

CLOUD SERVER HARDENING



Date: June 2025

Prepared By: Sahithi Gudigopuram

Table of Contents

S.No	Title	Page Number
1.	Introduction	03
2.	Architecture Overview	03
2.1	Infrastructure Components	03
2.2	Network Architecture	04
3.	Environment Setup	05
3.1	EC2 Instance Setup	05
3.2	Harden SSH and Network Access	06
3.3	AWS CLI Installation on Ubuntu EC2	07
3.4	IAM Access Key Creation for AWS CLI Configuration	07
4.	IAM Role Creation & Trust Policy Setup	10
4.1	Testing System Manager	11
5.	Logging & Monitoring Configuration	13
5.1	AWS CloudTrail Setup	13
5.2	Amazon GuardDuty Activation	14
5.3	Set up alerts for unauthorised access attempts	15
6.	Vulnerability Scanning & Findings	17
6.1	CVE-2025-4517 – Arbitrary File Write via Path Traversal	17
6.2	CVE-2025-4435- Bypass of Filtering via Errorlevel Override	18
6.3	CVE-2025-4138 – Symlink Attack via Filter Bypass	19
6.4	Alarm Response Validation & System Monitoring	20
6.5	Alarm Validation & Agent Activity Logs	21
7.	Key Observations	21
8.	Conclusion	22

1. Introduction

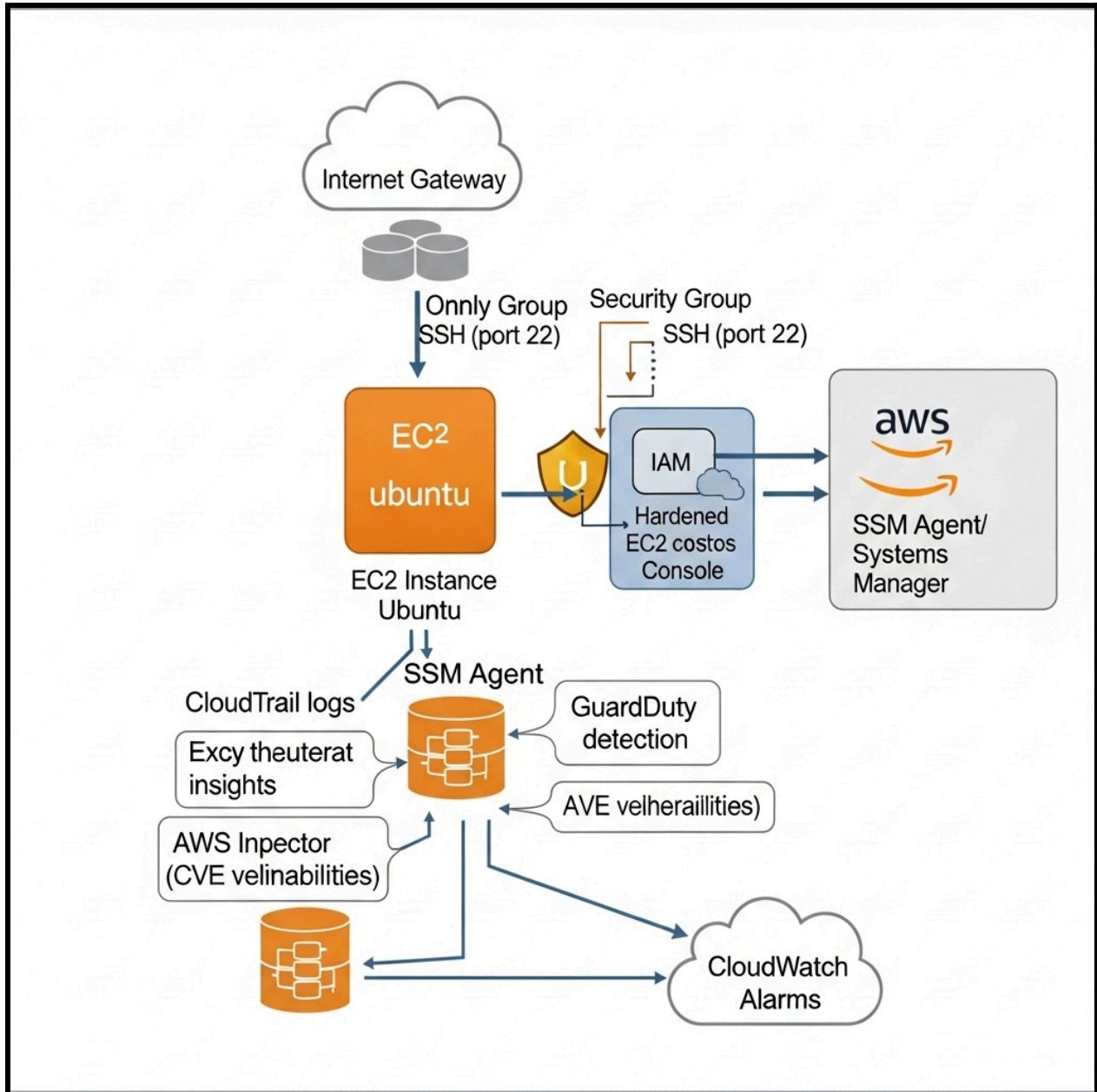
This project focuses on securing a cloud-based Ubuntu EC2 instance using various AWS-native tools and best practices. The primary goal is to identify vulnerabilities, monitor unauthorised access, and implement proactive security measures to harden the environment. Through tools like AWS Inspector, CloudTrail, and CloudWatch, the system was audited, monitored, and reinforced to ensure compliance and reduce attack surfaces. This report documents the process, findings, and improvements made during the security enhancement of the cloud infrastructure.

2. Architecture Overview

2.1. Infrastructure Components

- **EC2 Instance:** Ubuntu 24.04 LTS (t2.micro) in public subnet
- **Subnet:** Public subnet with access to the internet via the Internet Gateway
- **Security Groups:** Initially open SSH (port 22) to 0.0.0.0/0; later restricted
- **IAM Role:** EC2 instance profile with SSM permissions
- **SSM Agent:** Installed and used for secure agent-based access
- **S3:** Used for Inspector logging or storing snapshot/report data
- **CloudTrail:** Captures all management-level API activity across the AWS account
- **CloudWatch:** Custom metric filter and alarm (e.g. Spam) tied to unauthorised API activity
- **AWS Inspector:** Performs vulnerability assessments on EC2 instances.

2.2 Network Architecture

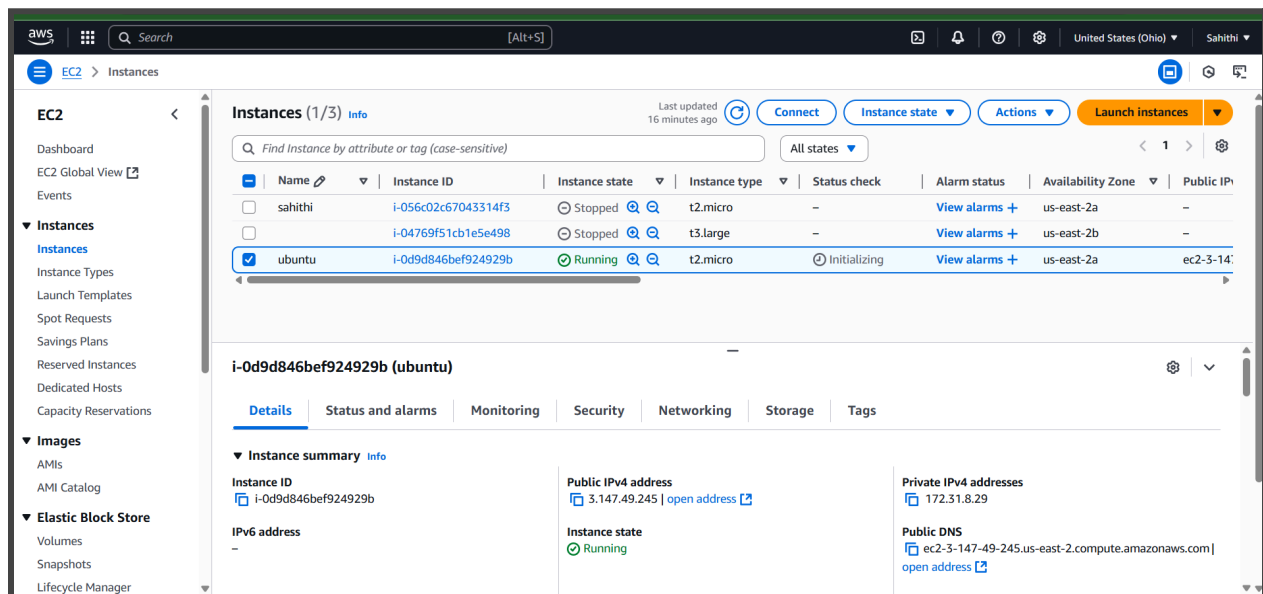


3. Environment Setup

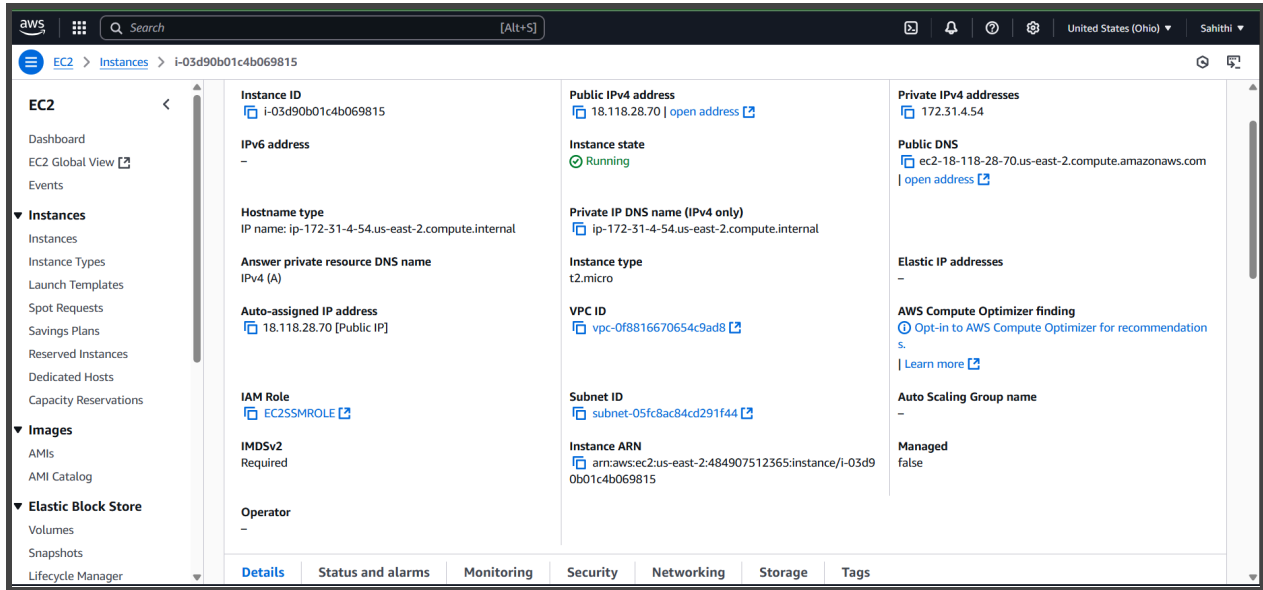
- **Platform:** AWS EC2 Ubuntu 24.04 LTS
- **Instance ID:** i-03d90b01c4b069815
- **Security Tools Used:**
 - AWS Inspector for vulnerability scanning
 - CloudTrail for activity logging
 - CloudWatch for real-time alerts
 - Systems Manager (SSM) for agent-based monitoring

3.1. EC2 Instance Setup

- Launched Ubuntu 24.04 LTS instance

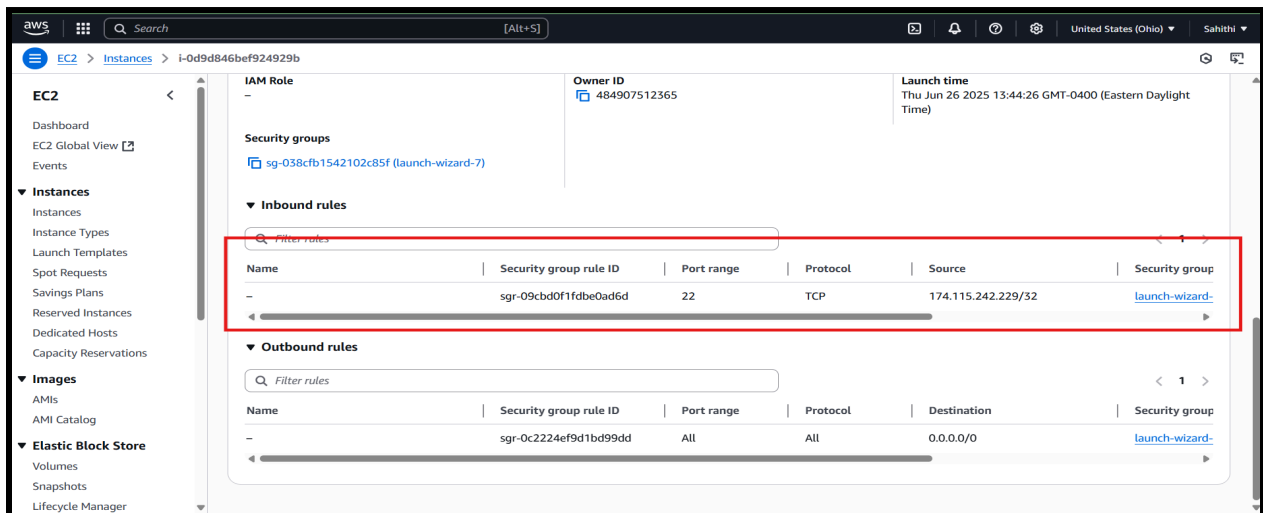


- Instance ID, region, subnet, and IP details



3.2.Harden SSH and Network Access

- Step 1: SSH configuration modification, edit the config file “ sudo nano /etc/ssh/sshd_config”
- Step 2: Applied these changes
PermitRootLogin no — Disables root login over SSH
PasswordAuthentication no — Enforces key-based authentication only
- Step 3: Restart the SSH server and check the applied changes “ sudo systemctl restart ssh”
- Step 4: Updated the EC2 **Security Group** to allow inbound SSH (port 22) access **only** from the public IP: 174.115.242.229



3.3 AWS CLI Installation on Ubuntu EC2

- Due to SSH limitations from the Ubuntu machine, AWS CLI operations were performed via a Windows terminal. A secondary CLI installation was also attempted directly on the Ubuntu EC2 instance.
- Initial step: `sudo apt update`
`sudo apt install awscli -y`
- Step 1: Download the AWS CLI bundle
`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
- Step 2: Unzip the package `"unzip awscliv2.zip"`
- Step 3: Run the installer `"sudo ./aws/install"`
- Step 4: Confirm it's working: `"aws --version"`
- Step 5: To unzip, I used this command: `"sudo apt install unzip -y"`
- Step 6: Reunzip the AWS CLI download: `unzip awscliv2.zip"`
- Step 7: Now run the installer `"sudo ./aws/install"`
- Step 8: Check the AWS version `"aws --version"`

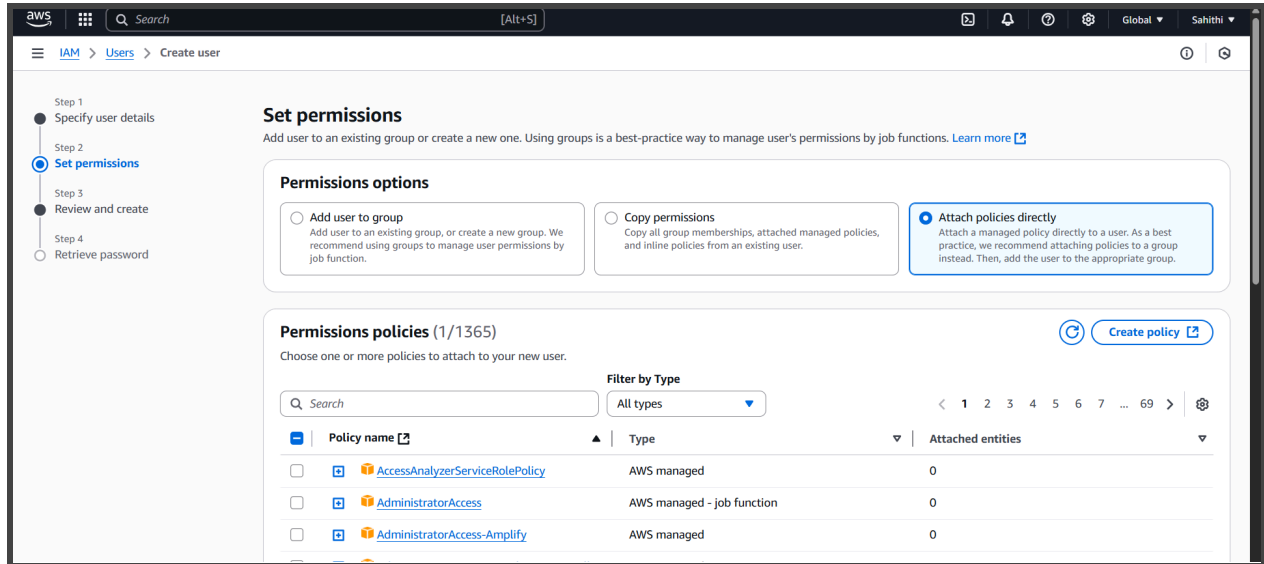
3.4. IAM Access Key Creation for AWS CLI Configuration

- Navigate to IAM Console
- From the AWS Console, search and open IAM
- In the sidebar, select Users → Create User
- Assigned a meaningful name

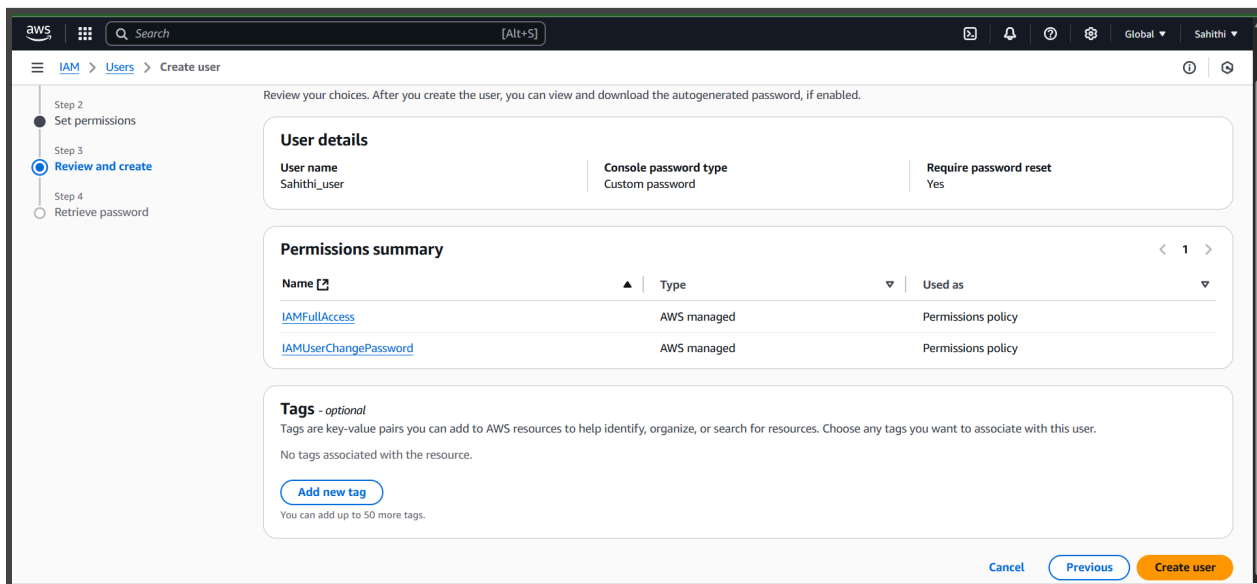
The screenshot shows the AWS IAM 'Create user' console. The left sidebar indicates the current step is 'Specify user details'. The main content area is titled 'Specify user details' and contains the following sections:

- User details**
 - User name**: A text input field containing 'Sahithi-user'. Below the field, a note states: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)'.
 - ☒ **Provide user access to the AWS Management Console - optional**
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.
- Are you providing console access to a person?**
 - User type**
 - ☐ **Specify a user in Identity Center - Recommended**
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.
 - ☒ **I want to create an IAM user**
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.
- Console password**
 - ☒ **Autogenerated password**
You can view the password after you create the user.
 - ☐ **Custom password**
Enter a custom password for the user.

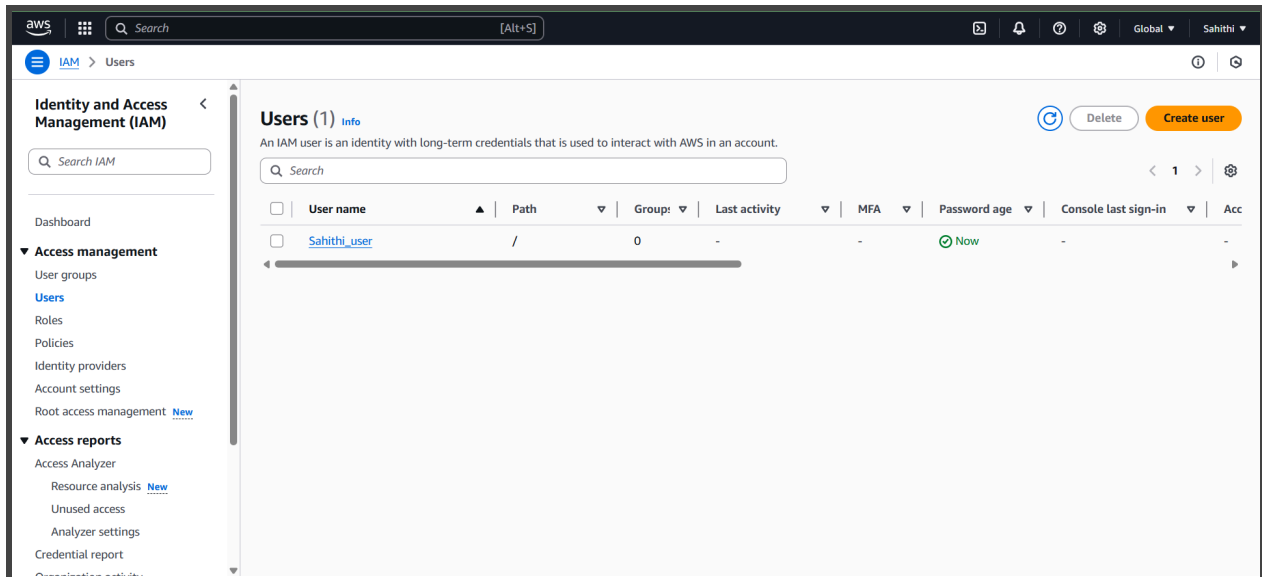
- Setting permissions and policies for the user as shown in the screenshot below.



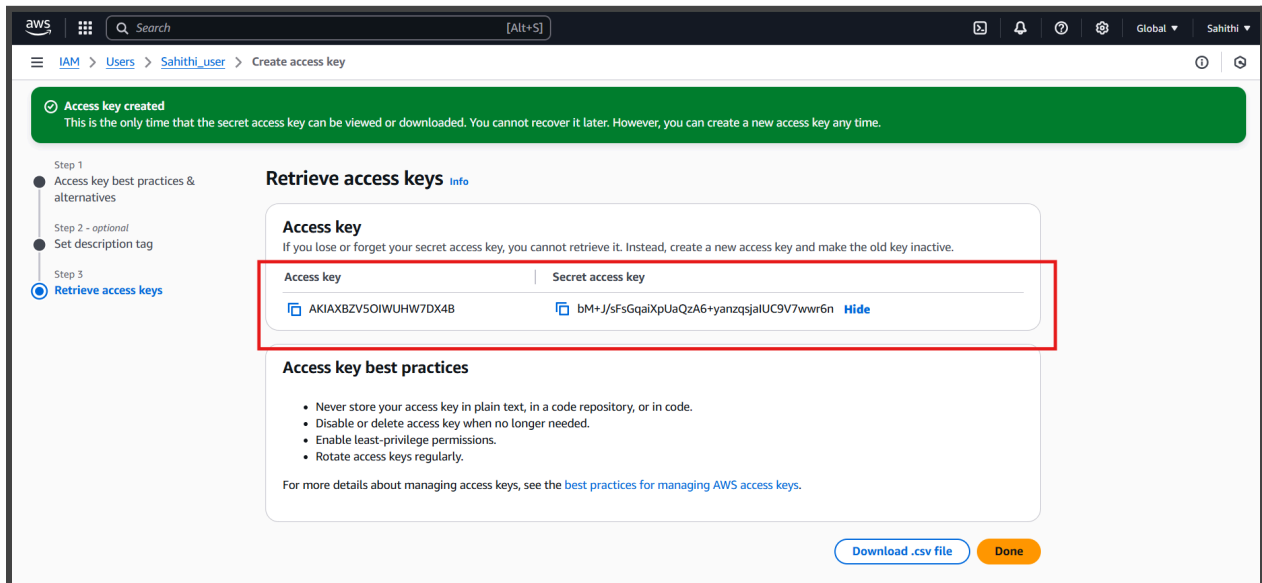
- Attaching the permissions based on the least privileges.



- The user has successfully created we can show in the screenshot below.



- Navigated to the **Security Credentials** tab for the user
- Clicked "Create access key"
- Used tag key like: "Key for CLI setup on Ubuntu EC2 instance"
- Downloaded the .csv file with credentials securely
- Both the access key and the secret key are generated successfully.



- AWS configuration is completed successfully.

```
ubuntu@ip-172-31-4-54: ~  
ubuntu@ip-172-31-4-54:~$ aws configure  
AWS Access Key ID [None]: AKIAXBZV50IWUHW7DX4B  
AWS Secret Access Key [None]: bM+J/sFsGgaiXpUaQzA6+yanzqsjaIUC9V7wwr6n  
Default region name [None]: ca-central-1  
Default output format [None]: json  
ubuntu@ip-172-31-4-54:~$
```

4. IAM Role Creation & Trust Policy Setup

- **Step 1:** Verify AWS CLI Identity “aws sts get-caller-identity”
- Confirmed CLI was authenticated and able to call AWS services using the configured IAM user

```
Last login: Thu Jun 26 19:04:04 2025 from 174.115.242.229  
ubuntu@ip-172-31-4-54:~$ aws sts get-caller-identity  
{  
  "UserId": "AIDAXBZV50IWU3E3G3QIX",  
  "Account": "484907512365",  
  "Arn": "arn:aws:iam::484907512365:user/Sahithi_user"  
}
```

- **Step 2:** Create Trust Policy Document “nano trust-policy.json”

```
GNU nano 7.2 trust-policy.json *  
{"Version": "2012-10-17",  
 "Statement": [  
   {  
     "Effect": "Allow",  
     "Principal": { "Service": "ec2.amazonaws.com" },  
     "Action": "sts:AssumeRole"  
   }  
 ]  
}
```

- **Step 3:** Create IAM Role Using Trust Policy “ aws iam create-role \

--role-name HardenedEC2Role \

--assume-role-policy-document file://trust-policy.json”
- Created a role named HardenedEC2Role with EC2 as the trusted entity.

```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Thu Jun 26 20:00:23 2025 from 3.16.146.5
ubuntu@ip-172-31-4-54:~$ aws iam attach-role-policy \
--role-name HardenedEC2Role \
--policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
ubuntu@ip-172-31-4-54:~$
```

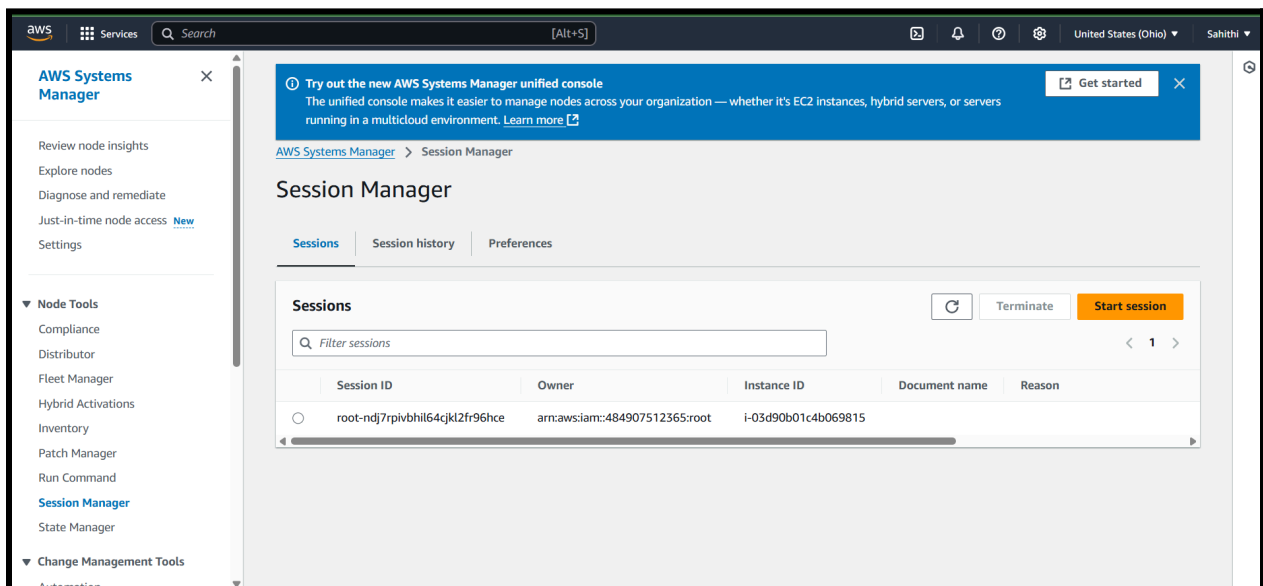
- Attach Required Policy to the Role “ aws iam attach-role-policy \

--role-name HardenedEC2Role \

--policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore”
- Attached **AmazonSSMManagedInstanceCore** to grant Systems Manager access for session management, inventory collection, and patching.

4.1. Testing System Manager

- Go to the System Manager and click Session Manager
- Start the session



- This screenshot below is from the AWS IAM console, and it shows the AmazonSSMManagedInstanceCore policy—an AWS-managed policy that grants permissions for Systems Manager to interact with your EC2 instances.

The screenshot displays the AWS IAM console interface. On the left, the 'Identity and Access Management (IAM)' sidebar is visible, with 'Policies' selected. The main content area shows the 'Policy details' for 'AmazonSSMManagedInstanceCore'. The policy is 'AWS managed', created on March 15, 2019, and edited on May 23, 2019. Its ARN is 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'. Below the details, the 'Permissions' tab is active, showing 'Permissions defined in this policy'. A table lists the permissions: 'EC2 Messages' (Full access, All resources), 'SSM Messages' (Full access, All resources), and 'Systems Manager' (Limited: List, Read, Write, All resources). The 'Systems Manager' permission is highlighted in blue.

Service	Access level	Resource	Request condition
EC2 Messages	Full access	All resources	None
SSM Messages	Full access	All resources	None
Systems Manager	Limited: List, Read, Write	All resources	None

5. Logging & Monitoring Configuration

To ensure traceability, threat detection, and audit readiness, key AWS security services were enabled and configured.

5.1. AWS CloudTrail Setup

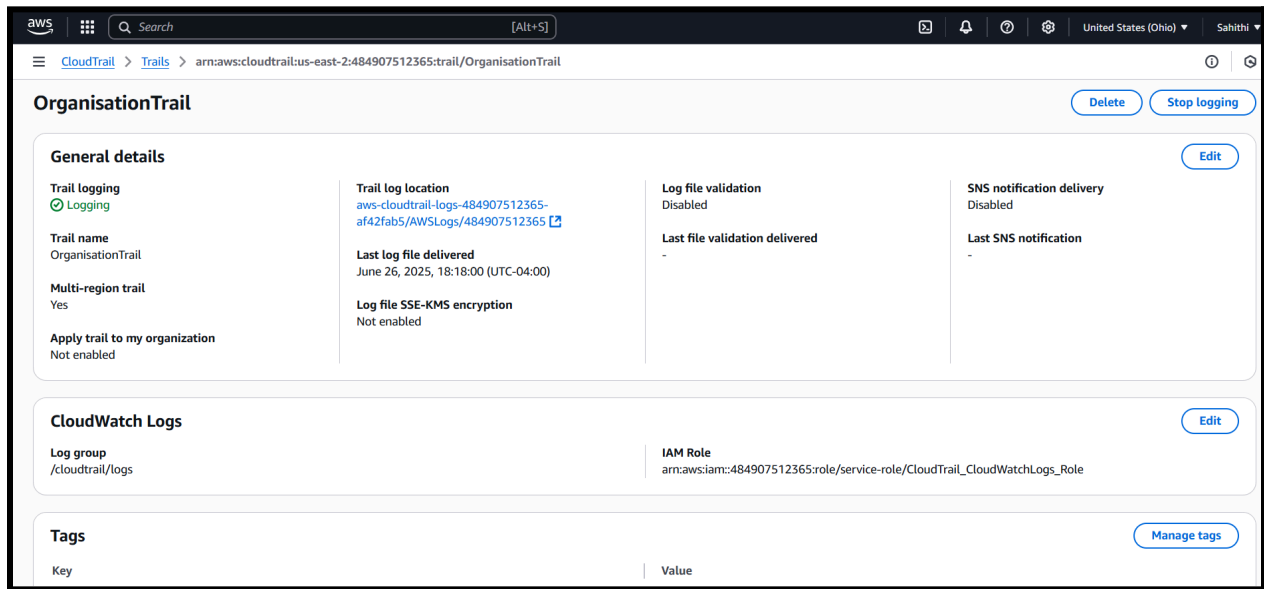
- Navigate to the CloudTrail and start creating a trail
- Step 1: As shown in the screenshot below,
- Trail Name: Organisation

The screenshot shows the 'Quick trail create' page in the AWS CloudTrail console. The page has a dark header with the AWS logo, a search bar, and navigation icons. The main content area is titled 'Quick trail create' and contains a 'Trail details' section. This section includes a description of trail logging, a 'Trail name' input field with the value 'OrganisationTrail', and a 'Trail log bucket and folder' section showing a default S3 bucket path. A note at the bottom states that there are no costs for logging events but charges for the S3 bucket. At the bottom right, there are 'Cancel' and 'Create trail' buttons.

- Step 2: Create cloudwatch logs in the couldtrail

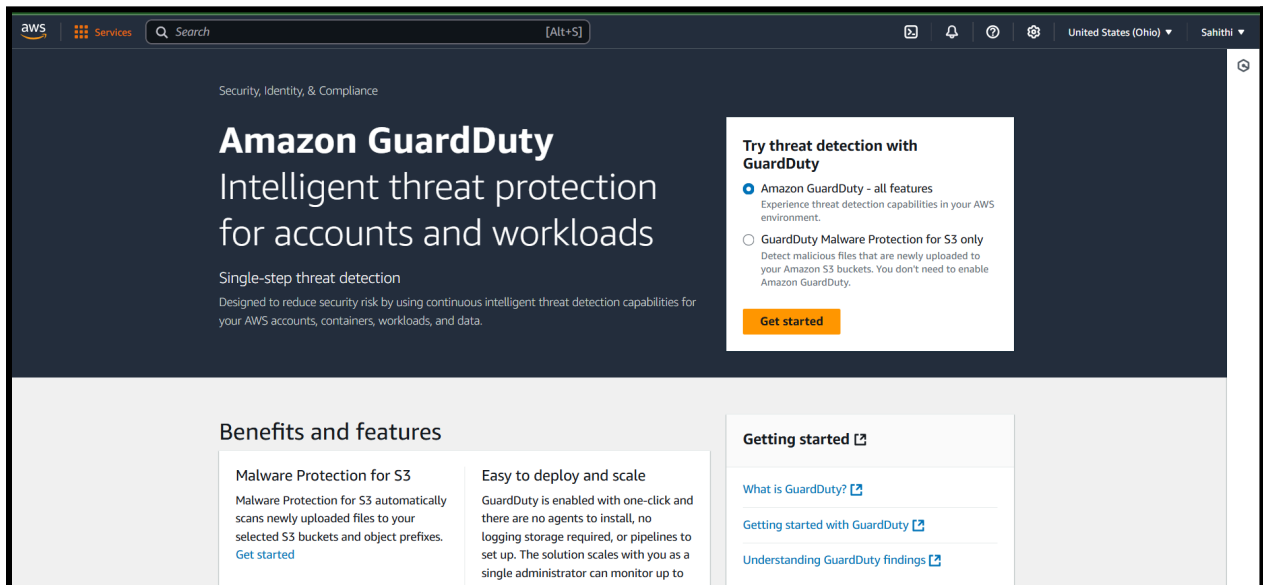
The screenshot shows the 'CloudWatch Logs - optional' configuration page for a specific trail. The page title is 'CloudWatch Logs - optional' and it includes a sub-header 'Configure CloudWatch Logs to monitor your trail logs and notify you when specific activity occurs'. The 'CloudWatch Logs' section is checked and set to 'Enabled'. Under 'Log group', the 'New' radio button is selected, and the 'Log group name' is set to '/cloudtrail/logs'. The 'IAM Role' section also has the 'New' radio button selected, with the 'Role name' set to 'CloudTrail_CloudWatchLogs_Role'. A 'Policy document' section is visible at the bottom.

- Step 3: Features enabled Management Events (Read/Write), Multi-region logging, Log file validation.
- Logs are stored in an S3 bucket for centralised analysis.

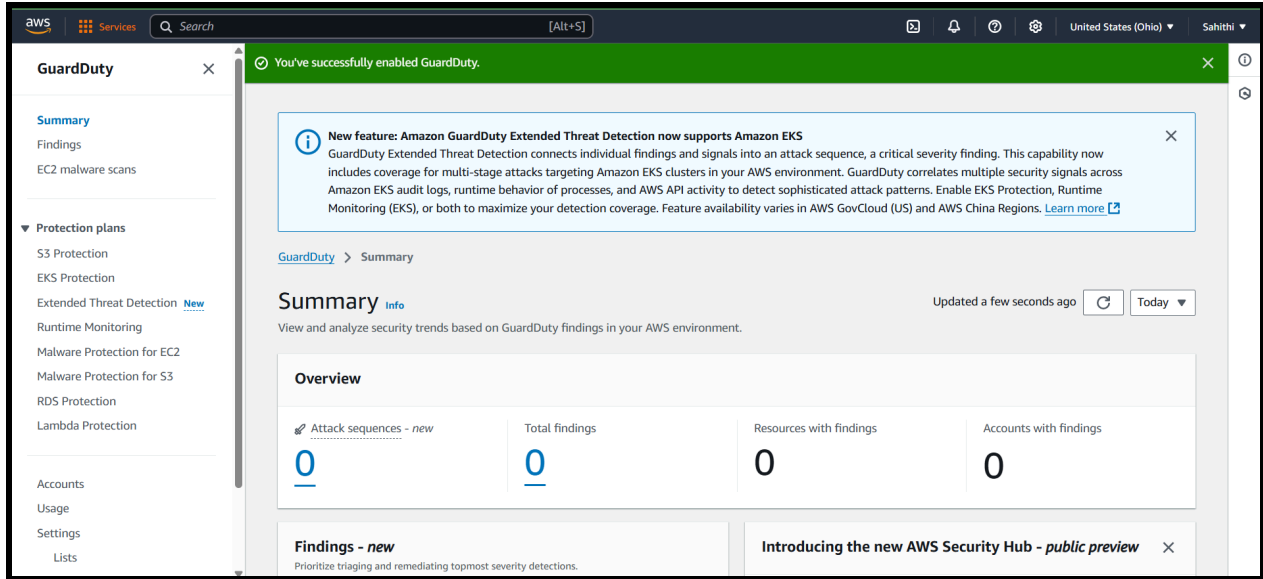


5.2. Amazon GuardDuty Activation

- Navigate to the Amazon GuardDuty console



- GuardDuty was successfully enabled via the AWS Console.

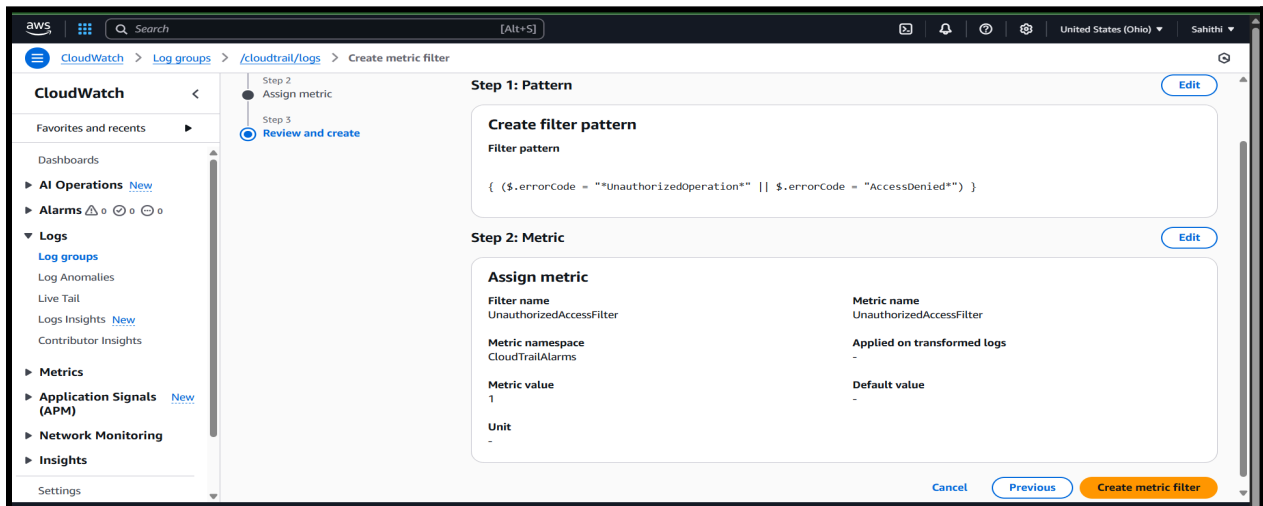


5.3. Set up alerts for unauthorised access attempts

To detect and respond to unauthorised API actions in real time, a CloudWatch alarm was created based on custom metrics from CloudTrail logs.

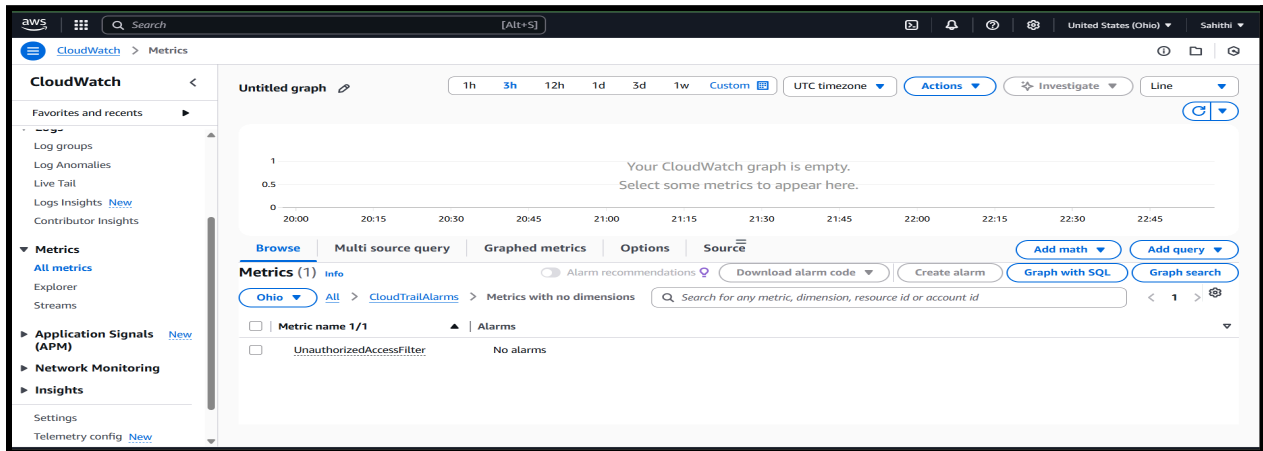
Create Metrix filter:

- Navigated to CloudWatch → Log groups → Selected the CloudTrail log group.
- Define the metrics
 - **Filter name:** UnauthorizedAccessFilter
 - **Metric namespace:** CloudTrailAlarms
 - **Metric name:** UnauthorizedCount
 - **Metric value :** 1

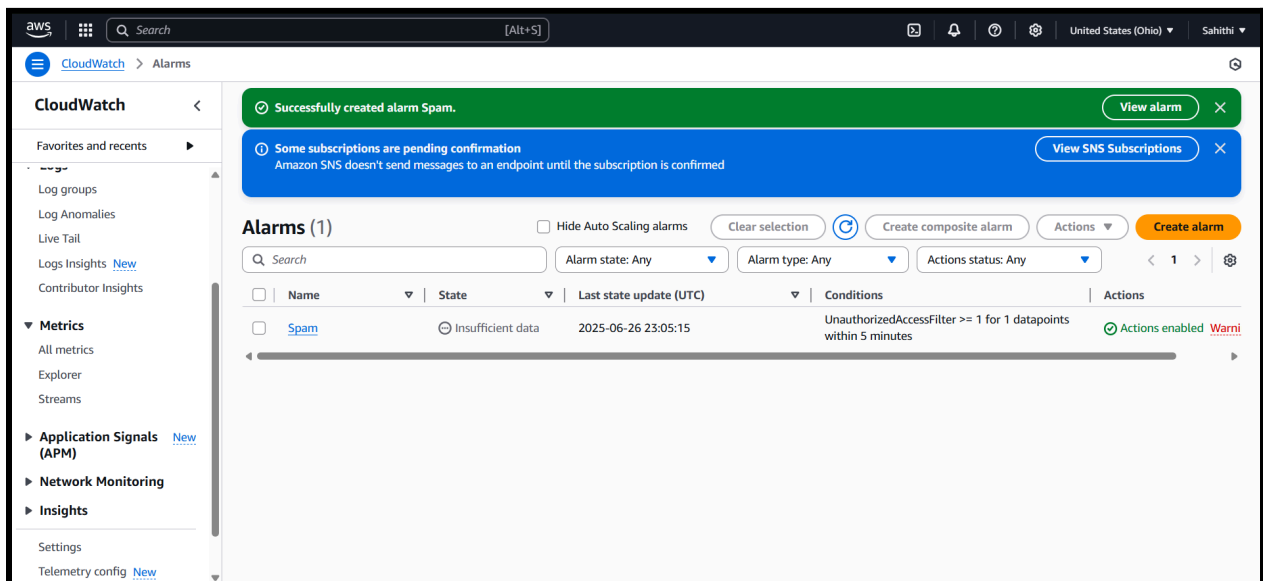


Create an Alarm Based on a Filter

- Navigated to All Metrics → Custom Namespaces → CloudTrailAlarms



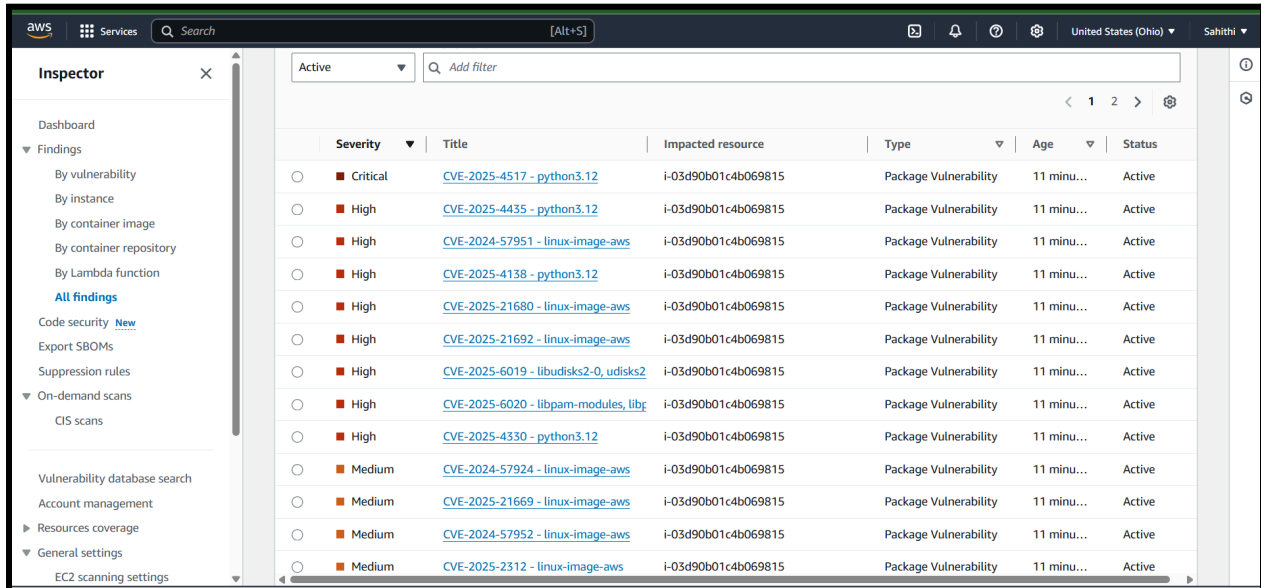
- Selected UnauthorizedCount under UnauthorizedAccessFilter .
- Created a new alarm:
- **Alarm Name:** UnauthorizedAccessAlarm (*you can rename as needed*)
- **Threshold:** When UnauthorizedCount ≥ 1 in 5 minutes
- **Actions:** Optional SNS notification or console alert



6. Vulnerability Scanning & Findings

As part of the hardening audit, **AWS Inspector** was used to scan the EC2 instance for known vulnerabilities (CVEs) affecting system libraries and runtime environments.

- This AWS Inspector dashboard lists security findings, including vulnerability titles, severity levels, affected resources, and their current status.



Severity	Title	Impacted resource	Type	Age	Status
Critical	CVE-2025-4517 - python3.12	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-4435 - python3.12	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2024-57951 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-4138 - python3.12	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-21680 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-21692 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-6019 - libudisks2-0, udisks2	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-6020 - libpam-modules, libpam	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
High	CVE-2025-4330 - python3.12	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
Medium	CVE-2024-57924 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
Medium	CVE-2025-21669 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
Medium	CVE-2024-57952 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active
Medium	CVE-2025-2312 - linux-image-aws	i-03d90b01c4b069815	Package Vulnerability	11 minu...	Active

6.1. CVE-2025-4517 – Arbitrary File Write via Path Traversal

- The vulnerability **CVE-2025-4517** affects Python's tarfile module and allows attackers to perform arbitrary file writes by extracting specially crafted .tar files. This could lead to overwriting sensitive system files, a classic **path traversal attack**. It's especially dangerous in applications that process untrusted archives.
- The screenshot below shows the

CVE-2025-4517 - python3.12
Finding ID: arn:aws:inspector:us-east-2:484907512365:finding/6a0b3d5494d9e18db5014aa02b5eb1e8

Allows arbitrary filesystem writes outside the extraction directory during extraction with filter="data". You are affected by this vulnerability if using the tarfile module to extract untrusted tar archives using TarFile.extractall() or TarFile.extract() using the filter= parameter with a value of "data" or "tar". See the tarfile extraction filters documentation <https://docs.python.org/3/library/tarfile.html#tarfile-extraction-filter> for more information. Note that for Python 3.14 or later the default value of filter= changed from "no filtering" to "data", so if you are relying on this new default behavior then your usage is also affected. Note that none of these vulnerabilities significantly affect the installation of source distributions which are tar archives as source distributions already allow arbitrary code execution during the build process. However when evaluating source distributions it's important to avoid installing source distributions with suspicious links.

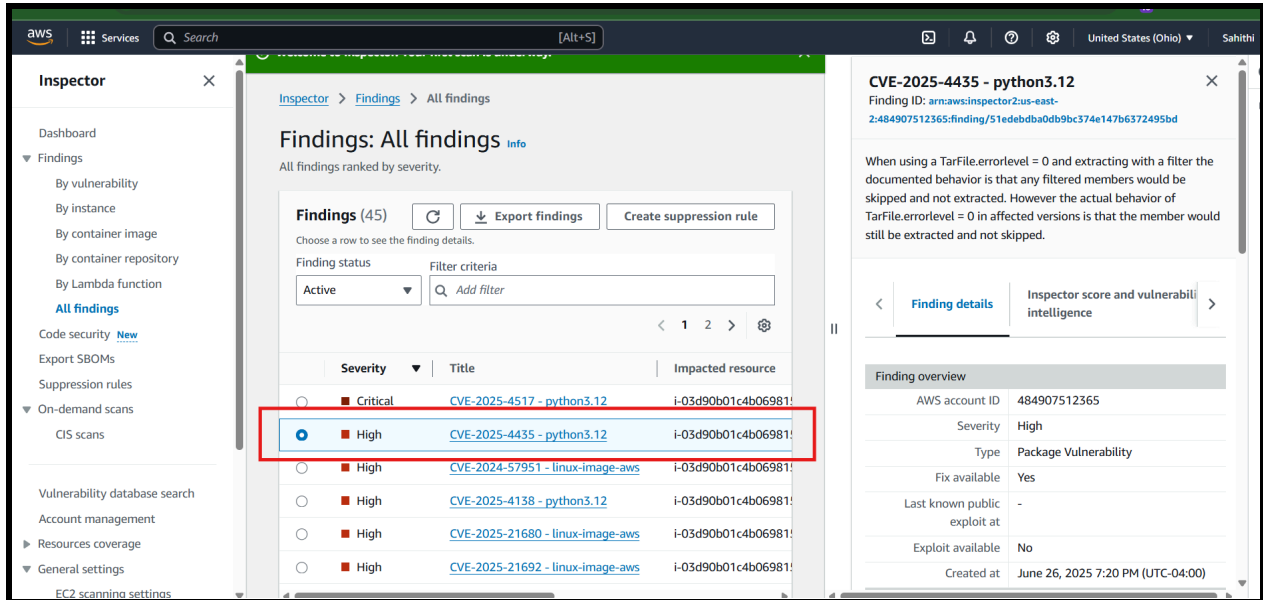
Inspector score and vulnerability intelligence

CVSS v3 (UBUNTU_CVE) Inspector

Mitigation: We need to avoid using insecure filters like filter="data" or filter="tar", apply the latest Python updates once available, and validate extraction paths before writing files. These steps harden the system against file manipulation exploits.

6.2.CVE-2025-4435- Bypass of Filtering via Errorlevel Override

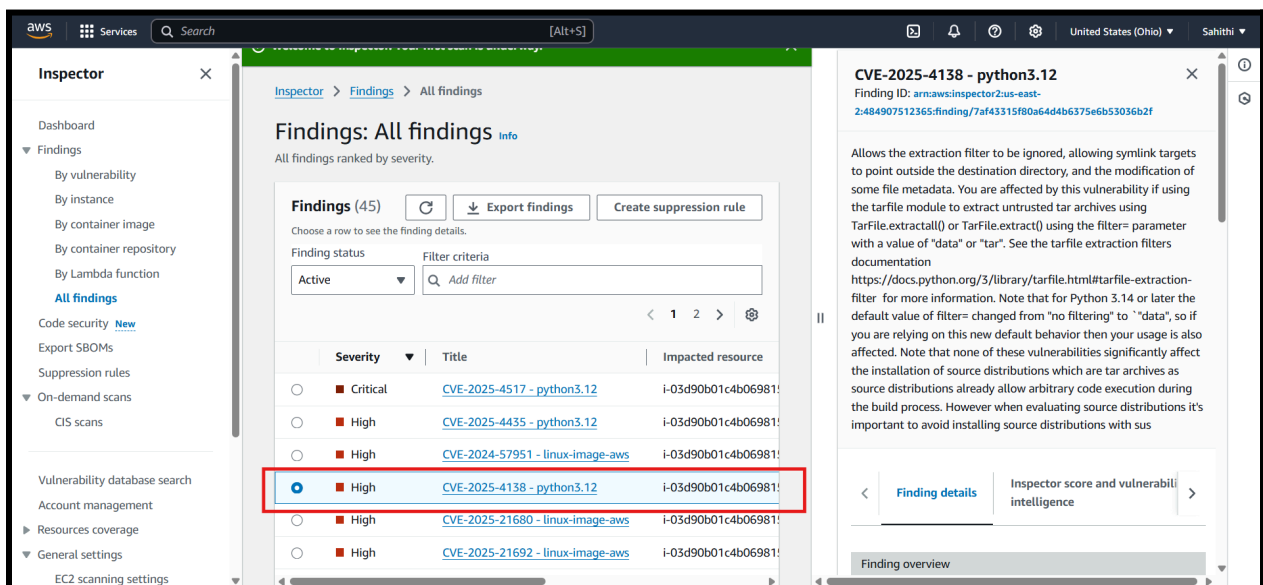
- The vulnerability **CVE-2025-4435** affects Python 3.12's **tarfile** module, where filtered members are incorrectly extracted even when **TarFile.errorlevel = 0**, bypassing expected safety behaviour. This flaw could be exploited to write arbitrary files during .tar extraction, posing a serious risk if handling untrusted archives.
- The screenshot below shows the latest vulnerabilities.



- Mitigation:** To mitigate this vulnerability, avoid relying solely on **errorlevel** as a safety check during **.tar** file extraction. Instead, ensure strict file path validation before writing any extracted content and upgrade Python to the latest patched release to eliminate known flaws.

6.3.CVE-2025-4138 – Symlink Attack via Filter Bypass

This vulnerability allows attackers to bypass extraction filters in Python's **tarfile** module, enabling **symlink attacks** that write files outside the intended extraction directory. It's especially risky when extracting untrusted **.tar** archives using **filter="data"** or **filter="tar"**—commonly used in Python 3.12 and above.

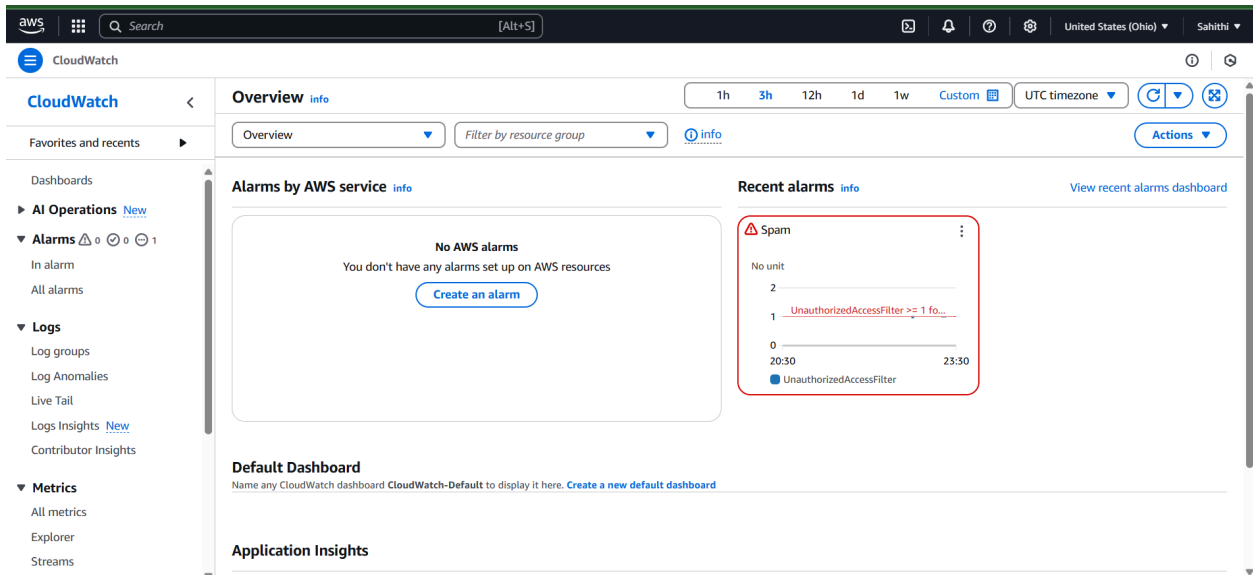


- **Mitigation:** To avoid this vulnerability, avoid using unsafe filters like **filter="tar"** or **filter="data"**, as they can be exploited during archive extraction. Upgrade to the latest patched Python version, and enforce strict path sanitisation to ensure all files are extracted only within the intended directory.

6.4. Alarm Response Validation & System Monitoring

To confirm the effectiveness of the monitoring setup, the **UnauthorizedAccessAlarm** was intentionally triggered by simulating an unauthorised API call. This validated that the CloudWatch metric filter and alarm were functioning as designed.

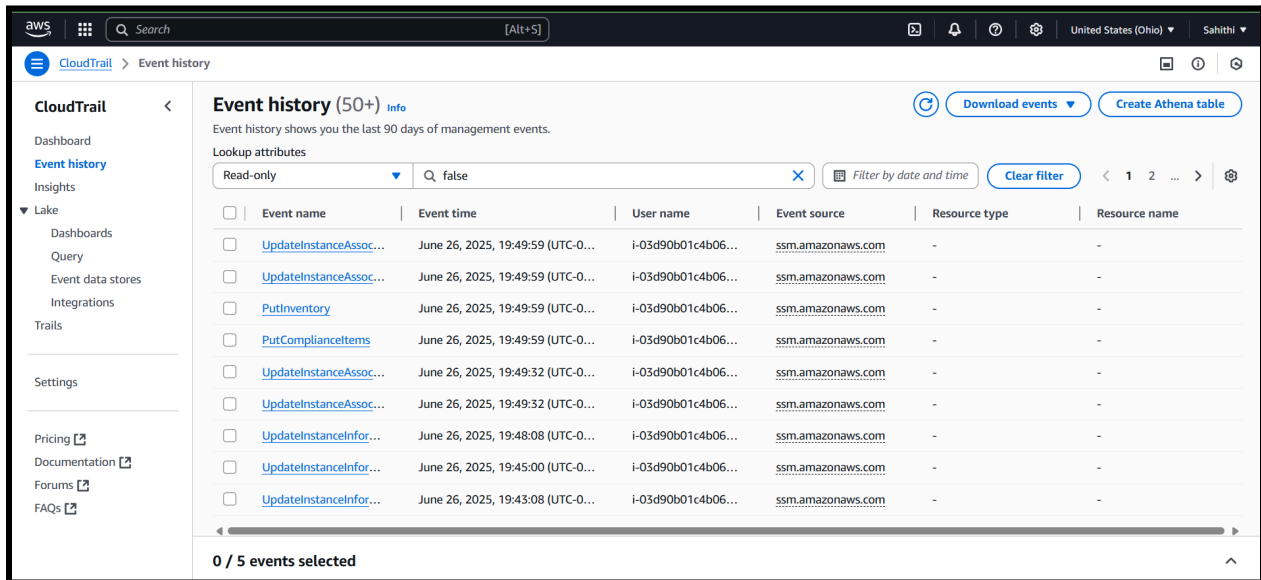
- **Alarm Trigger Condition: UnauthorizedAccessFilter ≥ 1**
- **Alarm Status:** The alarm transitioned to the “In alarm” state and appeared under CloudWatch → Alarms → Recent Alarms
- **Significance:** This confirms that the system is actively monitoring IAM policy violations and can provide timely alerts for unauthorised access attempts.



6.5. Alarm Validation & Agent Activity Logs

To verifying the CloudWatch alarm behaviour, logs captured by **AWS Systems Manager** confirm real-time agent activity.

- Systems Manager shows that the EC2 instance is actively communicating with **AWS Inspector** and **SSM agents**.
- These logs validate that periodic scans, software inventory checks, and patch compliance data are being collected without user intervention.



The screenshot displays the AWS CloudTrail 'Event history' console. The left sidebar shows the navigation menu with 'Event history' selected. The main panel shows a list of events with columns for Event name, Event time, User name, Event source, Resource type, and Resource name. The events listed are all from June 26, 2025, and involve the user 'i-03d90b01c4b06...' performing actions like 'UpdateInstanceAssoc...', 'PutInventory', and 'PutComplianceItems' on 'ssm.amazonaws.com' resources. The console also includes search filters, a 'Download events' button, and a 'Create Athena table' button.

Event name	Event time	User name	Event source	Resource type	Resource name
UpdateInstanceAssoc...	June 26, 2025, 19:49:59 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
UpdateInstanceAssoc...	June 26, 2025, 19:49:59 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
PutInventory	June 26, 2025, 19:49:59 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
PutComplianceItems	June 26, 2025, 19:49:59 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
UpdateInstanceAssoc...	June 26, 2025, 19:49:32 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
UpdateInstanceAssoc...	June 26, 2025, 19:49:32 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
UpdateInstanceInfor...	June 26, 2025, 19:48:08 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
UpdateInstanceInfor...	June 26, 2025, 19:45:00 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-
UpdateInstanceInfor...	June 26, 2025, 19:43:08 (UTC-0...)	i-03d90b01c4b06...	ssm.amazonaws.com	-	-

7. Key Observations

- SSH access was hardened by disabling root login and enforcing key-based authentication with restricted public IP access.
- IAM best practices were followed through least-privilege user roles and tightly scoped permissions.
- AWS Systems Manager enabled agent-based access and eliminated reliance on SSH or exposed credentials.
- CloudTrail and CloudWatch were successfully configured to detect and alert on unauthorised access attempts in real time.
- GuardDuty and Inspector were actively running, continuously monitoring threats and vulnerabilities.
- Simulated unauthorised actions triggered alarms as expected, validating full-stack monitoring functionality.

8. Conclusion

This project demonstrated a full-stack approach to EC2 security by combining access control, configuration hardening, real-time monitoring, and vulnerability management. By leveraging AWS native tools—CloudTrail, GuardDuty, Systems Manager, CloudWatch, and Inspector—the environment transitioned from an open, default state to a locked-down and audit-capable deployment.

The layered defence strategy ensures that unauthorised activity is both prevented and detected, while automation handles much of the alerting and inventory management. The implemented controls lay a strong foundation for compliance, governance, and future scaling of secure workloads in the cloud.