AccuKnox

ASPM User Playbook

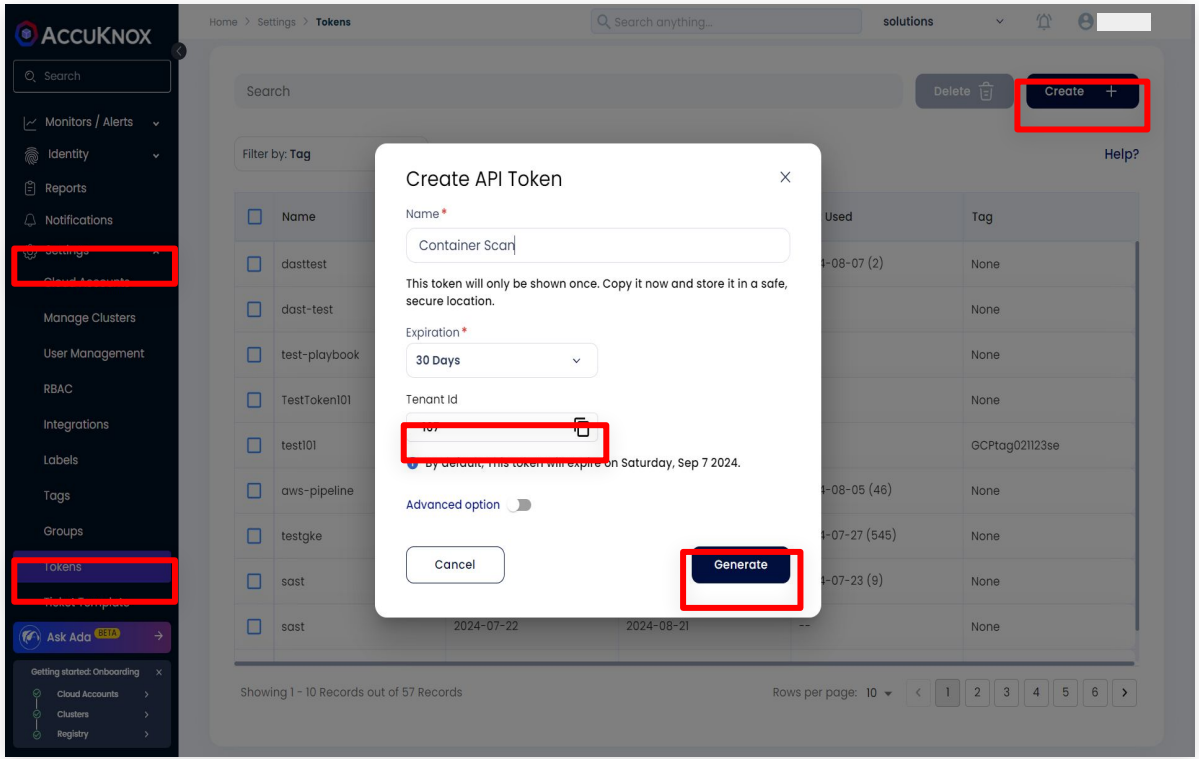
AccuKnox

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.

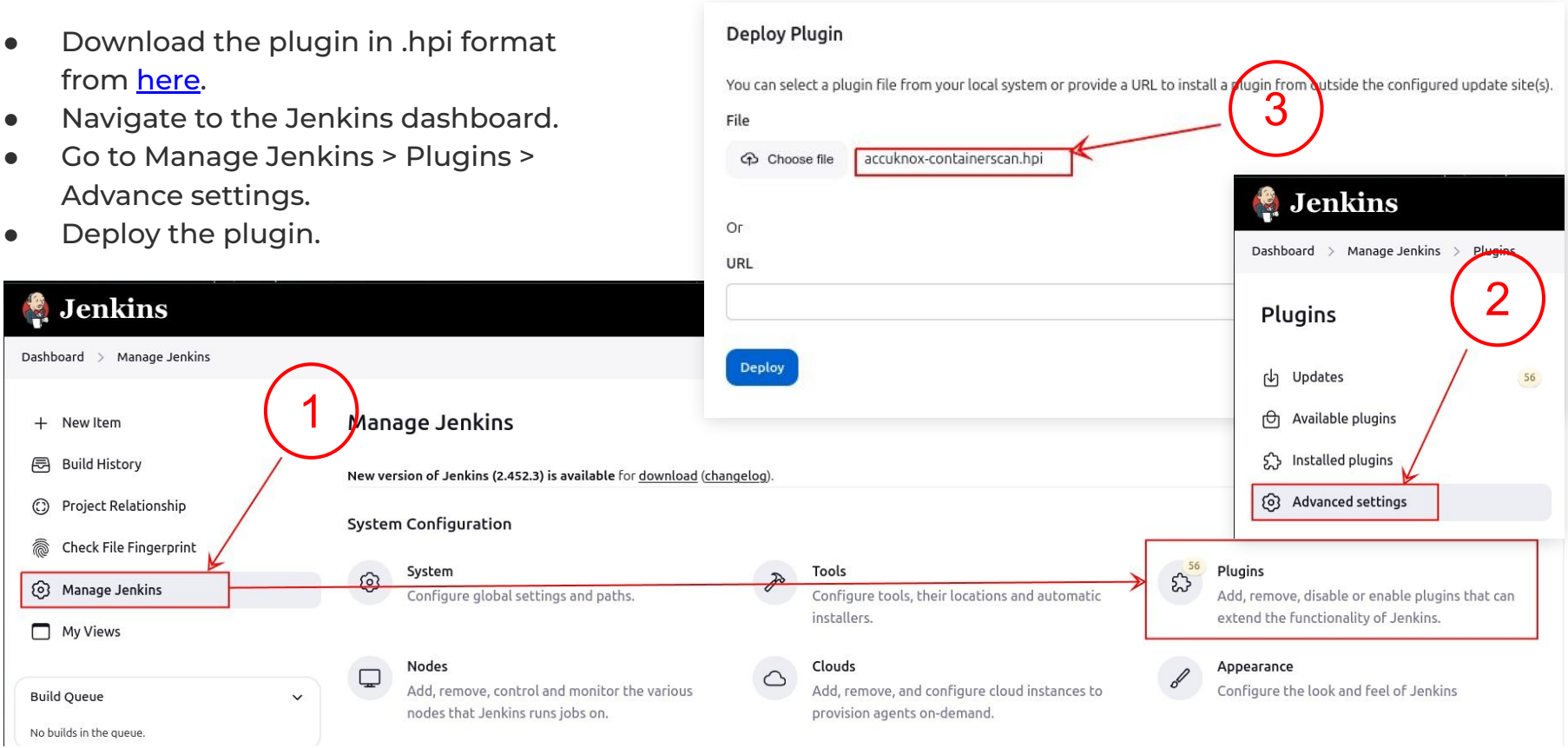
Step 3: Review Findings in AccuKnox

- 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.

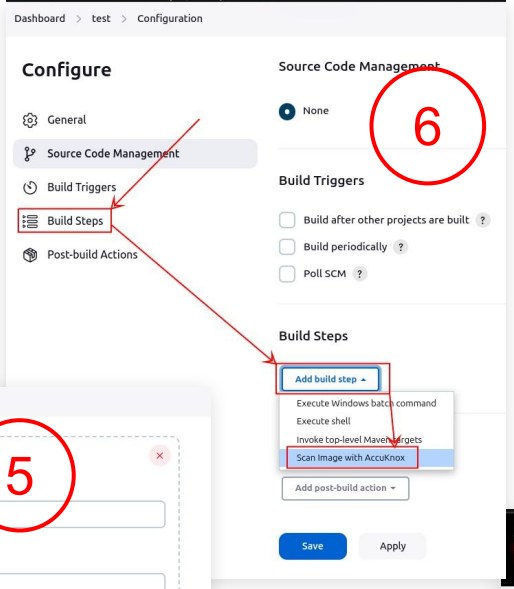
```yaml
jobs:
  accuknox-cicd:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@main

      - 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 }}
```

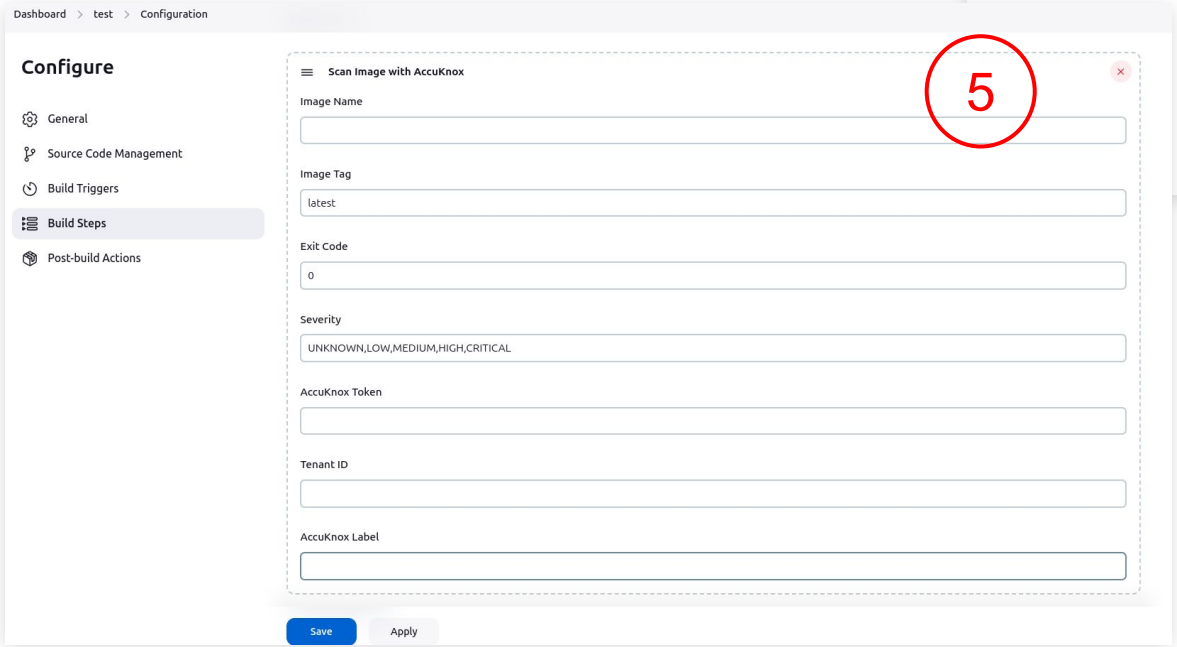
Configuring the container scan in Jenkins

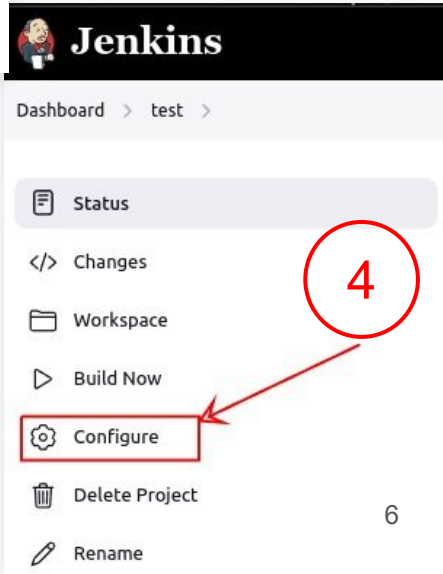
- Download the plugin in .hpi format from here.
- Navigate to the Jenkins dashboard.
- Go to Manage Jenkins > Plugins > Advance settings.
- Deploy the plugin.



- 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.

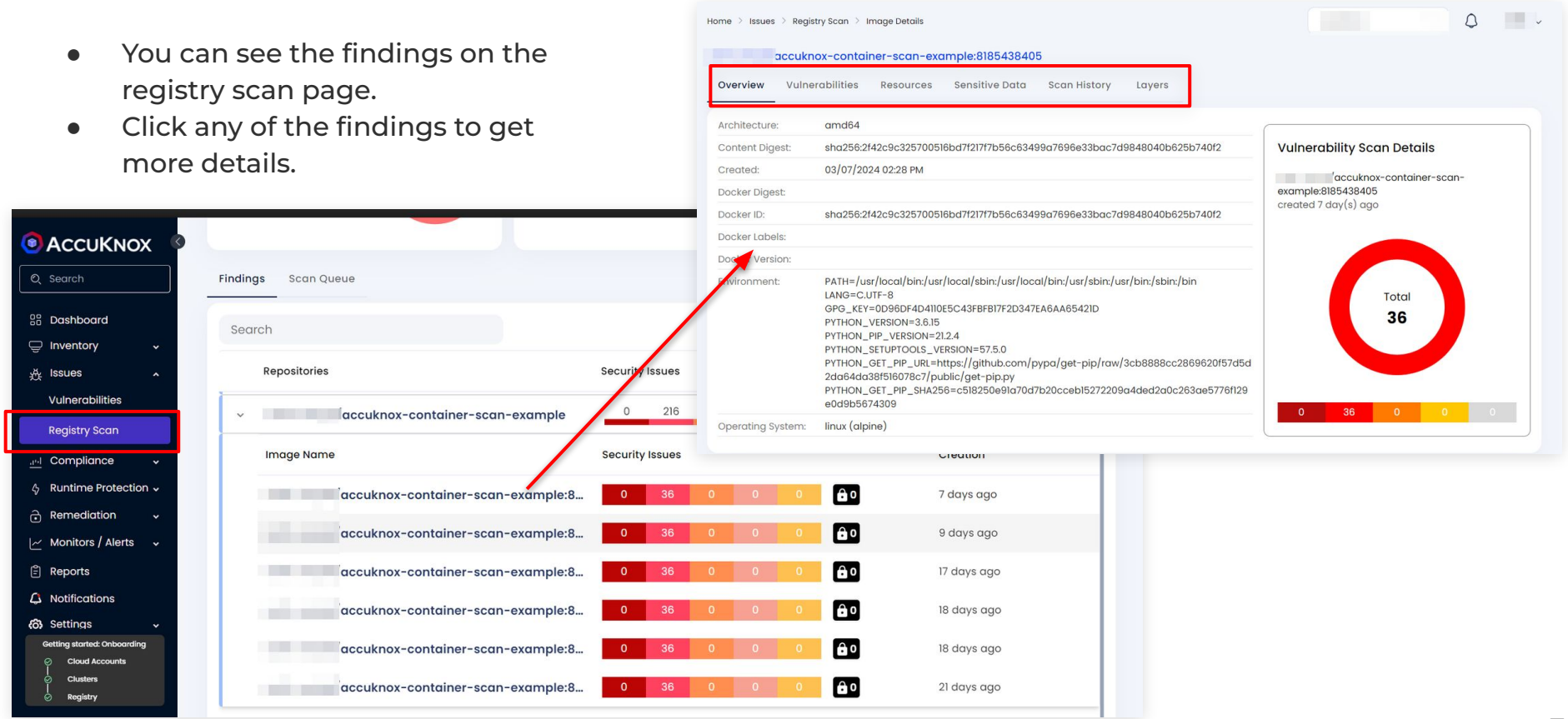






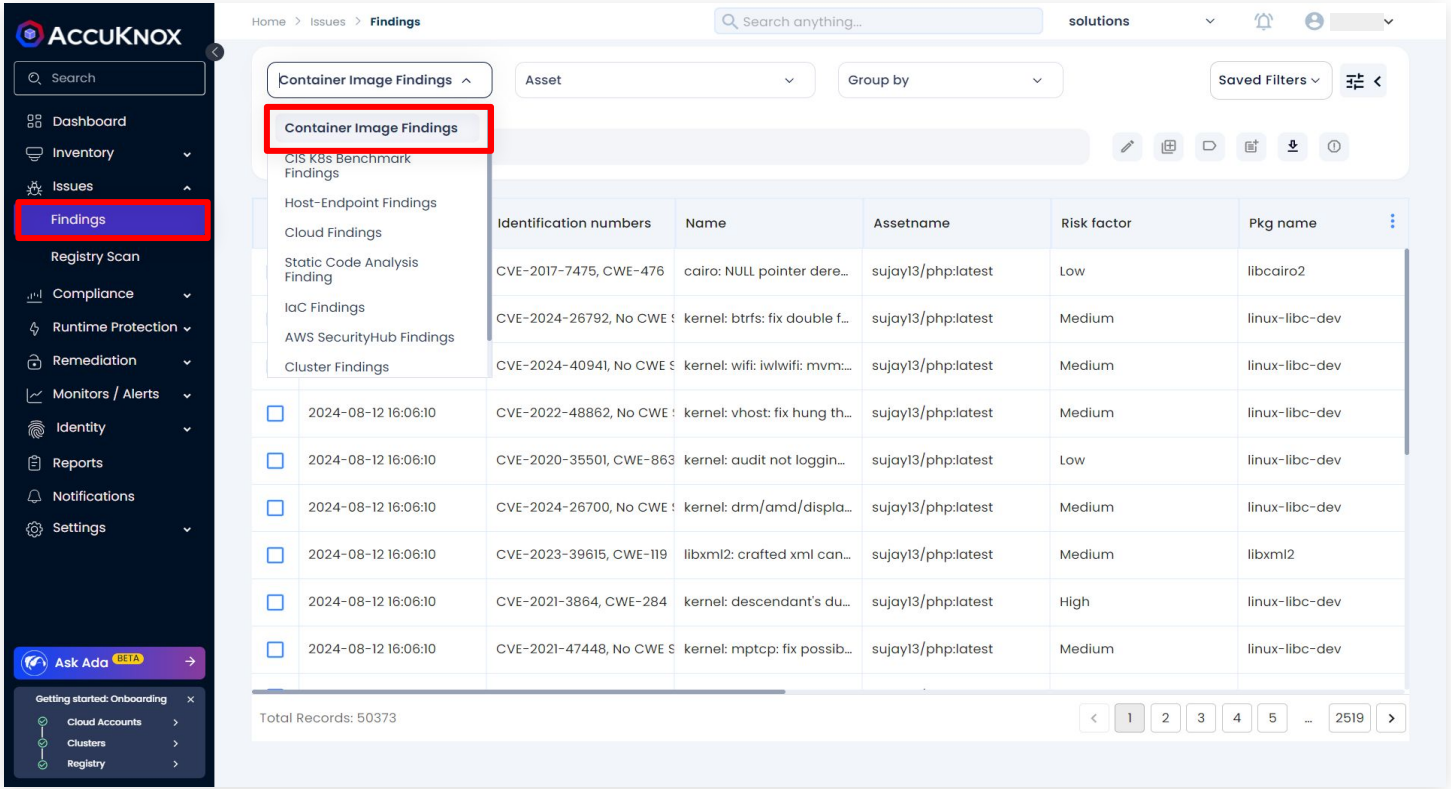
View Findings in registry scan page

- 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.



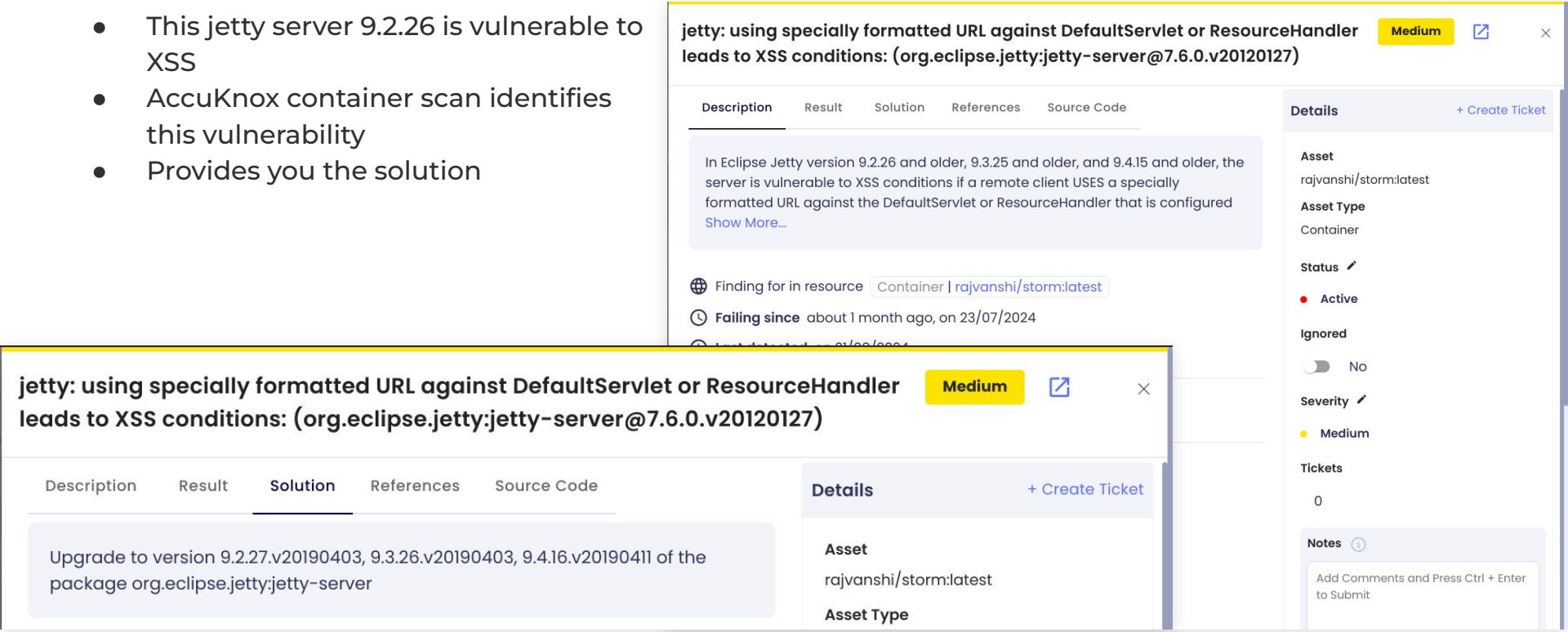
Container Scan Use Cases

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

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



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

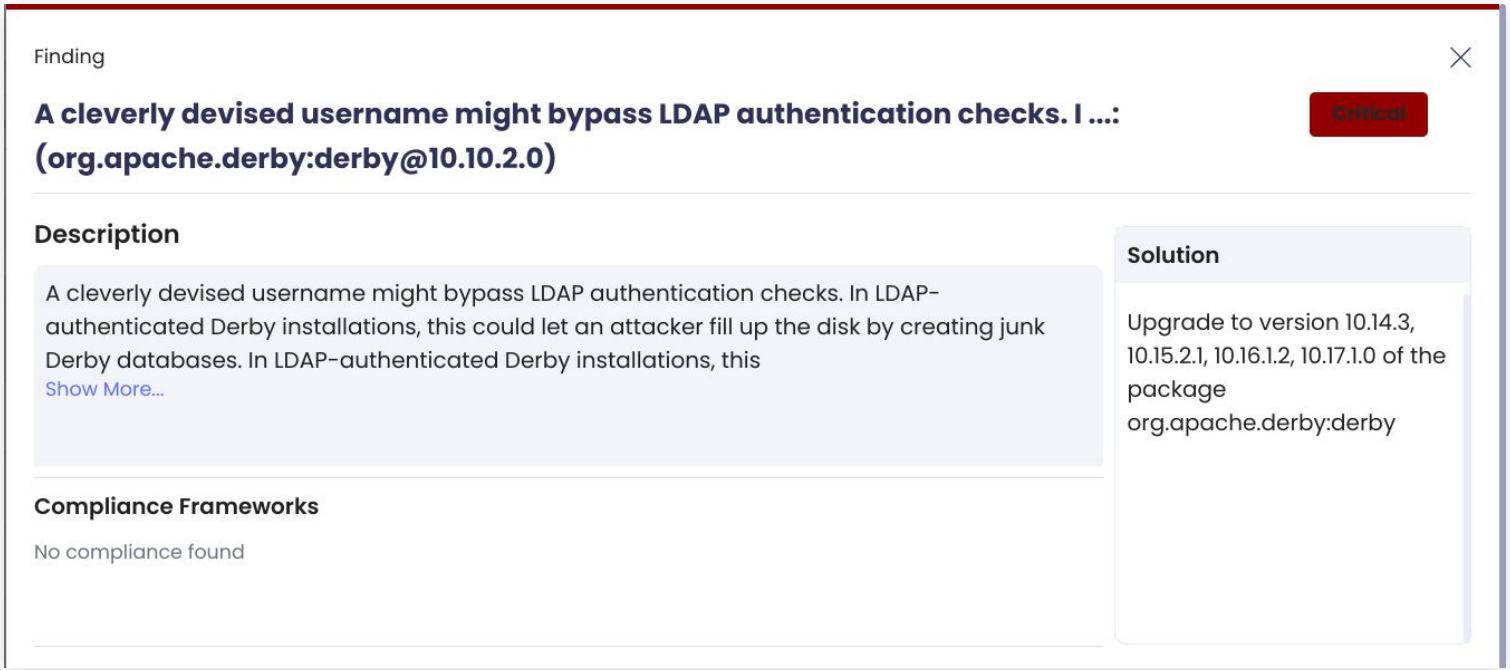
Overview | Vulnerabilities | Resources | **Sensitive Data** | Scan History | Layers

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

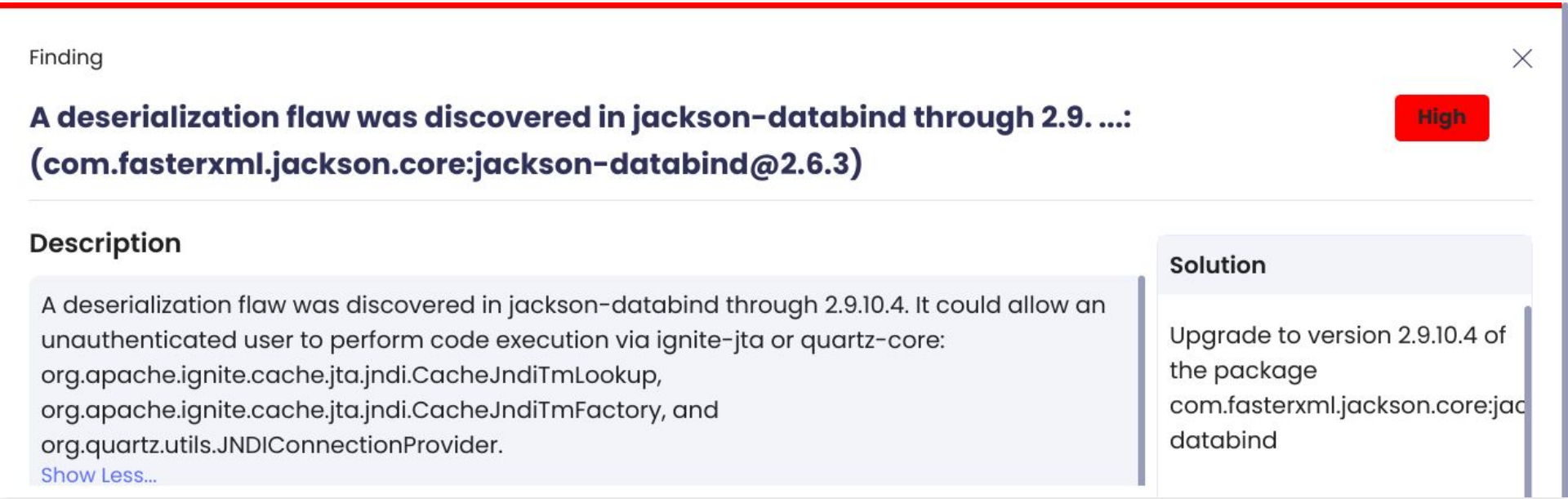
- The apache derby is a JDBC driver
- In this case it's vulnerable to a broken authentication
- AccuKnox proposes a solution to upgrading it to version 10.14.3



Use case 4: Remote Code Execution (RCE) vulnerabilities

ACCUKNOX

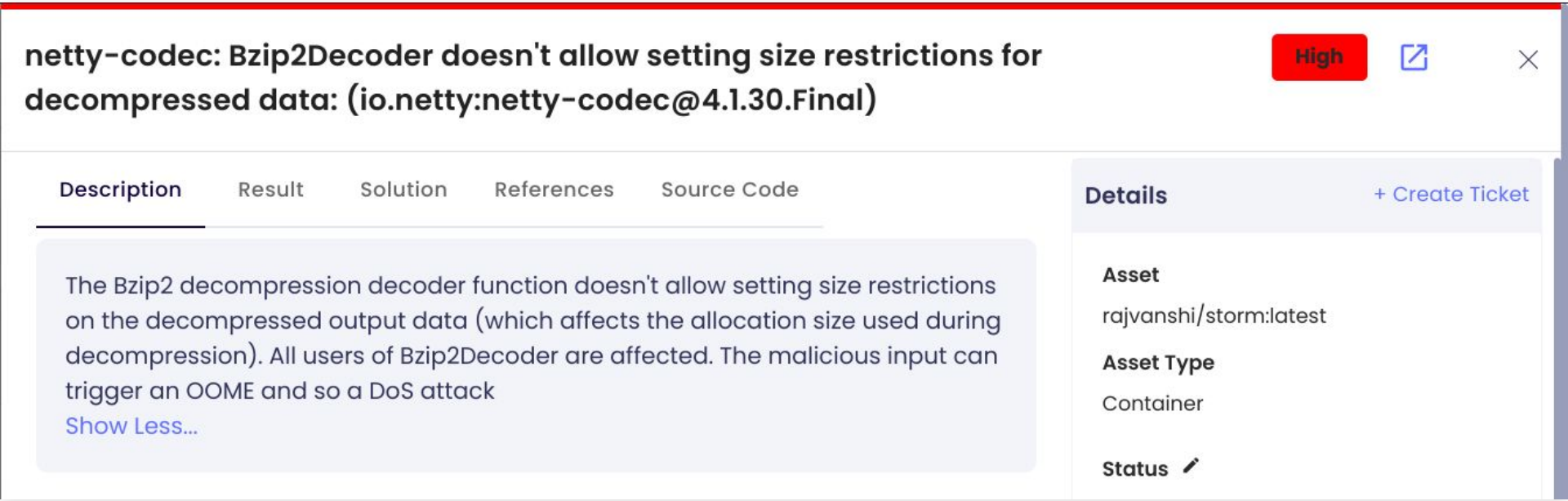
- Jackson is a popular Java library used for processing JSON, here it's vulnerable to code execution.
- AccuKnox identifies the vulnerability and suggests to updating Jackson to version 2.9.10.4



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.



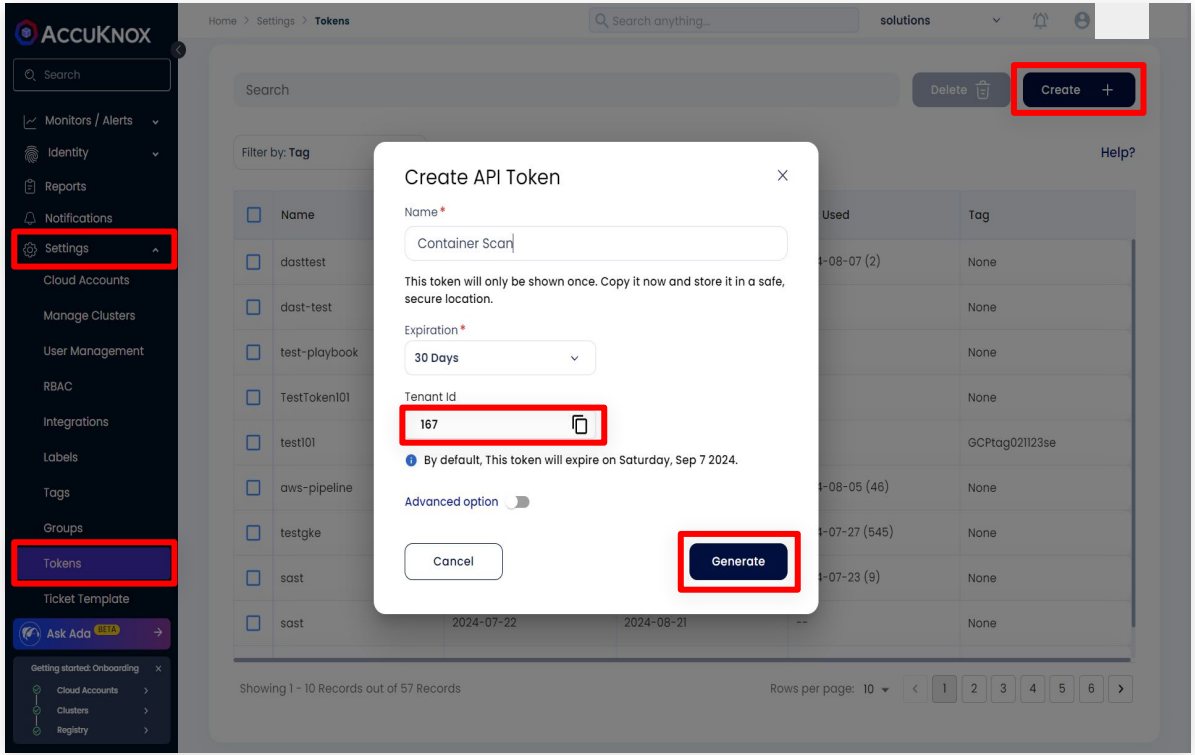
AccuKnox

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.

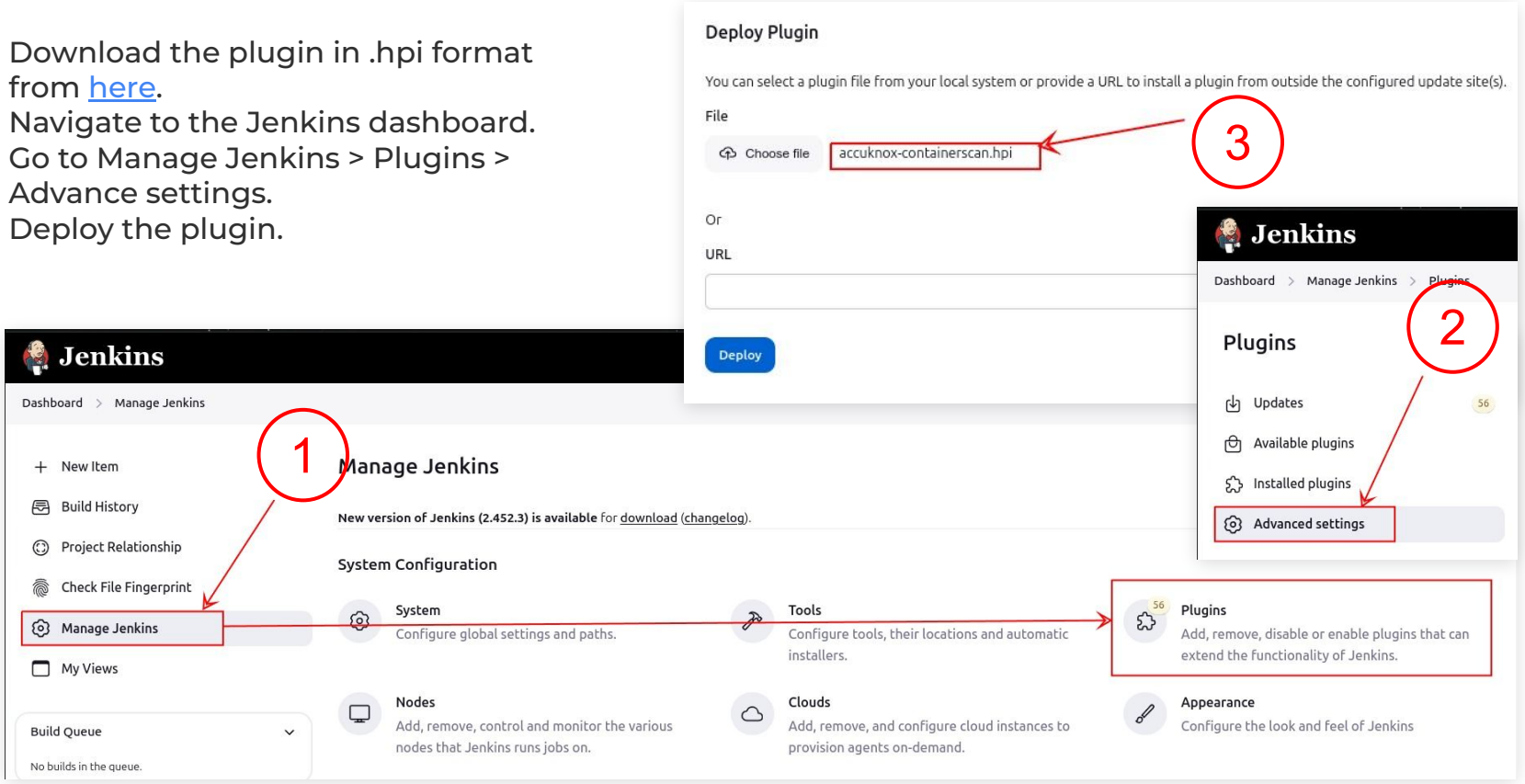
Step 3: Review Findings in AccuKnox

- 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.

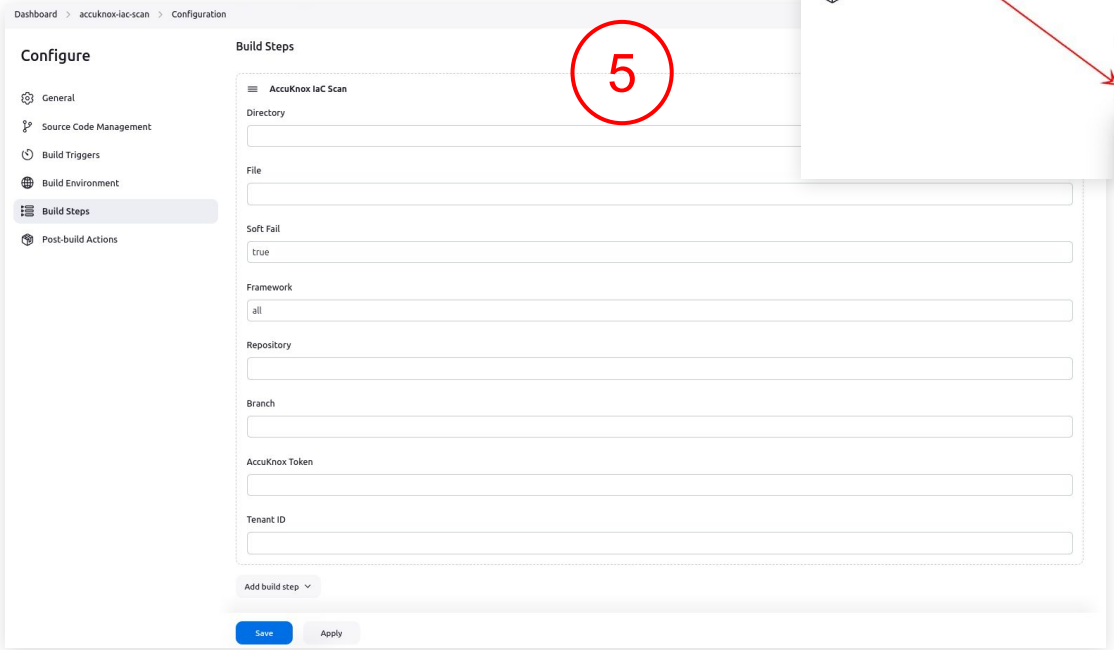
```yaml
jobs:
  tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@main

      - 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"
```

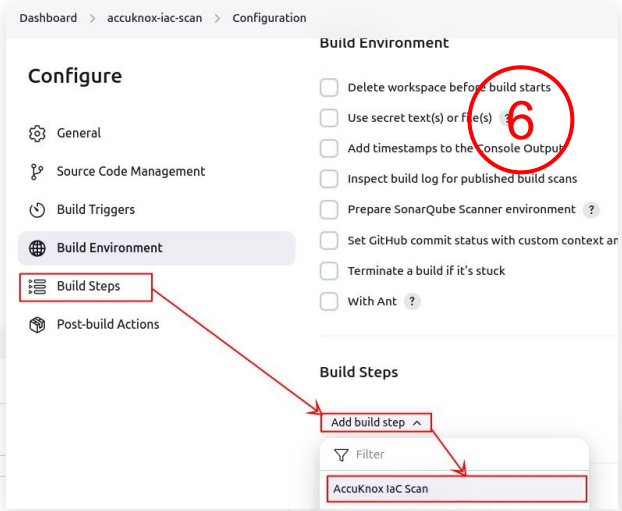
Configuring the IaC scan in Jenkins

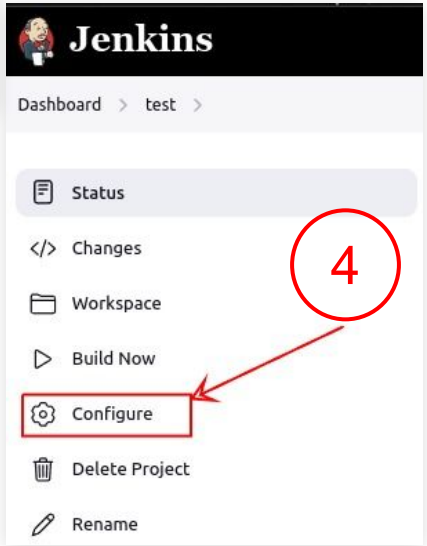
- Download the plugin in .hpi format from here.
- Navigate to the Jenkins dashboard.
- Go to Manage Jenkins > Plugins > Advance settings.
- Deploy the plugin.



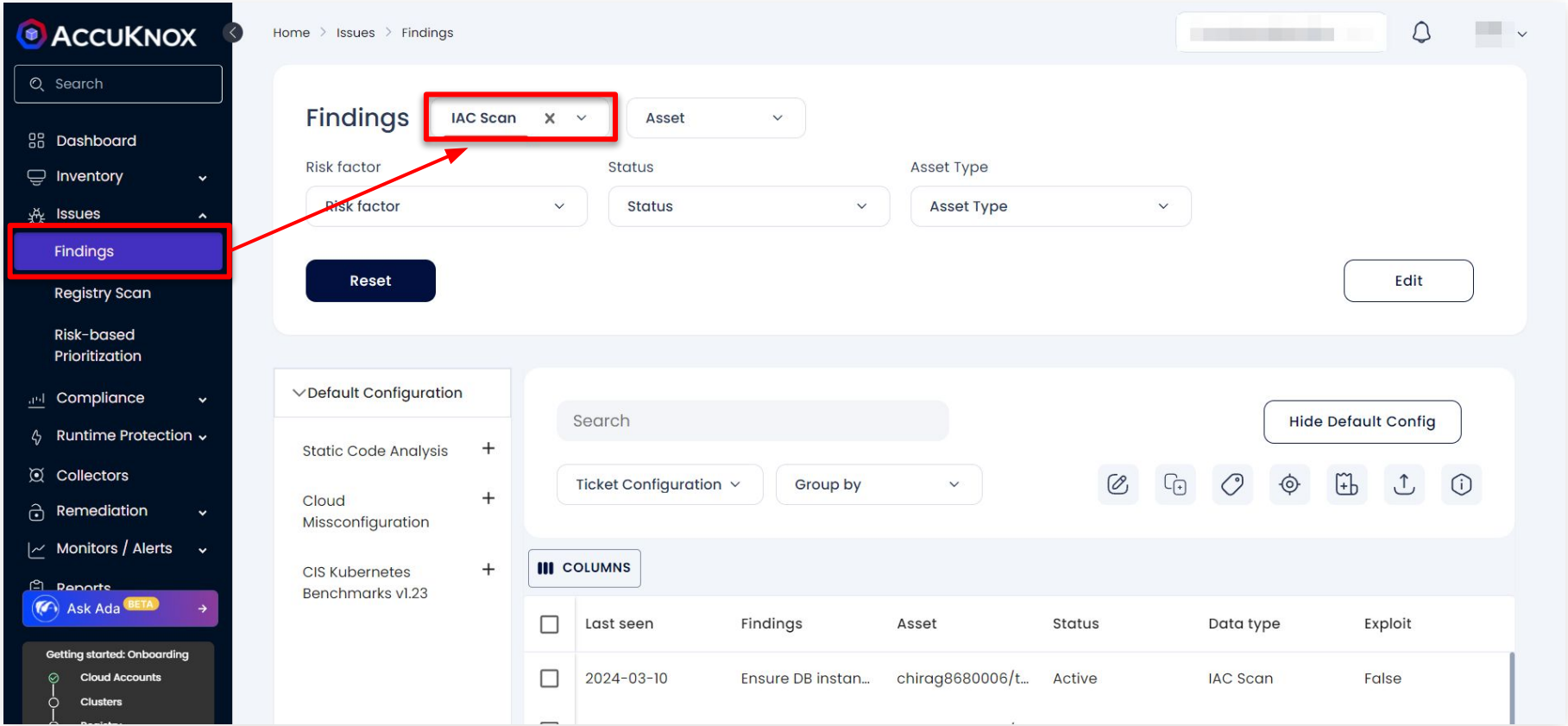
- Open the configuration page of your Jenkins job.
- Under the Build section, click on Add build step and select AccuKnox IaC scan.
- Fill all the required parameters.
- Review the findings in AccuKnox > Issues > Findings > Container scan.



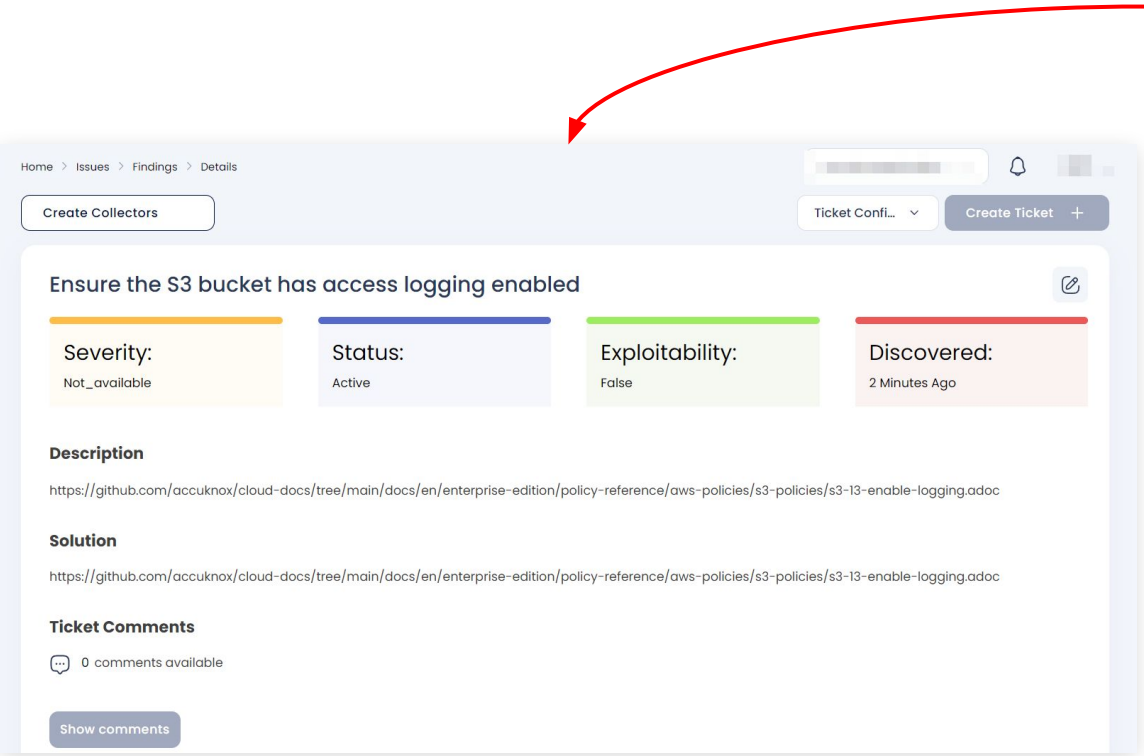


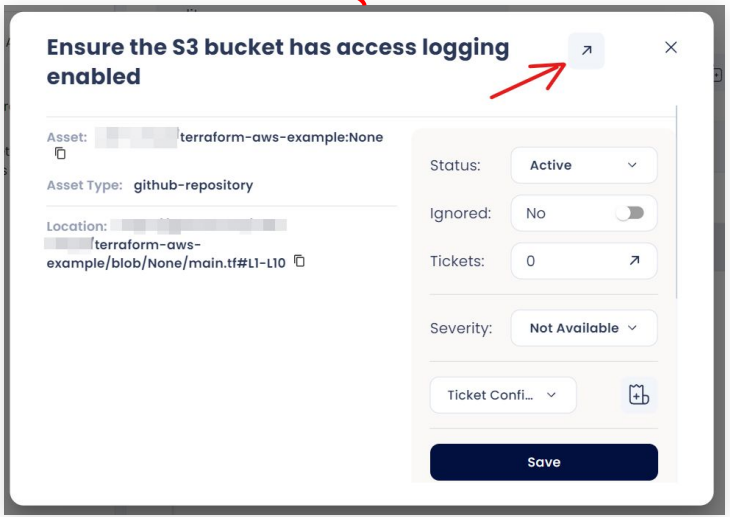


- Go to the AccuKnox > Findings and select the IAC Scan from the drop down menu



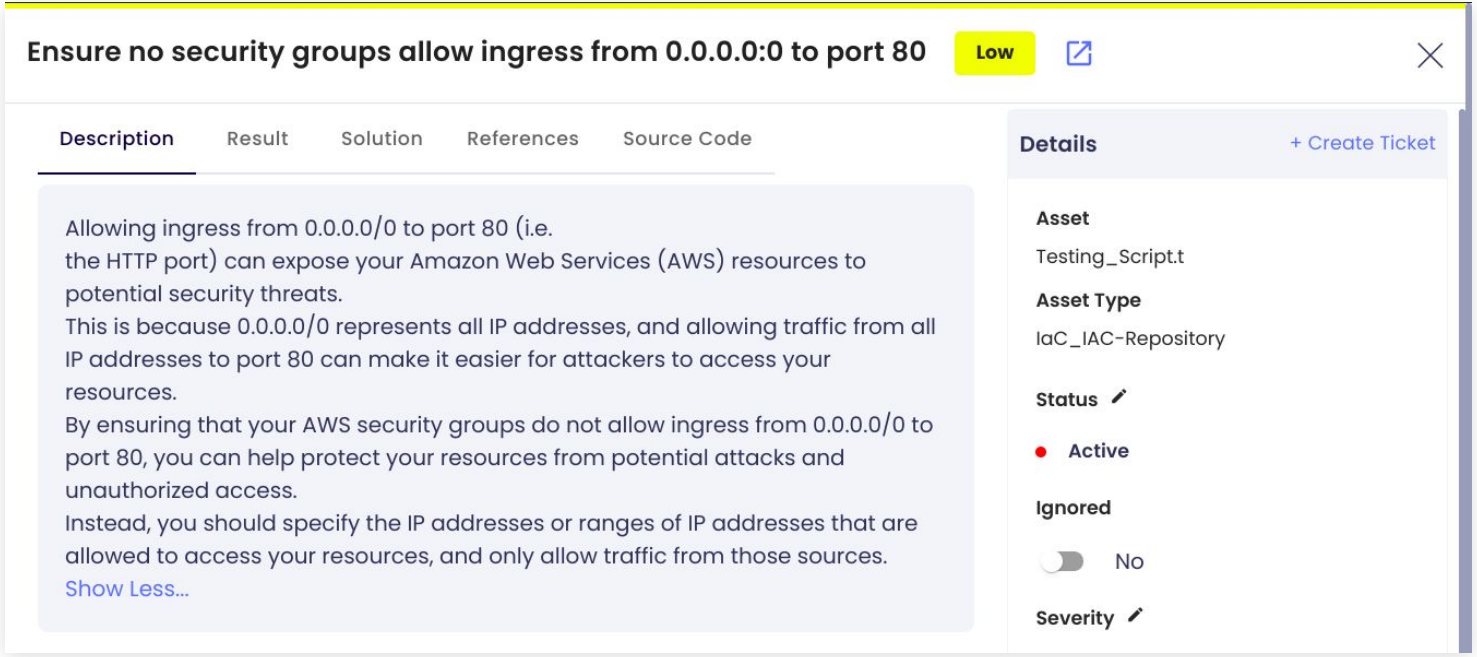
- To get a detailed view of a finding click on the finding and then click on the arrow icon.





Use case: Security group misconfigurations

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

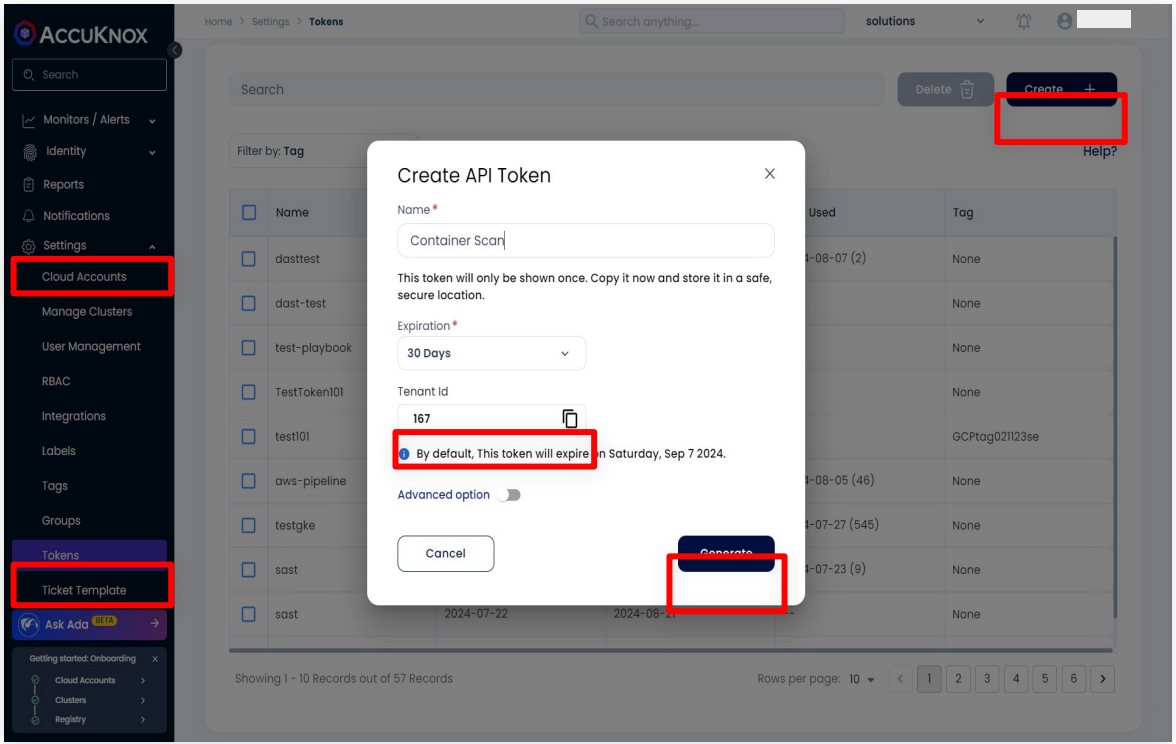


SAST Scan

ASPM Integration

Generate AccuKnox API token for CI/CD pipeline

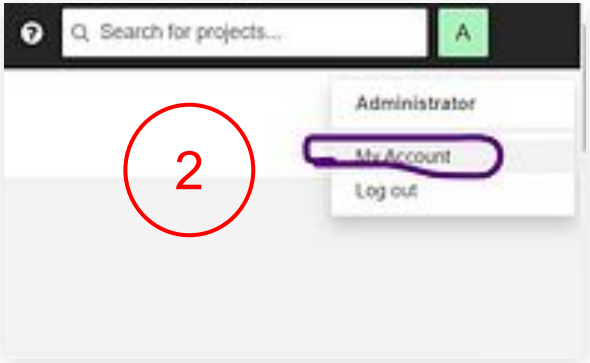
- To generate a token, open AccuKnox and navigate to **Settings > Tokens > Create**.
- Copy the token and tenant ID, then configure them as secrets in your CI/CD pipeline.

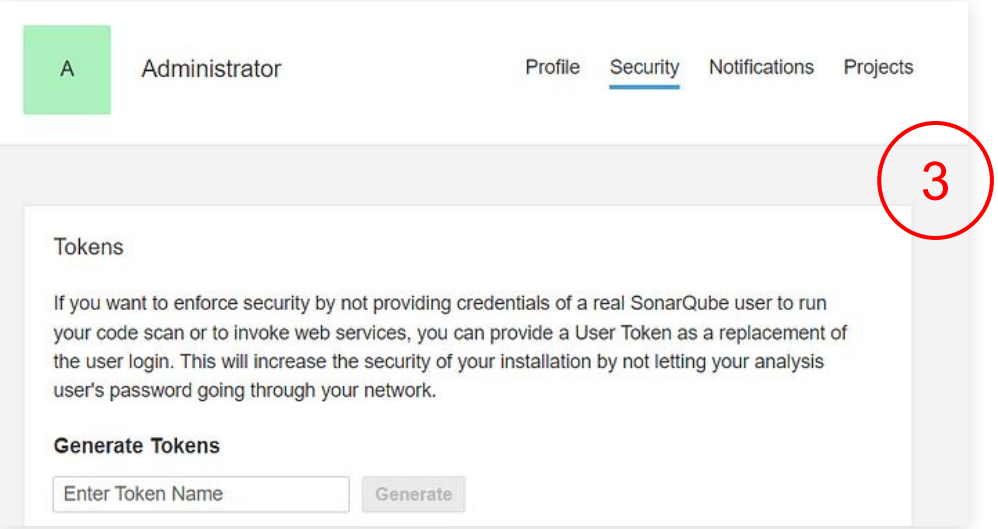


SonarQube configuration

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.

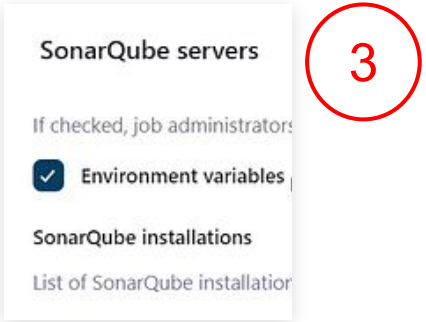
Step 4: Review Findings in AccuKnox

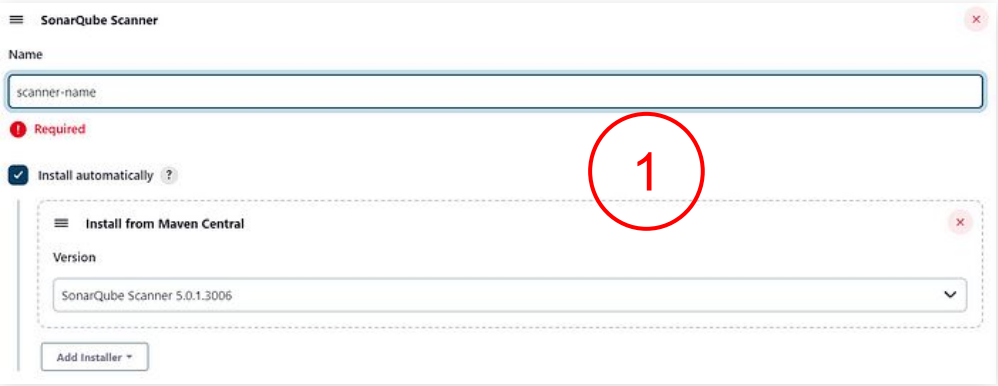
- Log in to your AccuKnox dashboard and navigate to the Issues section.
- Go to the "Findings" tab and select SAST Findings.
- Click on any finding that interests you to view detailed information and recommendations.

```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

- 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

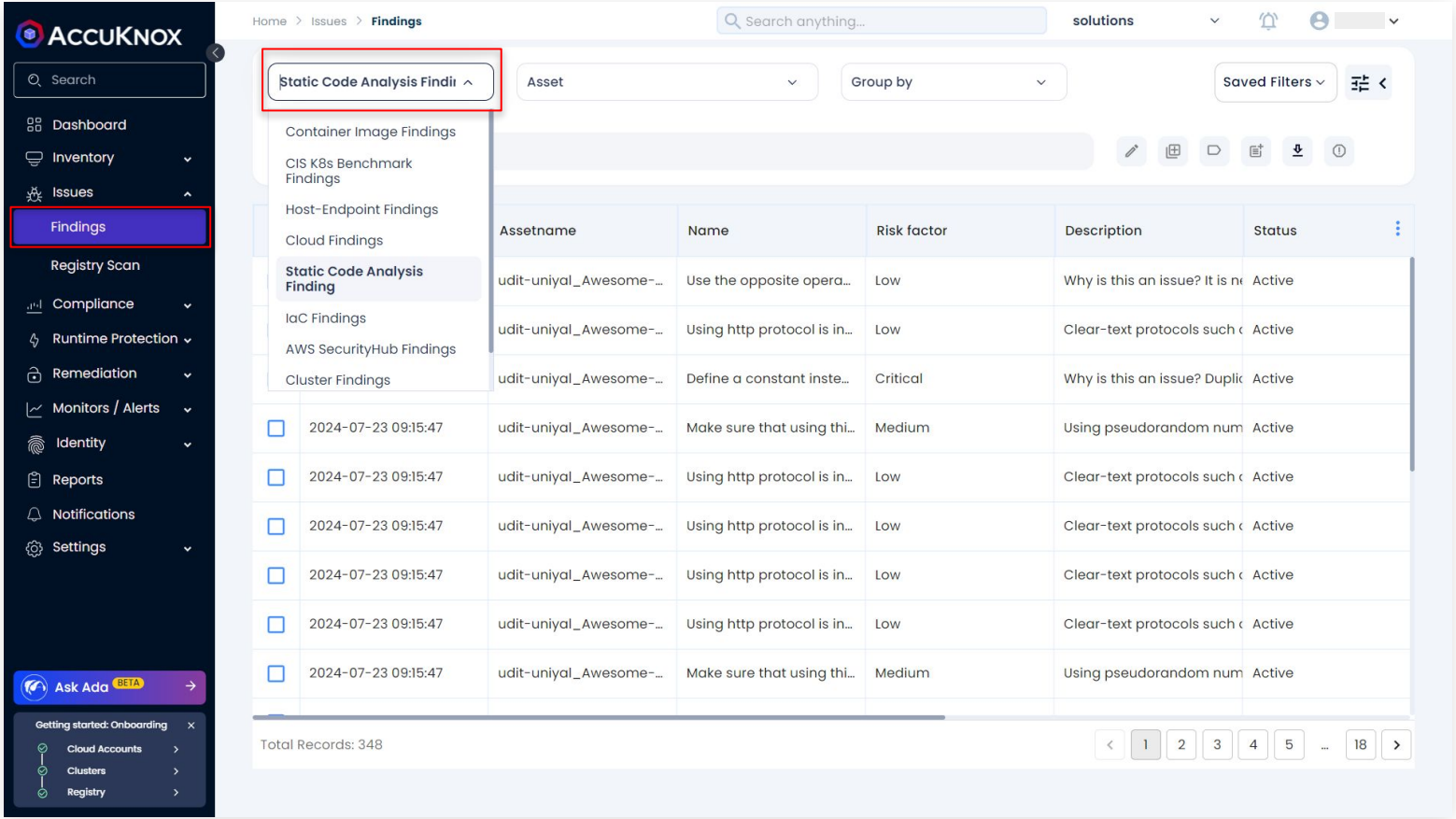




- Go to Manage Jenkins > Tools and add the SonarQube installation details.
- Create SonarQube credentials.
- Go to Manage Jenkins > System Configurations. Select the check-box of Injecting Environment variables and add the details of the SonarQube Server

```
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.



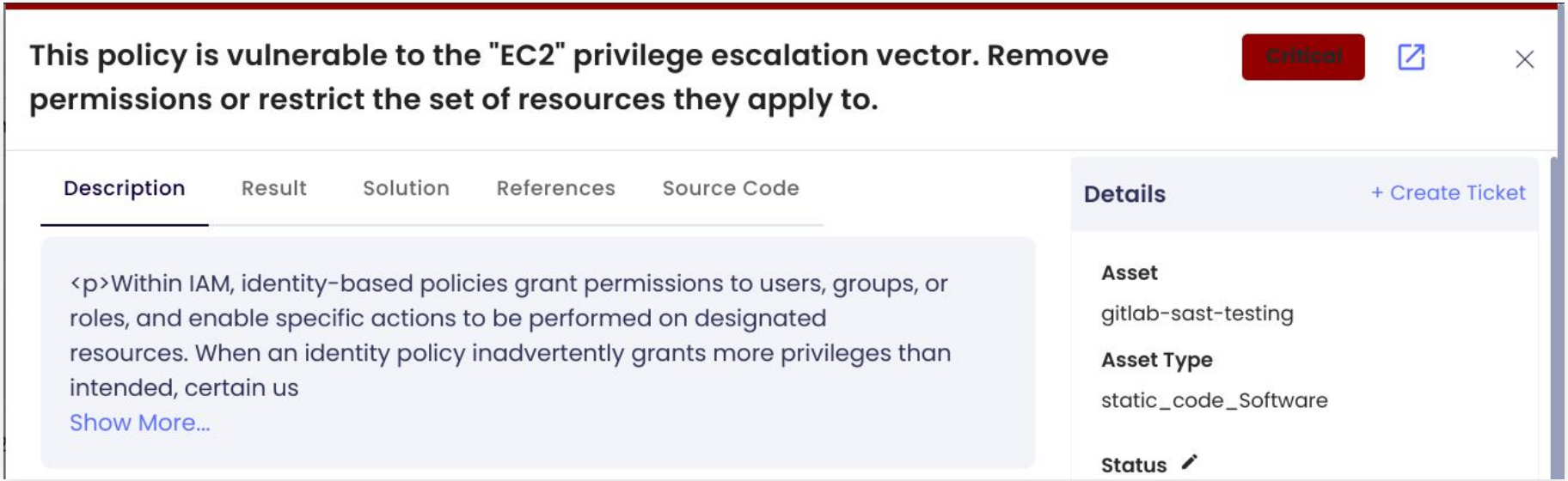
SAST use cases

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)

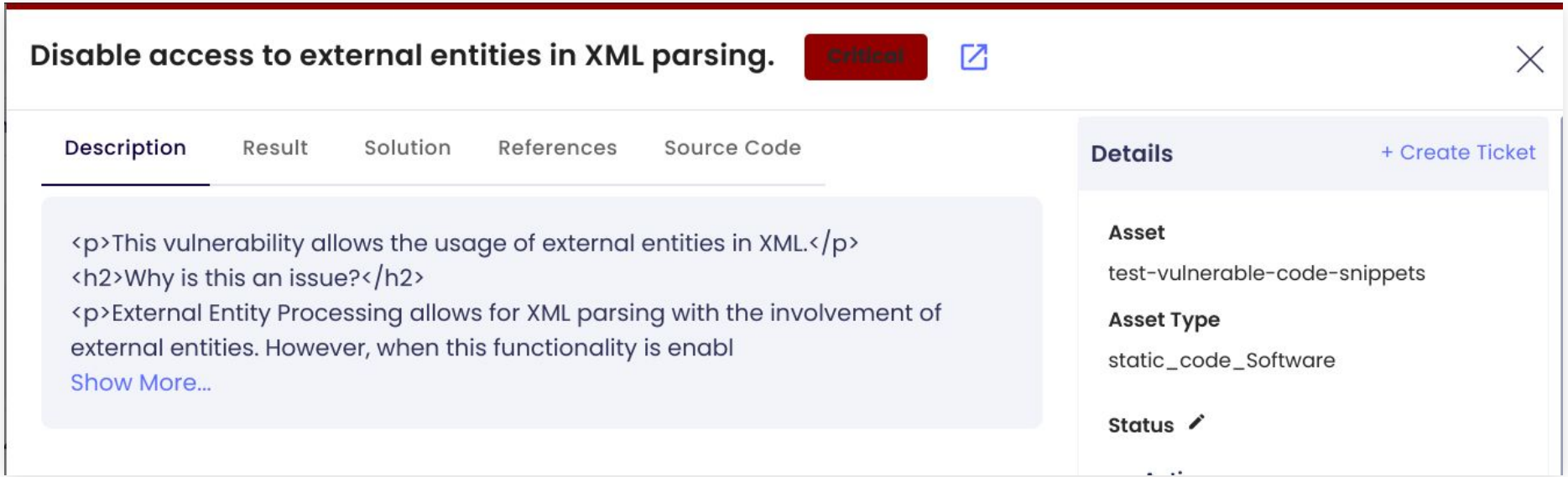
Use case 1: Privilege escalation

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



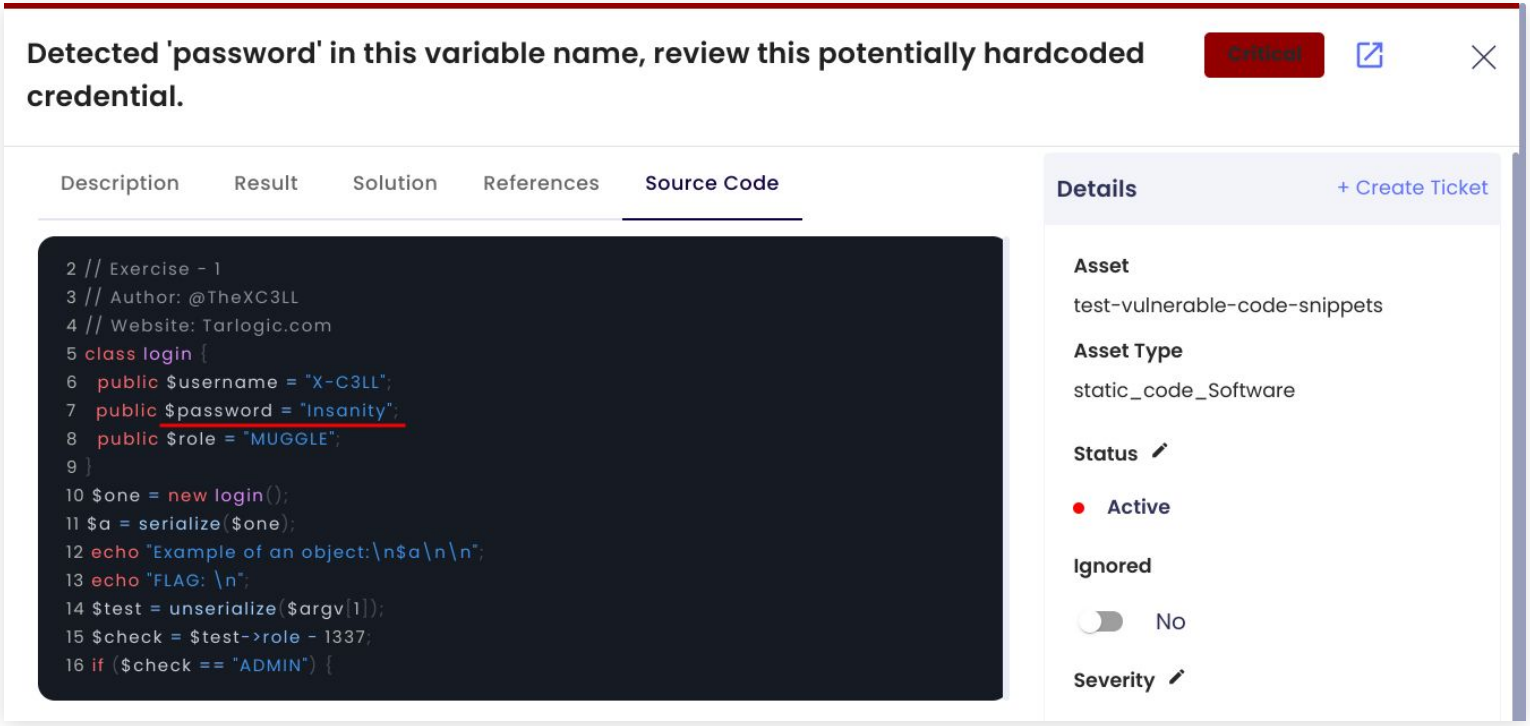
Use case 2: XML External Entity (XXE) injection

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



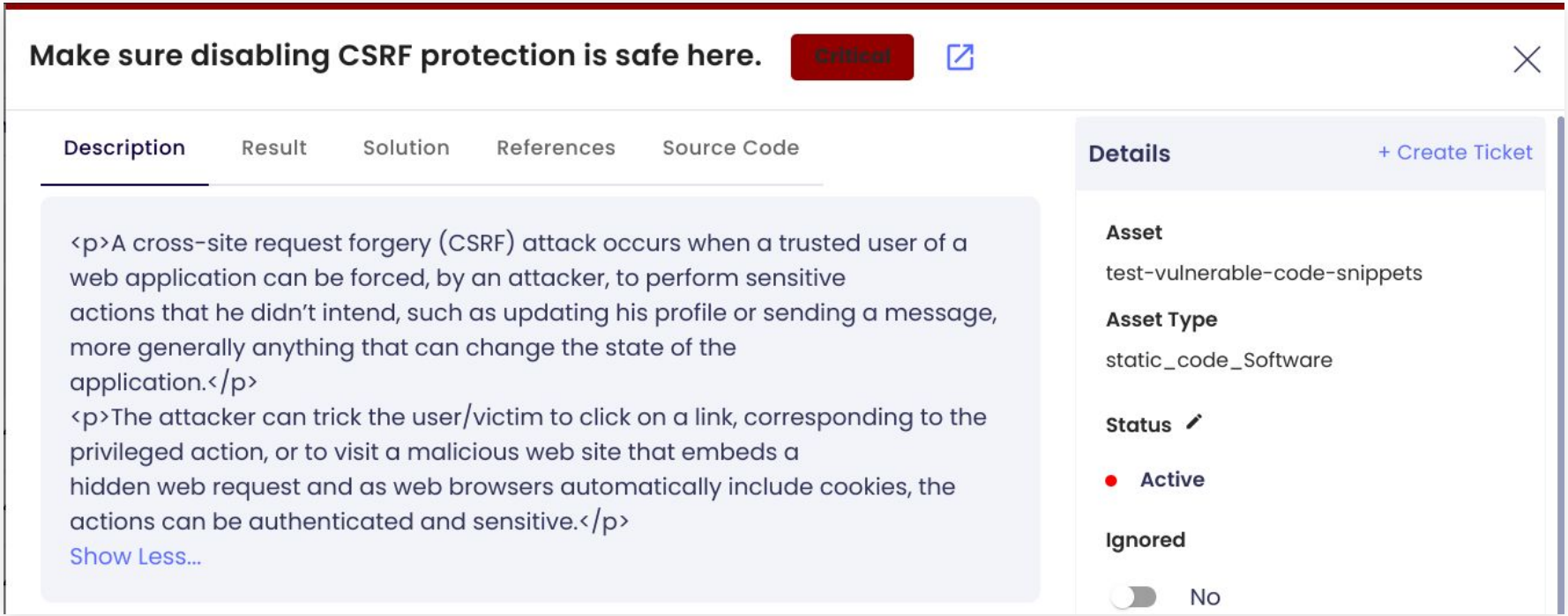
Use case 3: Hard coded password

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



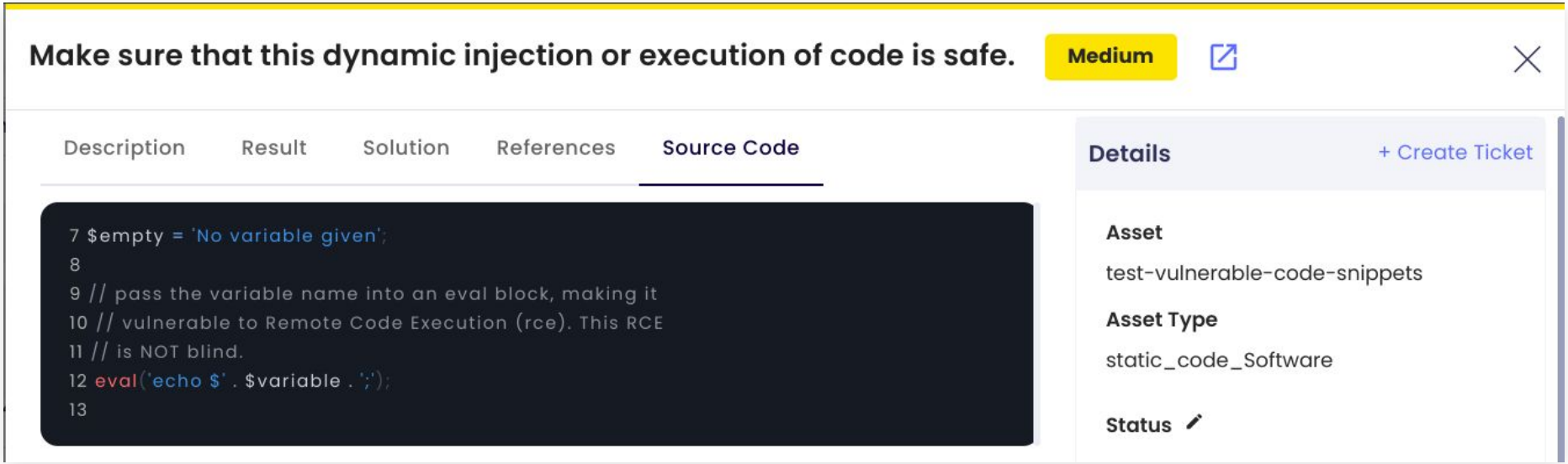
Use case 4: Cross Site Request Forgery (CSRF)

- 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.



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.



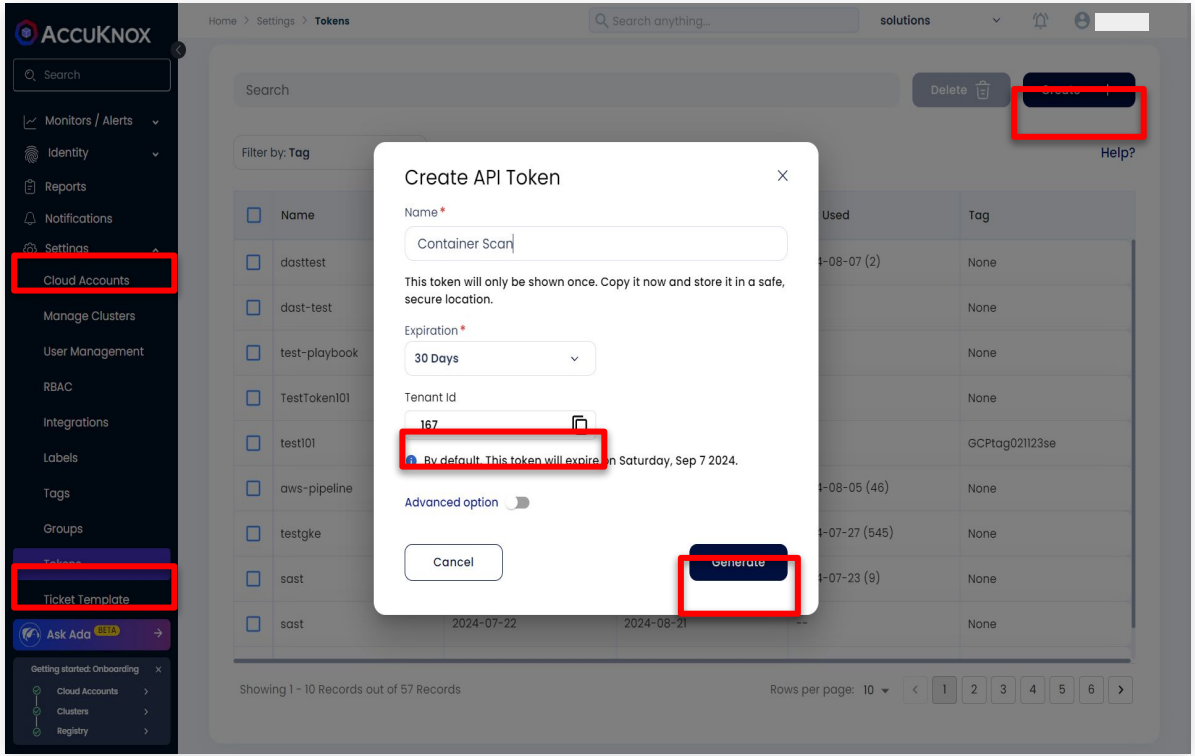
AccuKnox

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.

```yaml
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:

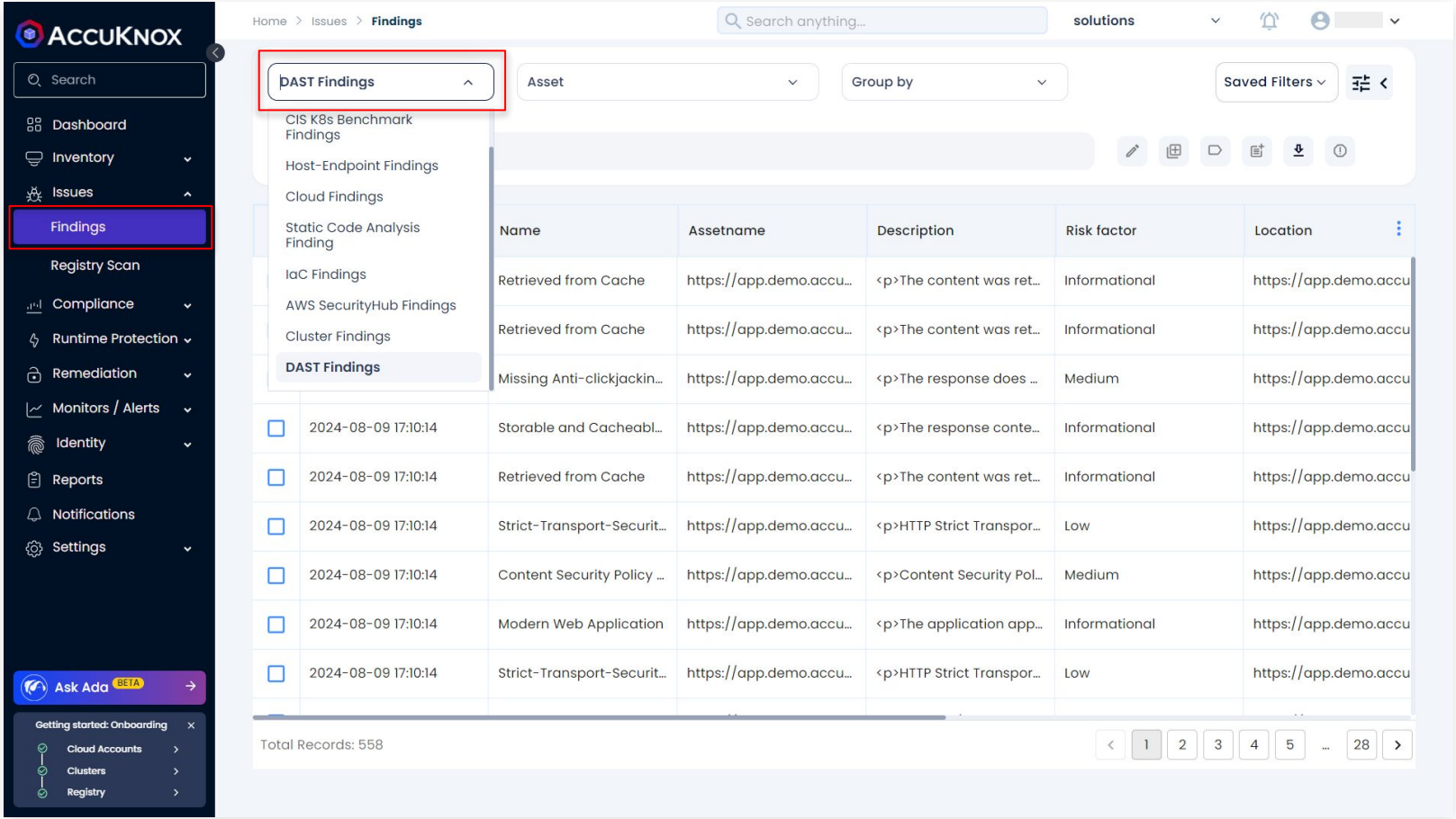
- After adding the script, click "Save" or "Apply."

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.



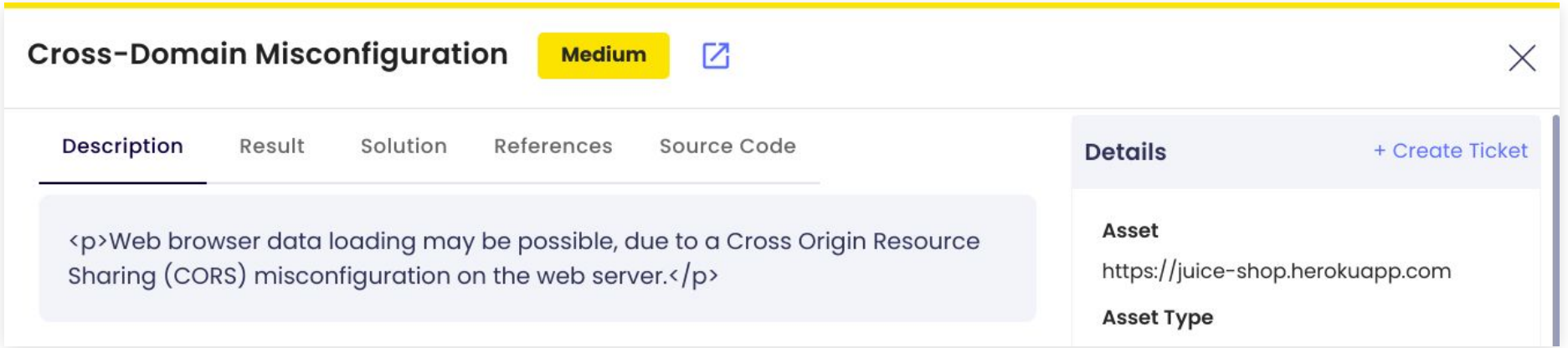
DAST Use Cases

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

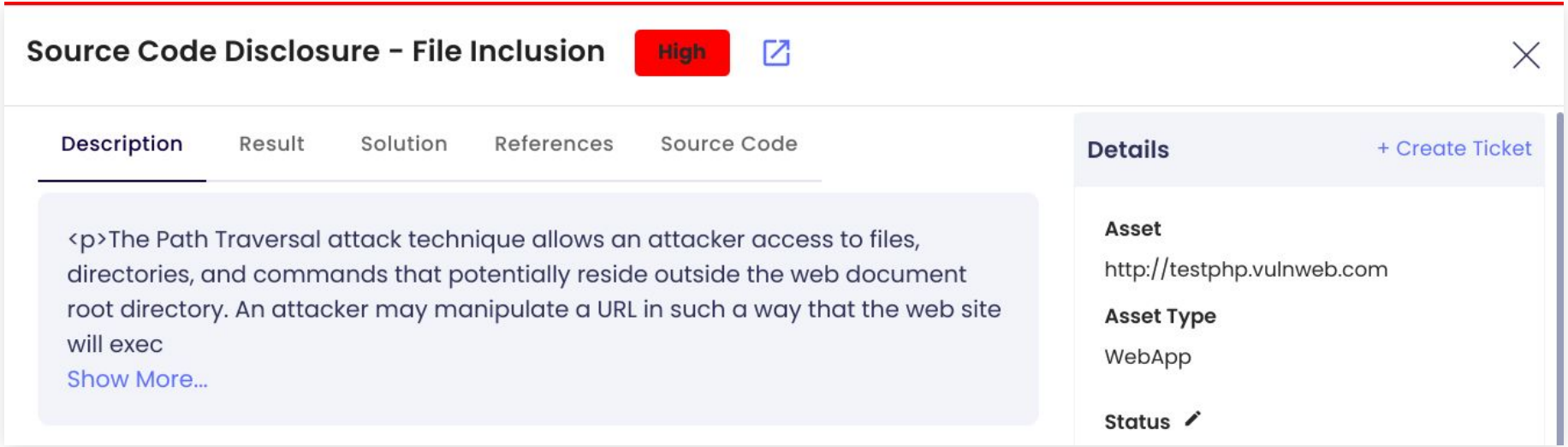
Use case 1: Cross Origin Resource Sharing (CORS) misconfiguration

- 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.



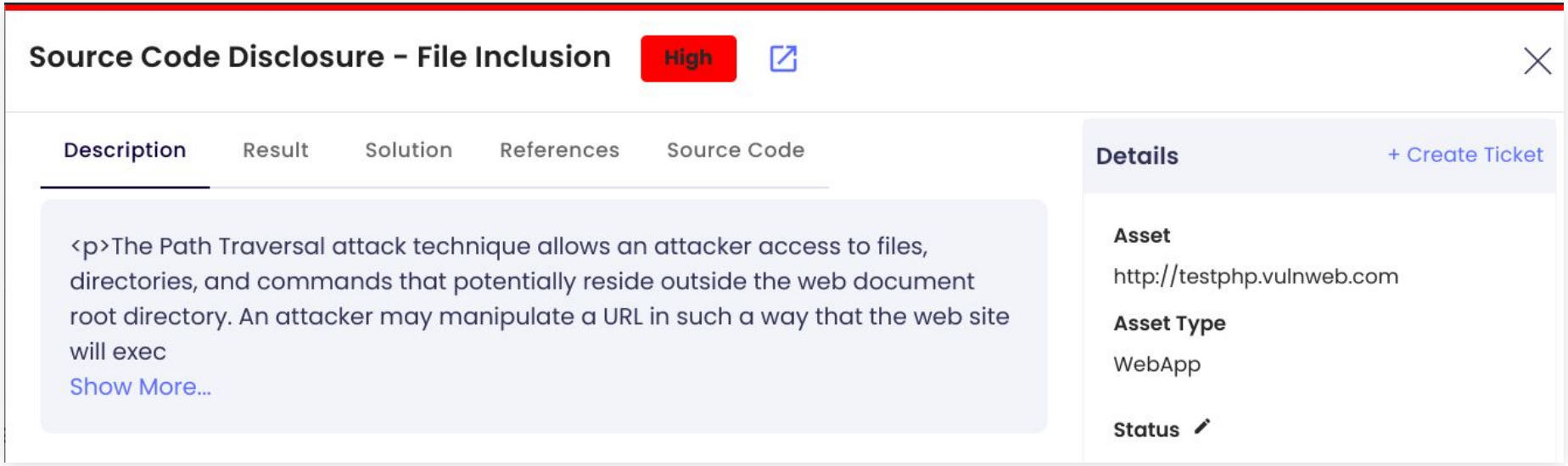
Use case 2: File inclusion vulnerability

- 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.



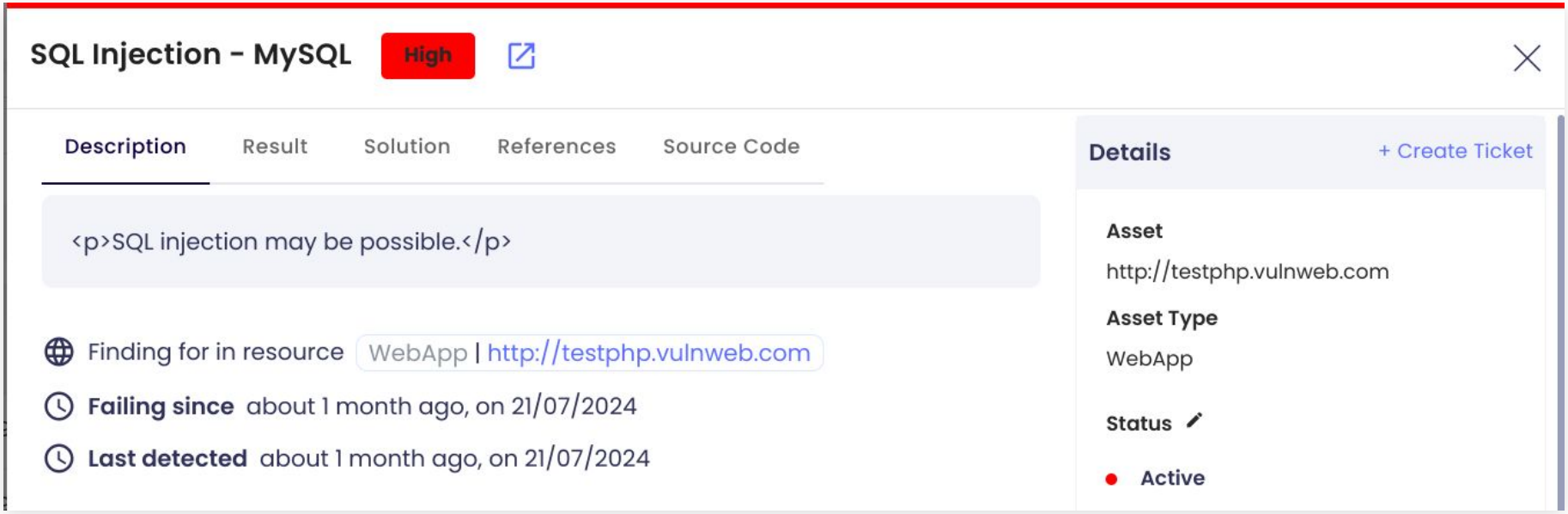
Use case 3: Cross site scripting vulnerability

- 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



Use case 4: SQL injection

- 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.



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.