Cloud Security Audit and Penetration Testing: A Case Study on AWS EC2

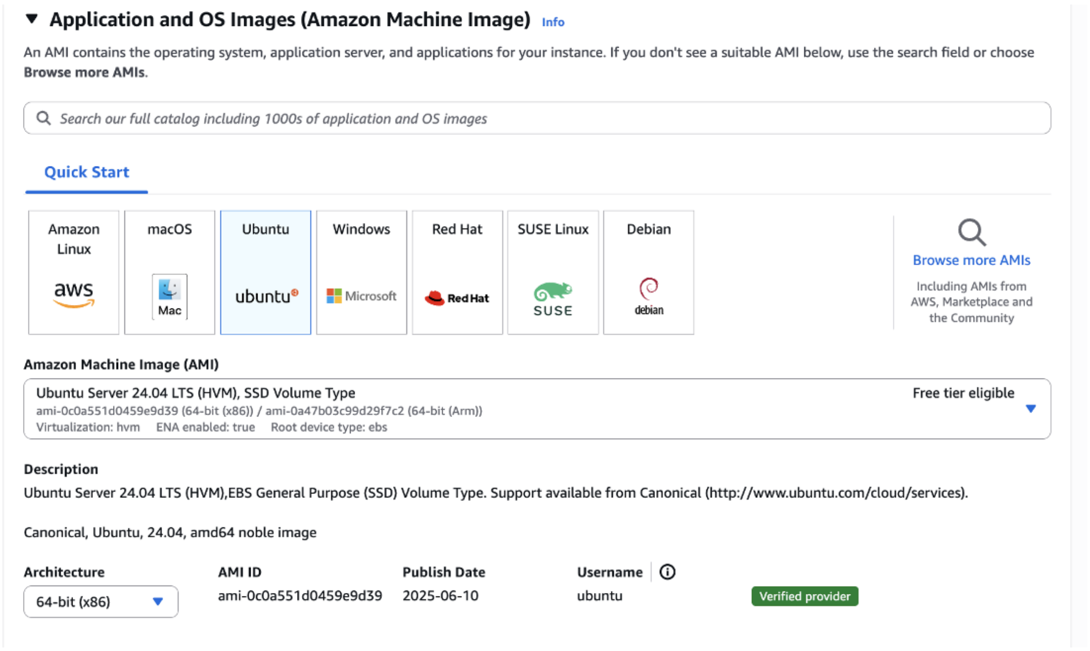
1.Introduction

This project focuses on performing a cloud security audit and penetration testing on an AWS EC2 instance to evaluate and enhance its security posture. We deployed a cloud-hosted web application on an Ubuntu-based EC2 instance and conducted a structured security review using both open-source tools (Nmap, Nikto) and AWS-native services (Amazon Inspector). The objective was to identify potential vulnerabilities, analyze the attack surface, and apply effective remediation strategies to harden the system against threats. By combining network scanning, web server analysis, and automated vulnerability management, we aimed to provide a comprehensive assessment of the EC2 instance and implement security best practices.

2. Set Up a Cloud-Hosted Web App (AWS EC2)

1) Launching an EC2 instance

We launched an EC2 instance under the AWS Free Tier, configured with Ubuntu 24.04 LTS as the operating system and the t3.micro instance type. To enable secure remote access, we used an SSH key pair named MyKEY-LIN.pem.



Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debian

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

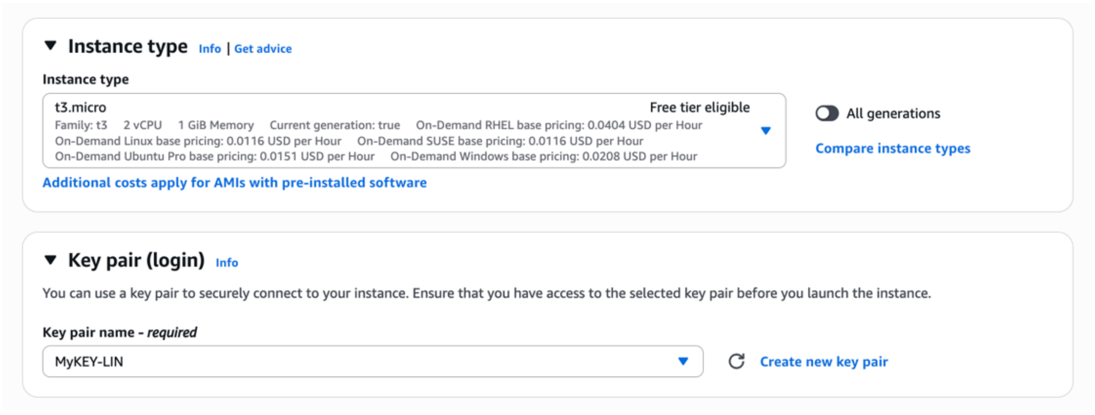
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0c0a551d0459e9d39 (64-bit (x86)) / ami-0a47b03c99d29f7c2 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture	AMI ID	Publish Date	Username	
64-bit (x86)	ami-0c0a551d0459e9d39	2025-06-10	ubuntu	Verified provider



Instance type Info | Get advice

Instance type

t3.micro — Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true On-Demand RHEL base pricing: 0.0404 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0151 USD per Hour On-Demand Windows base pricing: 0.0208 USD per Hour

Additional costs apply for AMIs with pre-installed software

All generations
Compare instance types

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

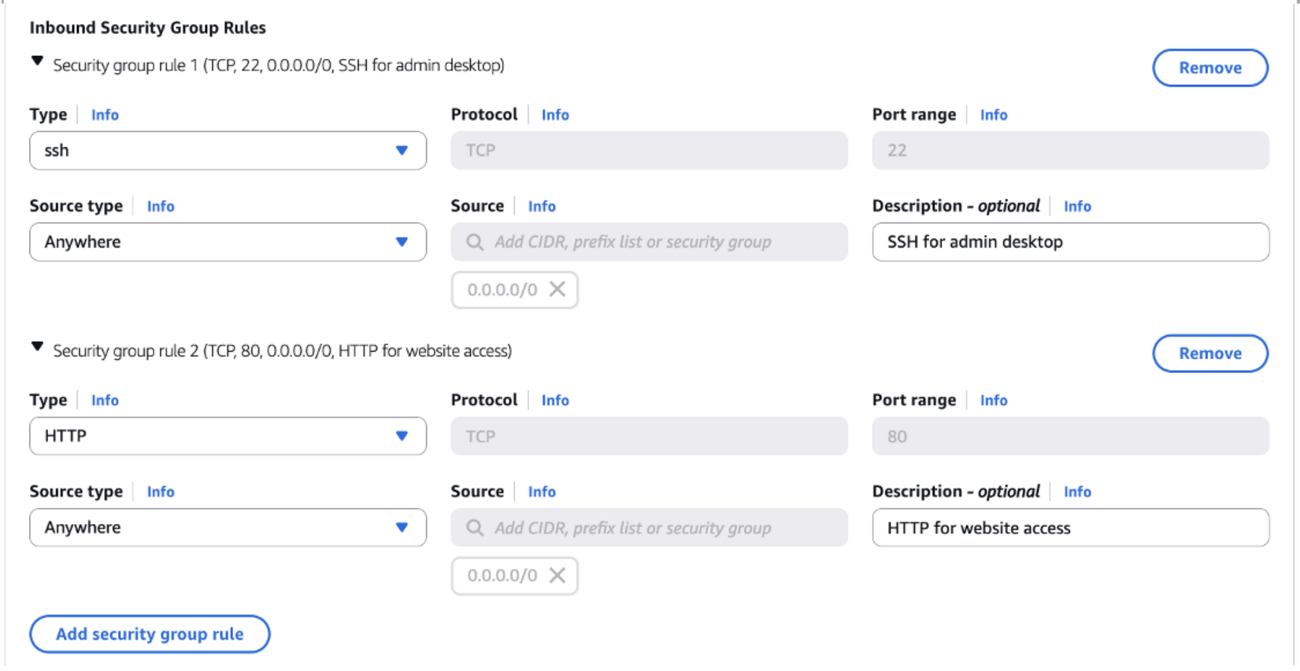
Key pair name - *required*

MyKEY-LIN

Create new key pair

We configured inbound Security Group rules to define which types of traffic are allowed into the EC2 instance. Specifically, TCP port 22 (SSH) was opened to allow secure remote terminal access, and TCP port 80 (HTTP) was opened to enable public access to the hosted web application. The source for both rules was set to 0.0.0.0/0, meaning access is permitted from any IP address. While this is acceptable for testing and demonstration purposes, it is not recommended in production environments due to security risks.

For outbound traffic**,** we retained the default Security Group configuration, which allows all outbound connections from the EC2 instance. This setting ensures that the instance can access external services such as package repositories, time servers, or other public APIs required during system setup and operation.



Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0, SSH for admin desktop)

Type: ssh	Protocol: TCP	Port range: 22
Source type: Anywhere	Source: 0.0.0.0/0	Description - *optional*: SSH for admin desktop

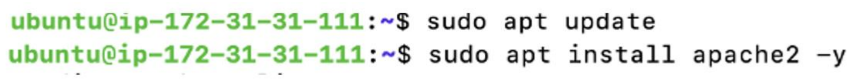
▼ Security group rule 2 (TCP, 80, 0.0.0.0/0, HTTP for website access)

Type: HTTP	Protocol: TCP	Port range: 80
Source type: Anywhere	Source: 0.0.0.0/0	Description - *optional*: HTTP for website access

2) Install Apache2 Web Server

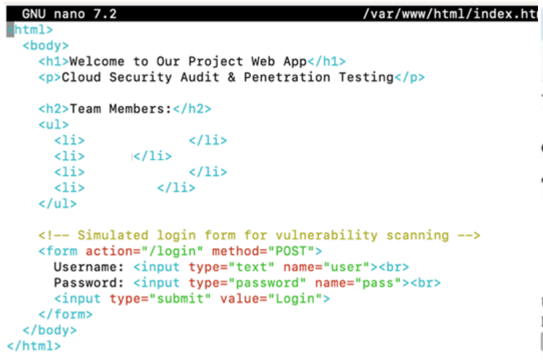
Then we installed the **Apache2 web server** using the following commands:

sudo apt update
sudo apt install apache2 -y



```
ubuntu@ip-172-31-31-111:~$ sudo apt update
ubuntu@ip-172-31-31-111:~$ sudo apt install apache2 -y
```

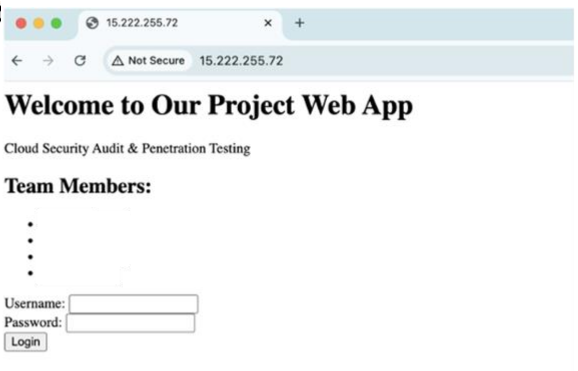
We modified the default index.html file to include our project title, the names of all team members, and a simulated login form designed specifically for testing vulnerability scanning tools. The web application was then publicly accessible at: **http://15.222.255.72**



```
  GNU nano 7.2                              /var/www/html/index.ht
<html>
  <body>
    <h1>Welcome to Our Project Web App</h1>
    <p>Cloud Security Audit & Penetration Testing</p>

    <h2>Team Members:</h2>
    <ul>
      <li>                 </li>
      <li>          </li>
      <li>                 </li>
      <li>             </li>
    </ul>

    <!-- Simulated login form for vulnerability scanning -->
    <form action="/login" method="POST">
      Username: <input type="text" name="user"><br>
      Password: <input type="password" name="pass"><br>
      <input type="submit" value="Login">
    </form>
  </body>
</html>
```



Welcome to Our Project Web App

Cloud Security Audit & Penetration Testing

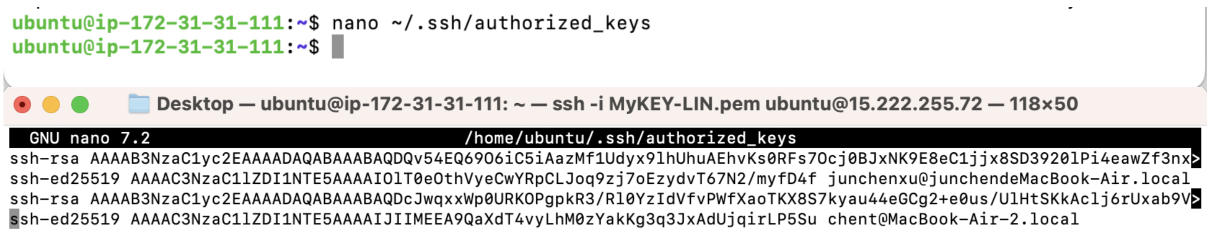
Team Members:

-
-
-
-

Username:
Password:
Login

3)SSH Key Configuration for Team Access

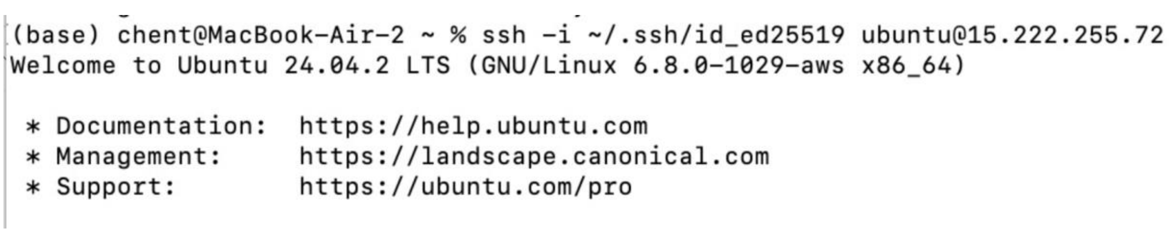
For both security and convenience, each team member generated their own **SSH key pair** locally using the ssh-keygen command. The **public keys** were collected and appended to the EC2 instance's /home/ubuntu/.ssh/authorized_keys file, allowing each member to authenticate individually without sharing credentials. This setup ensures secure access while enabling effective team collaboration.



```
ubuntu@ip-172-31-31-111:~$ nano ~/.ssh/authorized_keys
ubuntu@ip-172-31-31-111:~$
```

Desktop — ubuntu@ip-172-31-31-111: ~ — ssh -i MyKEY-LIN.pem ubuntu@15.222.255.72 — 118×50

```
  GNU nano 7.2                        /home/ubuntu/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQv54EQ69O6iC5iAazMf1Udyx9lhUhuAEhvKs0RFs7Ocj0BJxNK9E8eC1jjx8SD3920lPi4eawZf3nx>
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOlT0eOthVyeCwYRpCLJoq9zj7oEzydvT67N2/myfD4f junchenxu@junchendeMacBook-Air.local
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDcJwqxxWp0URKOPgpkR3/Rl0YzIdVfvPWfXaoTKX8S7kyau44eGCg2+e0us/UlHtSKkAclj6rUxab9V>
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJIIMEEA9QaXdT4vyLhM0zYakKg3q3JxAdUjqirLP5Su chent@MacBook-Air-2.local
```



```
(base) chent@MacBook-Air-2 ~ % ssh -i ~/.ssh/id_ed25519 ubuntu@15.222.255.72
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro
```

This configuration offers multiple security and operational advantages. First, it enhances overall security by eliminating the use of password-based authentication, which is more susceptible to brute-force attacks and credential theft. By relying solely on SSH key pairs, the system enforces a stronger and more resilient authentication mechanism.

Second, it allows for fine-grained access control, as each team member's public key is uniquely associated with their identity. This makes it easy to audit access logs, trace user activity, and revoke access for specific individuals without affecting others—ensuring better manageability in multi-user environments.

Finally, the approach facilitates seamless team collaboration, enabling all group members to access the shared EC2 instance securely and independently. This not only streamlines the development and testing process, but also upholds best practices for secure cloud operations.

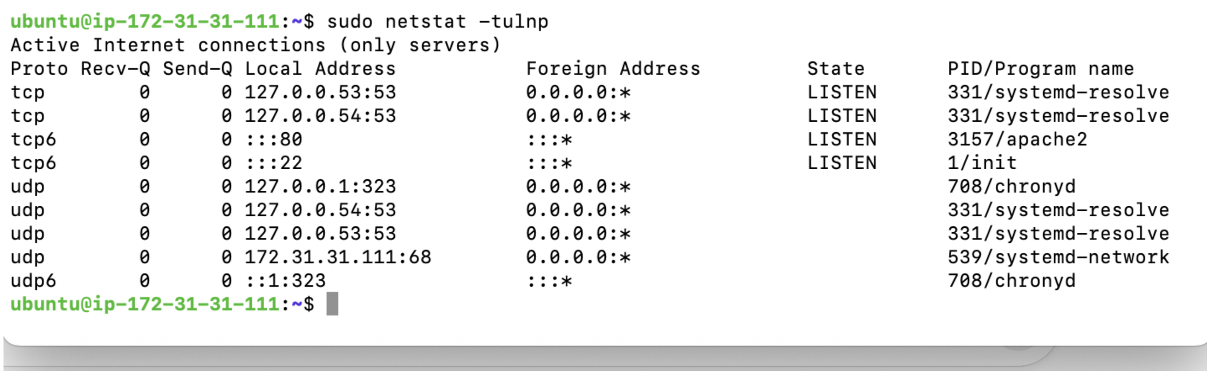
3. Conducting a Baseline Security Review

Before performing vulnerability scans, we conducted a baseline security review to understand the default state of our EC2 instance. This step aimed to identify which network services were actively listening, evaluate the exposure surface, and determine whether any unnecessary ports or protocols were enabled. By doing so, we established a security baseline against which later audit findings could be compared.

1） Command Used

To examine which network services were active and listening for connections, we executed the following command within the EC2 instance:

sudo netstat –tulnp



```
ubuntu@ip-172-31-31-111:~$ sudo netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      331/systemd-resolve
tcp        0      0 127.0.0.54:53           0.0.0.0:*               LISTEN      331/systemd-resolve
tcp6       0      0 :::80                   :::*                    LISTEN      3157/apache2
tcp6       0      0 :::22                   :::*                    LISTEN      1/init
udp        0      0 127.0.0.1:323           0.0.0.0:*                           708/chronyd
udp        0      0 127.0.0.54:53           0.0.0.0:*                           331/systemd-resolve
udp        0      0 127.0.0.53:53           0.0.0.0:*                           331/systemd-resolve
udp        0      0 172.31.31.111:68        0.0.0.0:*                           539/systemd-network
udp6       0      0 ::1:323                 :::*                                708/chronyd
ubuntu@ip-172-31-31-111:~$
```

2） Findings

The command output revealed the following open ports and services:

- **Port 80 (HTTP):** The Apache web server was actively listening, which is expected for hosting our web application.
- **Port 22 (SSH):** Enabled to allow secure remote terminal access for server management.
- **Port 53 (DNS):** The systemd-resolved service was listening on this port, providing local DNS resolution—enabled by default on Ubuntu.
- **Ports 323 & 68 (UDP):** These ports were used by the Chrony service and systemd-network for network time synchronization.
- **IPv4 and IPv6 Interfaces:** Both address types were active, indicating that services were potentially accessible over both network protocols, which could expand the attack surface if not properly managed.

3) Analysis

- **Expected Ports:** Ports 22 and 80 were necessary for remote access and web hosting respectively, and their presence was in line with our project design.
- **Optional Services:** The DNS resolver (Port 53) and time synchronization services (Ports 323 and 68) were not critical to our application. Disabling them would reduce the number of exposed services and potential vulnerabilities.
- **IPv6 Exposure:** As IPv6 was not required for our use case, disabling it could help reduce unnecessary exposure and simplify firewall configuration.

This baseline review provided us with a clear understanding of our EC2 instance's initial security posture, allowing us to move forward with vulnerability scanning and risk analysis in a more informed and structured manner.

4. Vulnerability Assessment

As part of the security audit, we conducted a vulnerability assessment on the deployed EC2 instance (15.222.255.72) to identify potential security risks at both the network and web server levels.

1) Tools and Methodology

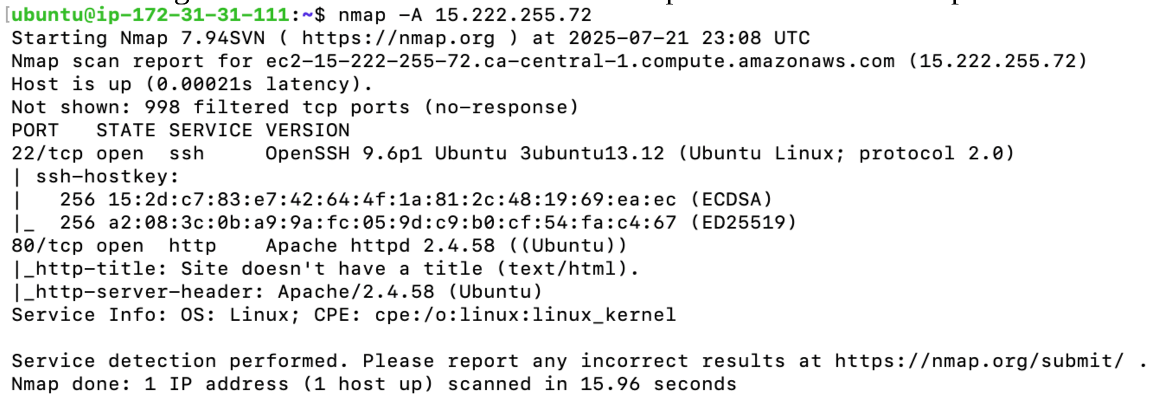
Two open-source tools were used to perform the scan:

- **Nmap**: A powerful network scanning tool used to detect open ports, running services, and operating system fingerprints.
- **Nikto**: A web server scanner that detects outdated software, misconfigurations, and missing HTTP security headers.

These tools were selected for their complementary focus—Nmap targets low-level network exposure, while Nikto focuses on HTTP-related vulnerabilities.

2) Nmap Scan Results

The following command was executed for a comprehensive scan: nmap -A 15.222.255.72



```
ubuntu@ip-172-31-31-111:~$ nmap -A 15.222.255.72
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-21 23:08 UTC
Nmap scan report for ec2-15-222-255-72.ca-central-1.compute.amazonaws.com (15.222.255.72)
Host is up (0.00021s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.12 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 15:2d:c7:83:e7:42:64:4f:1a:81:2c:48:19:69:ea:ec (ECDSA)
|_  256 a2:08:3c:0b:a9:9a:fc:05:9d:c9:b0:cf:54:fa:c4:67 (ED25519)
80/tcp open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.58 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.96 seconds
```

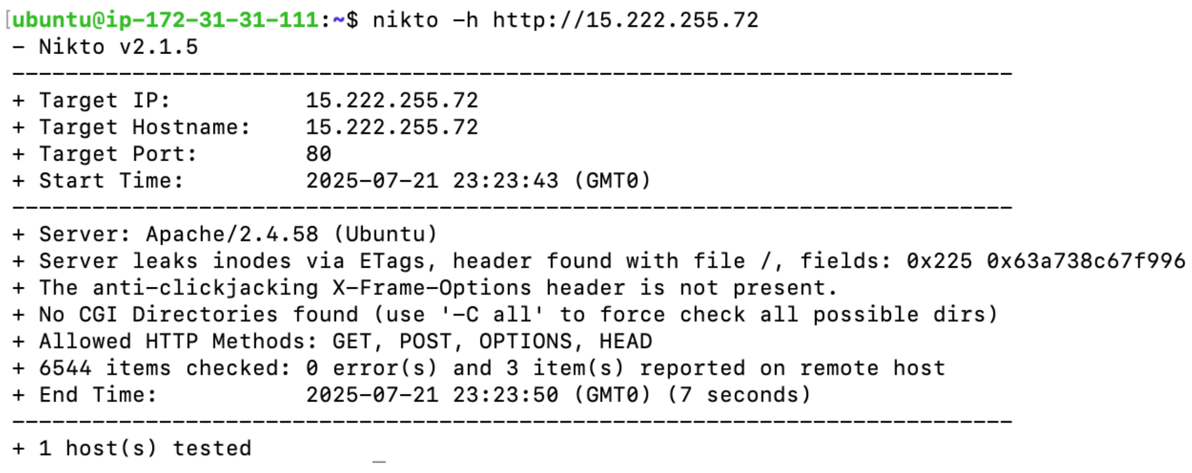
Key Findings:

- **Open Ports**: Port 22 (SSH) and 80 (HTTP) were open.
- **Running Services**:
  - **SSH**: OpenSSH 9.6p1
  - **Web Server**: Apache 2.4.58
- **OS Fingerprinting**: Detected as Ubuntu Linux.
- **HTTP Header Info**: The server banner and page title were accessible, suggesting possible information disclosure.

This scan also included a comparison with nmap -sV to observe the added value of the aggressive scan (-A), which revealed additional metadata like HTTP headers and OS info.

3) Nikto Scan Results

Command used: nikto -h http://15.222.255.72



```
[ubuntu@ip-172-31-31-111:~$ nikto -h http://15.222.255.72
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          15.222.255.72
+ Target Hostname:    15.222.255.72
+ Target Port:        80
+ Start Time:         2025-07-21 23:23:43 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache/2.4.58 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, fields: 0x225 0x63a738c67f996
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD
+ 6544 items checked: 0 error(s) and 3 item(s) reported on remote host
+ End Time:           2025-07-21 23:23:50 (GMT0) (7 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Key Findings:

- Apache version and server header exposed.
- HTTP security headers like X-Frame-Options were missing.
- Potential directory indexing detected (indicates public listing of files if enabled).

4) Severity Analysis

We followed a simplified severity classification inspired by CVSS principles:

- **High:** Vulnerabilities that could allow attackers to fully access or control the system.
- **Medium:** Issues that may help attackers exploit the system or gain more access.
- **Low:** Issues that only reveal some information but don't give direct access.

Vulnerability	Severity Level	Explanation
Open Ports (22, 80)	Medium	Harden with firewall & config
Apache version exposed	Medium	Can help attackers plan exploits
Missing HTTP headers	Medium	Lacks basic web protection
Directory indexing	Medium	May leak internal file structure

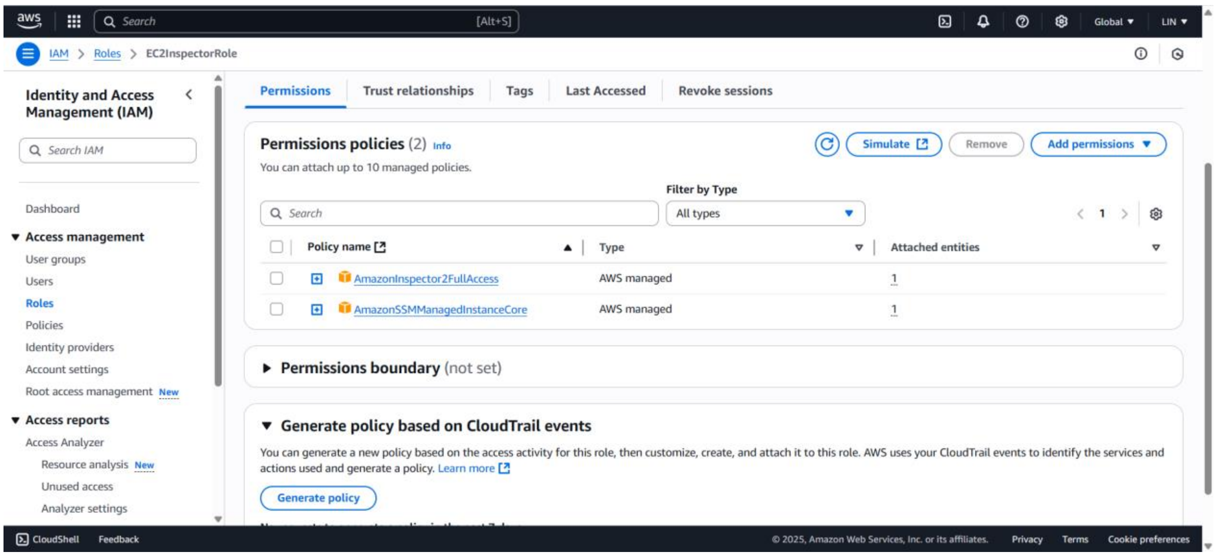
5. Cloud-Native Tool AWS Inspector to verify additional risks.

AWS Inspector is a vulnerability management service that automatically discovers workloads and continually scans them for software vulnerabilities and unintended network exposure. AWS Inspector discovers and scans Amazon EC2 instances, container images in

Amazon ECR, and Lambda functions. For this project we only focus on EC2 instances. When an Amazon Inspector detects a software vulnerability or unintended network exposure, it creates a finding, which is a detailed report about the issue.

1) Attached IAM Role with permissions.

Before activating the inspector we need to attach IAM role with permissions to the EC2 instance.



Permissions policies (2) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonInspector2FullAccess	AWS managed	1
AmazonSSMManagedInstanceCore	AWS managed	1

Permissions boundary (not set)

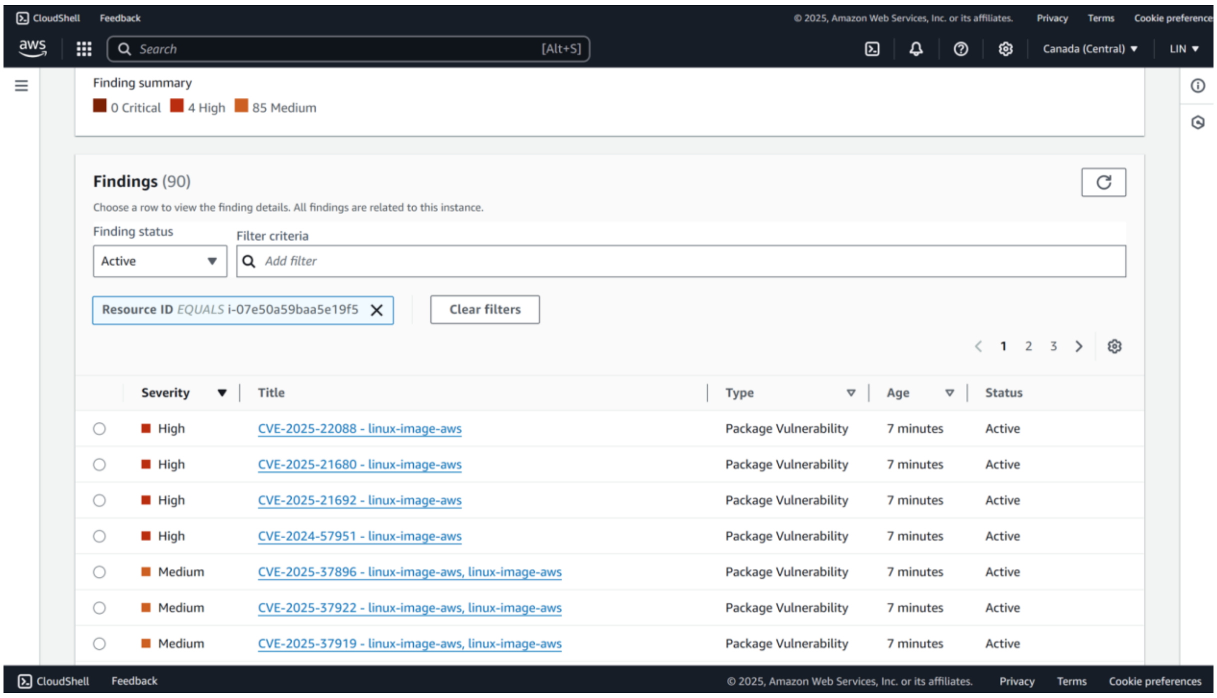
Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

This role includes two managed policies:

- **AmazonSSMManagedInstanceCore** – This is the essential permission for the EC2 instance itself. It allows the Inspector to communicate with the instance through AWS Systems Manager (SSM), enabling agent-based deep scanning. So that Inspector can use the SSM agent to install scanning components, collect data, and send results back.
- **AmazonInspector2FullAccess** – While this permission is not required by the EC2 instance directly, attaching it to the role can be useful if the same role is used by an IAM user or service that needs to interact with the Inspector service. It allows actions like viewing findings, starting scans manually, and configuring scan settings.

2) Findings – Vulnerability Summary



Finding summary
🟥 0 Critical 🟥 4 High 🟧 85 Medium

Findings (90)

Choose a row to view the finding details. All findings are related to this instance.

Finding status: Active
Filter criteria: *Add filter*

Resource ID *EQUALS* i-07e50a59baa5e19f5 ✕
Clear filters

1 2 3

	Severity	Title	Type	Age	Status
○	🟥 High	CVE-2025-22088 - linux-image-aws	Package Vulnerability	7 minutes	Active
○	🟥 High	CVE-2025-21680 - linux-image-aws	Package Vulnerability	7 minutes	Active
○	🟥 High	CVE-2025-21692 - linux-image-aws	Package Vulnerability	7 minutes	Active
○	🟥 High	CVE-2024-57951 - linux-image-aws	Package Vulnerability	7 minutes	Active
○	🟧 Medium	CVE-2025-37896 - linux-image-aws, linux-image-aws	Package Vulnerability	7 minutes	Active
○	🟧 Medium	CVE-2025-37922 - linux-image-aws, linux-image-aws	Package Vulnerability	7 minutes	Active
○	🟧 Medium	CVE-2025-37919 - linux-image-aws, linux-image-aws	Package Vulnerability	7 minutes	Active

Here is the vulnerability summary automatically generated by Amazon Inspector for our EC2 instance.

We had a total of 90 findings, categorized by severity: 4 High, 85 Medium and 1 Low.

Among these:

- 88 findings were Package Vulnerabilities, mostly from outdated packages such as linux-image-aws.
- The remaining 2 findings were related to Network Reachability, which means certain open ports could be reachable from the internet.

Each finding in this list can be clicked for more details, and AWS Inspector provides information about recommended remediation steps.

This summary provides a clear overview of which vulnerabilities require urgent attention and will guide our next steps in remediation.

3) Finding Details

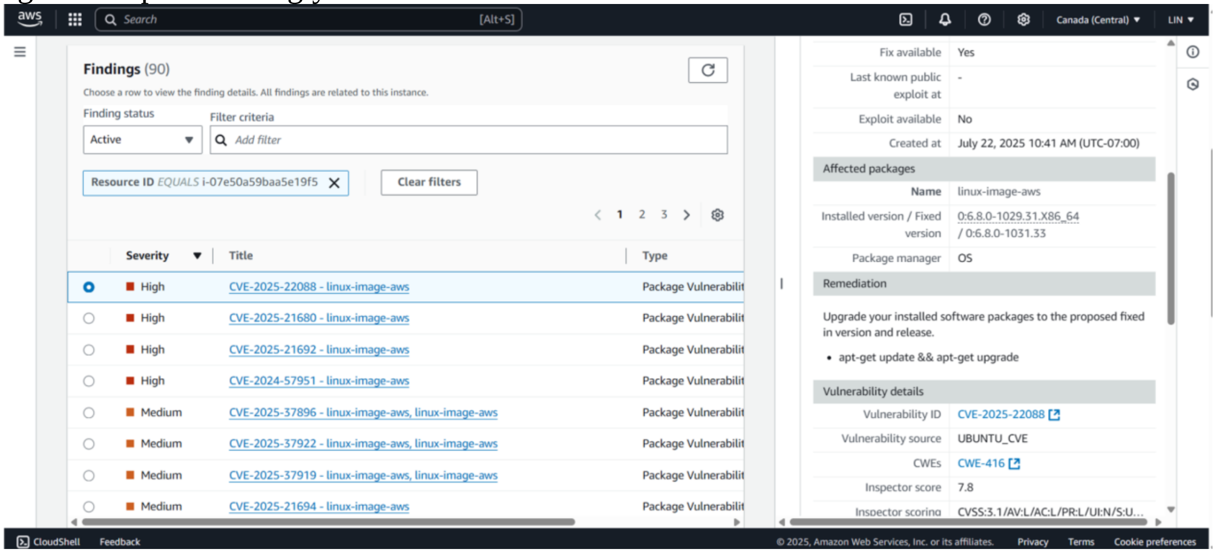
Each finding in list can be clicked for more details. Here, we take a closer look at one of the High severity findings identified by Amazon Inspector.
This specific finding is related to a known vulnerability: CVE-2025-22088, which affects the *linux-image-aws* package.

- The installed version on our EC2 instance is 6.8.0-1029.31, which is flagged as vulnerable.
- Amazon Inspector provides a clear remediation recommendation: we need to upgrade the package to a patched version using standard commands like *apt-get update && apt-get upgrade*.

In the vulnerability metadata, we can also see:

- The vulnerability source is UBUNTU_CVE.
- The CVSS score is 7.8, which classifies it as High severity.
- The CWE ID is CWE-416, which refers to a Use After Free vulnerability — a common and dangerous memory corruption flaw.

All this detailed information helps us assess the risk level more precisely and prioritize our mitigation steps accordingly.



Findings (90)

Choose a row to view the finding details. All findings are related to this instance.

Finding status: Active

Filter criteria: Add filter

Resource ID *EQUALS* i-07e50a59baa5e19f5

	Severity	Title	Type
●	🟥 High	CVE-2025-22088 - linux-image-aws	Package Vulnerabilit
○	🟥 High	CVE-2025-21680 - linux-image-aws	Package Vulnerabilit
○	🟥 High	CVE-2025-21692 - linux-image-aws	Package Vulnerabilit
○	🟥 High	CVE-2024-57951 - linux-image-aws	Package Vulnerabilit
○	🟧 Medium	CVE-2025-37896 - linux-image-aws, linux-image-aws	Package Vulnerabilit
○	🟧 Medium	CVE-2025-37922 - linux-image-aws, linux-image-aws	Package Vulnerabilit
○	🟧 Medium	CVE-2025-37919 - linux-image-aws, linux-image-aws	Package Vulnerabilit
○	🟧 Medium	CVE-2025-21694 - linux-image-aws	Package Vulnerabilit

Fix available	Yes
Last known public exploit at	-
Exploit available	No
Created at	July 22, 2025 10:41 AM (UTC-07:00)
Affected packages	
Name	linux-image-aws
Installed version / Fixed version	0:6.8.0-1029.31.X86_64 / 0:6.8.0-1031.33
Package manager	OS

Remediation

Upgrade your installed software packages to the proposed fixed in version and release.

- apt-get update && apt-get upgrade

Vulnerability details	
Vulnerability ID	CVE-2025-22088
Vulnerability source	UBUNTU_CVE
CWEs	CWE-416
Inspector score	7.8
Inspector scoring	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U...

6. Vulnerability Remediation and Security Improvement

To enhance the security posture of our AWS-hosted Ubuntu server, we implemented a comprehensive remediation plan to address vulnerabilities identified through multiple scanning tools, including Nmap, Nikto, and AWS Inspector. This section details the remediation strategies applied, the tools used, and the effectiveness of the solutions.

1) Nmap and Nikto Scan Findings and Remediation

To address the vulnerabilities identified during the initial Nmap and Nikto scans, we implemented several hardening measures on the Apache web server configuration.

First, to prevent the exposure of server version and operating system details, we modified the Apache configuration file (/etc/apache2/apache2.conf) by adding the following command:

- *ServerTokens Prod*
- *ServerSignature Off*

Next, to improve HTTP response security and mitigate missing headers, we added the following recommended headers to enhance protection against clickjacking, MIME sniffing, and XSS attacks:

- *Header always append X-Frame-Options SAMEORIGIN*
- *Header set X-XSS-Protection "1; mode=block"*
- *Header set X-Content-Type-Options nosniff*

To stop inode information leakage detected by Nikto, we disabled ETag with the following setting:

FileETag None

Additionally, to reduce the attack surface, we restricted allowed HTTP methods to only GET and POST using:

```
<LimitExcept GET POST>
    Deny from all
</LimitExcept>
```

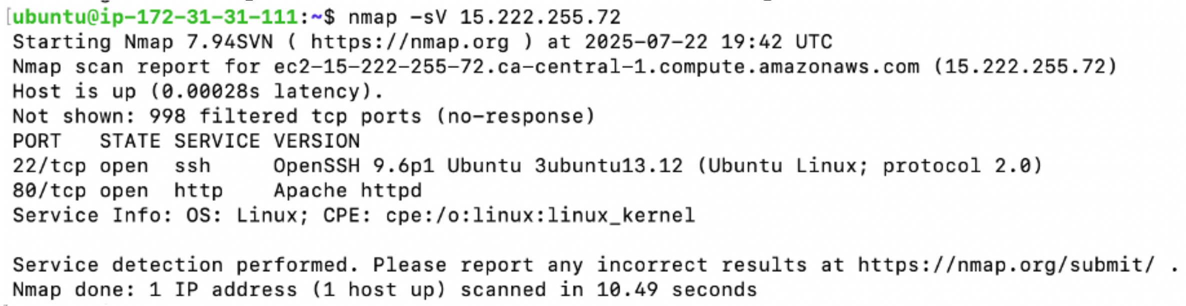
After applying these changes, we enabled the Apache headers module:

sudo a2enmod headers

We then restarted the Apache service to apply the configuration:

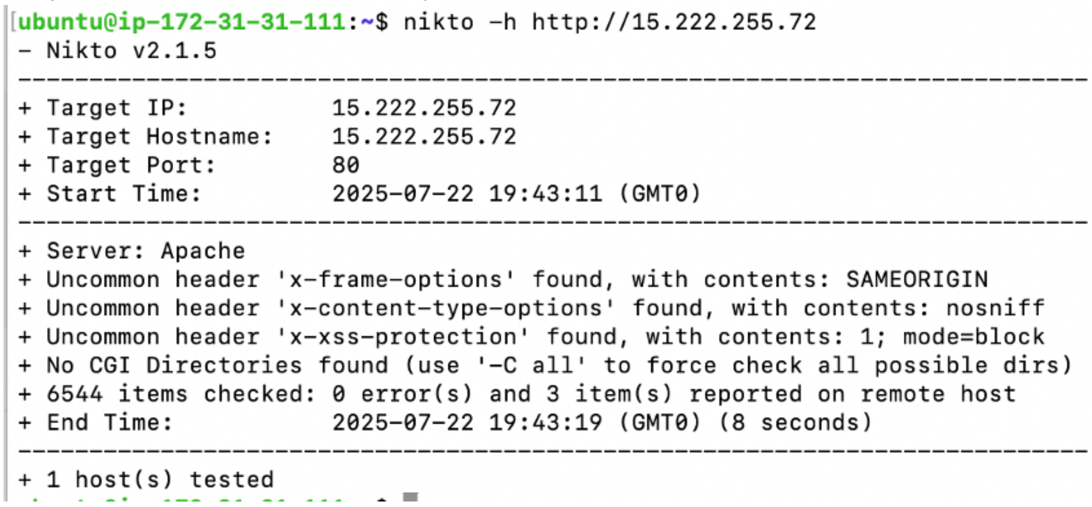
sudo systemctl restart apache2

Finally, we performed follow-up scans using Nmap and Nikto. The results confirmed that the server no longer exposed version details, the recommended headers were present, inode leaks were resolved, and only the intended HTTP methods were allowed. These remediations significantly improved the security posture of the system and resolved the issues flagged in the initial scans. The optimized scan results are shown in the figures below:



```
ubuntu@ip-172-31-31-111:~$ nmap -sV 15.222.255.72
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-22 19:42 UTC
Nmap scan report for ec2-15-222-255-72.ca-central-1.compute.amazonaws.com (15.222.255.72)
Host is up (0.00028s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.12 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.49 seconds
```



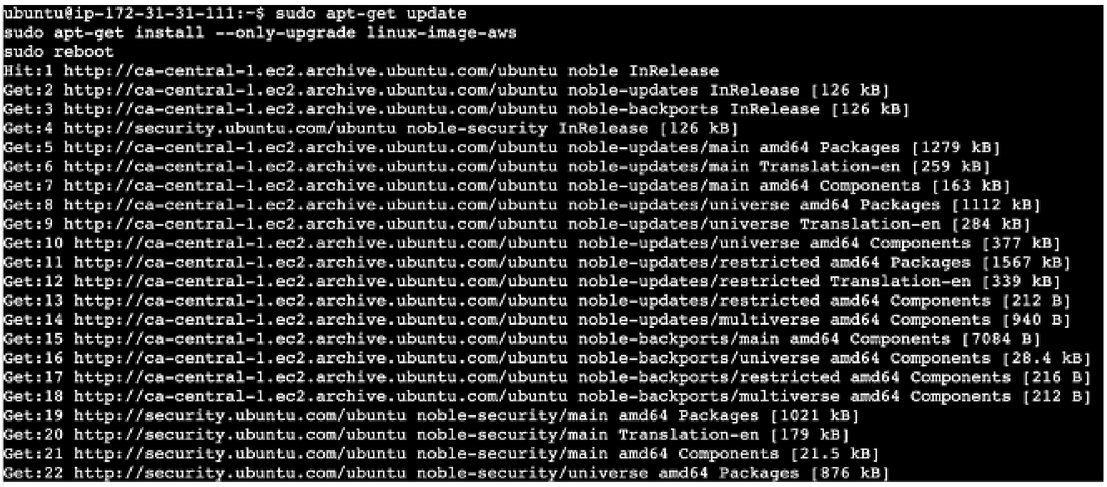
```
ubuntu@ip-172-31-31-111:~$ nikto -h http://15.222.255.72
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          15.222.255.72
+ Target Hostname:    15.222.255.72
+ Target Port:        80
+ Start Time:         2025-07-22 19:43:11 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache
+ Uncommon header 'x-frame-options' found, with contents: SAMEORIGIN
+ Uncommon header 'x-content-type-options' found, with contents: nosniff
+ Uncommon header 'x-xss-protection' found, with contents: 1; mode=block
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 6544 items checked: 0 error(s) and 3 item(s) reported on remote host
+ End Time:           2025-07-22 19:43:19 (GMT0) (8 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

2) AWS Inspector Vulnerability Remediation

To address the security issues discovered by AWS Inspector, we focused primarily on kernel-level vulnerabilities associated with the linux-image-aws package. The initial scan revealed 55 vulnerabilities, including multiple high-severity CVEs such as CVE-2025-22088, CVE-2025-21680, CVE-2025-21692, and CVE-2024-57951. These vulnerabilities were related to outdated kernel components and package configurations.

To remediate these, we updated the system by running the following commands:

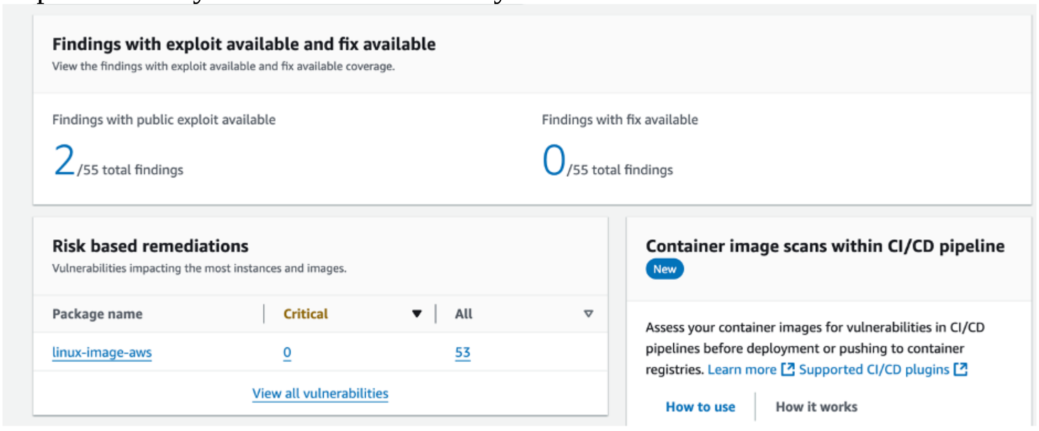
- *sudo apt-get update*
- *sudo apt-get install --only-upgrade linux-image-aws*
- *sudo reboot*



```
ubuntu@ip-172-31-31-111:~$ sudo apt-get update
sudo apt-get install --only-upgrade linux-image-aws
sudo reboot
Hit:1 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1279 kB]
Get:6 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [259 kB]
Get:7 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [163 kB]
Get:8 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1112 kB]
Get:9 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [284 kB]
Get:10 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [377 kB]
Get:11 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [1567 kB]
Get:12 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [339 kB]
Get:13 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:14 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:15 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7084 B]
Get:16 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [28.4 kB]
Get:17 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:18 http://ca-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:19 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1021 kB]
Get:20 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [179 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:22 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [876 kB]
```

This ensured the AWS EC2 instance was running the latest available kernel version. After rebooting, we verified the kernel version using uname -r, which confirmed the system was now operating with kernel version 6.8.0-1031-aws, indicating a successful upgrade.

A follow-up scan using AWS Inspector showed that 35 vulnerabilities were successfully resolved, including all the previously mentioned high-severity issues. This significantly improved the system's baseline security and reduced the attack surface at the OS level.



Findings with exploit available and fix available
View the findings with exploit available and fix available coverage.

Findings with public exploit available
2/55 total findings

Findings with fix available
0/55 total findings

Risk based remediations
Vulnerabilities impacting the most instances and images.

Package name	Critical	All
linux-image-aws	0	53

View all vulnerabilities

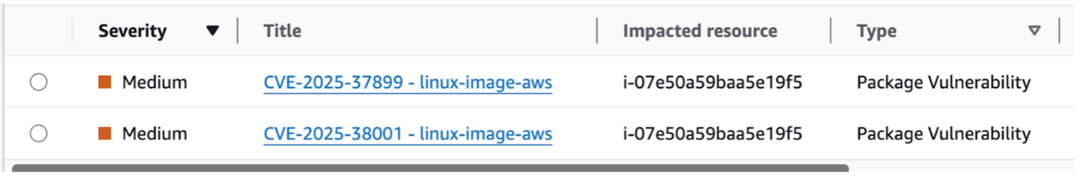
Container image scans within CI/CD pipeline

Assess your container images for vulnerabilities in CI/CD pipelines before deployment or pushing to container registries. Learn more Supported CI/CD plugins

3) Unresolved Vulnerabilities

However, two vulnerabilities remained unresolved:

- CVE-2025-37899
- CVE-2025-39001

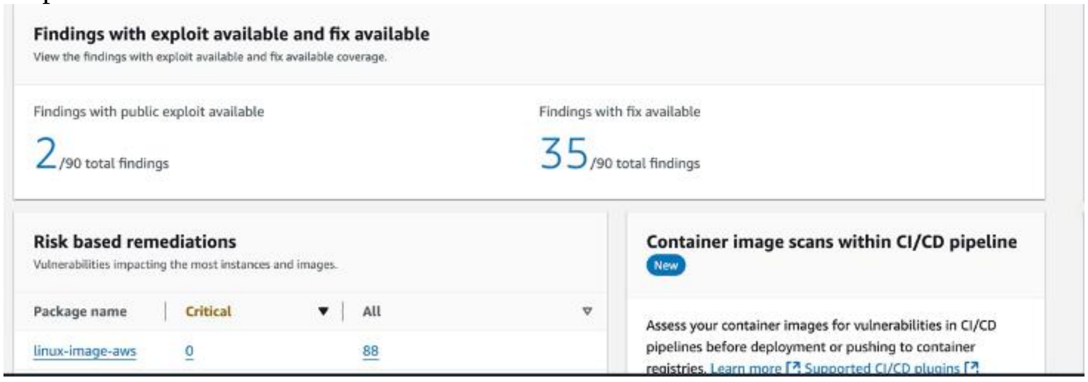


	Severity	Title	Impacted resource	Type
○	🟧 Medium	CVE-2025-37899 - linux-image-aws	i-07e50a59baa5e19f5	Package Vulnerability
○	🟧 Medium	CVE-2025-38001 - linux-image-aws	i-07e50a59baa5e19f5	Package Vulnerability

These are classified as medium severity, and both have public exploits available, increasing their risk level. Despite this, no official patch is currently available from AWS for these CVEs. This is confirmed by the Inspector's report stating -“Findings with fix available: 0/55”.

We will continue monitoring the patch releases and apply updates as soon as they are made available. Meanwhile, we have implemented additional compensating controls such as limiting network access, monitoring logs, and enforcing least privilege on services running on the affected instance.

The screenshots below demonstrate the pre- and post-remediation results captured by AWS Inspector.



Findings with exploit available and fix available

View the findings with exploit available and fix available coverage.

Findings with public exploit available

2/90 total findings

Findings with fix available

35/90 total findings

Risk based remediations

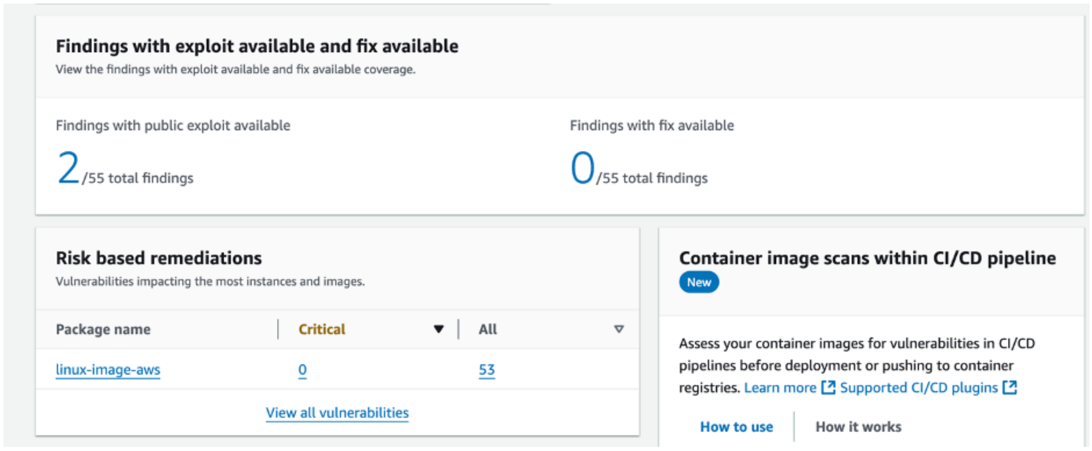
Vulnerabilities impacting the most instances and images.

Package name	Critical	All
linux-image-aws	0	88

Container image scans within CI/CD pipeline

New

Assess your container images for vulnerabilities in CI/CD pipelines before deployment or pushing to container registries. Learn more Supported CI/CD plugins



Findings with exploit available and fix available

View the findings with exploit available and fix available coverage.

Findings with public exploit available

2/55 total findings

Findings with fix available

0/55 total findings

Risk based remediations

Vulnerabilities impacting the most instances and images.

Package name	Critical	All
linux-image-aws	0	53

View all vulnerabilities

Container image scans within CI/CD pipeline

New

Assess your container images for vulnerabilities in CI/CD pipelines before deployment or pushing to container registries. Learn more Supported CI/CD plugins

How to use | How it works

7. Conclusion

Through this case study, we successfully identified and mitigated multiple security risks on our AWS EC2 instance. By configuring the Apache web server, applying HTTP security headers, and restricting unnecessary services, we significantly reduced network exposure. The use of AWS Inspector allowed us to detect and remediate high-severity kernel vulnerabilities, improving the server’s overall security posture. While a few medium-severity vulnerabilities remain unresolved due to the lack of available patches, we have applied compensating controls and continue to monitor updates. This project demonstrates the importance of proactive vulnerability assessment, continuous monitoring, and applying layered security measures to ensure cloud infrastructure resilience.

Then, we will re-execute the project steps to record a demo video. Please note that the IP address, vulnerability scan results, and timestamps in the video may differ slightly from those in the report or slides, due to changes during environment re-deployment.