

Using Amazon Inspector for vulnerability assessment and remediation

The screenshot shows the AWS Lambda Lab interface. At the top, there's a navigation bar with 'AWS' (with a green status icon), 'Used \$0 of \$1', '00:38', and buttons for 'Start Lab', 'End Lab', 'AWS Details', 'Details', 'Submit' (highlighted in blue), 'Submission Report', and 'Grades'. Below the navigation is a dropdown menu set to 'EN-US'. The main content area has a title 'Using Amazon Inspector for vulnerability assessment and remediation' and a 'Lab overview' section. The 'Lab overview' includes a paragraph about utilizing Amazon Inspector for AWS Lambda functions, a paragraph about AnyCompany's needs for automated security, and a bullet point about Amazon Inspector's capabilities. There's also a 'Objectives' section with a list of tasks to be completed.

Using Amazon Inspector for vulnerability assessment and remediation

Lab overview

In this lab, you utilize Amazon Inspector to scan for vulnerabilities in your AWS resources, specifically AWS Lambda functions. You learn how to activate Amazon Inspector, interpret the vulnerability reports, and remediate the findings.

The developers at AnyCompany are in the initial phases of building an application primarily using AWS Lambda. Throughout the development process, they need an automated security tool that not only scans for vulnerable software packages, but also scans within the code itself. You decide to utilize Amazon Inspector to fill this need.

Amazon Inspector meets the requirements of being able to scan AWS Lambda functions by quickly responding to new deployments. It also automatically scans additional resources such as EC2 instances, Amazon ECRs within AnyCompany's AWS account.

Objectives

After completing this lab, you should be able to:

- Activate Amazon Inspector.

Objectives

After completing this lab, you should be able to:

- Activate Amazon Inspector.
- Analyze and interpret vulnerability findings.
- Remediate the vulnerabilities found by Amazon Inspector.

Duration

This lab requires approximately **30 minutes** to complete.

Lab environment

The environment has Lambda functions with vulnerabilities which will be subsequently scanned by Amazon inspector and reported as per severity.

1: Activate the Amazon Inspector

MOKGADI SELEPE

Screenshot 1: AWS Home Page - Inspector Updates

The screenshot shows the AWS Home page with several promotional banners for Amazon Inspector:

- Inspector now supports deep inspection of EC2 Instances.
- Introducing the new AWS Security Hub - public preview
- Amazon Inspector now supports mapping of container images to running containers
- Amazon Inspector Launches Code Security for Comprehensive Software Development Lifecycle protection

A central callout box for "Amazon Inspector" highlights "automated and continual vulnerability management at scale".

Screenshot 2: AWS Home Page - Inspector Overview

The screenshot shows the AWS Home page with a large "Get started with Inspector" callout box. It includes a "Free 15-day trial for accounts new to Inspector" offer and a "Get Started" button.

Screenshot 3: Inspector Service Page

The screenshot shows the "Inspector" service page in the AWS Management Console. It features a "How it works" diagram illustrating the workflow from enabling Inspector to receiving customizable findings.

Pricing and Documentation sections are also visible on the right side of the page.

MOKGADI SELEPE

The screenshot shows two overlapping windows. The top window is titled 'Activate Inspector' and is part of the 'Inspector' service. It displays the 'Service permissions' section, which explains that activating Inspector grants it permission to discover, classify, and protect sensitive data in AWS. A 'View role permissions' button and an 'Activate Inspector' button are visible. The bottom window is titled 'Feedback for Amazon Inspector'. It starts with a thank you message and a request for feedback on the experience using Amazon Inspector. It includes a 5-point Likert scale for responses: Strongly disagree, Disagree, Neither disagree or agree, Agree, and Strongly agree. There are five statements to rate:

Statement	Strongly disagree	Disagree	Neither disagree or agree	Agree	Strongly agree
I found the summary overview very useful	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system was easy to navigate	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I found the type of findings that I was looking for	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We may want to contact you about your feedback. If you agree, please provide your email address. - <i>optional</i>	<input type="text"/>				

A note at the bottom states: 'Personal information you provide to us will be handled in accordance with the AWS Privacy Notice (<https://aws.amazon.com/privacy/>).'

At the bottom of the feedback window are 'Cancel' and 'Submit' buttons.

MOKGADI SELEPE

The image contains two screenshots of the AWS Inspector dashboard. The top screenshot shows the initial state where the environment coverage is at 0%. The bottom screenshot shows the final state where the environment coverage is at 100%.

Initial State (Top Screenshot):

- Environment coverage:** 0 / 0 instances, 0 / 0 repositories, 0 / 0 Lambda functions, 0 / 0 Code repositories.
- Critical findings:** 0 Critical, 0 total findings for ECR container, EC2 instance, Lambda functions, and Code repositories.

Final State (Bottom Screenshot):

- Environment coverage:** 100% coverage achieved across all categories.
- Critical findings:** 0 Critical, 0 total findings for ECR container, EC2 instance, Lambda functions, and Code repositories.

I just turned on Amazon Inspector so it can keep checking my Lambda functions.

-First I typed “Inspector” in the console search bar and opened the service. In the left-hand menu I clicked “Activate Inspector” and then pressed the button to turn it on for my account – that’s only needed the first time.

-After it started, a welcome message appeared and a scan began automatically.

-I closed the survey prompt and any banner messages, then refreshed the page a few times until the dashboard showed that all my Lambda functions were covered (100 %).

The dashboard also shows my account number and that Inspector is now active, with standard scanning enabled for EC2, ECR and Lambda by default.

2: Reviewing the inspected resources

2.1: Reviewing your Lambda functions

Findings: All findings

Severity	Title	Impacted resource	Type	Age	Status
Medium	CVE-2023-32681 - requests	get-request	Package Vulnerability	an hour	Active
Medium	CVE-2024-47081 - requests	get-request	Package Vulnerability	an hour	Active
Medium	CVE-2024-35195 - requests	get-request	Package Vulnerability	an hour	Active

CVE-2023-32681 - requests

Finding ID: arn:aws:inspector:us-west-2:415302997546:finding/05ebe1eb912cdd51376a94e8abcc8498

Requests is a HTTP library. Since Requests 2.3.0, Requests has been leaking Proxy-Authorization headers to destination servers when redirected to an HTTPS endpoint. This is a product of how we use 'rebuild_proxies' to reattach the 'Proxy-Authorization' header to requests. For HTTP connections sent through the tunnel, the proxy will identify the header in the request itself and remove it prior to forwarding to the destination server. However when sent over HTTPS, the 'Proxy-Authorization' header must be sent in the CONNECT request as the proxy has no visibility into the tunneled requests. This results in Requests forwarding proxy credentials to the destination server unintentionally, allowing a malicious actor to potentially exfiltrate sensitive information. This issue has been patched in version 2.31.0.

Finding details

Inspector score and vulnerability intelligence

Finding overview

AWS account ID: 415302997546
Severity: Medium

MOKGADI SELEPE

The screenshot shows the NIST National Vulnerability Database (NVD) website. At the top, there's a header with the NIST logo and "Information Technology Laboratory" text. Below the header, the "NATIONAL VULNERABILITY DATABASE" is prominently displayed. A green navigation bar at the bottom left includes the "VULNERABILITIES" tab. The main content area shows a detailed view of CVE-2023-32681. On the left, under "Current Description", it says: "Requests is a HTTP library. Since Requests 2.3.0, Requests has been leaking Proxy-Authorization headers to destination servers when redirected to an HTTPS endpoint. This is a product of how we use `rebuild_proxies` to reattach the `Proxy-Authorization` header to requests. For HTTP connections sent through the tunnel, the proxy will identify the header in the request itself and remove it prior to forwarding to the destination server. However when sent over HTTPS, the `Proxy-Authorization` header must be sent in the CONNECT request as the proxy has no visibility into the tunneled request. This results in Requests forwarding proxy credentials to the destination server unintentionally, allowing a malicious actor to potentially exfiltrate sensitive information. This issue has been patched in version 2.31.0." To the right, a "QUICK INFO" sidebar lists details: "CVE Dictionary Entry: CVE-2023-32681", "NVD Published Date: 05/26/2023", "NVD Last Modified: 02/13/2025", and "Source: GitHub, Inc.". At the bottom of the main content area, there's a "Metrics" section with tabs for CVSS Version 4.0, CVSS Version 3.x (selected), and CVSS Version 2.0. It shows a CNA icon for GitHub, Inc., a base score of 6.1 MEDIUM, and a vector string of CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:N/A:N.

I opened the “Findings” section on the left and clicked “All findings”.

-Three rows showed up, one for each vulnerability that was found in my Lambda functions. Each row told me the severity (it was medium), which Lambda function was affected, and a short title explaining the issue.

-I selected the radio button next to the finding titled “CVE-2023-32681 – requests”. That opened a side pane with more details about the vulnerability. In that pane I clicked the external link next to the Vulnerability ID, which took me to a page on the National Vulnerability Database (NVD) – a site run by NIST that lists standardized info about security flaws. That page gave me a deeper look at the problem.

MOKGADI SELEPE

Back in the pane I saw the “Remediation” section, which said the requests package I was using is outdated and has a security hole, and that I should upgrade it. I’ll use that recommendation to fix the issue in my Lambda function.

3: Remediating the vulnerabilities findings

3.1: Remediating your Lambda function’s Package Vulnerabilities

The screenshot shows two overlapping AWS Lambda console pages. The top page is a detailed view of a vulnerability finding for the 'get-request' Lambda function. It highlights the 'Lambda' service entry and provides remediation steps, including upgrading the Python package manager and file paths. The right side of this panel displays vulnerability details such as CVE-2023-32681, NVD source, and a CVSS score of 6.1. The bottom page shows a list of functions under the 'Functions' tab, with 'get-request' and 'generate-password-for-new-account' listed. The sidebar on the left includes sections for Dashboard, Findings (with 'All findings' selected), Services (Lambda, CodeBuild, AWS Signer), Features (Lambda Insights, Object Lambda Access Points, Local processing), and Additional resources (Code signing configurations, Event source mappings, Layers, Replicas). A 'Related AWS resources' section at the bottom links to Step Functions state machines.

MOKGADI SELEPE

The screenshots illustrate the process of creating and configuring a Lambda function named 'get-request'.

Screenshot 1: Function Overview

This screenshot shows the 'Function overview' page for the 'get-request' function. It includes:

- A diagram view showing the function's structure.
- A 'Layers' section indicating 0 layers.
- A 'Related functions' dropdown set to 'Select a function'.
- A 'Description' field.
- A 'Last modified' timestamp: 2 hours ago.
- A 'Function ARN' field: arn:aws:lambda:us-west-2:415302997546:function:get-request.
- An 'Application' field: c183903a476869712744372t1w415302997546.
- A 'Function URL' field.

Screenshot 2: Code Source

This screenshot shows the 'Code source' tab. The code editor displays the 'index.py' file:

```
import requests
from requests.exceptions import RequestException
def lambda_handler(event, context):
    try:
        response = requests.get("https://api.example.com/data")
    except RequestException as e:
        return {"statusCode": 500, "body": str(e)}
    if response.status_code == 200:
        data = response.json()
        return data
    else:
        return {"statusCode": response.status_code, "body": response.text}
```

The sidebar shows the project structure with 'index.py' and 'requirements.txt' in the 'GET-REQUEST' folder. Deployment options include 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'.

Screenshot 3: Requirements

This screenshot shows the 'Code source' tab with the 'requirements.txt' file open. The file contains:

```
requests==2.20.0
```

The sidebar shows the project structure with 'index.py' and 'requirements.txt' in the 'GET-REQUEST' folder. Deployment options include 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'.

MOKGADI SELEPE

The image displays three separate screenshots of the AWS Lambda, Amazon Inspector, and AWS Inspector interfaces.

AWS Lambda Screenshot: Shows the Lambda console with a green success message: "Successfully updated the function get-request." The code source editor shows a file structure with "requirements.txt" and "index.py". Deployment options include "Deploy (Ctrl+Shift+U)" and "Test (Ctrl+Shift+I)".

Amazon Inspector Screenshot: Shows the search results for "Amazon Inspector". It lists services like Amazon Inspector, AWS Private Certificate Authority, and Amazon Redshift, along with features like Create Macie allow list, Amazon Chime, and AMIs.

AWS Inspector Screenshot: Shows the AWS Inspector dashboard with a message about a 15-day free trial for EC2 scanning. It displays the "Findings" section, which is currently empty, showing a table header for Severity, Title, Impacted resource, Type, Age, and Status.

MOKGADI SELEPE

Findings:

Findings ranked by severity.

Findings (3)

Choose a row to see the finding details.

Severity	Title	Impacted resource	Type	Age	Status
Medium	CVE-2023-32681 - requests	get-request	Package Vulnerability	2 hours	Closed
Medium	CVE-2024-47081 - requests	get-request	Package Vulnerability	2 hours	Closed
Medium	CVE-2024-35195 - requests	get-request	Package Vulnerability	2 hours	Closed

Account management:

Manage your accounts, and review the coverage of your instances, repositories, images and Lambda functions.

Instances (0)

No instances

Lambda functions:

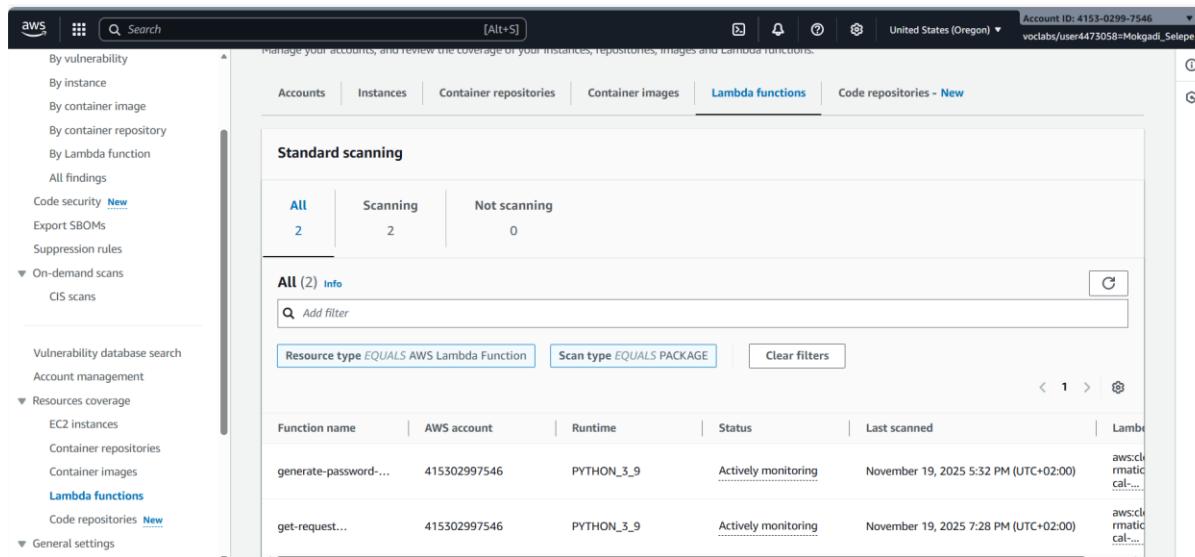
Manage your accounts, and review the coverage of your instances, repositories, images and Lambda functions.

Standard scanning

All (2)

Function name	AWS account	Runtime	Status	Last scanned	Lambda function tags
generate-password...	415302997546	PYTHON_3_9	Actively monitoring	November 19, 2025 ...	aws:cloudformation:logical...

MOKGADI SELEPE



The screenshot shows the AWS Lambda functions page. On the left, there's a sidebar with various navigation options like 'By vulnerability', 'Accounts', 'Instances', 'Container repositories', 'Container images', 'Lambda functions' (which is selected), and 'Code repositories - New'. The main content area is titled 'Standard scanning' and shows a table with two rows. The columns are 'Function name', 'AWS account', 'Runtime', 'Status', and 'Last scanned'. The first row has 'generate-password...' as the function name, '415302997546' as the AWS account, 'PYTHON_3_9' as the runtime, 'Actively monitoring' as the status, and 'November 19, 2025 5:32 PM (UTC+02:00)' as the last scan time. The second row has 'get-request...' as the function name, '415302997546' as the AWS account, 'PYTHON_3_9' as the runtime, 'Actively monitoring' as the status, and 'November 19, 2025 7:28 PM (UTC+02:00)' as the last scan time.

I opened the AWS console and searched for Lambda.

I picked the get-request function, opened its code editor, and found the requirements.txt file. I removed the version number (==2.20.0) from the line that said requests==2.20.0 so it just read requests. That tells Lambda to use the latest version of the package. I hit Deploy and a banner popped up saying the function was updated successfully.

Because the function changed, Amazon Inspector started a new scan automatically. I went back to the console, searched for Inspector, and opened the Findings page. I switched the filter from “Active” to “Closed” and saw that CVE-2023-32681 – requests now appears in the closed list, which means the vulnerability has been fixed.

Finally, I looked under Resources coverage → Lambda functions and saw that the last-scanned timestamp for the function was updated, confirming the new scan finished.

Conclusion

I turned on Amazon Inspector and set it up for my account, then I looked at the security problems it found in my Lambda function. After that, I fixed the issues by updating the vulnerable package, so the findings are now closed and my function is secure.