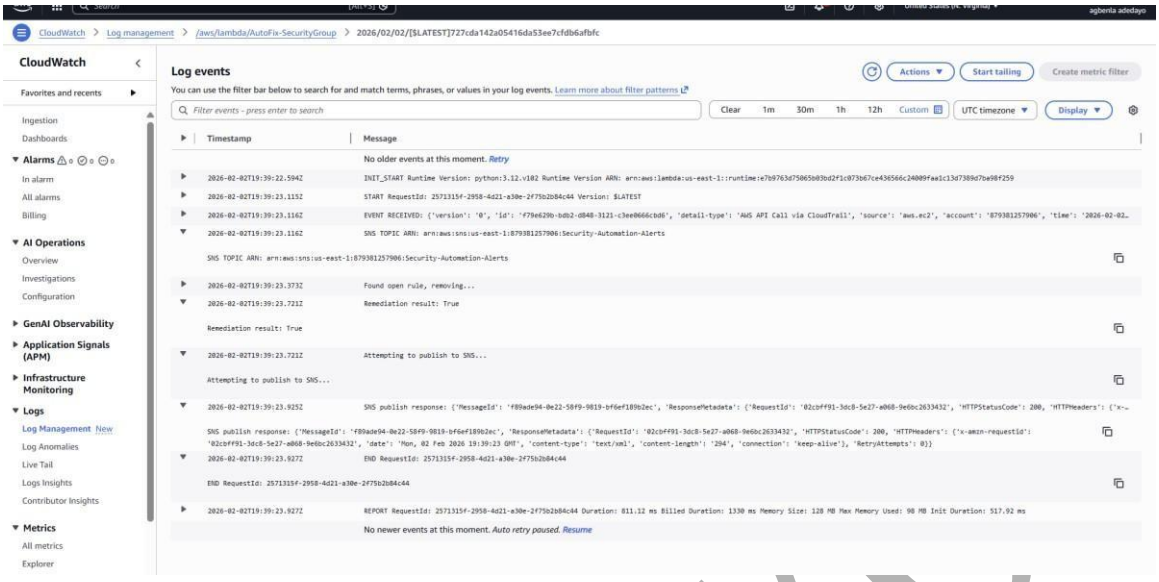
- Amazon GuardDuty
- Amazon EventBridge
- AWS Lambda
- Amazon SNS
- AWS Systems Manager
- AWS Config
- Hardened Amazon Machine Images (AMIs)

Security findings from GuardDuty and CloudTrail were routed through EventBridge to trigger automated remediation workflows.

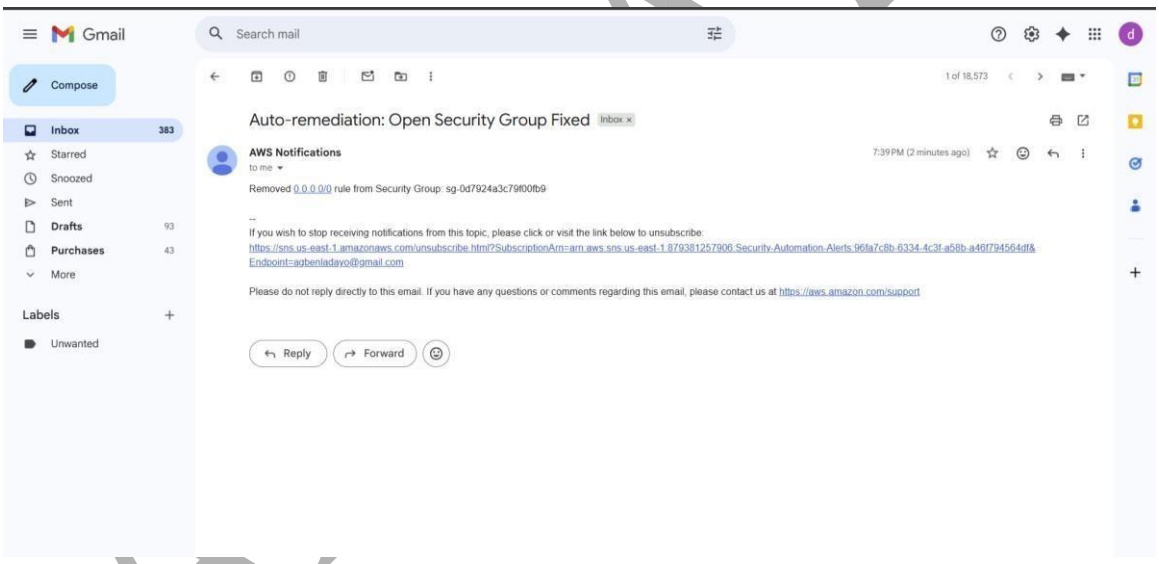
5. Automated Remediation Implementation

5.1 Security Group Remediation

A Lambda function monitored CloudTrail events related to security group changes.

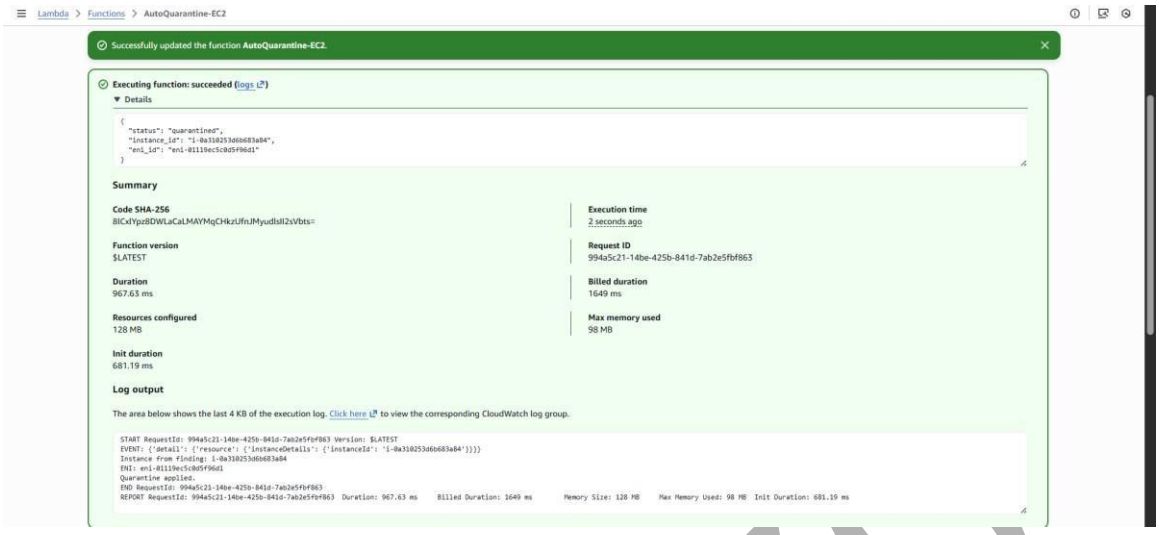


When unrestricted access (0.0.0.0/0) was detected, the rule was automatically removed and administrators were notified.

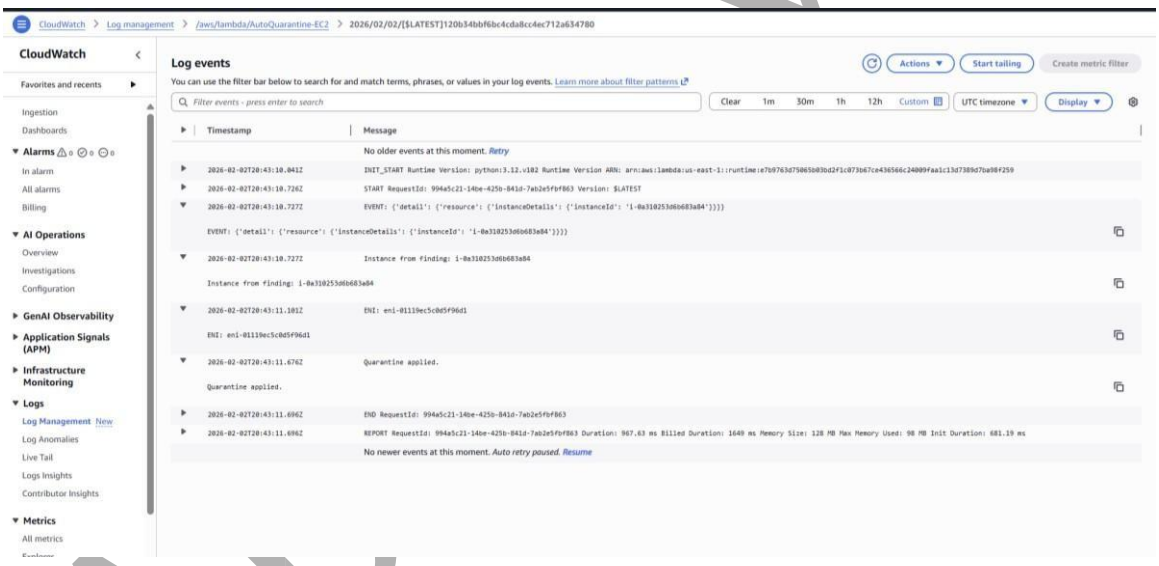


5.2 EC2 Quarantine Automation

A dedicated Lambda function processed GuardDuty findings related to compromised instances.



High-risk instances were isolated by attaching a restrictive quarantine security group.



This limited lateral movement and reduced impact.

5.3 Alerting and Logging

All remediation actions were logged in CloudWatch Logs.

SNS notifications were sent to security administrators for visibility and escalation.

6. Event Monitoring and Detection

EventBridge rules were configured to monitor:

- Unauthorized API calls
- Privilege escalation attempts - Security group modifications
- Malware indicators
- Network anomalies

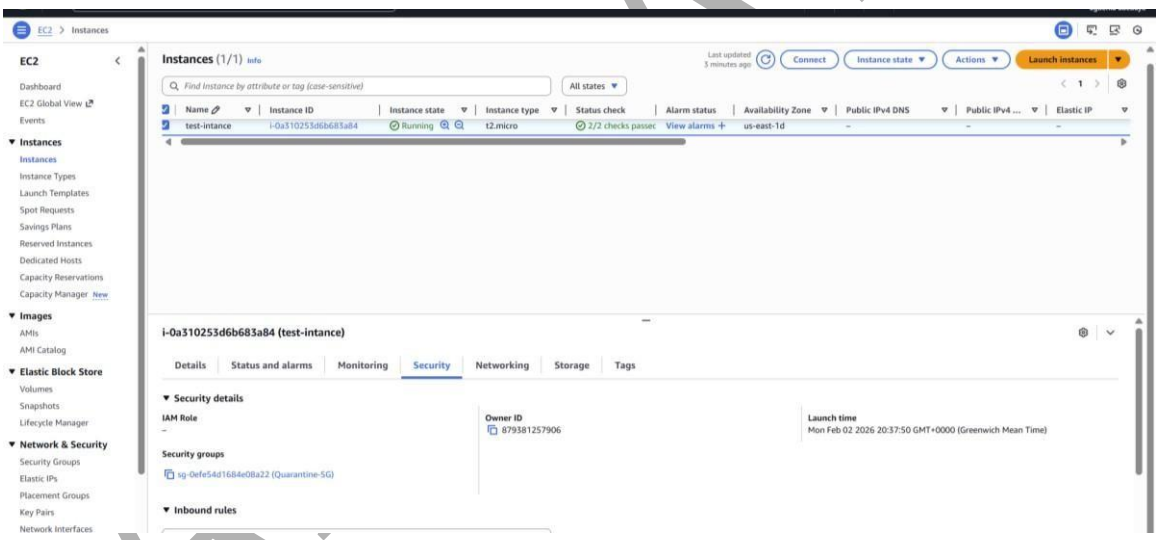
Each rule triggered automated actions or alerts.

CloudWatch metrics were monitored to validate automation reliability.

7. EC2 Hardening and Golden AMI Management

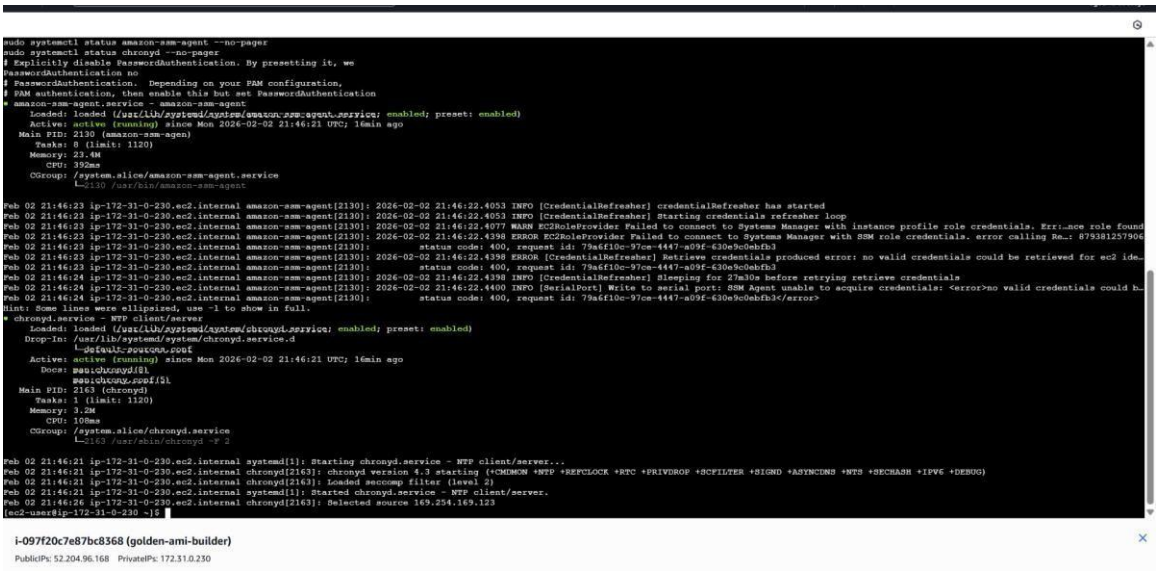
7.1 Baseline Hardening

A dedicated EC2 instance was used to build hardened baselines.



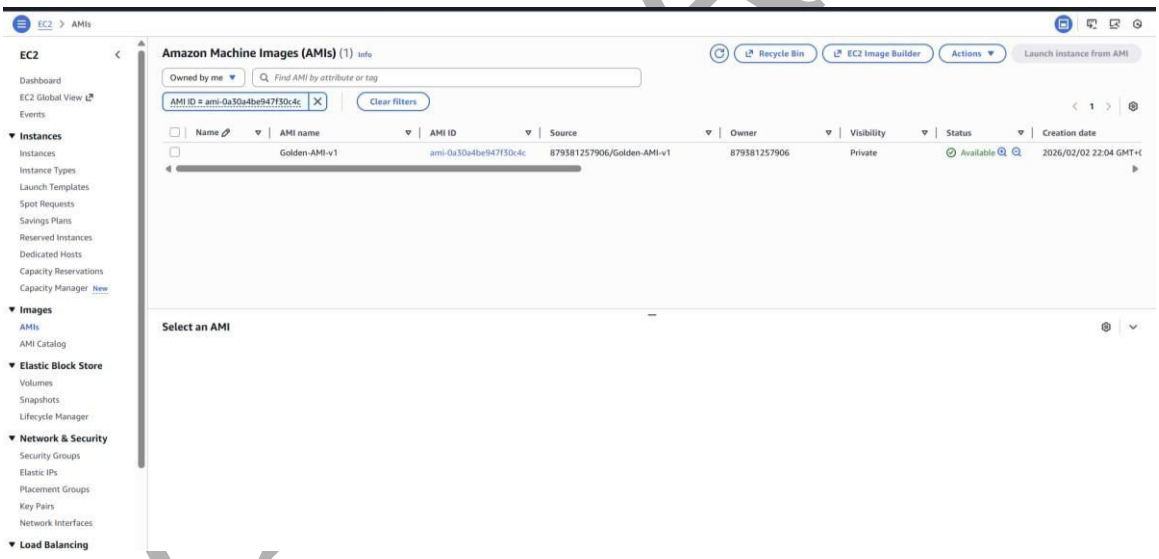
Hardening actions included:

- Installing latest security patches
- Configuring Systems Manager Agent
- Disabling password-based SSH
- Enabling time synchronization
- Attaching least-privilege IAM roles



7.2 Golden AMI Creation

After validation, hardened instances were converted into Golden AMIs.



These images became the standard deployment baseline.

7.3 Validation

New instances launched from Golden AMIs were validated to ensure security controls were preserved.

8. Automated Patch Management

AWS Systems Manager Patch Manager was configured to maintain operating system updates.

Patch baselines were defined to:

- Scan instances daily
- Apply critical patches weekly
- Reboot automatically when required

Compliance reports were reviewed regularly.

9. Baseline Compliance Enforcement

AWS Config was enabled for continuous configuration evaluation.

Key rules included:

- No public IPs on EC2 instances
- Mandatory EBS encryption
- Restricted SSH access
- Mandatory MFA for IAM users

Non-compliant resources were flagged and remediated where possible.

10. Testing and Validation Framework
The framework was validated through controlled simulations, including:

- GuardDuty sample findings
- Insecure security group creation
- EventBridge trigger testing
- Lambda execution review
- Quarantine verification
- Patch compliance testing
- AMI validation

These tests confirmed end-to-end automation effectiveness.

11. Challenges and Lessons Learned

Challenges encountered included:

- Private instance connectivity issues
- Missing IAM roles for SSM
- Sample findings using dummy IDs
- SNS permission errors

These issues were resolved through policy tuning, network adjustments, and service integration improvements.

They strengthened operational understanding.

12. Outcomes and Impact

This implementation delivered:

- Reduced manual security workload
- Faster incident containment
- Standardized hardened baselines
- Improved compliance visibility
- Enhanced monitoring coverage
- Reduced attack surface

Overall security maturity improved significantly.

13. Professional Impact

This engagement strengthened practical skills in:

- Security automation
- Incident response
- Infrastructure hardening - Compliance enforcement
- Cloud-native security operations

It demonstrates the ability to design, implement, and manage enterprise-scale security automation frameworks on AWS.