AccuKnox

ASPM User Playbook

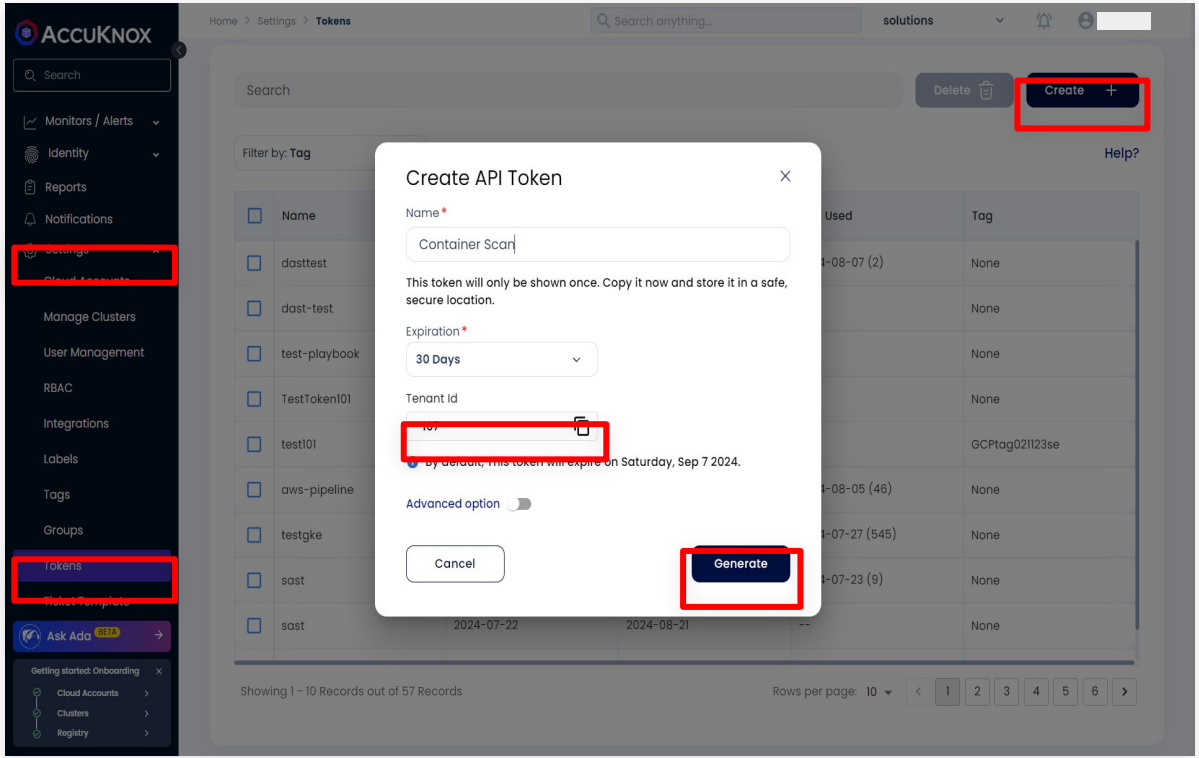
AccuKnox

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.



Configuring the container scan in GitHub actions

ACCUKNOX

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

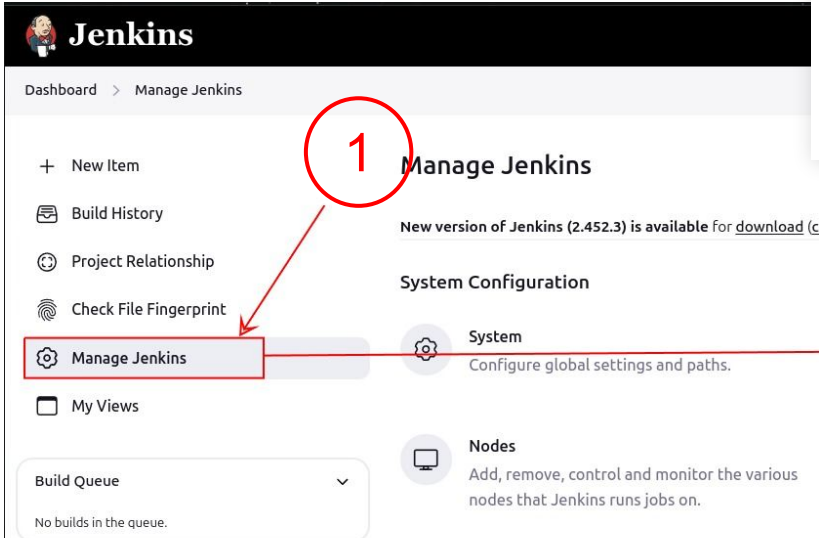
- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select Container Image Findings.
- Click on any finding that interests you to view detailed information and recommendations.

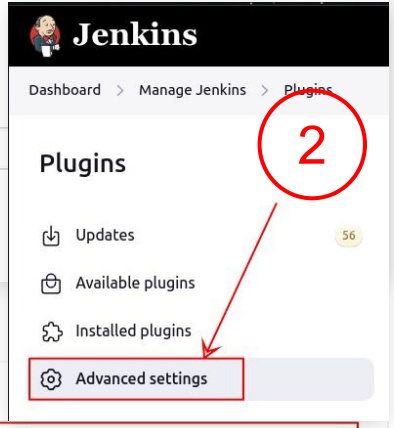
```yaml
jobs:
  accuknox-cicd:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@main

      - 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 }}
```

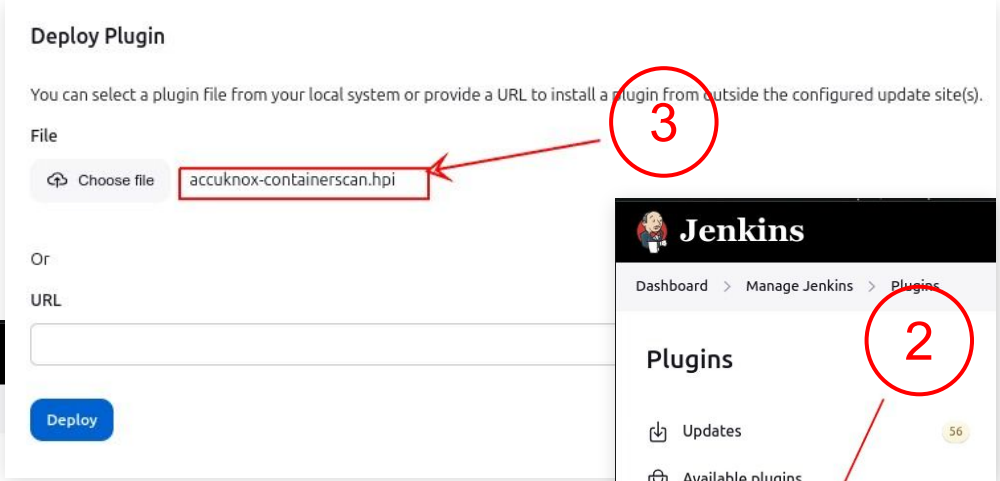
Configuring the container scan in Jenkins

ACCUKNOX

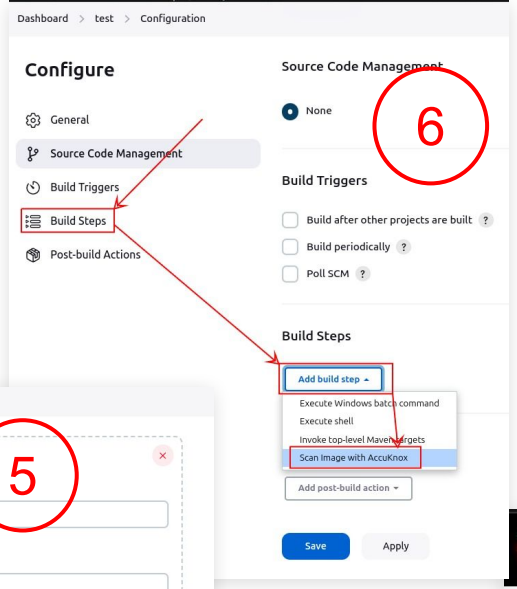
- Download the plugin in .hpi format from [here](#).
- Navigate to the Jenkins dashboard.
- Go to Manage Jenkins > Plugins > Advance settings.
- Deploy the plugin.

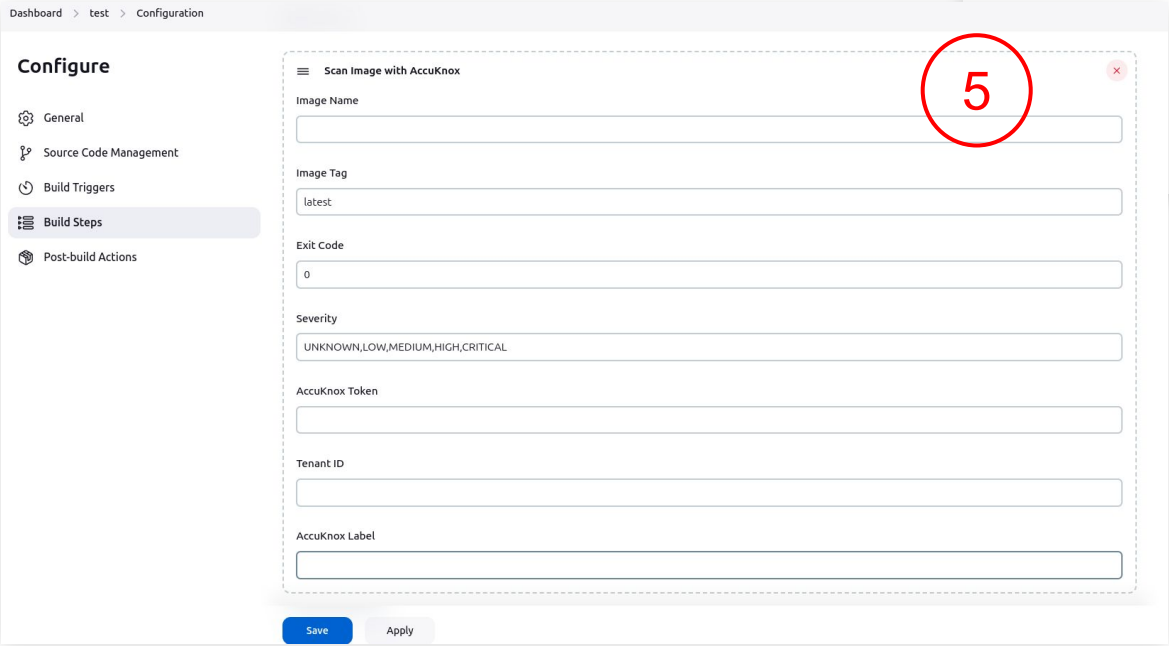


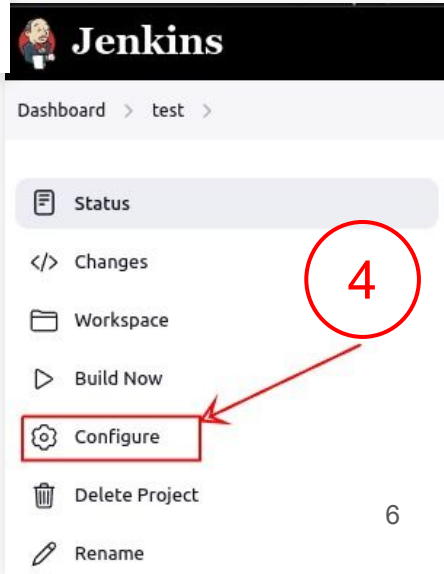




- Open the configuration page of your Jenkins job.
- Under the Build section, click on Add build step and select scan image with AccuKnox.
- Fill all the required parameters and trigger the pipeline.
- Review the findings in AccuKnox > Issues > Findings > Container scan.



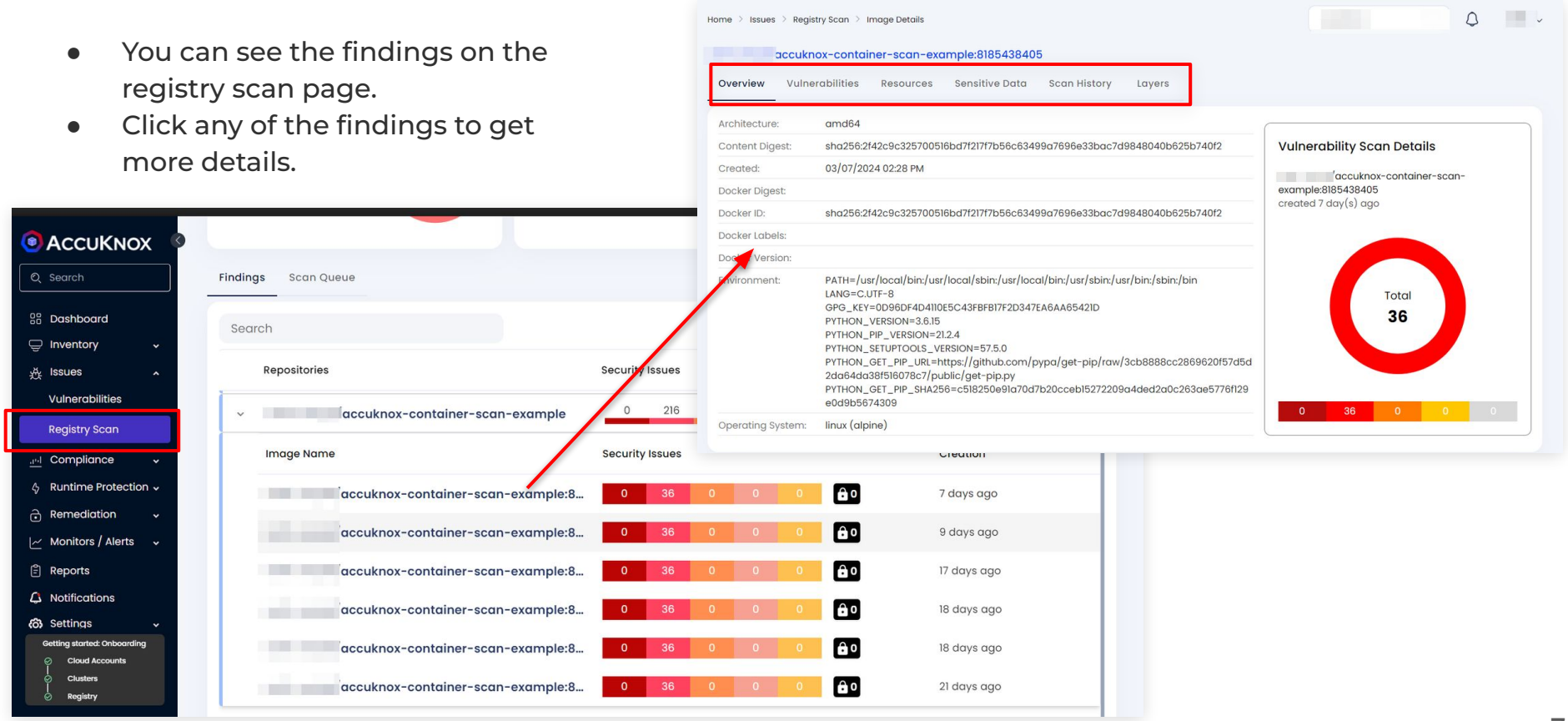




View Findings in registry scan page

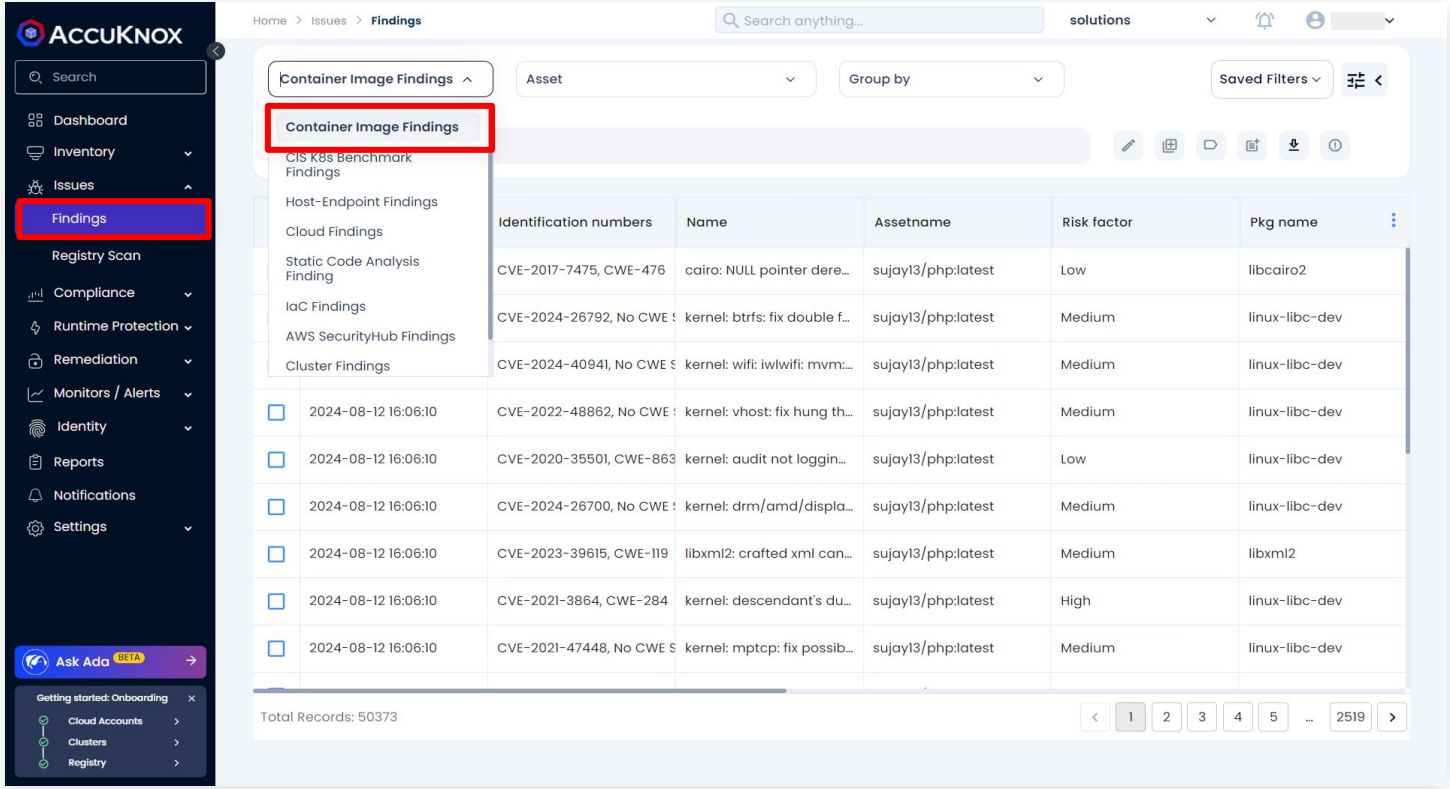
ACCUKNOX

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



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.



Container Scan Use Cases

ACCUKNOX

- 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

ACCUKNOX

- 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

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

Details + Create Ticket

Asset
rajvanshi/storm:latest

Asset Type
Container

Status
Active

Ignored
No

Severity
Medium

Tickets
0

Notes
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) Medium

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

ACCUKNOX

- 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

ACCUKNOX

- 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

A cleverly devised username might bypass LDAP authentication checks. I ...: (org.apache.derby:derby@10.10.2.0)

Critical

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

Use case 4: Remote Code Execution (RCE) vulnerabilities

ACCUKNOX

- 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

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:jac
databind

Use case 5: Denial of Service (DoS) vulnerabilities

ACCUKNOX

- Netty codec is a java 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

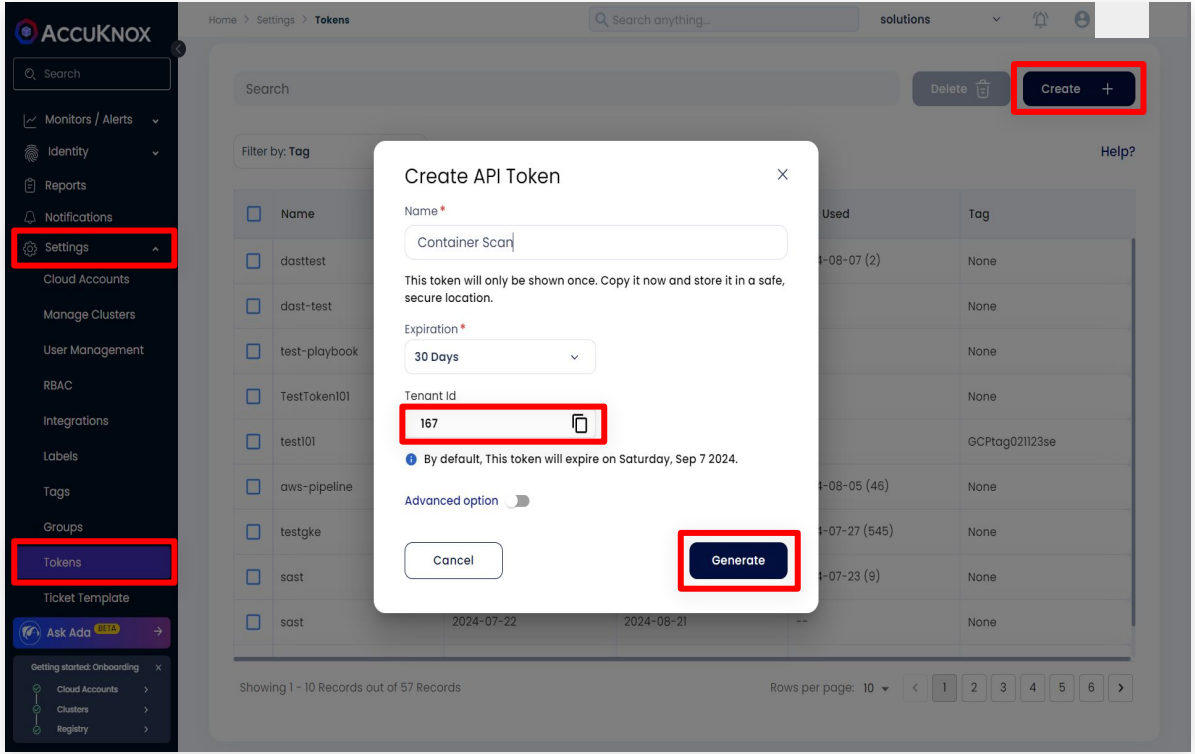
AccuKnox

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.



Configuring the IAC scan in GitHub actions

ACCUKNOX

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

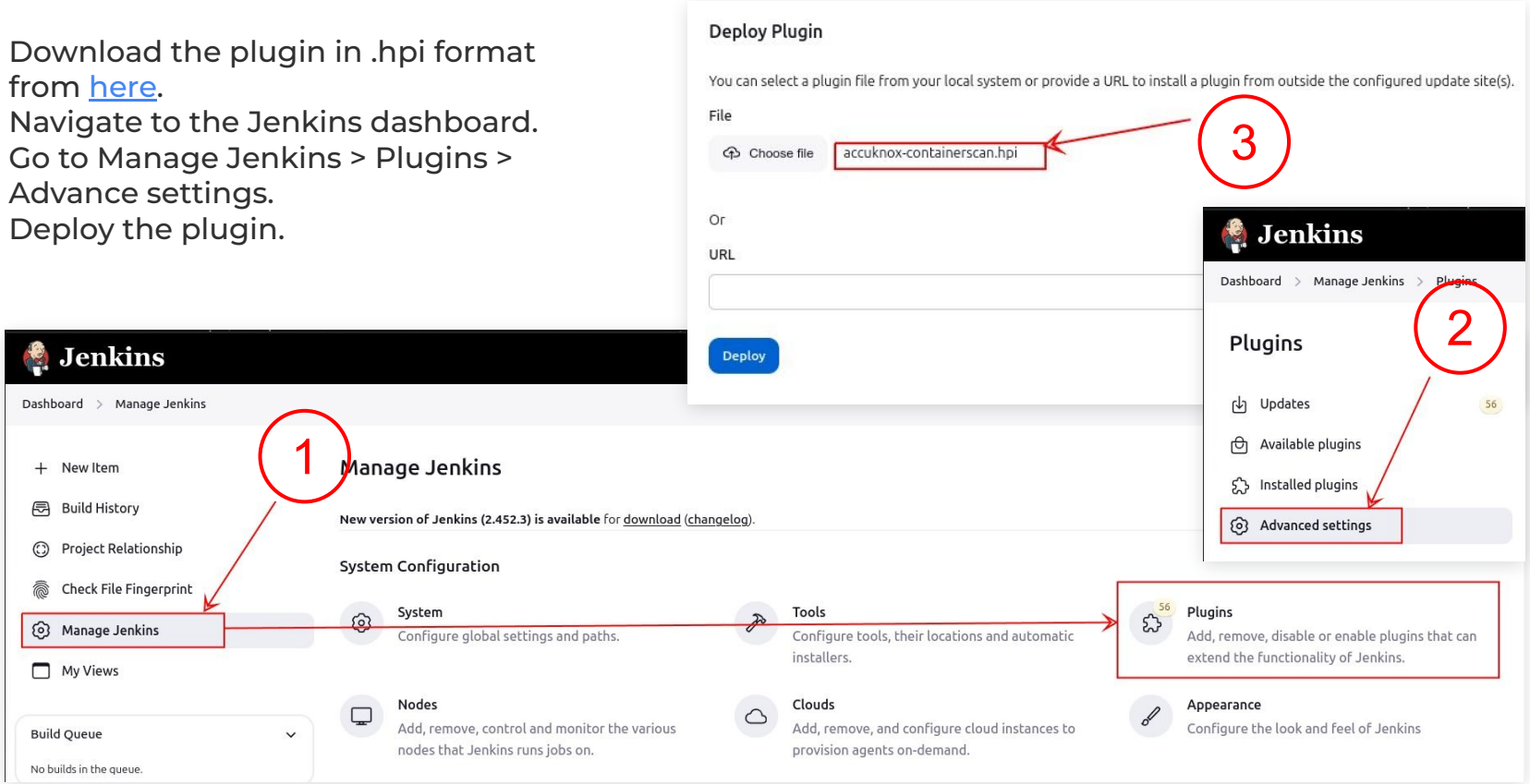
- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select IaC Findings.
- Click on any finding that interests you to view detailed information and recommendations.

```yaml
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: ./
          output_file_path: ./results
          token: ${{ secrets.TOKEN }}
          endpoint: ${{ secrets.ENDPOINT}}
          tenant_id: ${{ secrets.TENANT_ID}}
          quiet: "true"
          soft_fail: "true"
```

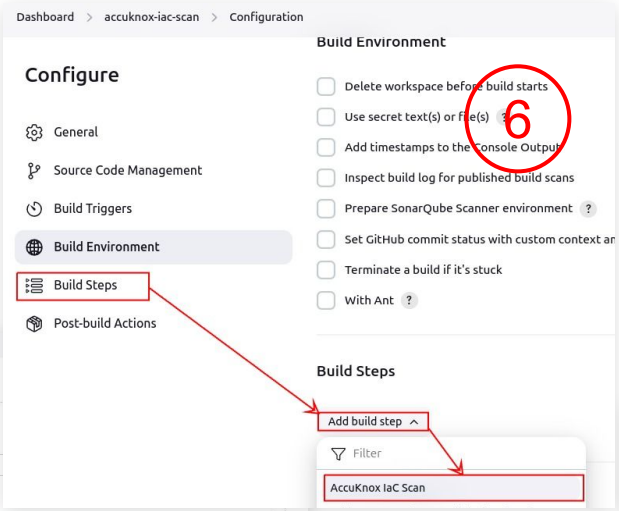
Configuring the IaC scan in Jenkins

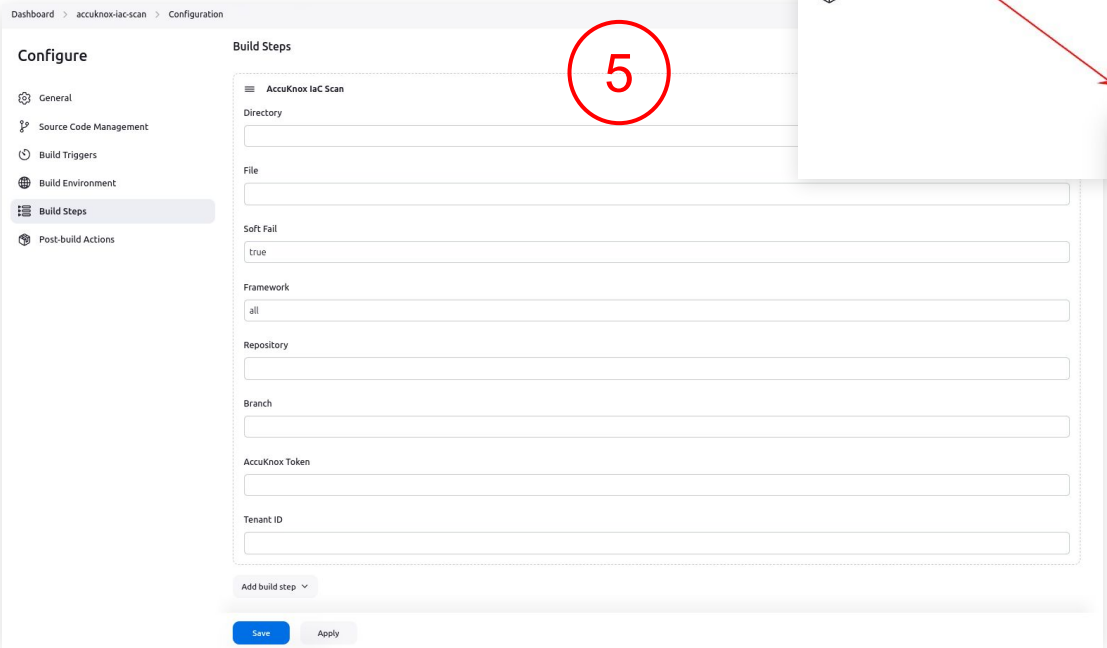
ACCUKNOX

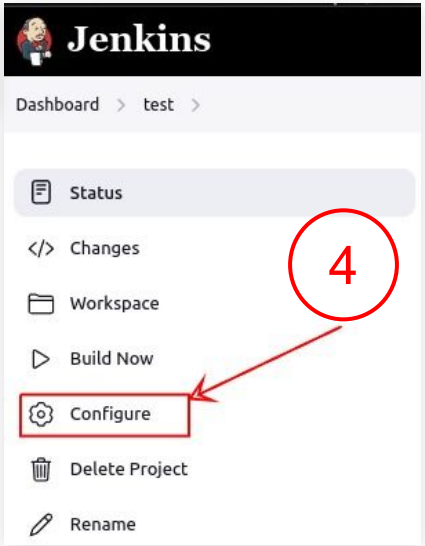
- Download the plugin in .hpi format from here.
- Navigate to the Jenkins dashboard.
- Go to Manage Jenkins > Plugins > Advance settings.
- Deploy the plugin.



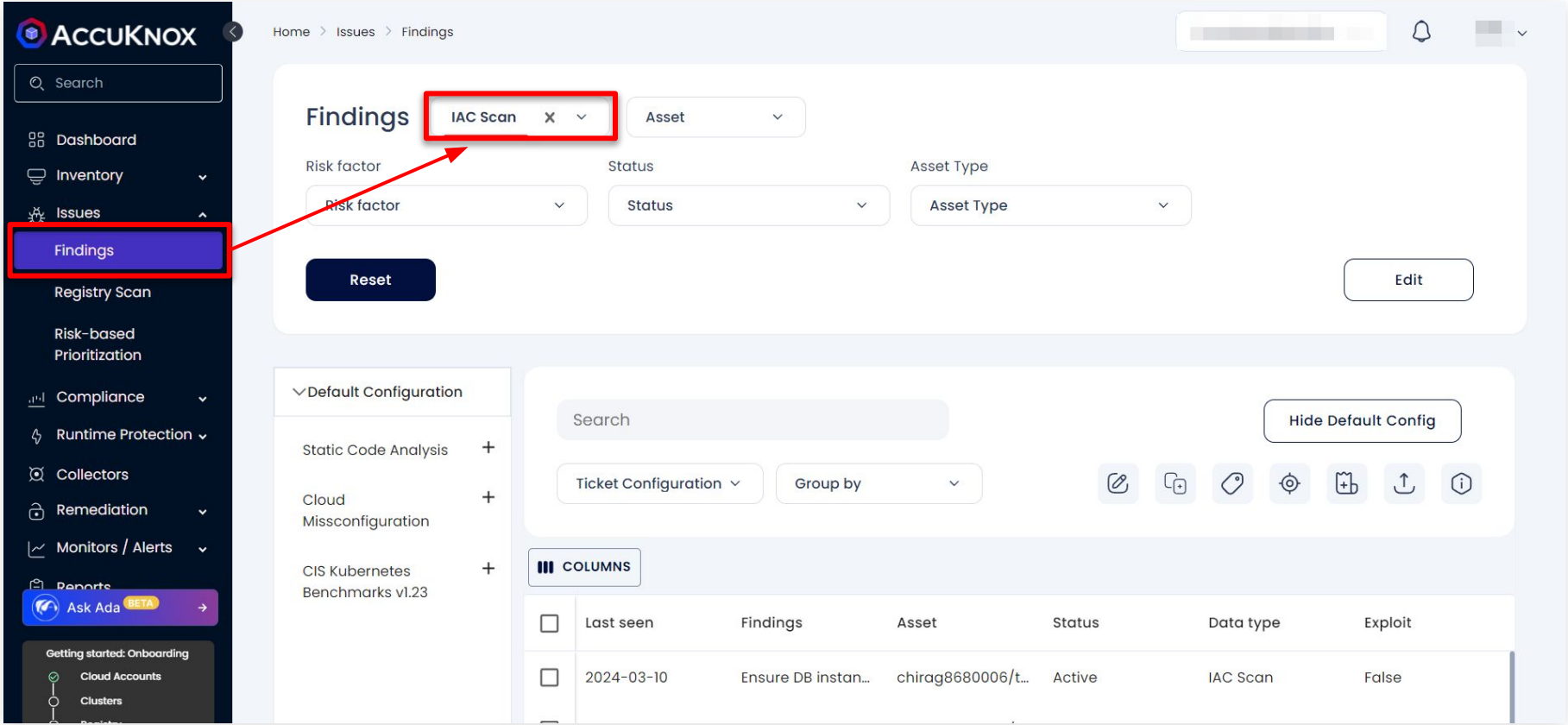
- Open the configuration page of your Jenkins job.
- Under the Build section, click on Add build step and select AccuKnox IaC scan.
- Fill all the required parameters.
- Review the findings in AccuKnox > Issues > Findings > Container scan.



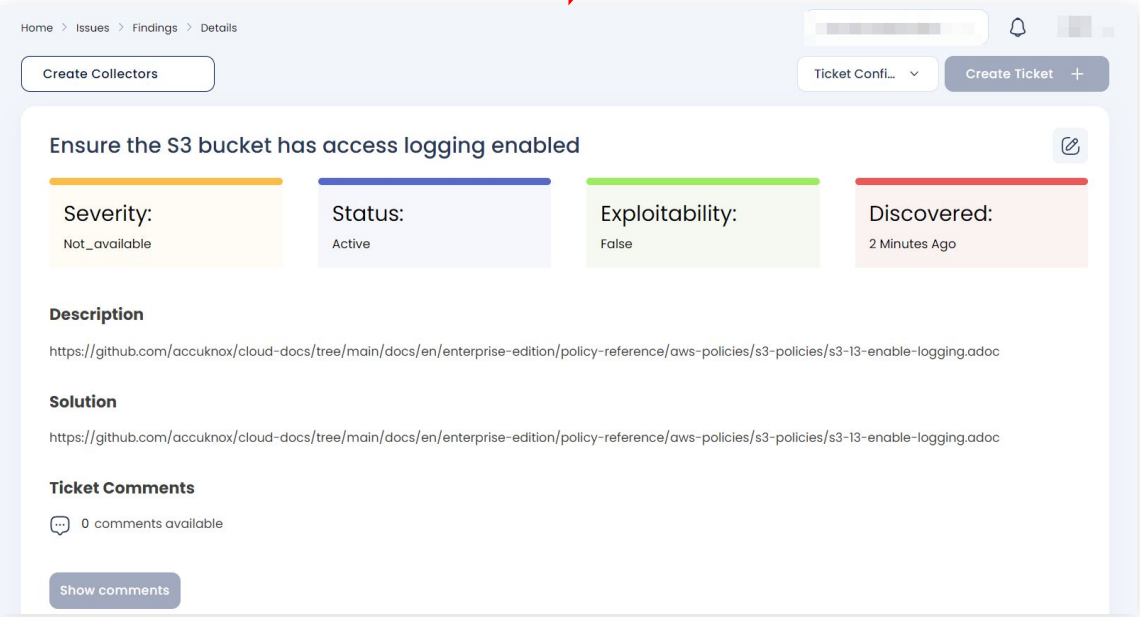


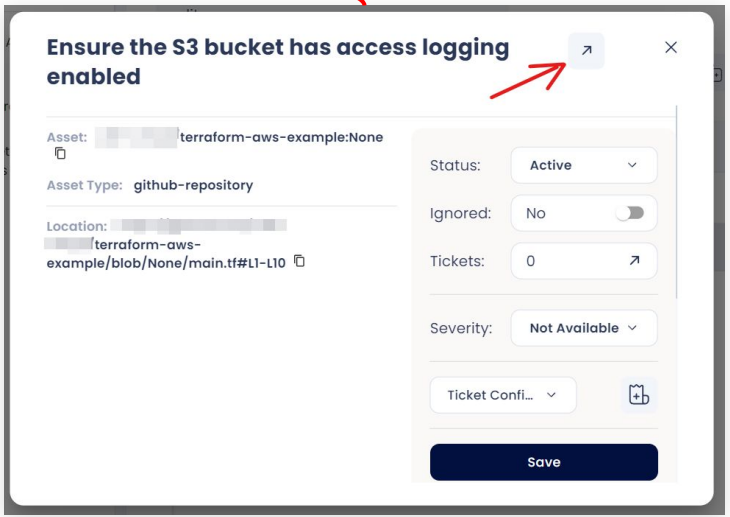


- Go to the AccuKnox > Findings and select the IAC Scan from the drop down menu



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





Use case: Security group misconfigurations

ACCUKNOX

- 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

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

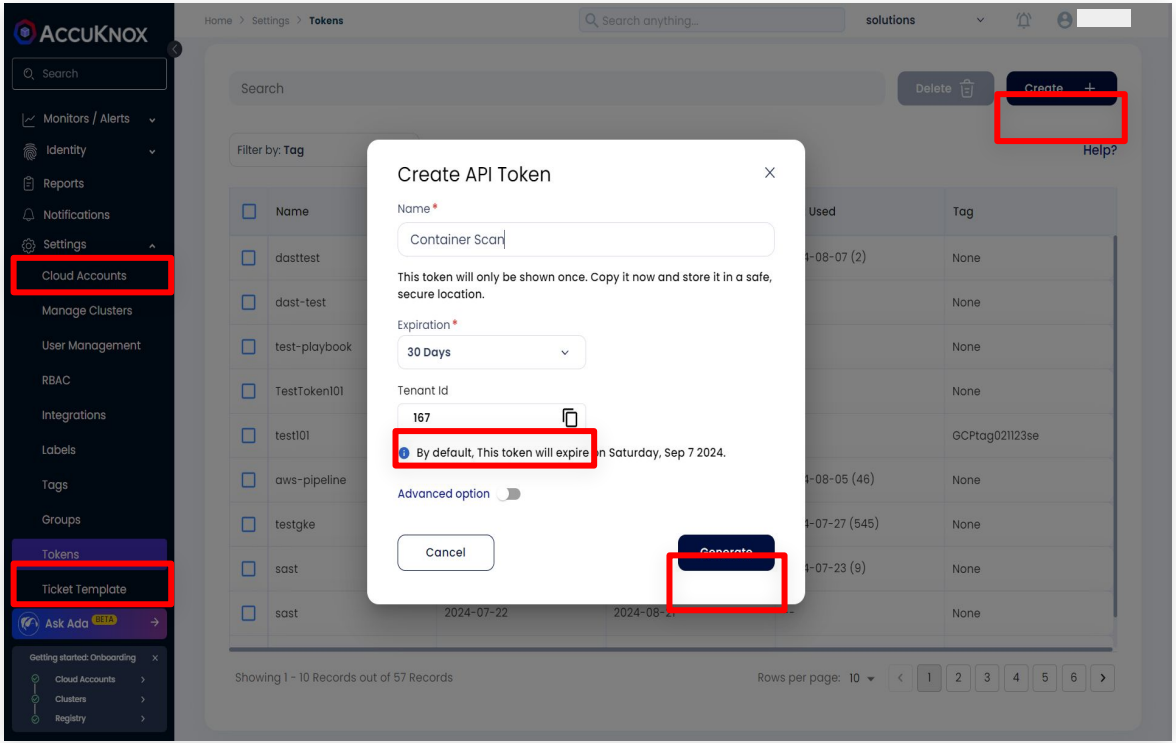
AccuKnox

SAST Scan

ASPM Integration

Generate AccuKnox API token for CI/CD pipeline

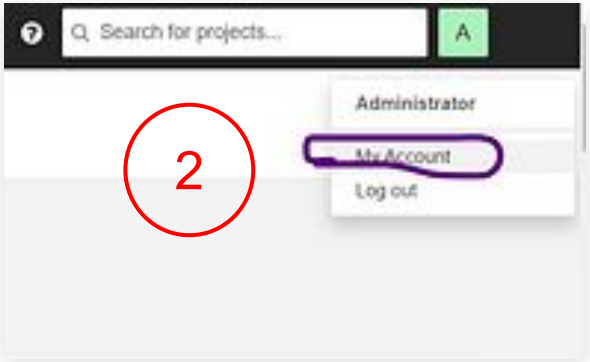
- To generate a token, open AccuKnox and navigate to **Settings > Tokens > Create**.
- Copy the token and tenant ID, then configure them as secrets in your CI/CD pipeline.

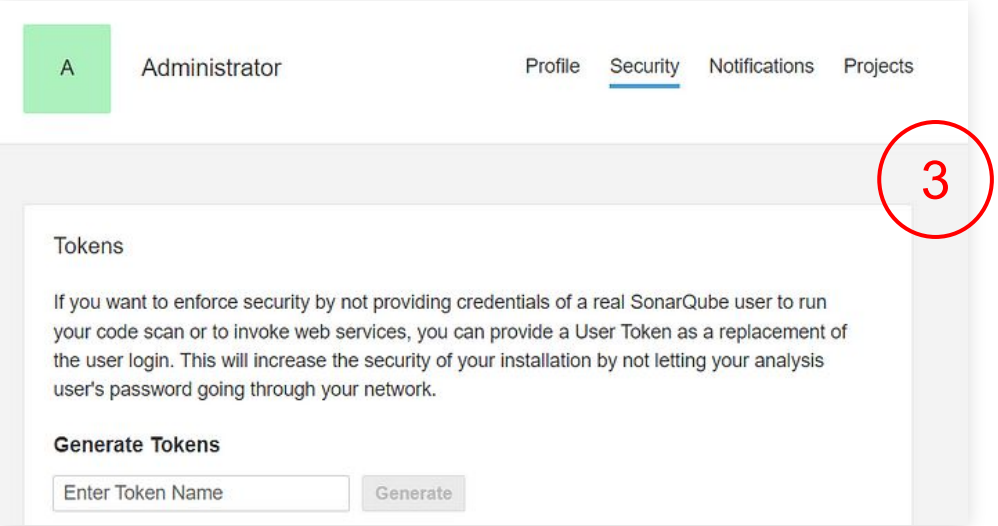


SonarQube configuration

ACCUKNOX

- 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.






```
# 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

ACCUKNOX

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_IS`, `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

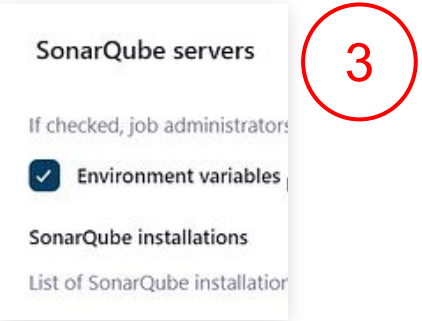
- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select SAST Findings.
- Click on any finding that interests you to view detailed information and recommendations.

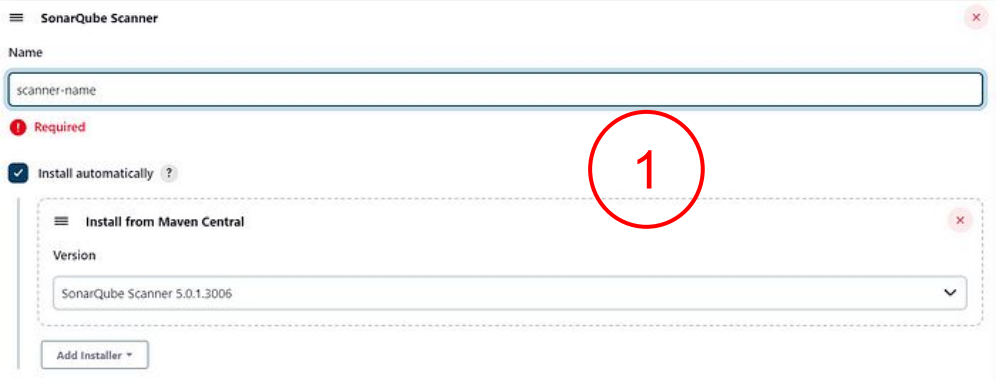
```yaml
jobs:
  sonarqube_sast:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a better relevancy of
analysis
      - uses: sonarsource/sonarqube-scan-action@master
        env:
          SONAR_TOKEN: ${{ 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.tenat_id }}&data_type=SQ&save_to_s3=false>" \
              --header "Tenant-Id: ${{ secrets.tenat_id }}" \
              --header "Authorization: ${{ secrets.accuknox_token }}" \
              --form "file=@\"$file\""
          done
```

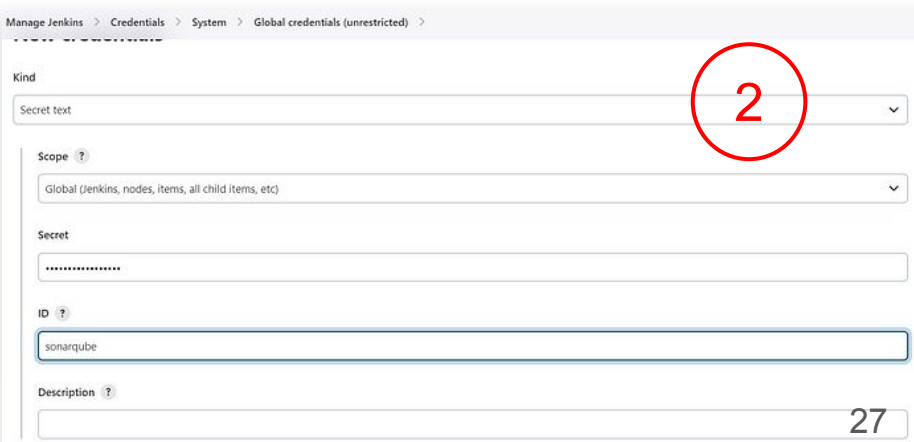
Configuring the SAST scan in Jenkins

ACCUKNOX

- 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



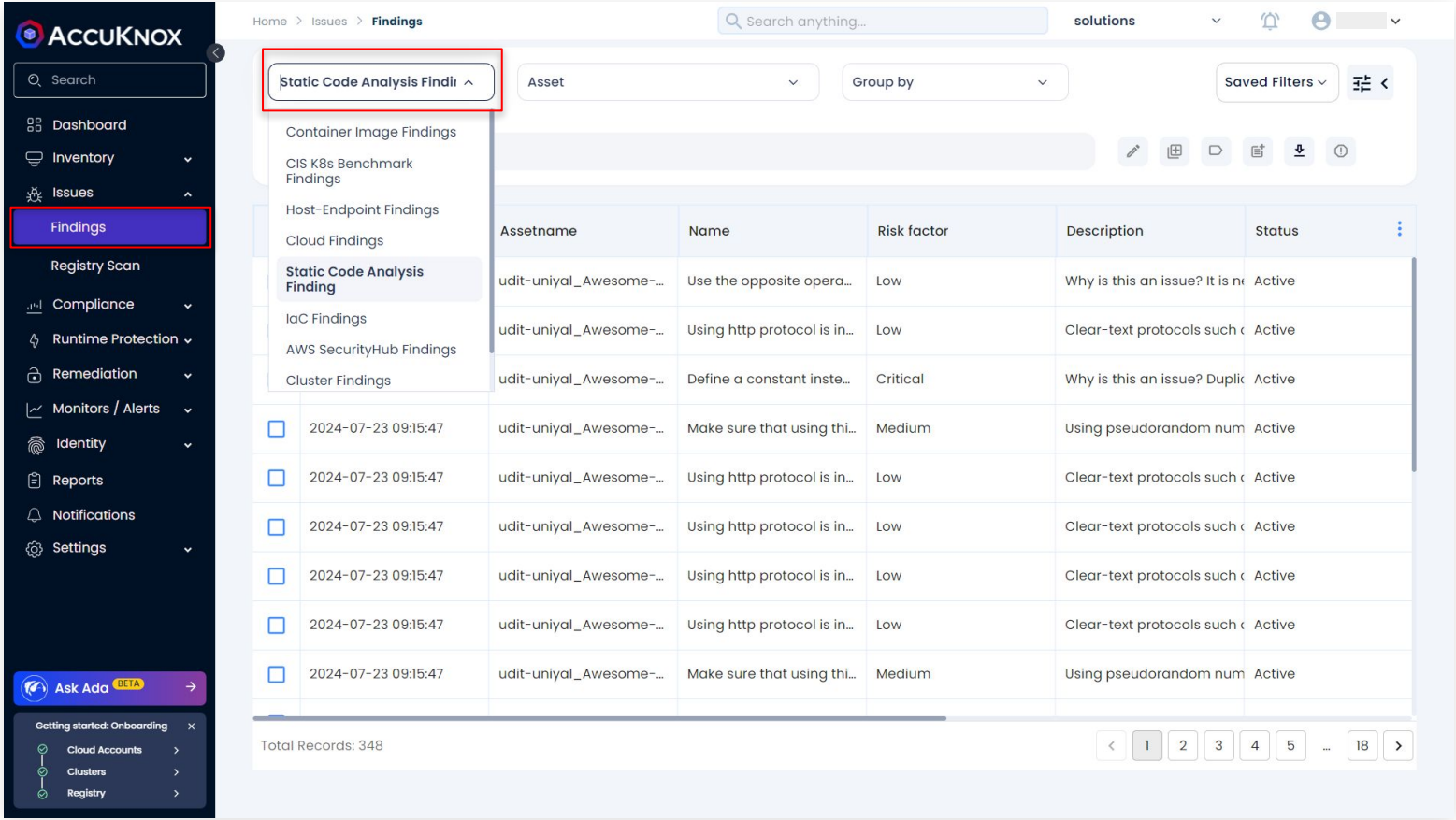




- Go to Manage Jenkins > Tools and add the SonarQube installation details.
- Create SonarQube credentials.
- Go to Manage Jenkins > System Configurations. Select the check-box of Injecting Environment variables and add the details of the SonarQube Server

```
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$" \
                        -v \$(pwd):/app/data/ \
                        accuknox/sastjob:latest
                """
            }
        }
        stage('Upload SAST Reports') {
            steps {
                script {
                    def files = sh(script: 'ls -1 SQ-*.json', returnStdout:
true).trim().split('\n')
                    for (file in files) {
                        sh """
                            curl --location --request POST
"https://${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.



SAST use cases

ACCUKNOX

- 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

ACCUKNOX

- 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

<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

Show More...

Details + Create Ticket

Asset

gitlab-sast-testing

Asset Type

static_code_Software

Status

Use case 2: XML External Entity (XXE) injection

ACCUKNOX

- 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

This vulnerability allows the usage of external entities in XML.
<h2>Why is this an issue?</h2>
External Entity Processing allows for XML parsing with the involvement of external entities. However, when this functionality is enabl
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

```
2 // Exercise - 1
3 // Author: @TheXC3LL
4 // Website: Tarlogic.com
5 class login {
6   public $username = "X-C3LL";
7   public $password = "Insanity";
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") {
```

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)

ACCUKNOX

- 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

Description | Result | Solution | References | Source Code

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.
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.
Show Less...

Details + Create Ticket

Asset
test-vulnerable-code-snippets

Asset Type
static_code_Software

Status
Active

Ignored
No

Use case 5: Remote Code Execution (RCE)

ACCUKNOX

- 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

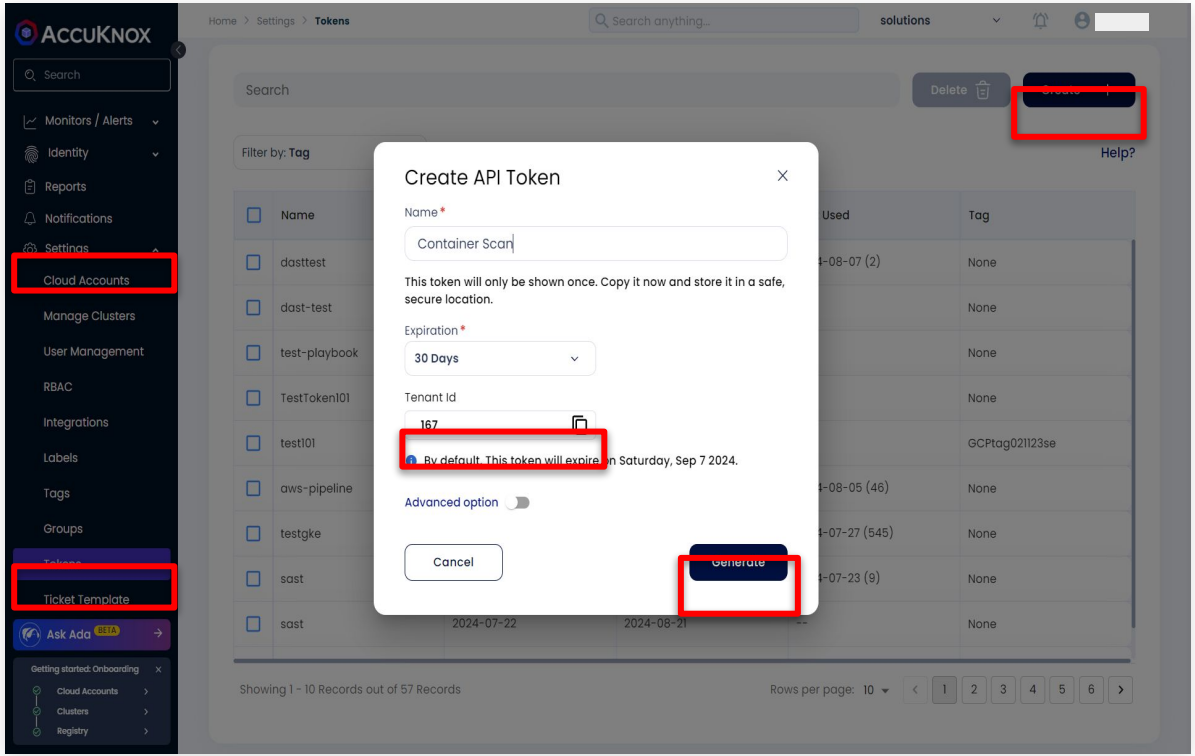
AccuKnox

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.



Configuring the DAST scan in GitHub actions

ACCUKNOX

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

ACCUKNOX

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:

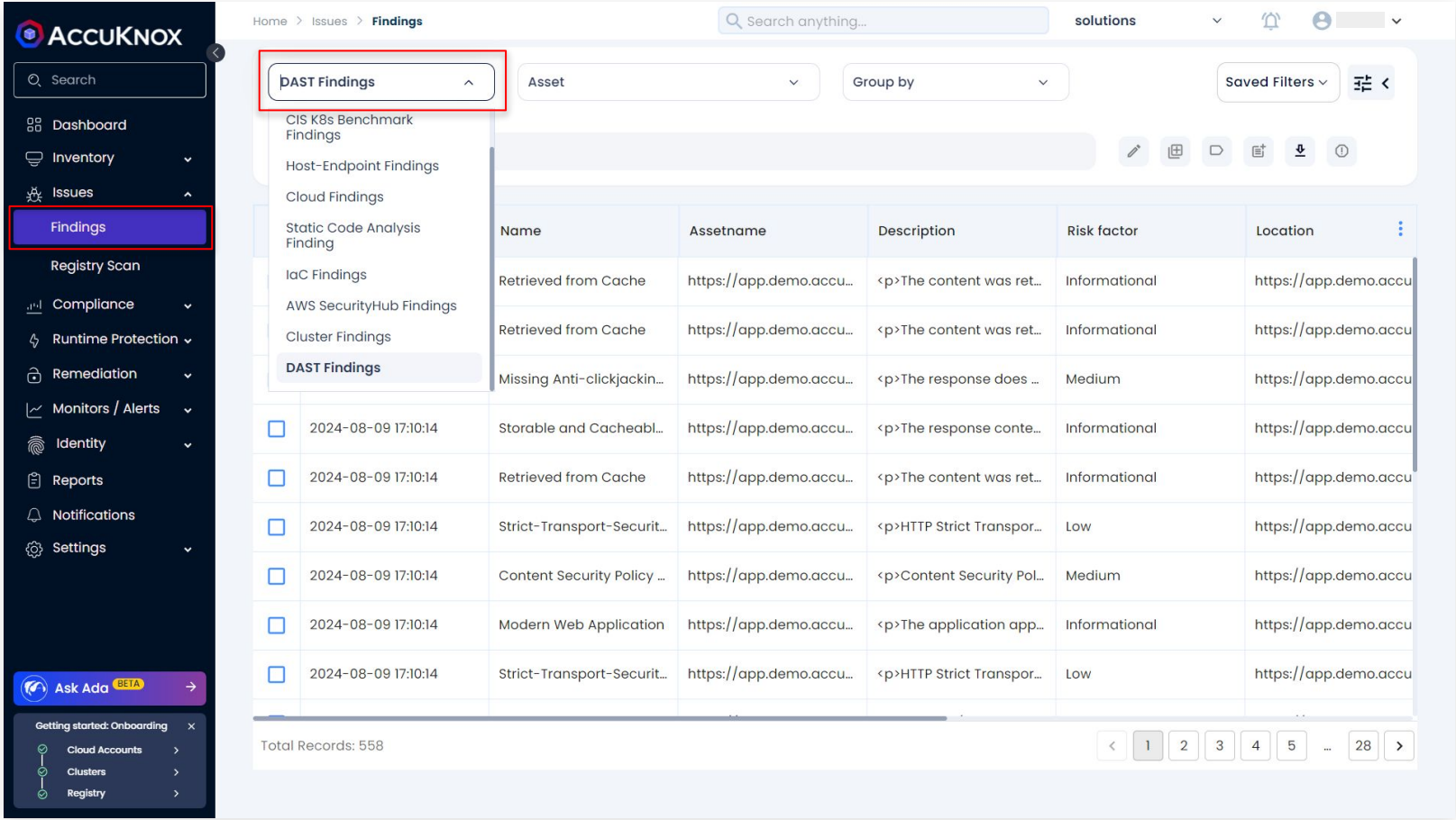
- After adding the script, click "Save" or "Apply."

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
ID for the tenant_id
        BEARER_TOKEN = credentials('bearer-token')  // Replace with your Jenkins credentials
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 $PWD:/zap/wrk/:rw -t \
                        zaproxy/zap-stable zap-baseline.py \
                        -t <your-website> \
                        -J report.json \
                        -I
                """
            }
        }
        stage('Upload ZAP Report') {
            steps {
                sh """
                    curl --location --request POST
"https://cspm.demo.accuknox.com/api/v1/artifact/?
tenant_id=${TENANT_ID}&data_type=ZAP&label_id=dasttest&save_to_s3=false" \
                        --header "Tenant-Id: ${TENANT_ID}" \
                        --header "Authorization: Bearer ${BEARER_TOKEN}" \
                        --form "file=@report.json"
                """
            }
        }
    }
}
```

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



DAST Use Cases

ACCUKNOX

- 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

ACCUKNOX

- 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

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

Details + Create Ticket

Asset

https://juice-shop.herokuapp.com

Asset Type

Use case 2: File inclusion vulnerability

ACCUKNOX

- 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

<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

Show More...

Details + Create Ticket

Asset

http://testphp.vulnweb.com

Asset Type

WebApp

Status

Use case 3: Cross site scripting vulnerability

ACCUKNOX

- 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

<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

Show More...

Details + Create Ticket

Asset

http://testphp.vulnweb.com

Asset Type

WebApp

Status

Use case 4: SQL injection

ACCUKNOX

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

SQL Injection - MySQL High

Description | Result | Solution | References | Source Code

SQL injection may be possible.

Finding for in resource WebApp | http://testphp.vulnweb.com

Failing since about 1 month ago, on 21/07/2024

Last detected about 1 month ago, on 21/07/2024

Details + Create Ticket

Asset
http://testphp.vulnweb.com

Asset Type
WebApp

Status
Active

Use case 5: Missing content security policy

ACCUKNOX

- 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

Description | Result | Solution | References | Source Code

<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 thef

Show More...

Details

+ Create Ticket

Asset

https://juice-shop.herokuapp.com

Asset Type

WebApp