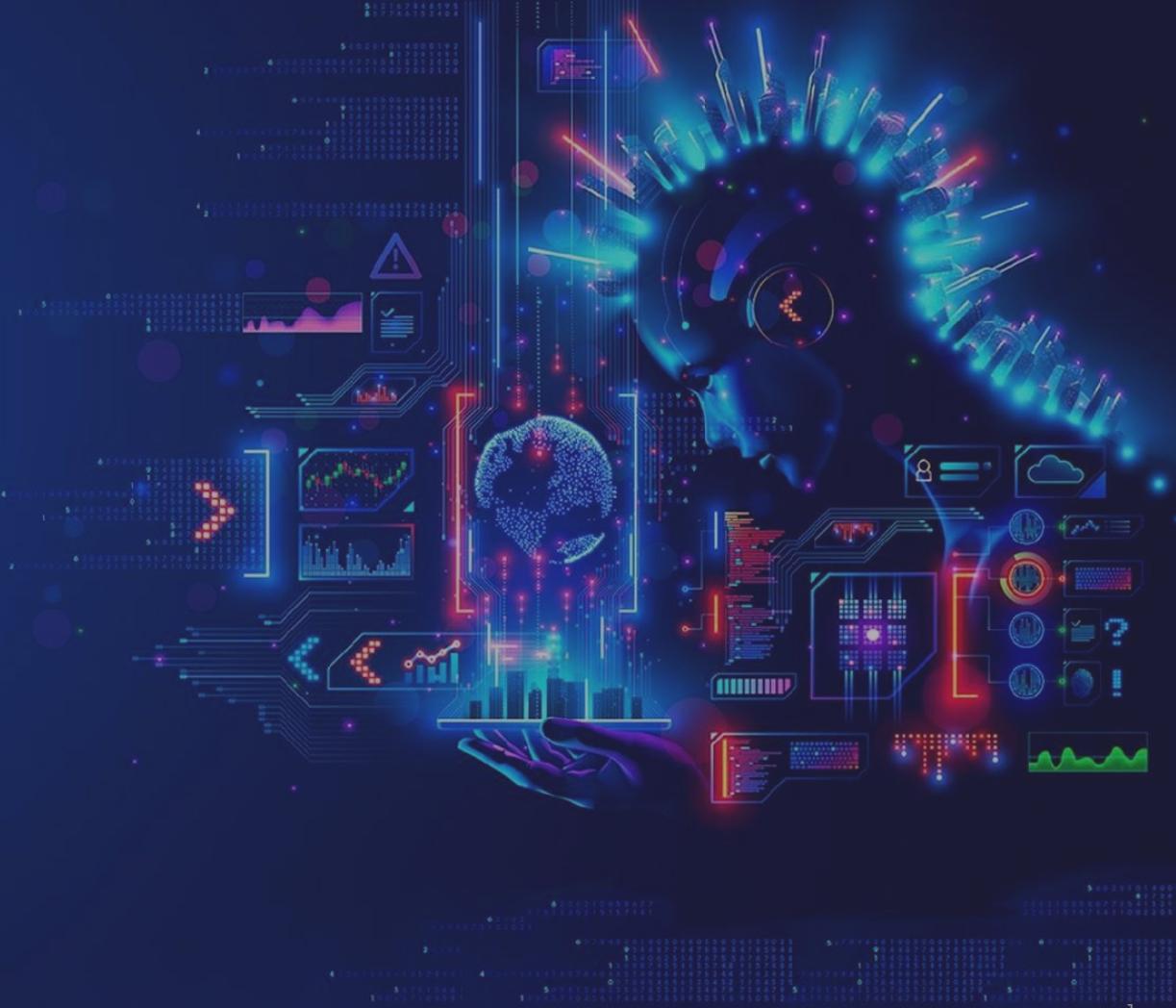




ASPM User Playbook





Container Scan

ASPM Integration

Generate AccuKnox API token for CI/CD pipeline



- To generate a token, open AccuKnox and navigate to Settings > Tokens > Create.
- Copy the token and tenant ID, then configure them as secrets in your CI/CD pipeline.

The screenshot shows the AccuKnox web interface. On the left, a sidebar menu is visible with items like Monitors / Alerts, Identity, Reports, Notifications, Settings (which is highlighted with a red box), Manage Clusters, User Management, RBAC, Integrations, Labels, Tags, Groups, and Tokens (which is highlighted with a blue box). Below the sidebar, there's an 'Ask Ada' button and a 'Getting started: Onboarding' section with Cloud Accounts, Clusters, and Registry options.

The main content area shows a 'Tokens' page with a search bar and a 'Create' button (also highlighted with a red box). A modal window titled 'Create API Token' is open. In the modal, the 'Name' field contains 'Container Scan'. The 'Expiration' dropdown is set to '30 Days'. The 'Tenant Id' field contains '107' (also highlighted with a red box). A note below the expiration field states: 'This token will only be shown once. Copy it now and store it in a safe, secure location.' It also includes a warning: 'By default, this token will expire on Saturday, Sep 7 2024.' At the bottom of the modal are 'Cancel' and 'Generate' buttons (the latter is highlighted with a red box).

In the background, the main tokens list shows several entries with columns for 'Used', 'Tag', and dates like '2024-08-07 (2)', '2024-08-05 (46)', '2024-07-27 (545)', and '2024-07-23 (9)'. The 'Rows per page' dropdown is set to 10, and there are navigation buttons at the bottom.

Configuring the container scan in GitHub actions



Step 1: Add [AccuKnox Container Scan](#) to Your GitHub Workflow

- Open your GitHub repository and navigate to your workflow file (typically .github/workflows/your-workflow.yml).
- After the build step, add the AccuKnox container scan GitHub Action.

Step 2: Run the Workflow

- Push your changes to trigger the workflow, or manually run it from the "Actions" tab in your repository.

Step 3: Review Findings in AccuKnox

- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select Container Image Findings.
- Click on any finding that interests you to view detailed information and recommendations.

```
jobs:  
  accuknox-cicd:  
    runs-on: ubuntu-latest  
    steps:  
      - name: Checkout code  
        uses: actions/checkout@main  
  
      - name: Build Docker image  
        run: |  
          docker buildx build .  
  
      - name: AccuKnox Container Scan  
        uses: accuknox/container-scan-action@v0.0.1  
        with:  
          token: ${{ secrets.TOKEN }}  
          tenant_id: ${{ secrets.TENANT_ID }}
```

Configuring the container scan in Jenkins



- Download the plugin in .hpi format from [here](#).
- Navigate to the Jenkins dashboard.
- Go to Manage Jenkins > Plugins > Advance settings.
- Deploy the plugin.

Jenkins

Dashboard > Manage Jenkins

- + New Item
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins**
- My Views

New version of Jenkins (2.452.3) is available for download ([changelog](#)).

System Configuration

- System Configure global settings and paths.
- Nodes Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Tools Configure tools, their locations and automatic installers.
- Clouds Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance Configure the look and feel of Jenkins

Build Queue No builds in the queue.

Deploy Plugin

You can select a plugin file from your local system or provide a URL to install a plugin from outside the configured update site(s).

File

Choose file **accuknox-containerscan.hpi**

Or

URL

Deploy

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings**

56

56

Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

- Open the configuration page of your Jenkins job.
- Under the Build section, click on Add build step and select scan image with AccuKnox.
- Fill all the required parameters and trigger the pipeline.
- Review the findings in AccuKnox > Issues > Findings > Container scan.

Dashboard > test > Configuration

Configure

Build Steps

Scan Image with AccuKnox

Image Name:

Image Tag: latest

Exit Code: 0

Severity: UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL

AccuKnox Token:

Tenant ID:

AccuKnox Label:

Buttons: Save, Apply

Dashboard > test > Configuration

Configure

Source Code Management

None **(6)**

Build Triggers

Build after other projects are built ?
 Build periodically ?
 Poll SCM ?

Build Steps

Add build step ▾

- Execute Windows batch command
- Execute shell
- Invoke top-level Maven targets
- Scan Image with AccuKnox**

Buttons: Save, Apply

Jenkins

Dashboard > test >

Configure **(4)**

Status, Changes, Workspace, Build Now, Configure, Delete Project, Rename

6

View Findings in registry scan page



- You can see the findings on the registry scan page.
- Click any of the findings to get more details.

The screenshot illustrates the ACCUKNOX platform's user interface for viewing container scan findings. On the left, the main navigation bar includes links for Dashboard, Inventory, Issues, Vulnerabilities, and Registry Scan (which is highlighted with a red box). Below this, there are sections for Compliance, Runtime Protection, Remediation, Monitors / Alerts, Reports, Notifications, and Settings, along with an 'Onboarding' guide for Cloud Accounts, Clusters, and Registry.

The central part of the interface shows the 'Findings' tab selected under the 'Scan Queue' section. It displays a list of repositories, specifically focusing on the 'accuknox-container-scan-example' repository. For each image entry, there is a color-coded bar indicating the number of findings: 0 (green), 36 (red), 0 (orange), 0 (yellow), and 0 (grey). A red arrow points from the '36' value in the first row to a detailed view of one of the findings.

The detailed view on the right is titled 'accuknox-container-scan-example:8185438405'. It includes tabs for Overview, Vulnerabilities, Resources, Sensitive Data, Scan History, and Layers. The 'Overview' tab is active. Key details shown include:

- Architecture: amd64
- Content Digest: sha256:2142c9c325700516bd7f217f7b56c63499a7696e33bac7d9848040b625b740f2
- Created: 03/07/2024 02:28 PM
- Docker Digest: sha256:2f42c9c325700516bd7f217f7b56c63499a7696e33bac7d9848040b625b740f2
- Docker Labels: (empty)
- Docker Version: (empty)
- Environment: PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
LANG=C.UTF-8
GPG_KEY=0D96DF4D4110E5C43FBFB17F2D347EA6AA65421D
PYTHON_VERSION=3.6.15
PYTHON_PIP_VERSION=21.2.4
PYTHON_SETUPTOOLS_VERSION=57.5.0
PYTHON_GET_PIP_URL=https://github.com/pypa/get-pip/raw/3cb8888cc2869620f57d5d2da64ada38f516078c7/public/get-pip.py
PYTHON_GET_PIP_SHA256=c51b250a91a70d7b20ccebf5272209a4ded2a0c263ae5776f129e0d9b5674309
- Operating System: linux (alpine)

A large red circle highlights the 'Total 36' count in the 'Vulnerability Scan Details' section, which also lists the same findings as the detailed view.

View Findings in Findings page



- Alternatively, you can view the findings on the **Findings** page. Select the **Container Image Findings** to access the relevant details.

The screenshot shows the AccuKnox web application interface. On the left, there is a dark sidebar with various navigation options: Dashboard, Inventory, Issues, Findings (which is highlighted with a red box), Registry Scan, Compliance, Runtime Protection, Remediation, Monitors / Alerts, Identity, Reports, Notifications, and Settings. At the bottom of the sidebar, there is an 'Ask Ada' button and a 'Getting started: Onboarding' section with links for Cloud Accounts, Clusters, and Registry.

The main content area has a header with 'Home > Issues > Findings'. Below the header, there is a search bar and several filter options: 'Container Image Findings' (which is also highlighted with a red box), 'Asset', 'Group by', 'Saved Filters', and a sorting/filtering icon. The main table is titled 'Container Image Findings' and lists various findings categorized under CIS K8s Benchmark Findings, Host-Endpoint Findings, Cloud Findings, Static Code Analysis Finding, IaC Findings, AWS SecurityHub Findings, and Cluster Findings. The table columns are: Identification numbers, Name, Assetname, Risk factor, Pkg name, and a more options icon. The table shows 10 rows of findings, each with a checkbox and some descriptive text. At the bottom of the table, it says 'Total Records: 50373' and has a page navigation bar with buttons for 1, 2, 3, 4, 5, ..., 2519, and >.

- Use case 1: Dependency analysis - scanning for supply chain vulnerabilities
- Use case 2: Scan for sensitive data exposure
- Use case 3: Authentication Vulnerabilities
- Use case 4: Remote Code Execution (RCE) vulnerabilities
- Use case 5: Denial of Service (DoS) vulnerabilities

Use case 1: Dependency analysis - scanning for supply chain vulnerabilities



- This jetty server 9.2.26 is vulnerable to XSS
- AccuKnox container scan identifies this vulnerability
- Provides you the solution

jetty: using specially formatted URL against DefaultServlet or ResourceHandler leads to XSS conditions: (org.eclipse.jetty:jetty-server@7.6.0.v20120127) Medium Edit X

Description	Result	Solution	References	Source Code
In Eclipse Jetty version 9.2.26 and older, 9.3.25 and older, and 9.4.15 and older, the server is vulnerable to XSS conditions if a remote client USES a specially formatted URL against the DefaultServlet or ResourceHandler that is configured Show More...				

● Finding for in resource Container | rajvanshi/storm:latest
● Failing since about 1 month ago, on 23/07/2024
● In progress since 23/07/2024

Details [+ Create Ticket](#)

Asset
rajvanshi/storm:latest
Asset Type
Container
Status ● Active
Ignored No
Severity ● Medium
Tickets 0
Notes (1)
Add Comments and Press Ctrl + Enter to Submit

jetty: using specially formatted URL against DefaultServlet or ResourceHandler leads to XSS conditions: (org.eclipse.jetty:jetty-server@7.6.0.v20120127)

Description	Result	Solution	References	Source Code
Upgrade to version 9.2.27.v20190403, 9.3.26.v20190403, 9.4.16.v20190411 of the package org.eclipse.jetty:jetty-server				

Details [+ Create Ticket](#)

Asset
rajvanshi/storm:latest
Asset Type

Use case 2: Scan for sensitive data exposure



- This container image have multiple RSA private keys
- AccuKnox container scans for the sensitive data exposure and reports it.

rajvanshi/juice-shop:latest

Overview Vulnerabilities Resources **Sensitive Data** Scan History Layers

RSA private Key

File Name	Full Path
last-login-ip.component.spec.ts	/juice-shop/frontend/src/app/last-login-ip/last-login-ip.component.spec.ts
app.guard.spec.ts	/juice-shop/frontend/src/app/app.guard.spec.ts
insecurity.js	/juice-shop/build/lib/insecurity.js
insecurity.ts	/juice-shop/lib/insecurity.ts

Use case 3: Authentication Vulnerabilities



- The apache derby is a JDBC driver
- In this case it's vulnerable to a broken authentication
- AccuKnox proposes a solution to upgrading it to version 10.14.3

Finding X

A cleverly devised username might bypass LDAP authentication checks. I ...: Critical

(org.apache.derby:derby@10.10.2.0)

Description

A cleverly devised username might bypass LDAP authentication checks. In LDAP-authenticated Derby installations, this could let an attacker fill up the disk by creating junk Derby databases. In LDAP-authenticated Derby installations, this

[Show More...](#)

Compliance Frameworks

No compliance found

Solution

Upgrade to version 10.14.3, 10.15.2.1, 10.16.1.2, 10.17.1.0 of the package org.apache.derby:derby

- Jackson is a popular Java library used for processing JSON, here it's vulnerable to code execution.
- AccuKnox identifies the vulnerability and suggests to updating Jackson to version 2.9.10.4

Finding

X

**A deserialization flaw was discovered in jackson-databind through 2.9.:
(com.fasterxml.jackson.core:jackson-databind@2.6.3)**

High

Description

A deserialization flaw was discovered in jackson-databind through 2.9.10.4. It could allow an unauthenticated user to perform code execution via ignite-jta or quartz-core:
org.apache.ignite.cache.jta.jndi.CacheJndiTmLookup,
org.apache.ignite.cache.jta.jndi.CacheJndiTmFactory, and
org.quartz.utils.JNDIConnectionProvider.

Show Less...

Solution

Upgrade to version 2.9.10.4 of the package
com.fasterxml.jackson.core:jackson-databind

Use case 5: Denial of Service (DoS) vulnerabilities



- Netty codec is a java library, here it's vulnerable to a Denial of Service attack via a memory leakage.
- AccuKnox identifies the issue and reports it.

netty-codec: Bzip2Decoder doesn't allow setting size restrictions for decompressed data: (io.netty:netty-codec@4.1.30.Final)

High



Description Result Solution References Source Code

The Bzip2 decompression decoder function doesn't allow setting size restrictions on the decompressed output data (which affects the allocation size used during decompression). All users of Bzip2Decoder are affected. The malicious input can trigger an OOME and so a DoS attack

Show Less...

Details

+ Create Ticket

Asset

rajvanshi/storm:latest

Asset Type

Container

Status



IaC Scan

ASPM Integration

Generate AccuKnox API token for CI/CD pipeline



- To generate a token, open AccuKnox and navigate to **Settings > Tokens > Create**.
- Copy the token and tenant ID, then configure them as secrets in your CI/CD pipeline.

The screenshot shows the AccuKnox web interface. On the left, a dark sidebar menu is visible with the following items: Monitors / Alerts, Identity, Reports, Notifications, Settings (which is expanded), Cloud Accounts, Manage Clusters, User Management, RBAC, Integrations, Labels, Tags, Groups, Tokens (which is highlighted with a red box), and Ticket Template. At the bottom of the sidebar, there's an 'Ask Ada' button and a 'Getting started: Onboarding' section with links for Cloud Accounts, Clusters, and Registry.

The main content area shows a 'Tokens' page with a search bar and a 'Create' button. A modal window titled 'Create API Token' is open. In the modal, the 'Name' field contains 'Container Scan', the 'Expiration' dropdown is set to '30 Days', and the 'Tenant Id' field contains '167'. Below the Tenant Id field, a note says: 'By default, This token will expire on Saturday, Sep 7 2024.' At the bottom of the modal is a 'Generate' button, which is also highlighted with a red box. The background of the tokens list shows several entries with columns for 'Used' (e.g., '+08-07 (2)'), 'Tag' (e.g., 'None'), and 'Created' date (e.g., '+08-05 (46)').

Configuring the IAC scan in GitHub actions



Step 1: Add [AccuKnox IAC scan GitHub action](#) to your workflow like this image.

Step 2: Run the Workflow

- Push your changes to trigger the workflow, or manually run it from the "Actions" tab in your repository.

Step 3: Review Findings in AccuKnox

- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select IaC Findings.
- Click on any finding that interests you to view detailed information and recommendations.

```
...
jobs:
tests:
  runs-on: ubuntu-latest
  steps:
    - name: Checkout code
      uses: actions/checkout@main

    - name: Run IaC scan
      uses: accuknox/iac-scan-action@v0.0.1
      with:
        directory: ./results
        output_file_path: ./results
        token: ${{ secrets.TOKEN }}
        endpoint: ${{ secrets.ENDPOINT}}
        tenant_id: ${{ secrets.TENANT_ID}}
        quiet: "true"
        soft_fail: "true"
```

Configuring the IaC scan in Jenkins



- Download the plugin in .hpi format from [here](#).
- Navigate to the Jenkins dashboard.
- Go to Manage Jenkins > Plugins > Advance settings.
- Deploy the plugin.

The screenshot shows the Jenkins dashboard with the 'Manage Jenkins' menu item highlighted by a red circle labeled 1. The 'Manage Jenkins' menu is open, showing options like 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', and 'Manage Jenkins'. The 'Manage Jenkins' option is also highlighted with a red box.

The screenshot shows the Jenkins 'Deploy Plugin' dialog and the 'Plugins' section of the 'Manage Jenkins' page. The 'Deploy Plugin' dialog has a 'File' input field containing 'accuknox-containerscan.hpi' highlighted with a red box and circled with a red number 3. The 'Plugins' section of the 'Manage Jenkins' page has an 'Advanced settings' button highlighted with a red box and circled with a red number 2.

- Open the configuration page of your Jenkins job.
- Under the Build section, click on Add build step and select AccuKnox IaC scan.
- Fill all the required parameters.
- Review the findings in AccuKnox > Issues > Findings > Container scan.

Dashboard > accuknox-iac-scan > Configuration

Configure

Build Steps

AccuKnox IaC Scan

Directory:

File:

Soft Fail: true

Framework: all

Repository:

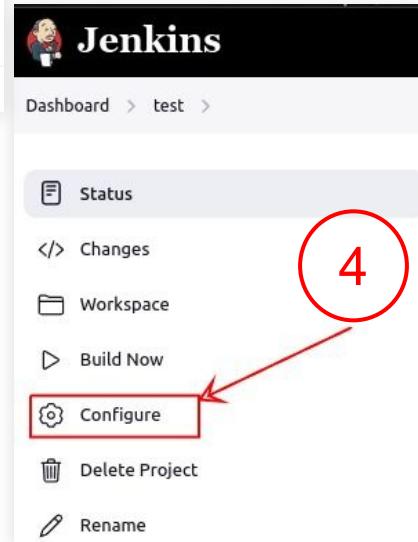
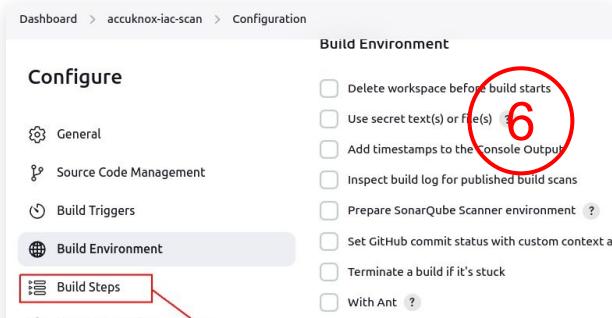
Branch:

AccuKnox Token:

Tenant ID:

Add build step ▾

Save Apply



- Go to the AccuKnox > Findings and select the IAC Scan from the drop down menu

The screenshot shows the AccuKnox web interface. On the left, a dark sidebar contains various navigation items: Dashboard, Inventory, Issues (with a red box around it), Findings (also with a red box around it), Registry Scan, Risk-based Prioritization, Compliance, Runtime Protection, Collectors, Remediation, Monitors / Alerts, Reports (with a blue 'Ask Ada BETA' button), and Getting started: Onboarding (with Cloud Accounts, Clusters, and Regions). The main content area is titled 'Findings'. At the top right of this area is a dropdown menu currently set to 'IAC Scan' (also highlighted with a red box), with other options like 'Asset' and 'X' available. Below this are three filter dropdowns: 'Risk factor', 'Status', and 'Asset Type'. A 'Reset' button is located at the bottom left of the findings search area. To the right of the search area is an 'Edit' button. The main content area displays a table titled 'Default Configuration' with three rows: 'Static Code Analysis', 'Cloud Missconfiguration', and 'CIS Kubernetes Benchmarks v1.23'. To the right of the table is a search bar, a 'Ticket Configuration' dropdown, a 'Group by' dropdown, and several icons for edit, copy, delete, and more. Below the table is a 'COLUMNS' section with headers: Last seen, Findings, Asset, Status, Data type, and Exploit. One row is visible in the table, showing data for an IAC Scan.

	Last seen	Findings	Asset	Status	Data type	Exploit
<input type="checkbox"/>	2024-03-10	Ensure DB instan...	chirag8680006/t...	Active	IAC Scan	False

- To get a detailed view of a finding click on the finding and then click on the arrow icon.

Home > Issues > Findings > Details

Create Collectors

Ensure the S3 bucket has access logging enabled

Severity: Not_available

Status: Active

Exploitability: False

Discovered: 2 Minutes Ago

Description
<https://github.com/accuknox/cloud-docs/tree/main/docs/en/enterprise-edition/policy-reference/aws-policies/s3-13-enable-logging.adoc>

Solution
<https://github.com/accuknox/cloud-docs/tree/main/docs/en/enterprise-edition/policy-reference/aws-policies/s3-13-enable-logging.adoc>

Ticket Comments
0 comments available

Show comments

Ensure the S3 bucket has access logging enabled

Asset: [REDACTED] terraform-aws-example:None

Asset Type: github-repository

Location: [REDACTED] terraform-aws-example/blob/None/main.tf#L1-L10

Status: Active

Ignored: No

Tickets: 0

Severity: Not Available

Ticket Config **+**

Save

Use case: Security group misconfigurations



- This security group is misconfigured and allows ingress connections from any IP in the world to the port 80

Ensure no security groups allow ingress from 0.0.0.0:0 to port 80 Low X

Description	Result	Solution	References	Source Code
Allowing ingress from 0.0.0.0/0 to port 80 (i.e. the HTTP port) can expose your Amazon Web Services (AWS) resources to potential security threats. This is because 0.0.0.0/0 represents all IP addresses, and allowing traffic from all IP addresses to port 80 can make it easier for attackers to access your resources. By ensuring that your AWS security groups do not allow ingress from 0.0.0.0/0 to port 80, you can help protect your resources from potential attacks and unauthorized access. Instead, you should specify the IP addresses or ranges of IP addresses that are allowed to access your resources, and only allow traffic from those sources. Show Less...				

Details	+ Create Ticket
Asset Testing_Script.t Asset Type IaC_IAC-Repository Status ● Active Ignored No Severity ●	



AST Scan

ASPM Integration

Generate AccuKnox API token for CI/CD pipeline



- To generate a token, open AccuKnox and navigate to **Settings > Tokens > Create**.
- Copy the token and tenant ID, then configure them as secrets in your CI/CD pipeline.

The screenshot shows the AccuKnox web interface. On the left, the sidebar includes options like Monitors / Alerts, Identity, Reports, Notifications, Settings (with Cloud Accounts highlighted), Manage Clusters, User Management, RBAC, Integrations, Labels, Tags, Groups, Tokens (highlighted), Ticket Template, and Ask Add. The main area shows a list of existing tokens with columns for Name, Used, and Tag. A modal window titled "Create API Token" is open in the center. It contains fields for Name (Container Scan), Expiration (30 Days), Tenant Id (167), and a note about the token expiring on Saturday, Sep 7 2024. There are "Cancel" and "Generate" buttons at the bottom of the modal. The "Generate" button is highlighted with a red box. The "Create" button in the top right of the main interface is also highlighted with a red box.

SonarQube configuration



- Deploy a SonarQube VM. Refer this [guide here](#).
- Create a `sonar-project.properties` file into your GitHub repository.
- To generate a SonarQube token, go to SonarQube > My Account > security and click on generate button.

A screenshot of the SonarQube interface. At the top, there's a navigation bar with tabs: Profile, Security (which is underlined), Notifications, and Projects. On the left, a sidebar shows the user name "Administrator". Below the navigation bar, the main content area has a heading "Tokens". A text block explains how tokens can be used for security instead of user credentials. At the bottom, there's a form with a text input field labeled "Enter Token Name" and a "Generate" button.

3

A screenshot of the SonarQube "My Account" page. The user "Administrator" is logged in. The navigation menu includes "My Account" (which is highlighted with a purple background and circled in red with the number 2) and "Log out".

2

Required metadata
sonar.projectKey=github_sonar_example
sonar.projectName=GitHub Sonar Example
sonar.projectVersion=1.0
Path to source directories (required)
sonar.sources=.
Encoding of the source files
sonar.sourceEncoding=UTF-8
Additional settings
sonar.host.url=<http://your_sonarqube_server:9000>
sonar.login=your_project_token

1

Configuring the SAST scan in GitHub actions



Step 1: Add these steps to your GitHub workflow.

Step 2: Configure these Parameters as GitHub Secrets `SONAR_TOKEN`, `SQ_URL`, `SQ_PROJECTS`, `AK_URL`, `TENANT_ID`, `ACCUKNOX_TOKEN`

Step 3: Run the Workflow

- Push your changes to trigger the workflow, or manually run it from the "Actions" tab in your repository.

Step 4: Review Findings in AccuKnox

- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select SAST Findings.
- Click on any finding that interests you to view detailed information and recommendations.

```
...  
  
jobs:  
  sonarqube_sast:  
    runs-on: ubuntu-latest  
    steps:  
      - uses: actions/checkout@v4  
        with:  
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis  
      - uses: sonarsource/sonarqube-scan-action@master  
        env:  
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}  
          SONAR_HOST_URL: ${{ secrets.SQ_URL }}  
      - name: Run AccuKnox SAST job  
        run: |  
          docker run --rm \  
            -e SQ_URL=${{ secrets.SQ_URL }} \  
            -e SQ_AUTH_TOKEN=${{ secrets.SONAR_TOKEN }} \  
            -e REPORT_PATH=/app/data/ \  
            -e SQ_PROJECTS="^github_sonar_example$" \  
            -v $PWD:/app/data/ \  
            accuknox/sastjob:latest  
      - name: Upload SAST reports  
        env:  
        run: |  
          cd ${GITHUB_WORKSPACE}  
          for file in `ls -1 SQ-*.json`; do  
            curl --location --request POST "<https://$AK_URL/api/v1/artifact/?tenant_id=${secrets.tenant_id}&data_type=SQ&save_to_s3=false>" \  
              --header "Tenant-Id: ${secrets.tenant_id}" \  
              --header "Authorization: ${secrets.accuknox_token}" \  
              --form "file=@\"$file\""  
          done
```

Configuring the SAST scan in Jenkins



- Go to Manage Jenkins > Tools and add the SonarQube installation details.
- Create SonarQube credentials.
- Go to Manage Jenkins > System Configurations. Select the check-box of Injecting Environment variables and add the details of the SonarQube Server

SonarQube Scanner

Name: scanner-name

Required

Install automatically

Install from Maven Central

Version: SonarQube Scanner 5.0.1.3006

Add Installer

Manage Jenkins > Credentials > System > Global credentials (unrestricted)

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: sonarqube

ID: sonarqube

Description:

SonarQube servers

If checked, job administrators can inject environment variables defined here into their builds.

Environment variables

SonarQube installations

List of SonarQube installations

- Go to Manage Jenkins > Tools and add the SonarQube installation details.
- Create SonarQube credentials.
- Go to Manage Jenkins > System Configurations. Select the check-box of Injecting Environment variables and add the details of the SonarQube Server

```

pipeline {
    agent any
    environment {
        SONAR_TOKEN = credentials('sonar-token') // Replace with your Jenkins credential ID
    for SonarQube token
        SQ_URL = credentials('sq-url')           // Replace with your Jenkins credential ID
    for SonarQube URL
        AK_URL = credentials('ak-url')          // Replace with your Jenkins credential ID
    for AccuKnox URL
        TENANT_ID = credentials('tenant-id')     // Replace with your Jenkins credential ID
    for Tenant ID
        AK_TOK = credentials('ak-tok')          // Replace with your Jenkins credential ID
    for AccuKnox token
    }
    stages {
        stage('Checkout Code') {
            steps [
                checkout([$class: 'GitSCM', branches: [[name: '/main']]],
doGenerateSubmoduleConfigurations: false, extensions: [$class: 'CloneOption', depth: 0]],
userRemoteConfigs: [[url: 'YOUR_GIT_REPO_URL']]] // Replace with your Git repository URL
            }
        stage('SonarQube Analysis') {
            steps [
                withSonarQubeEnv('SonarQube') { // Replace 'SonarQube' with your SonarQube
server configuration name in Jenkins
                    sh """
                        sonar-scanner \
                            -Dsonar.projectKey=your_project_key \
                            -Dsonar.sources=. \
                            -Dsonar.host.url=${SQ_URL} \
                            -Dsonar.login=${SONAR_TOKEN}
                    """
                }
            }
        stage('Run AccuKnox SAST') {
            steps [
                sh """
                    docker run --rm \
                        -e SQ_URL=${SQ_URL} \
                        -e SQ_AUTH_TOKEN=${SONAR_TOKEN} \
                        -e REPORT_PATH=/app/data/ \
                        -e SQ_PROJECTS="AccuKnox-Sonarqube-Usecase$" \
                        -v ${pwd}:/app/data/ \
                        accuknox/sastjob:latest
                """
            }
        stage('Upload SAST Reports') {
            steps [
                script {
                    def files = sh(script: 'ls -1 SQ-x.json', returnStdout: true).trim().split('\n')
                    for (file in files) {
                        sh """
                            curl --location --request POST
                            "https://${AK_URL}/api/v1/artifact/?tenant_id=${TENANT_ID}&data_type=SQ&save_to_s3=false" \
                                --header "Tenant-Id: ${TENANT_ID}" \
                                --header "Authorization: Bearer ${AK_TOK}" \
                                --form "file=@${file}"
                        """
                    }
                }
            }
        }
    }
}

```

- Trigger the workflow, go to the AccuKnox > Findings and select the Static Code Analysis findings here.

The screenshot shows the AccuKnox interface with the 'Findings' menu item selected. The 'Static Code Analysis Finding' dropdown is open, and the first finding in the list is expanded to show detailed information:

Assetname	Name	Risk factor	Description	Status
udit-uniyal_Awesome-...	Use the opposite opera...	Low	Why is this an issue? It is n...	Active
udit-uniyal_Awesome-...	Using http protocol is in...	Low	Clear-text protocols such as ...	Active
udit-uniyal_Awesome-...	Define a constant instea...	Critical	Why is this an issue? Duplicat...	Active
udit-uniyal_Awesome-...	Make sure that using thi...	Medium	Using pseudorandom num...	Active
udit-uniyal_Awesome-...	Using http protocol is in...	Low	Clear-text protocols such as ...	Active
udit-uniyal_Awesome-...	Using http protocol is in...	Low	Clear-text protocols such as ...	Active
udit-uniyal_Awesome-...	Using http protocol is in...	Low	Clear-text protocols such as ...	Active
udit-uniyal_Awesome-...	Make sure that using thi...	Medium	Using pseudorandom num...	Active

Total Records: 348

- Use case 1: Privilege escalation
- Use case 2: XML External Entity (XXE) injection
- Use case 3: Hard coded password
- Use case 4: Cross Site Request Forgery (CSRF)
- Use case 5: Remote Code Execution (RCE)

Use case 1: Privilege escalation

- An I am policy in this code is vulnerable to privilege escalation attack.
- AccuKnox identifies this vulnerability and suggests a solution.

This policy is vulnerable to the "EC2" privilege escalation vector. Remove permissions or restrict the set of resources they apply to.

Critical  

Description	Result	Solution	References	Source Code	Details	+ Create Ticket
<p>Within IAM, identity-based policies grant permissions to users, groups, or roles, and enable specific actions to be performed on designated resources. When an identity policy inadvertently grants more privileges than intended, certain us</p> <p>Show More...</p>					Asset gitlab-sast-testing	
					Asset Type static_code_Software	
					Status 	

Use case 2: XML External Entity (XXE) injection



- This code have an XML parsing vulnerability XML External Entity injection.
- AccuKnox identifies the vulnerability and proposes a solution.

Disable access to external entities in XML parsing.

Critical



Description Result Solution References Source Code

<p>This vulnerability allows the usage of external entities in XML.</p>

<h2>Why is this an issue?</h2>

<p>External Entity Processing allows for XML parsing with the involvement of external entities. However, when this functionality is enabled, it can lead to security issues such as Denial of Service (DoS) attacks, information disclosure, and remote code execution. It's important to disable this feature or restrict its use to prevent these risks.</p>

Show More...

Details

+ Create Ticket

Asset

test-vulnerable-code-snippets

Asset Type

static_code_Software

Status

Use case 3: Hard coded password

- There is a hardcoded password in the source code.
- AccuKnox identifies the vulnerability, and proposes a solution.

Detected 'password' in this variable name, review this potentially hardcoded credential.

Critical  

Description	Result	Solution	References	Source Code
				<pre>2 // Exercise - 1 3 // Author: @TheXC3LL 4 // Website: Tarlogic.com 5 class login { 6 public \$username = "X-C3LL"; 7 public <u>\$password = "Insanity"</u>; 8 public \$role = "MUGGLE"; 9 } 10 \$one = new login(); 11 \$a = serialize(\$one); 12 echo "Example of an object:\n\$a\n\n"; 13 echo "FLAG: \n"; 14 \$test = unserialize(\$argv[1]); 15 \$check = \$test->role - 1337; 16 if (\$check == "ADMIN") {</pre>

Details [+ Create Ticket](#)

Asset
test-vulnerable-code-snippets

Asset Type
static_code_Software

Status  Active

Ignored  No

Severity 

Use case 4: Cross Site Request Forgery (CSRF)



- This is a CSRF vulnerability. Here an attacker can send arbitrary HTTP or HTTPS requests behalf of this web server.
- AccuKnox identifies this critical vulnerability and proposes a solution.

Make sure disabling CSRF protection is safe here. Critical 🔗 X

Description	Result	Solution	References	Source Code	Details	+ Create Ticket
<p>< p > A cross-site request forgery (CSRF) attack occurs when a trusted user of a web application can be forced, by an attacker, to perform sensitive actions that he didn't intend, such as updating his profile or sending a message, more generally anything that can change the state of the application. </ p ></p> <p>< p > The attacker can trick the user/victim to click on a link, corresponding to the privileged action, or to visit a malicious web site that embeds a hidden web request and as web browsers automatically include cookies, the actions can be authenticated and sensitive. </ p ></p> <p>Show Less...</p>					<p>Asset test-vulnerable-code-snippets</p> <p>Asset Type static_code_Software</p> <p>Status Active</p> <p>Ignored</p>	

Use case 5: Remote Code Execution (RCE)



- This code uses the eval() function, which is vulnerable to RCE.
- AccuKnox identifies the vulnerability within the source code and suggests a solution.

Make sure that this dynamic injection or execution of code is safe. Medium 🔗 ×

Description	Result	Solution	References	Source Code
				7 \$empty = 'No variable given'; 8 9 // pass the variable name into an eval block, making it 10 // vulnerable to Remote Code Execution (rce). This RCE 11 // is NOT blind. 12 eval('echo \$' . \$variable . ';') 13

Details + Create Ticket

Asset
test-vulnerable-code-snippets

Asset Type
static_code_Software

Status ✓



DAST Scan

ASPM Integration

Generate AccuKnox API token for CI/CD pipeline



- To generate a token, open AccuKnox and navigate to **Settings > Tokens > Create**.
- Copy the token and tenant ID, then configure them as secrets in your CI/CD pipeline.

The screenshot shows the AccuKnox web interface. On the left, a sidebar menu includes options like Cloud Accounts, Tickets, and Tokens, with the Tokens option highlighted by a red box. The main content area shows a list of existing tokens with columns for Used, Tag, and expiration dates. A modal window titled "Create API Token" is open in the center. It contains fields for Name (set to "Container Scan"), Expiration (set to "30 Days"), and Tenant Id (set to "167"). A note at the bottom of the modal states: "By default, This token will expire on Saturday, Sep 7 2024." The "Generate" button at the bottom right of the modal is also highlighted with a red box. The entire modal window is centered over the list of tokens.

Configuring the DAST scan in GitHub actions



Step 1: Add these steps to your GitHub workflow.

Step 2: Run the Workflow

- Push your changes to trigger the workflow, or manually run it from the "Actions" tab in your repository.

Step 3: Review Findings in AccuKnox

- Log in to your AccuKnox dashboard and navigate to the Issues section.

- Go to the "Findings" tab and select DAST Findings.
- Click on any finding that interests you to view detailed information and recommendations.

```
jobs:  
  zap:  
    runs-on: ubuntu-latest  
    steps:  
      - name: set permissions  
        run: sudo chmod -R 777 .  
  
      - name: workspace permissions  
        run: sudo chmod -R 777 ${github.workspace}  
  
      - name: zap scan  
        run: |  
          docker run --rm -v ${github.workspace}:/zap/wrk/:rw -t zaproxy/zap-stable zap-  
          baseline.py \  
            -t <url that you want to scan> \  
            -J report.json  
            -I  
  
      - name: zap  
        run: |  
          cd ${GITHUB_WORKSPACE}  
          curl --location --request POST 'https://${{ secrets.accuknox_url }}/api/v1/artifact/?tenant_id=${{ secrets.tenant_id }}&data_type=ZAP&label_id=dasttest&save_to_s3=false' \  
            --header 'Tenant-Id: ${{ secrets.tenant_id }}' \  
            --header 'Authorization: Bearer ${{ secrets.token }}' \  
            --form 'file=@\"report.json\"'
```

Configuring the DAST scan in Jenkins



Create a New Pipeline:

- Go to the Jenkins dashboard.
- Click on "New Item" to create a new pipeline.
- Name your pipeline and select "Pipeline" from the list, then click "OK."

Configure the Pipeline:

- In the pipeline configuration page, scroll down to the "Pipeline" section.
- Under "Definition," select "Pipeline script" from the dropdown.

Add the Script:

- In the script box that appears, paste your pipeline script.

Save the Configuration:

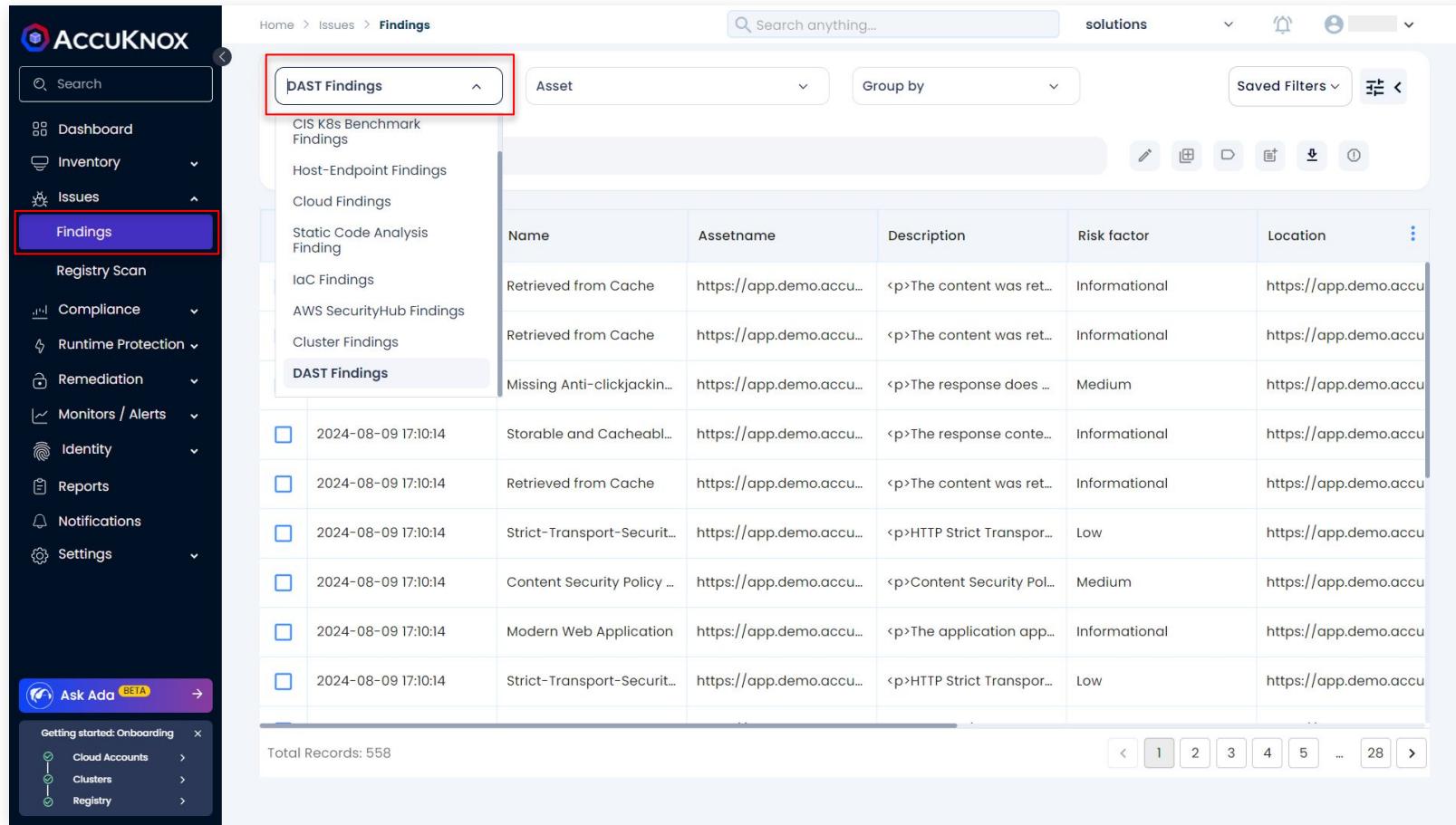
- After adding the script, click "Save" or "Apply."

Run the Pipeline:

- Go to the newly created pipeline and click "Build Now" to run the script.

```
...
pipeline {
    agent any
    environment {
        TENANT_ID = credentials('bearer-tenant_id') // Replace with your Jenkins credentials
        BEARER_TOKEN = credentials('bearer-token') // Replace with your Jenkins credentials
        ID for the tenant_id
        ID for the Bearer token
    }
    stages {
        stage('Checkout') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/main']], doGenerateSubmoduleConfigurations: false, extensions: [], userRemoteConfigs: [[url: 'YOUR_GIT_REPO_URL']]]) // Replace with your Git repository URL
            }
        }
        stage('Set Workspace Permissions') {
            steps {
                sh "sudo chmod -R 777 $PWD"
            }
        }
        stage('List Files') {
            steps {
                sh 'ls'
            }
        }
        stage('Run ZAP Scan') {
            steps {
                sh """
                    docker run --rm -v $PWD:/zap/wrk/:rw -t \
                        zaproxy/zap-stable zap-baseline.py \
                        -t <your-website> \
                        -r report.json \
                        -I
                """
            }
        }
        stage('Upload ZAP Report') {
            steps {
                sh """
                    curl --location --request POST
                    "https://cspn.demo.accuknox.com/api/v1/artifact/?"
                    tenant_id=${TENANT_ID}&data_type=ZAP&label_id=dasttest&save_to_s3=false" \
                    -header "Tenant-Id: ${TENANT_ID}" \
                    -header "Authorization: Bearer ${BEARER_TOKEN}" \
                    -F "file=@report.json"
                """
            }
        }
    }
}
```

- To get the findings, go to the AccuKnox > Findings and select the DAST here.



The screenshot shows the AccuKnox interface. On the left, there's a dark sidebar with various navigation options like Dashboard, Inventory, Issues, Findings (which is highlighted with a red box), Registry Scan, Compliance, Runtime Protection, Remediation, Monitors / Alerts, Identity, Reports, Notifications, and Settings. Below the sidebar, there's an 'Ask Ada' BETA button and sections for Cloud Accounts, Clusters, and Registry. The main area is titled 'Findings' and has a sub-section 'DAST Findings' also highlighted with a red box. The interface includes a search bar, dropdown filters for Asset and Group by, and a 'Saved Filters' button. A table lists various findings, including CIS K8s Benchmark Findings, Host-Endpoint Findings, Cloud Findings, Static Code Analysis Finding, IaC Findings, AWS SecurityHub Findings, Cluster Findings, and DAST Findings. The DAST Findings table has columns for Name, Assetname, Description, Risk factor, and Location. At the bottom, it says 'Total Records: 558' and shows a page navigation with numbers 1 through 28.

Name	Assetname	Description	Risk factor	Location
Missing Anti-clickjacking Headers	https://app.demo.accu...	<p>The response does not contain a Content-Security-Policy header that...	Medium	https://app.demo.accu...
Storable and Cacheable Response Headers	https://app.demo.accu...	<p>The response contains a Cache-Control header with a value that...	Informational	https://app.demo.accu...
Retrieved from Cache	https://app.demo.accu...	<p>The content was retrieved from a cache.	Informational	https://app.demo.accu...
Strict-Transport-Security Header	https://app.demo.accu...	<p>HTTP Strict Transport Security (HSTS) is not enabled for this site.	Low	https://app.demo.accu...
Content Security Policy Header	https://app.demo.accu...	<p>Content Security Policy (CSP) is not enabled for this site.	Medium	https://app.demo.accu...
Modern Web Application	https://app.demo.accu...	<p>The application appears to be a modern web application.	Informational	https://app.demo.accu...
Strict-Transport-Security Header	https://app.demo.accu...	<p>HTTP Strict Transport Security (HSTS) is not enabled for this site.	Low	https://app.demo.accu...

- Use case 1: Cross Origin Resource Sharing (CORS) misconfiguration
- Use case 2: File inclusion vulnerability
- Use case 3: Cross site scripting vulnerability
- Use case 4: SQL injection vulnerability
- Use case 5: Missing content security policy

Use case 1: Cross Origin Resource Sharing (CORS) misconfiguration



- This web application is vulnerable to CORS.
- CORS can lead to so many issues such as unauthorized access, cross site scripting, session hijacking and many other issues.
- AccuKnox identifies this vulnerability and proposes the solution.

Cross-Domain Misconfiguration

Medium



Description Result Solution References Source Code

<p>Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server.</p>

Details

+ Create Ticket

Asset

<https://juice-shop.herokuapp.com>

Asset Type

Use case 2: File inclusion vulnerability



- This web application is vulnerable to file inclusion.
- In file inclusion attack, and attacker can access files stored on a web server.
- This can lead to sensitive data leakage and source code leakage.
- AccuKnox identifies this vulnerability, and proposes a solution.

Source Code Disclosure - File Inclusion

High

[Description](#) [Result](#) [Solution](#) [References](#) [Source Code](#) [+ Create Ticket](#)

The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will exec

[Show More...](#)

Details	+ Create Ticket
Asset http://testphp.vulnweb.com	
Asset Type WebApp	
Status	

Use case 3: Cross site scripting vulnerability



- This web application have a cross site scripting vulnerability (XSS)
- XSS allows an attacker to inject arbitrary javascript code into a webpage
- AccuKnox identifies the vulnerability and proposes a solution to it

Source Code Disclosure - File Inclusion

High

Description	Result	Solution	References	Source Code
<pre><p>The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will exec</p></pre>				

[Show More...](#)

Details [+ Create Ticket](#)

Asset
http://testphp.vulnweb.com

Asset Type
WebApp

Status

Use case 4: SQL injection

- This web application is vulnerable to SQL injection attacks.
- SQL injection vulnerability allows an attacker to access the database of the website.
- AccuKnox identifies the vulnerability and proposes a solution.

SQL Injection - MySQL

High 

X

Description	Result	Solution	References	Source Code	Details	+ Create Ticket
<pre><p>SQL injection may be possible.</p></pre>					Asset http://testphp.vulnweb.com	
 Finding for in resource WebApp http://testphp.vulnweb.com					Asset Type WebApp	
 Failing since about 1 month ago, on 21/07/2024					Status 	
 Last detected about 1 month ago, on 21/07/2024					 Active	

Use case 5: Missing content security policy



- This web application don't have a content security policy.
- The CSP adds a layer of security to the application.
- AccuKnox identifies this misconfiguration and proposes a solution.

Content Security Policy (CSP) Header Not Set Medium 🔗 X

Description	Result	Solution	References	Source Code	Details	+ Create Ticket
<p><p>Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (xss) and data injection attacks. These attacks are used for everything from data theft.</p> <p>Show More...</p>					<p>Asset https://juice-shop.herokuapp.com</p> <p>Asset Type WebApp</p>	