Enterprise Automated Security Remediation and Server Hardening Framework on AWS

Author: Adedayo
Specialization: Cloud Security & Security Automation
Platform: Amazon Web Services (AWS)

1. Introduction

This document describes the design, implementation, and validation of an automated security remediation, monitoring, and server hardening framework in AWS.

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:

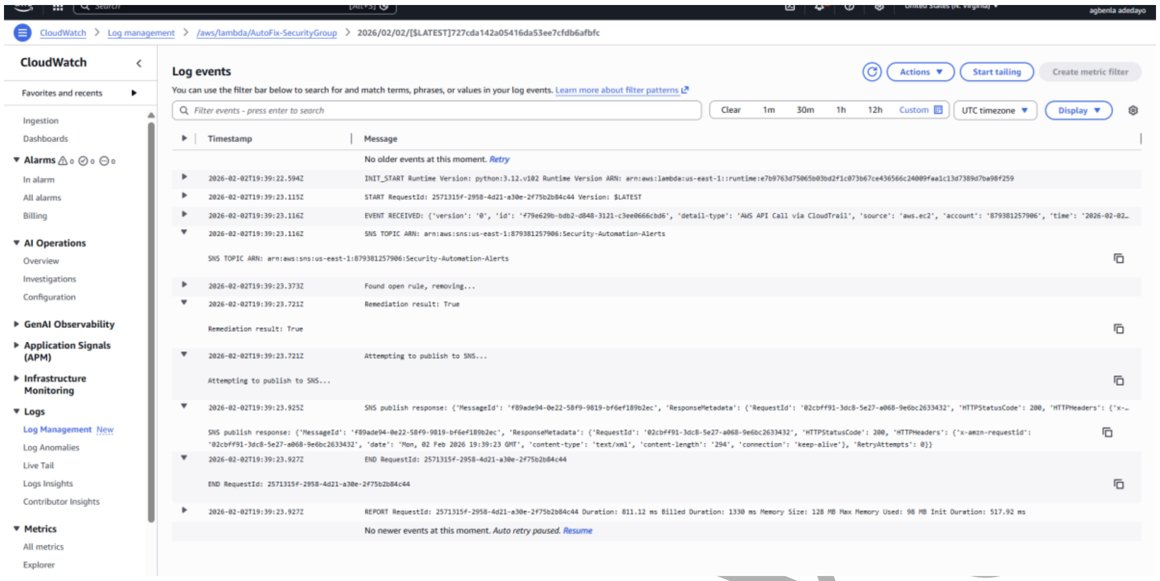
- Amazon GuardDuty
- Amazon EventBridge
- AWS Lambda
- Amazon SNS
- AWS Systems Manager
- AWS Config
- Hardened Amazon Machine Images (AMIs)

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

5. Automated Remediation Implementation

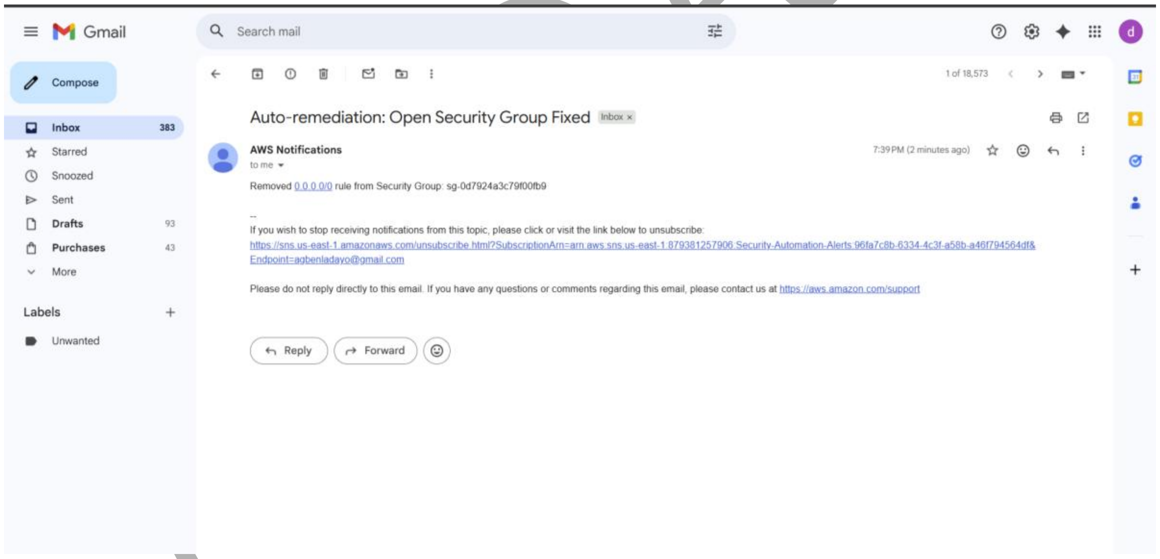
5.1 Security Group Remediation

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


CloudWatch > Log management > /aws/lambda/AutoFix-SecurityGroup > 2026/02/02/[$LATEST]727cda142a05416da53ee7cfdb6afbfc
Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Actions
Start tailing
Create metric filter
Clear 1m 30m 1h 12h Custom
UTC timezone
Display

```
Timestamp                   Message
                            No older events at this moment. Retry
2026-02-02T19:39:22.594Z    INIT_START Runtime Version: python:3.12.v102 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:e7b9763d75065b03bd2f1c073b67ce436566c24009faa1c13d7389d7ba98f259
2026-02-02T19:39:23.115Z    START RequestId: 2571315f-2958-4d21-a30e-2f75b2b84c44 Version: $LATEST
2026-02-02T19:39:23.116Z    EVENT RECEIVED: {'version': '0', 'id': 'f79e629b-bdb2-d848-3121-c3ee0666cbd6', 'detail-type': 'AWS API Call via CloudTrail', 'source': 'aws.ec2', 'account': '879381257906', 'time': '2026-02-02…
2026-02-02T19:39:23.116Z    SNS TOPIC ARN: arn:aws:sns:us-east-1:879381257906:Security-Automation-Alerts
SNS TOPIC ARN: arn:aws:sns:us-east-1:879381257906:Security-Automation-Alerts
2026-02-02T19:39:23.373Z    Found open rule, removing...
2026-02-02T19:39:23.721Z    Remediation result: True
Remediation result: True
2026-02-02T19:39:23.721Z    Attempting to publish to SNS...
Attempting to publish to SNS...
2026-02-02T19:39:23.925Z    SNS publish response: {'MessageId': 'f89ade94-0e22-58f9-9819-bf6ef189b2ec', 'ResponseMetadata': {'RequestId': '02cbff91-3dc8-5e27-a068-9e6bc2633432', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-…
SNS publish response: {'MessageId': 'f89ade94-0e22-58f9-9819-bf6ef189b2ec', 'ResponseMetadata': {'RequestId': '02cbff91-3dc8-5e27-a068-9e6bc2633432', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid':
'02cbff91-3dc8-5e27-a068-9e6bc2633432', 'date': 'Mon, 02 Feb 2026 19:39:23 GMT', 'content-type': 'text/xml', 'content-length': '294', 'connection': 'keep-alive'}, 'RetryAttempts': 0}}
2026-02-02T19:39:23.927Z    END RequestId: 2571315f-2958-4d21-a30e-2f75b2b84c44
END RequestId: 2571315f-2958-4d21-a30e-2f75b2b84c44
2026-02-02T19:39:23.927Z    REPORT RequestId: 2571315f-2958-4d21-a30e-2f75b2b84c44 Duration: 811.12 ms Billed Duration: 1330 ms Memory Size: 128 MB Max Memory Used: 98 MB Init Duration: 517.92 ms
                            No newer events at this moment. Auto retry paused. Resume
```

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



Auto-remediation: Open Security Group Fixed Inbox ×

AWS Notifications 7:39 PM (2 minutes ago)
to me

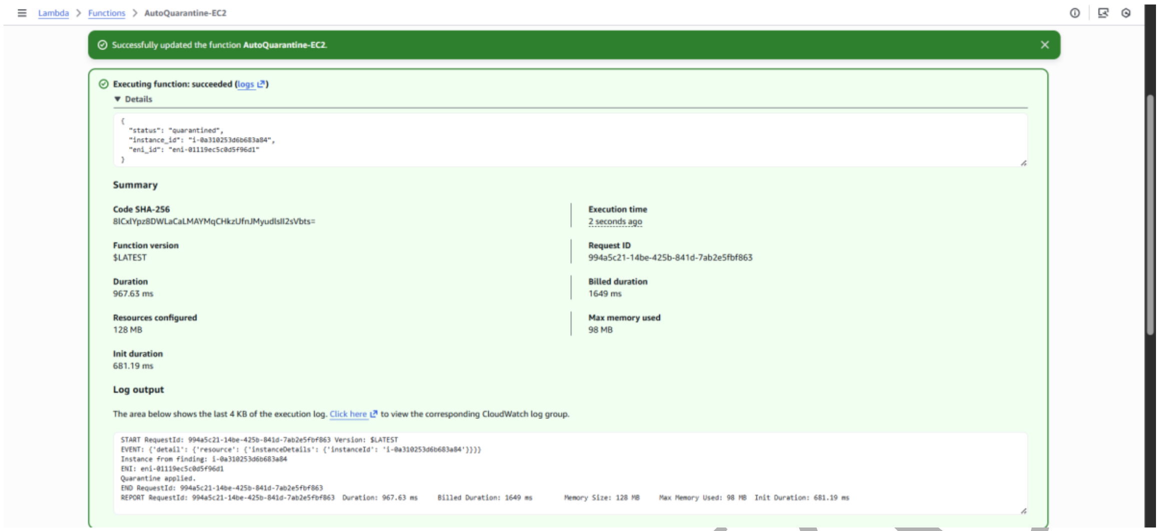
Removed 0.0.0.0/0 rule from Security Group: sg-0d7924a3c79f00fb9

--
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:96fa7c8b-6334-4c3f-a58b-a46f794564df&Endpoint=agbenladayo@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.


Lambda > Functions > AutoQuarantine-EC2
Successfully updated the function AutoQuarantine-EC2.
Executing function: succeeded (logs)
Details

```
{
  "status": "quarantined",
  "instance_id": "i-0a310253d6b683a84",
  "eni_id": "eni-01119ec5c0d5f96d1"
}
```

Summary

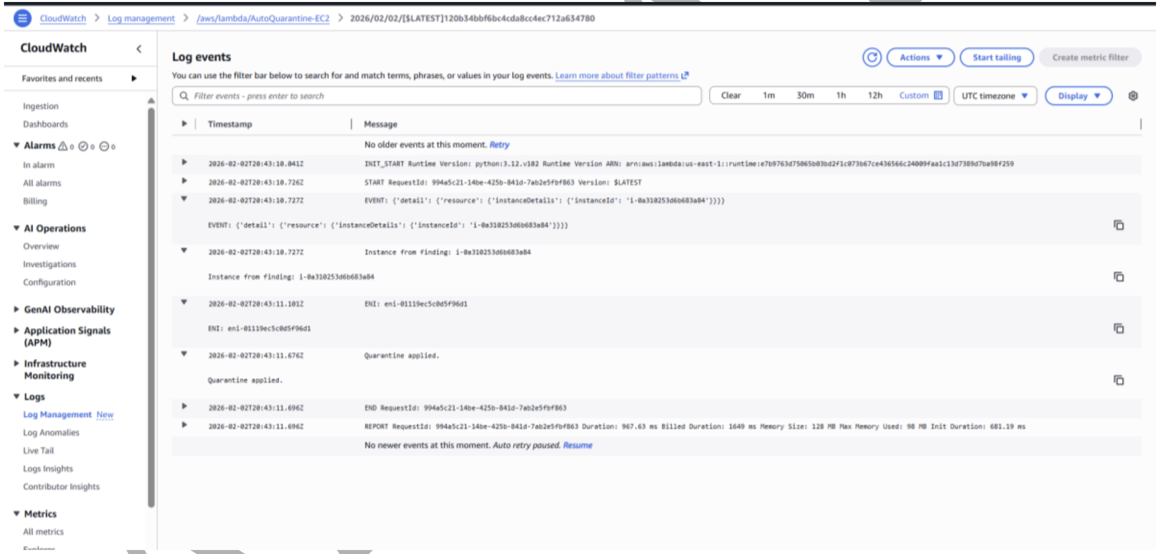
Code SHA-256 8lCxlYpz8DWLaCaLMAYMqCHkzUfnJMyudlsll2sVbts=	Execution time 2 seconds ago
Function version $LATEST	Request ID 994a5c21-14be-425b-841d-7ab2e5fbf863
Duration 967.63 ms	Billed duration 1649 ms
Resources configured 128 MB	Max memory used 98 MB
Init duration 681.19 ms	

Log output

The area below shows the last 4 KB of the execution log. Click here to view the corresponding CloudWatch log group.

```
START RequestId: 994a5c21-14be-425b-841d-7ab2e5fbf863 Version: $LATEST
EVENT: {'detail': {'resource': {'instanceDetails': {'instanceId': 'i-0a310253d6b683a84'}}}}
Instance from finding: i-0a310253d6b683a84
ENI: eni-01119ec5c0d5f96d1
Quarantine applied.
END RequestId: 994a5c21-14be-425b-841d-7ab2e5fbf863
REPORT RequestId: 994a5c21-14be-425b-841d-7ab2e5fbf863  Duration: 967.63 ms    Billed Duration: 1649 ms      Memory Size: 128 MB    Max Memory Used: 98 MB  Init Duration: 681.19 ms
```

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



Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                 Message
                          No older events at this moment. Retry
2026-02-02T20:43:10.041Z  INIT_START Runtime Version: python:3.12.v102 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:e7b9763d75065b03bd2f1c073b67ce436566c24009faa1c13d7389d7ba98f259
2026-02-02T20:43:10.726Z  START RequestId: 994a5c21-14be-425b-841d-7ab2e5fbf863 Version: $LATEST
2026-02-02T20:43:10.727Z  EVENT: {'detail': {'resource': {'instanceDetails': {'instanceId': 'i-0a310253d6b683a84'}}}}
    EVENT: {'detail': {'resource': {'instanceDetails': {'instanceId': 'i-0a310253d6b683a84'}}}}
2026-02-02T20:43:10.727Z  Instance from finding: i-0a310253d6b683a84
    Instance from finding: i-0a310253d6b683a84
2026-02-02T20:43:11.101Z  ENI: eni-01119ec5c0d5f96d1
    ENI: eni-01119ec5c0d5f96d1
2026-02-02T20:43:11.676Z  Quarantine applied.
    Quarantine applied.
2026-02-02T20:43:11.696Z  END RequestId: 994a5c21-14be-425b-841d-7ab2e5fbf863
2026-02-02T20:43:11.696Z  REPORT RequestId: 994a5c21-14be-425b-841d-7ab2e5fbf863 Duration: 967.63 ms Billed Duration: 1649 ms Memory Size: 128 MB Max Memory Used: 98 MB Init Duration: 681.19 ms
                          No newer events at this moment. Auto retry paused. Resume
```

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

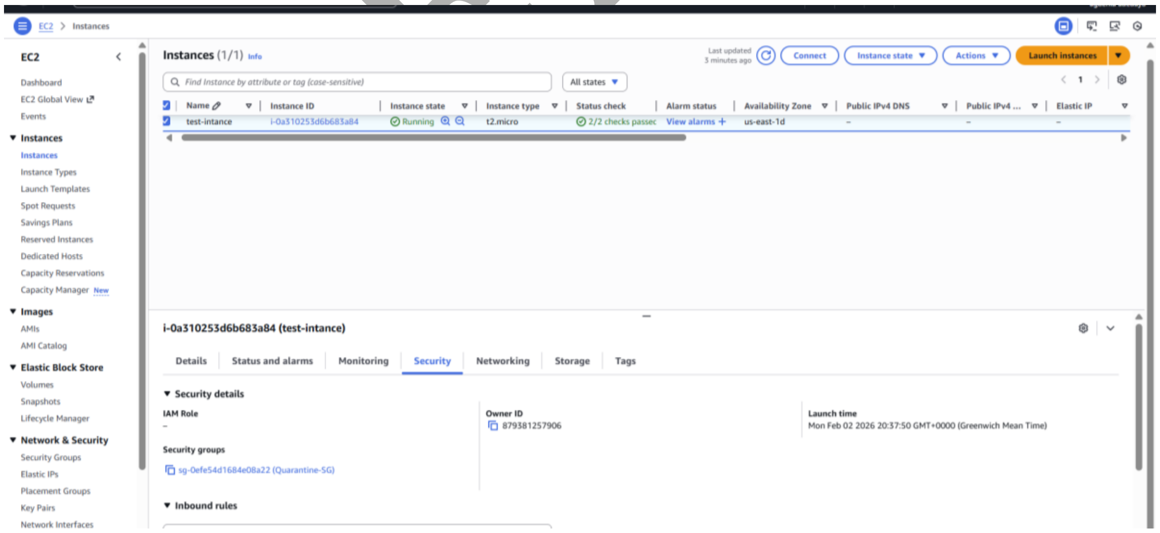
Each rule triggered automated actions or alerts.

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.



Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
test-intance	i-0a310253d6b683a84	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1d	–	–	–

i-0a310253d6b683a84 (test-intance)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Security details

IAM Role
–

Owner ID
879381257906

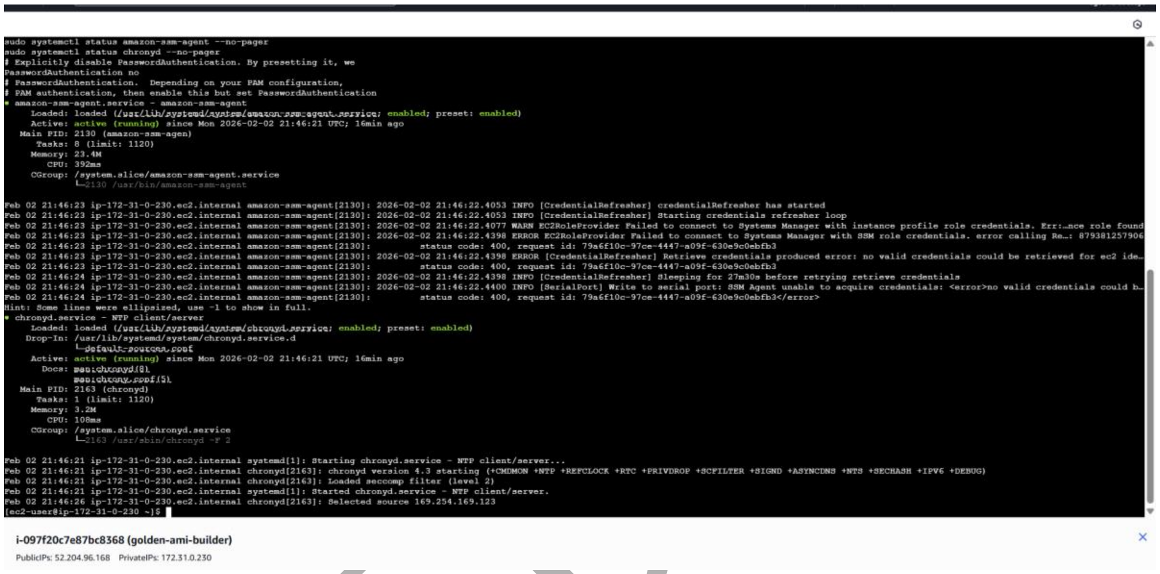
Launch time
Mon Feb 02 2026 20:37:50 GMT+0000 (Greenwich Mean Time)

Security groups
sg-0efe54d1684e08a22 (Quarantine-SG)

Inbound rules

Hardening actions included:

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

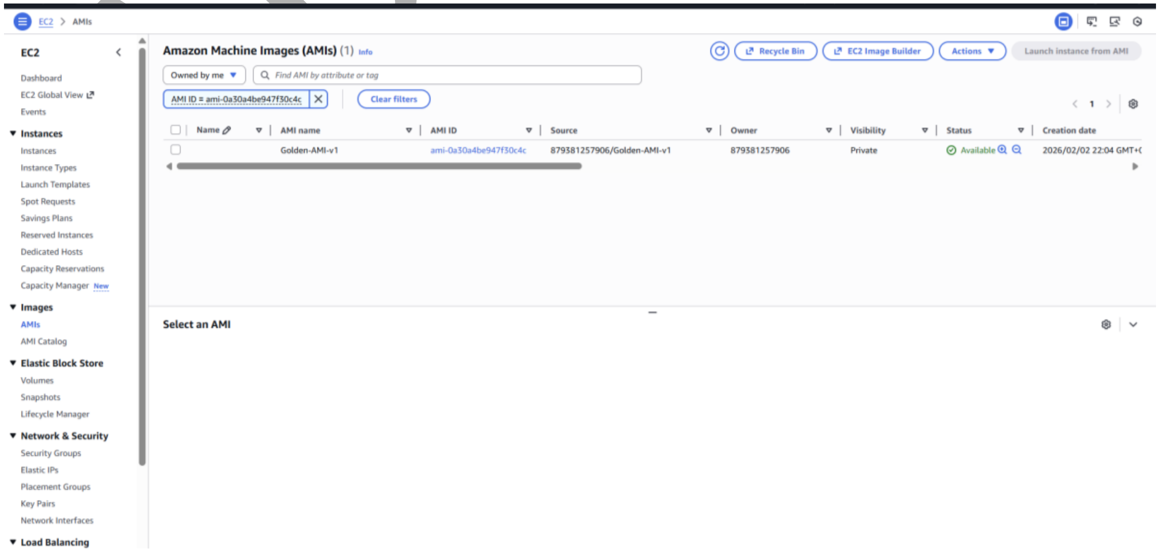


```
sudo systemctl status amazon-ssm-agent --no-pager
sudo systemctl status chronyd --no-pager
# Explicitly disable PasswordAuthentication. By presetting it, we
PasswordAuthentication no
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication, then enable this but set PasswordAuthentication
● amazon-ssm-agent.service - amazon-ssm-agent
     Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-02-02 21:46:21 UTC; 16min ago
   Main PID: 2130 (amazon-ssm-agen)
      Tasks: 8 (limit: 1120)
     Memory: 23.4M
        CPU: 392ms
     CGroup: /system.slice/amazon-ssm-agent.service
             └─2130 /usr/bin/amazon-ssm-agent

Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4053 INFO [CredentialRefresher] credentialRefresher has started
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4053 INFO [CredentialRefresher] Starting credentials refresher loop
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4077 WARN EC2RoleProvider Failed to connect to Systems Manager with instance profile role credentials. Err:…nce role found
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4398 ERROR EC2RoleProvider Failed to connect to Systems Manager with SSM role credentials. error calling Re…: 879381257906
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]:         status code: 400, request id: 79a6f10c-97ce-4447-a09f-630e9c0ebfb3
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4398 ERROR [CredentialRefresher] Retrieve credentials produced error: no valid credentials could be retrieved for ec2 ide…
Feb 02 21:46:23 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]:         status code: 400, request id: 79a6f10c-97ce-4447-a09f-630e9c0ebfb3
Feb 02 21:46:24 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4398 INFO [CredentialRefresher] Sleeping for 27m30s before retrying retrieve credentials
Feb 02 21:46:24 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]: 2026-02-02 21:46:22.4400 INFO [SerialPort] Write to serial port: SSM Agent unable to acquire credentials: <error>no valid credentials could b…
Feb 02 21:46:24 ip-172-31-0-230.ec2.internal amazon-ssm-agent[2130]:         status code: 400, request id: 79a6f10c-97ce-4447-a09f-630e9c0ebfb3</error>
Hint: Some lines were ellipsized, use -l to show in full.
● chronyd.service - NTP client/server
     Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/chronyd.service.d
             └─default-sources.conf
     Active: active (running) since Mon 2026-02-02 21:46:21 UTC; 16min ago
       Docs: man:chronyd(8)
             man:chrony.conf(5)
   Main PID: 2163 (chronyd)
      Tasks: 1 (limit: 1120)
     Memory: 3.2M
        CPU: 108ms
     CGroup: /system.slice/chronyd.service
             └─2163 /usr/sbin/chronyd -F 2

Feb 02 21:46:21 ip-172-31-0-230.ec2.internal systemd[1]: Starting chronyd.service - NTP client/server...
Feb 02 21:46:21 ip-172-31-0-230.ec2.internal chronyd[2163]: chronyd version 4.3 starting (+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS +SECHASH +IPV6 +DEBUG)
Feb 02 21:46:21 ip-172-31-0-230.ec2.internal chronyd[2163]: Loaded seccomp filter (level 2)
Feb 02 21:46:21 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-097f20c7e87bc8368 (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.



Amazon Machine Images (AMIs) (1) Info

AMI ID = ami-0a30a4be947f30c4c

Name	AMI name	AMI ID	Source	Owner	Visibility	Status	Creation date
	Golden-AMI-v1	ami-0a30a4be947f30c4c	879381257906/Golden-AMI-v1	879381257906	Private	Available	2026/02/02 22:04 GMT+(

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.