Introduction

**Estimated time needed: 20-25 minutes

Welcome to the hands-on lab for **Evaluating Software Composition Analysis (SCA)**.

Learning Objectives

After completing this lab, you will be able to:

- Download and install the **OWASP SCA Tool**
- Use the SCA tool to detect vulnerabilities in a project's components
- Output scan results to JSON and HTML formats
- Understand how to analyze the output of the tool
- Discover where the reports are stored

What is Software Composition Analysis (SCA)?

Modern applications may be developed using 3rd-party and open-source components. Other people code these components. How do you know if the components are safe? How can you verify their security? You can do this by learning how to use software composition analysis tools to inspect the components used in your applications. You can ensure your code is safe from any known component vulnerabilities.

Software Composition Analysis (SCA)

Software Composition Analysis (SCA) is the process of identifying areas of risk that result from the use of third-party and open-source components during application development.

SCA tools can identify several risk factors, including:

- Outdated components.
- Components with known vulnerabilities.
- Component quality.
  - From a security standpoint, a component might be considered lower quality if it is poorly maintained or has a very small community supporting it.
- Transitive dependencies.
  - SCA tools can track vulnerabilities from **transitive dependencies**. When a component relies upon another component, that dependency is referred to as **transitive**.
- External services
  - A component may interact with external services, such as Web APIs. SCA tools can identify when this interaction might be a vulnerability.

Dependency-Check is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies. Dependencies are the software components that your code relies on for additional functionality. The SCA tool will generate a report listing the dependency, any identified Common Platform Enumeration (CPE) identifiers, and the associated Common Vulnerability and Exposure (CVE) entries.

In this hands-on lab, we will explore the use of the **OWASP SCA Dependency-checker** tool.

Step 1: Installing the OWASP SCA Tool

You have a little preparation to do before you can start the hands-on lab.

Your Task

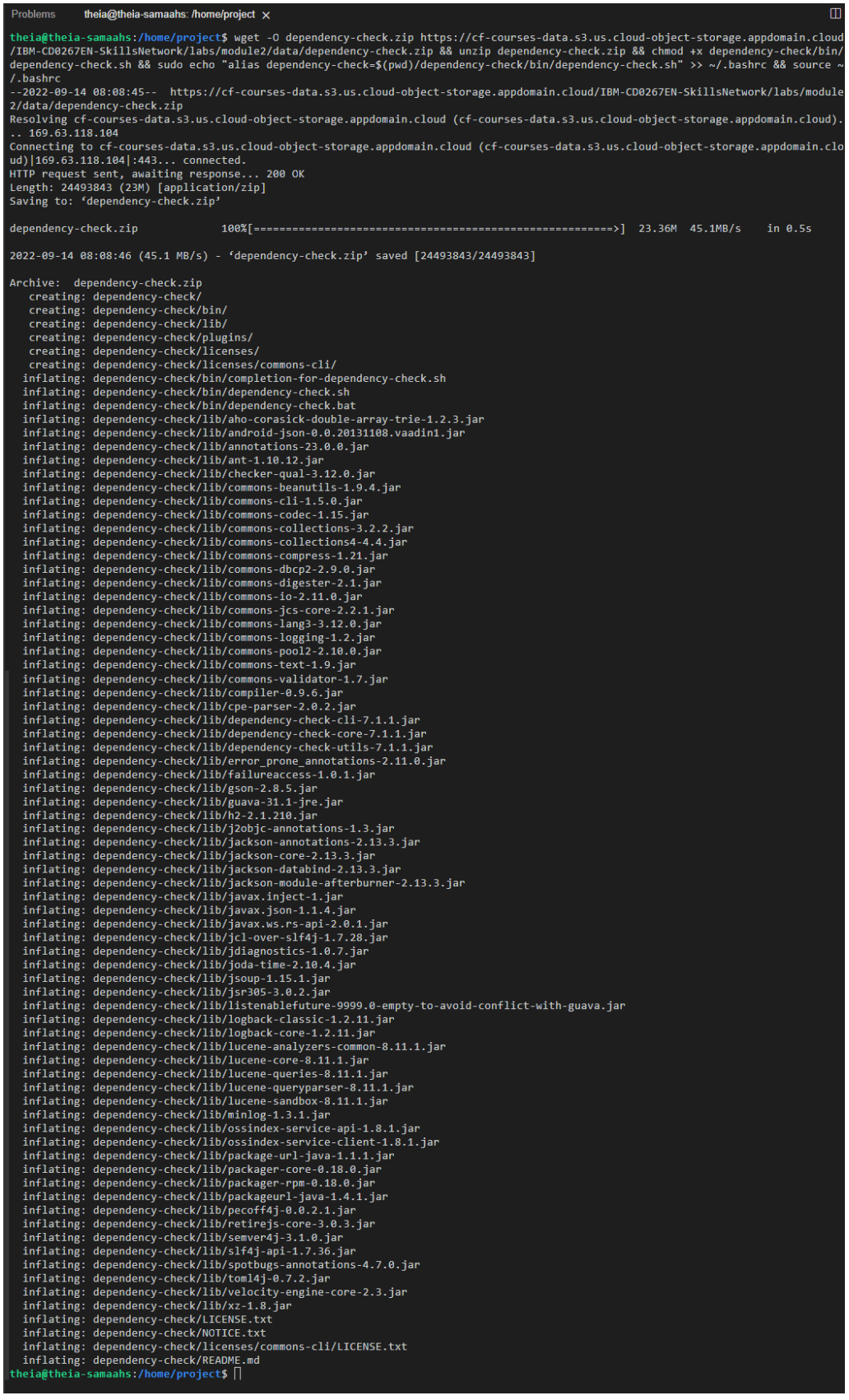
1. Run the `wget` command to download and install the OWASP dependency-check script:

```
1. 1
1. wget -O dependency-check.zip https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0267EN-SkillsNetwork/labs/module2/data/dependency-check.zip && unzip dependency-check.zip && chmod
```

Copied! Executed!

Results

You should see an output similar to this:



Now you'll be able to run the OWASP SCA tool using `dependency-check` on the command line.

Step 2: Download the Source Code

Next, we'll need some source code to scan. We'll use a popular example of a vulnerable application called **OWASP Juice Shop**, explicitly created for security training purposes.

Your Task

1. Run the following `git clone` command to download the source code:

```
1. 1
1. git clone https://github.com/juice-shop/juice-shop.git
```

Copied! Executed!

Results

Your output should look something like this:

```
theia@theia-samaahs:/home/project$ git clone https://github.com/juice-shop/juice-shop.git
Cloning into 'juice-shop'...
remote: Enumerating objects: 115562, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 115562 (delta 12), reused 52 (delta 11), pack-reused 115505
Receiving objects: 100% (115562/115562), 179.76 MiB | 34.31 MiB/s, done.
Resolving deltas: 100% (89637/89637), done.
theia@theia-samaahs:/home/project$
```

Now you're ready to analyze the **Juice Shop** application.

Step 3: Run SCA on Juice Shop Components

The OWASP SCA tool can create a Hypertext Markup Language (HTML) report or JavaScript Object Notation (JSON) output. In this step, we'll run the tool and output the results to a JSON output.

Your Task

1. Use the `dependency-check` command on the **Juice Shop** source code with the following options to produce a JSON output:

```
1. 1
1. dependency-check -f JSON --prettyPrint --scan juice-shop
```

Copied! Executed!

> Note: This may take a while because the first time you run it, it must download all of the updates from the CVE database. Subsequent runs will complete more quickly.

The command will produce a file called `dependency-check-report.json` which may contain information about any vulnerable components found by the OWASP SCA Tool.

Scan Results

Results

The results of the scan will look similar to this:

```
Terminal  Help
Problems  theia@theia-samaahs: /home/project/juice-shop
Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependenc
ies; false positives and false negatives may exist in the analysis performed by the tool. Use of
 the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and t
here are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of
 the tool and the reporting provided is at the user's risk. In no event shall the copyright hold
er or OWASP be held liable for any damages whatsoever arising out of or in connection with the u
se of this tool, the analysis performed, or the resulting report.

   About ODC: https://jeremylong.github.io/DependencyCheck/general/internals.html
   False Positives: https://jeremylong.github.io/DependencyCheck/general/suppression.html

💔Sponsor: https://github.com/sponsors/jeremylong

[INFO] Analysis Started
[WARN] Exception extracting archive 'arbitraryFileWrite.zip'.
[WARN] Exception extracting archive 'videoExploit.zip'.
[INFO] Finished Archive Analyzer (0 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[WARN] An error occurred with the .NET AssemblyAnalyzer, please see the log for more details.
[ERROR] Exception occurred initializing Assembly Analyzer.
[WARN] No lock file exists - this will result in false negatives; please run `npm install --pack
age-lock`
[WARN] No lock file exists - this will result in false negatives; please run `npm install --pack
age-lock`
[WARN] Analyzing `/home/project/juice-shop/package.json` - however, the node_modules directory d
oes not exist. Please run `npm install` prior to running dependency-check
[WARN] Analyzing `/home/project/juice-shop/frontend/package.json` - however, the node_modules di
rectory does not exist. Please run `npm install` prior to running dependency-check
[INFO] Finished Node.js Package Analyzer (0 seconds)
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Created CPE Index (3 seconds)
[INFO] Finished CPE Analyzer (4 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished RetireJS Analyzer (0 seconds)
[WARN] Unable to determine Package-URL identifiers for 20 dependencies
[INFO] Finished Sonatype OSS Index Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Analysis Complete (5 seconds)
[INFO] Writing report to: /home/project/juice-shop/./dependency-check-report.json
[ERROR] Could not execute .NET AssemblyAnalyzer
theia@theia-samaahs:/home/project/juice-shop$
```

Step 4: Analyzing JSON Results

When we ran the command in Step 3, the report `dependency-check-report.json` was saved in the `/home/project` folder. You can view the file by opening it manually or by pressing this button:

Open **dependency-check-report.json** in IDE

Since JSON can be challenging to read, you might want to modify the output to only display the name of the affected files.

To do this you can use the `jq` command and append the `'.dependencies[].filePath'` parameter to filter the JSON output to only display the path of the affected files.

Your Task

1. Use the following `jq` command to list the software components that were found to be vulnerable by the OWASP tool:

```
1. 1
1. jq '.dependencies[].filePath' dependency-check-report.json
```

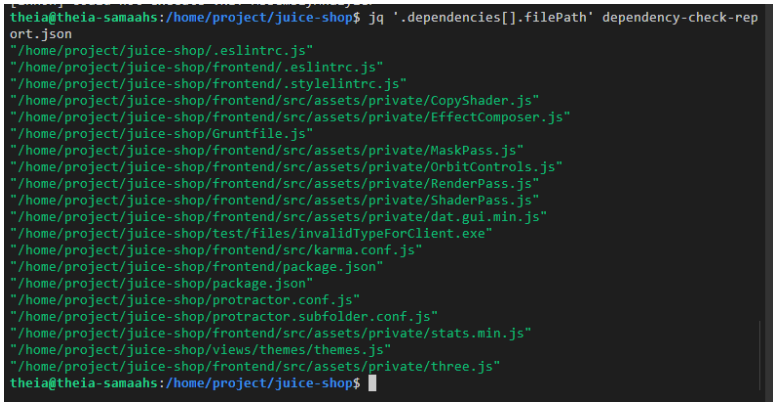
Copied! Executed!

Note: This search will return the `filePath` attribute of all of the items in the `dependencies[]` array in the file `dependency-check-report.json`.

Filter Output Results

Results

The output should look like this:



In a secure Software Development Life Cycle (SDLC), each component should be thoroughly checked and verified for security. Any dependencies that might be vulnerable should be upgraded or replaced.

Step 5: Creating an HTML report

Reports outputted to JSON format can be difficult to read. It may be preferable to send your scan output to an HTML report instead. HTML reports are easier for us to read and interpret.

Your Task

1. Use the `dependency-check` command to create an HTML report from a `--scan` of the `juice-shop` folder:

```
1. 1
1. dependency-check --scan juice-shop
```

Copied! Executed!

Once the process has completed, a report called `dependency-check-report.html` will be located in your file explorer. Unfortunately, the Cloud IDE environment does not have a way to render this HTML file, so you must download the report and view it in your web browser.

Let's take a closer look at an example HTML report.

Results

The **Project:** section of the report displays **Scan Information**. This section contains the version of the dependency checker used, the date and time that the report was run, some additional information about the number of dependencies scanned, and how many vulnerabilities were found.

The **Summary** section of the report displays dependencies by name, the Vulnerability ID, and the package name. It also shows the level of criticality for each vulnerability and other details associated with the **Juice Shop** project.

DEPENDENCY-CHECK

How to read the report | **Suppressing false positives** | **Getting Help: github issues**

Sponsor

Project:

Scan Information (show all):

- *dependency-check version*: 7.1.1
- *Report Generated On*: Wed, 20 Jul 2022 21:09:42 GMT
- *Dependencies Scanned*: 15675 (10954 unique)
- *Vulnerable Dependencies*: 6
- *Vulnerabilities Found*: 11
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Eviden
express-jwt:0.1.3	cpe:2.3:a:auth0:express-jwt:0.1.3:*:*:*:*:*:*:*	pkg:npm/express-jwt@0.1.3	CRITICAL	1	Highest	10
hbs:4.2.0	cpe:2.3:a:hbs_project:hbs:4.2.0:*:*:*:*:*:*:*	pkg:npm/hbs@4.2.0	MEDIUM	1	Highest	10
jquery.js		pkg:javascript/jquery@2.1.0	MEDIUM	4		3
jsonwebtoken:0.4.0	cpe:2.3:a:auth0:jsonwebtoken:0.4.0:*:*:*:*:*:*:*	pkg:npm/jsonwebtoken@0.4.0	CRITICAL	1	Highest	8
notevil:1.3.3	cpe:2.3:a:notevil_project:notevil:1.3.3:*:*:*:*:*:*:*	pkg:npm/notevil@1.3.3	MEDIUM	1	Highest	8
sanitize-html:1.4.2	cpe:2.3:a:punkave:sanitize-html:1.4.2:*:*:*:*:*:*:*	pkg:npm/sanitize-html@1.4.2	MEDIUM	3	Highest	8

Dependencies

express-jwt:0.1.3

Description:

JWT authentication middleware.

File Path: /home/project/juice-shop_14.1.1/package.json?/express-jwt:0.1.3

Referenced In Project/Scope:juice-shop:14.1.1

Evidence

Identifiers

- pkg:npm/express-jwt@0.1.3 (*Confidence*:Highest)
- cpe:2.3:a:auth0:express-jwt:0.1.3:*:*:*:*:*:*:* (*Confidence*:Highest) suppress

As you can see, the html version is easier to interpret. You would click on all of the vulnerabilities to understand them and remediate them one by one.

Conclusion

Congratulations! You have learned how **Software Composition Analysis (SCA)** can help you detect known vulnerabilities in any dependencies your code is dependent on. This is an integral part of secure software application development.

In this lab, you have learned to download, install, and configure the **OWASP SCA Dependency-check Tool** to perform dependency analysis on components used in a software application. You learned how to output scan results to JSON and HTML formats and where the reports are stored. You familiarized yourself with the layout of the HTML report and learned how to analyze the results.

Next Steps

Detecting different vulnerabilities is just one of the first steps in secure app development. It is helpful to understand the meaning behind those vulnerabilities to take corrective actions to mitigate them. There is no better way to learn than by doing.

Your next challenge is installing OWASP SCA Dependency-Check in your development environment, performing dependency scans on your code, and fixing any problems it may find. You are well on your way to writing more secure code!

Author(s)

Sam Prokopchuk

Other Contributor(s)

Samaah Sarang
John J. Rofrano
Michelle R. Sanchez, Instructional Designer at Skill-up Technologies with over 25 years of enterprise-level technical support and enterprise technical training.

Changelog

Date	Version	Changed by	Change Description
2022-07-20	0.1	Sam Prokopchuk	Initial version created
2022-09-14	0.2	Samaah Sarang	Screenshots added
2022-09-16	0.3	John Rofrano	Added additional content and formatting
2022-09-19	0.4	Michelle Sanchez	Corrective edits to spelling, grammar. Added LO, Summary, Next Steps
2022-09-20	0.4	Steve Hord	QA pass edits