

Enterprise Automated Security Remediation and Server Hardening Framework on AWS

Author: Adedayo

Specialization: Cloud Security & Security Automation

Platform: Amazon Web Services (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.

The screenshot shows the AWS CloudWatch Log Management interface. The left sidebar includes sections for CloudWatch, Favorites and recents, Ingestion, Dashboards, Alarms, AI Operations, GenAI Observability, Application Signals (APM), Infrastructure Monitoring, Logs (Log Management New, Log Anomalies, Live Tail, Logs Insights, Contributor Insights), Metrics (All metrics, Explorer), and Metrics. The main pane displays 'Log events' with a timestamp filter set to '2026-02-02T19:39:23.594Z'. The logs show the following sequence of events:

- INIT_START Runtime Version: python:3.12-v102 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:re709765d75065bb03d2f3c073b67ce436566c24000fae3c13d7398d7be8f259
- START RequestId: 2571315f-2958-4d21-a30e-2f75b2bdc44 Version: \$LATEST
- EVENT RECEIVED: {"version": "0", "id": "F79ae6290-bd02-d848-3121-3ee8666cd86", "detail-type": "AWS API Call via CloudTrail", "source": "aws.ec2", "account": "879381257906", "time": "2026-02-02T19:39:23.116Z", "region": "us-east-1", "awsRegion": "us-east-1", "eventName": "SecurityAutomation-Alerts", "userIdentity": "arn:aws:sts::879381257906:SecurityAutomation-Alerts", "requestParameters": null}
- SNS TOPIC ARN: arn:aws:sns:us-east-1:879381257906:Security-Automation-Alerts
- Found open rule, removing...
- Remediation result: True
- Remediation result: True
- Attempting to publish to SNS...
- Attempting to publish to SNS...
- SNS publish response: {"MessageId": "#09ade94-0e12-5bf9-9819-bf6ef189b2c", "ResponseMetadata": {"RequestId": "02cf0f91-3ec8-5e27-e008-9e0c2633432", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "02cf0f91-3ec8-5e27-e008-9e0c2633432", "date": "Mon, 02 Feb 2026 19:39:23 GMT", "content-type": "text/xml", "content-length": "204", "connection": "keep-alive", "retry-attempts": "0"}}, END RequestId: 2571315f-2958-4d21-a30e-2f75b2bdc44
- REPORT RequestId: 2571315f-2958-4d21-a30e-2f75b2bdc44 Duration: 811.12 ms Billed Duration: 1330 ms Memory Size: 128 MB Max Memory Used: 98 MB Init Duration: 517.92 ms

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

The screenshot shows a Gmail inbox with 383 unread messages. The email in focus is from 'AWS Notifications' to 'me' at '7:39PM (2 minutes ago)'. The subject is 'Auto-remediation: Open Security Group Fixed'. The email body contains the following text:

Removed 0.0.0.0/0 rule from Security Group: sg-0d7924a3c79f00fb9

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:879381257906:Security-Automation-Alerts.96fa7c0b-0334-4c3f-a50b-a40f794564df&EndPointArn=auto:lambda:879381257906:Security-Automation-Alerts@gmail.com>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>.

5.2 EC2 Quarantine Automation

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

The screenshot shows the AWS Lambda function AutoQuarantine-EC2 details page. A green banner at the top indicates "Successfully updated the function AutoQuarantine-EC2". Below this, the "Executing function: succeeded" status is shown with a link to logs. The "Details" section contains a JSON object with fields: "status": "quarantined", "instance_id": "i-0a310253d0b683a84", and "encl_id": "enl-01119ec5cb05f96d1". The "Summary" section provides execution statistics: Code SHA-256 (BICXHpxzDWAicCaIcMAYMqCChkzUfnJMyudlsI2iVbts=), Function version (\$LATEST), Duration (967.63 ms), Resources configured (128 MB), Init duration (681.19 ms), and Log output. The log output shows the execution of the Lambda function, including the start and end requests, instance details, and quarantine applied.

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

The screenshot shows the CloudWatch Log management interface for the /aws/lambda/AutoQuarantine-EC2 log stream. The left sidebar includes sections for CloudWatch, Log management, Favorites and recents, Ingestion, Dashboards, Alarms, AI Operations, GenAI Observability, Application Signals (APM), Infrastructure Monitoring, Logs (Log Management New, Log Anomalies, Live Tail, Logs Insights, Contributor Insights), and Metrics. The main area displays log events with a timestamp filter from 2026-02-02T20:43:10.841Z to 2026-02-02T20:43:11.696Z. The log entries show the Lambda function starting, finding an instance, applying quarantine, and reporting the result. The interface includes filters, actions, and metrics creation options.

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

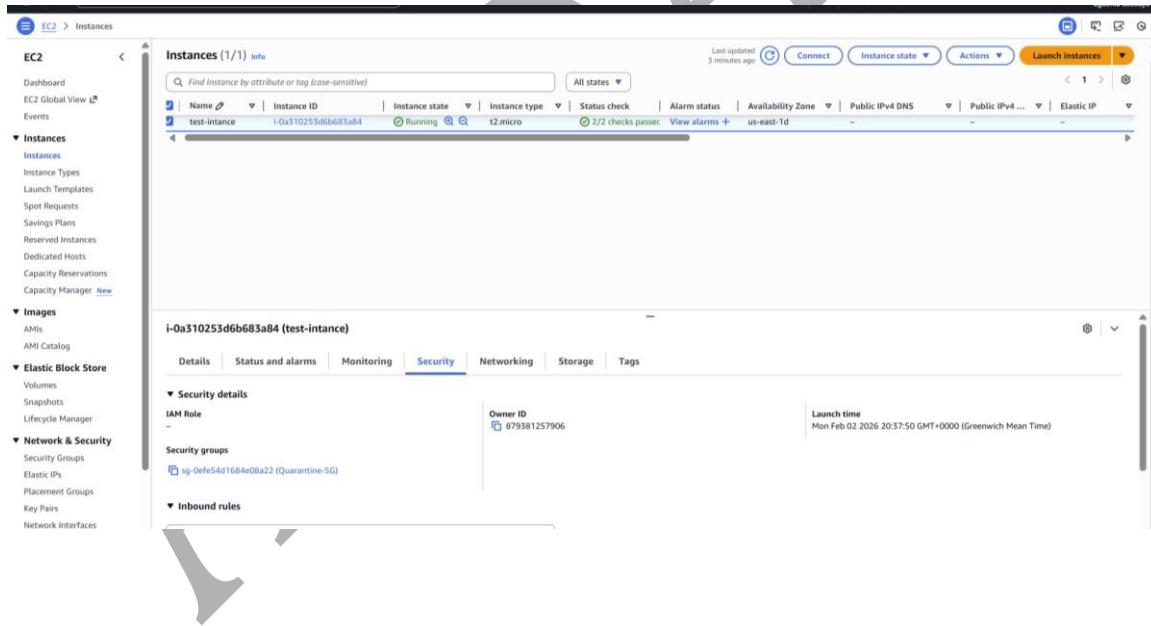
Each rule triggered automated actions or alerts.

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

```
sudo systemctl status amazon-smm-agent --no-pager
# This system does not have password authentication. By presetting it, we
# PasswordAuthentication no
# PAM authentication, etc., enable this but set PasswordAuthentication
# AmazonSSMRoleProvider
# Loadeds loaded (/usr/lib/systemd/system/amazon-smm-agent.service; enabled; preset: enabled)
# Active: active (running) since Mon 2026-02-02 21:46:21 UTC; 1min ago
# Main PID: 1030 (amazon-smm-agent)
# Tasks: 0 (limit: 1120)
# Memory: 23.4M
# CPU: 39ms
# CGroup: /system.slice/amazon-smm-agent.service
#           └─ 1030 /var/bin/amazon-smm-agent

Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4053 INFO [CredentialRefresh] credentialRefresher has started
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4053 INFO [CredentialRefresh] Starting credentials refresh loop
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4077 WARN EC2RoleProvider Failed to connect to Systems Manager with instance profile role credentials. Err:None role found
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4398 ERROR EC2RoleProvider Failed to connect to Systems Manager with instance profile role credentials. error calling Re..: 879381257906
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4398 ERROR EC2RoleProvider Failed to connect to Systems Manager with instance profile role credentials. produced error: no valid credentials could be retrieved for ec2 id...
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4398 status code: 400, request id: 79a6f10c-97ce-4447-a09f-630e9c0abfb3
Feb 02 21:46:24 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4398 INFO [CredentialRefresh] Sleeping for 270s before retrying retrieve credentials
Feb 02 21:46:24 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4398 INFO [CredentialRefresh] Failed to retrieve credentials from instance profile role Agent unable to acquire credentials: <error>no valid credentials could b...
Feb 02 21:46:24 ip-172-31-0-230.ec2.internal amazon-smm-agent[2130]: 2026-02-02 21:46:22.4398 status code: 400, request id: 79a6f10c-97ce-4447-a09f-630e9c0abfb3</error>
Hint: Some lines were ellipsized, use -l to show in full.
* chronyd loaded (/usr/lib/systemd/system/chronyd.service; enabled; preset: enabled)
Drop-In: /usr/lib/systemd/system/chronyd.service.d
  _default.conf
  _default.conf.d
  Active: active (running) since Mon 2026-02-02 21:46:21 UTC; 1min ago
    Docs: man:chrony(8),
          man:chrony.conf(5)
  Main PID: 2130 (chronyd)
  Tasks: 1 (limit: 1120)
  Memory: 3.2MiB
  CPU: 10ms
  CGroup: /system.slice/chronyd.service
          └─ 2130 /usr/sbin/chronyd -f -2

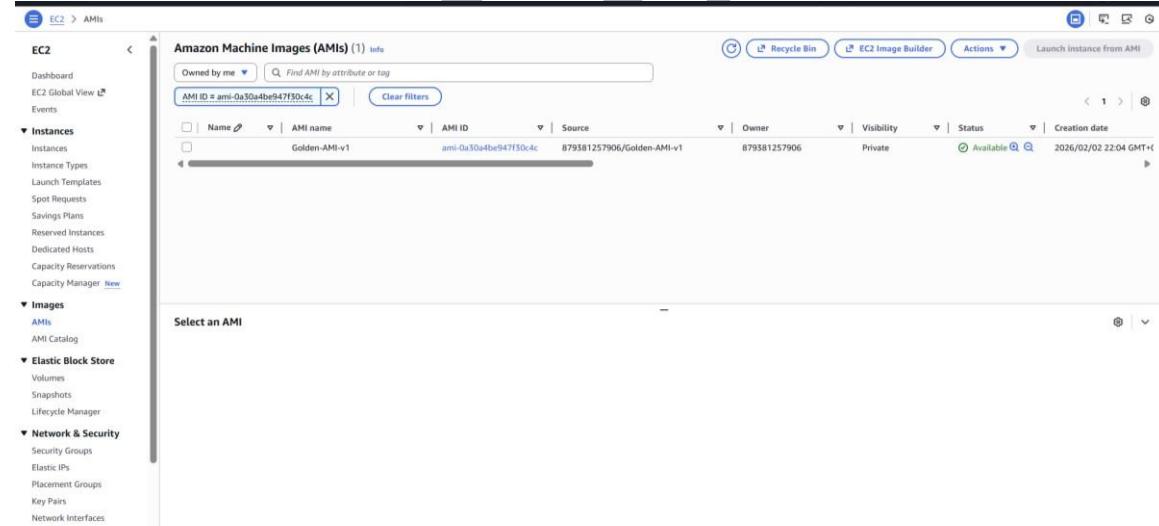
Feb 02 21:46:25 ip-172-31-0-230.ec2.internal systemd[1]: Starting chronyd.service - NTP client/server...
Feb 02 21:46:25 ip-172-31-0-230.ec2.internal chronyd[2163]: chronyd version 4.3 starting (+CMON +NTP +REFCLOCK +RTC +PRIVDROP +SFILTER +SIGN +ASYNCNS +NTS +GCHASH +IPV6 +DEBUG)
Feb 02 21:46:25 ip-172-31-0-230.ec2.internal chronyd[2163]: Loaded seccomp filter (level 2)
Feb 02 21:46:25 ip-172-31-0-230.ec2.internal systemd[1]: Started chronyd.service - NTP client/server.
Feb 02 21:46:26 ip-172-31-0-230.ec2.internal chronyd[2163]: Selected source 169.254.169.123
[ec2-user@ip-172-31-0-230 ~]$
```

i-09f720c7e87bc8368 (golden-ami-builder)

PublicIPs: 52.204.96.168 PrivateIPs: 172.31.0.230

7.2 Golden AMI Creation

After validation, hardened instances were converted into Golden AMIs.



Name	AMI name	AMI ID	Source	Owner	Visibility	Status	Creation date
Golden-AMI-v1	ami-03a04be947f30c4c	879381257906/Golden-AMI-v1	879381257906	Private	Available	2026/02/02 22:04:00 GMT+0	

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.

ADEDAYO