Evaluating Vulnerability Analysis

Estimated time needed: 20 minutes

Introduction

Welcome to the hands-on lab for **Evaluating Vulnerability Analysis**.

When you are tasked with developing a secure application, building security into the software development lifecycle is important. However, that does not guarantee that your app will be free from vulnerabilities when it is time to deploy it. Performing a vulnerability analysis can help you identify vulnerabilities lurking inside your app before it goes live.

A vulnerability analysis involves a systematic and thorough review of possible known weaknesses or vulnerabilities in a system or application. Revealing vulnerabilities is the first step in remedying the issue and developing a secure system.

Learning Objectives

After completing this lab, you will be able to:

- Perform a vulnerability scan on a sample web application using the Jake tool
- Evaluate the results of the vulnerability scan
- Explain why performing vulnerability analyses are essential to developing secure software applications

Set up the Lab Environment

You have a little preparation to do before you can start the lab. In the following steps, you will download and install a vulnerability scanner along with a sample web application to test against.

For this lab, the applications you will use are:

- Jake, a vulnerability scanner developed by the SonaType Nexus Community
- Hit Counter (a sample web application)

Both of these software applications are available as Docker images.

What is Jake?

Jake is a community-created, open source tool that is designed to check Python environments for vulnerabilities. While Jake is not created by or supported by Sonatype, it uses the Sonatype OSS Index.

What is Hit Counter?

Hit Counter is a Python Flask application. It has not been updated in a while. This makes it a good application check for security vulnerabilities. It currently uses an older version of Flask 1.1.4, and its dependencies have, unfortunately, become vulnerable over the years.

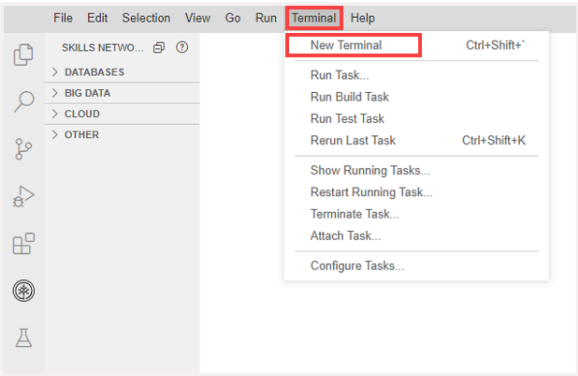
Step 1: Install the Jake tool

You will install `jake` using the Python package manager (`pip`) by running the following command in the command line interface.

Your Task

1. Open a new terminal from the top menu with: **Terminal New Terminal** and `cd` to `home/projects`:

    ```
    cd /home/project
    ```



2. Install Python virtual environment support:

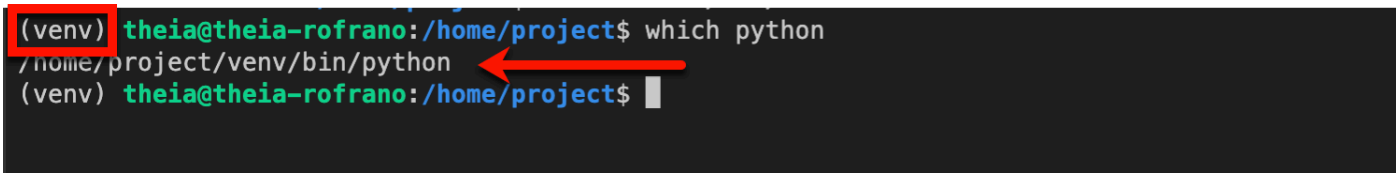
    ```
    sudo apt-get update && sudo apt-get install -y python3-venv
    ```

3. Create a Python virtual environment called `venv` and activate it:

    ```
    python3 -m venv venv
    source venv/bin/activate
    ```

4. Run the `which python` command to make sure that the environment is active:

    ```
    which python
    ```



You should see `(venv)` before the prompt and `which python` should return `/home/project/venv/bin/python`. If you see both of these, everything is working properly.

5. Run the following `pip install` command to install `jake` into that virtual environment:

```
pip install jake
```

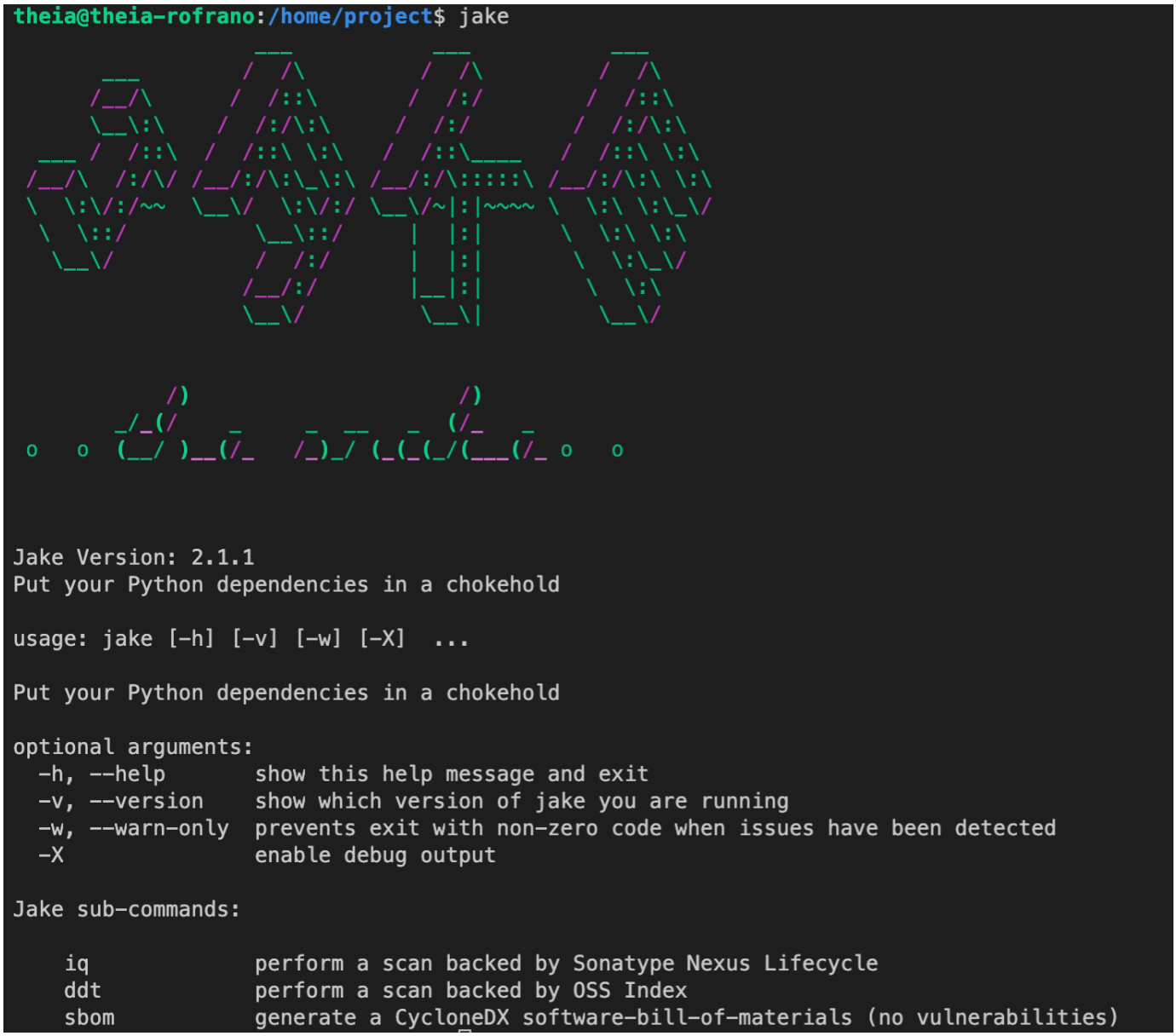
You will see a variety of packages being installed that `jake` depends on.

6. Run `jake` to make sure that it works:

```
jake
```

Results

You should see the following output:



Step 2: Install the code to scan

Now that you have your vulnerability scanning tool installed, you will also need an application to scan for vulnerabilities. You will use a Python Flask application that is called **Hit Counter**. It has some outdated dependencies, which will provide a good example for vulnerability analysis.

Your Task

To get started, first install the application by pulling its Docker image.

1. Use the following `git clone` command in the terminal to pull the code:

   ```
   git clone https://github.com/ibm-developer-skills-network/ycuer-flask-hitcounter.git
   ```

2. Then `cd` into the `ycuer-flask-hitcounter` folder.

   ```
   cd ycuer-flask-hitcounter
   ```

3. Use the `pip install` command to install the Python dependencies for Hit Counter from the `requirements.txt` file:

   ```
   pip install -r requirements.txt
   ```

You are now ready to scan the Hit Counter application.

Step 3: Running a Vulnerability Scan

Now that we have a local copy of the application, we can run `jake` on the code to see what vulnerabilities we might find. Jake can run in several modes. The `ddt` mode does static scanning for vulnerable dependencies, which is what we are looking to find.

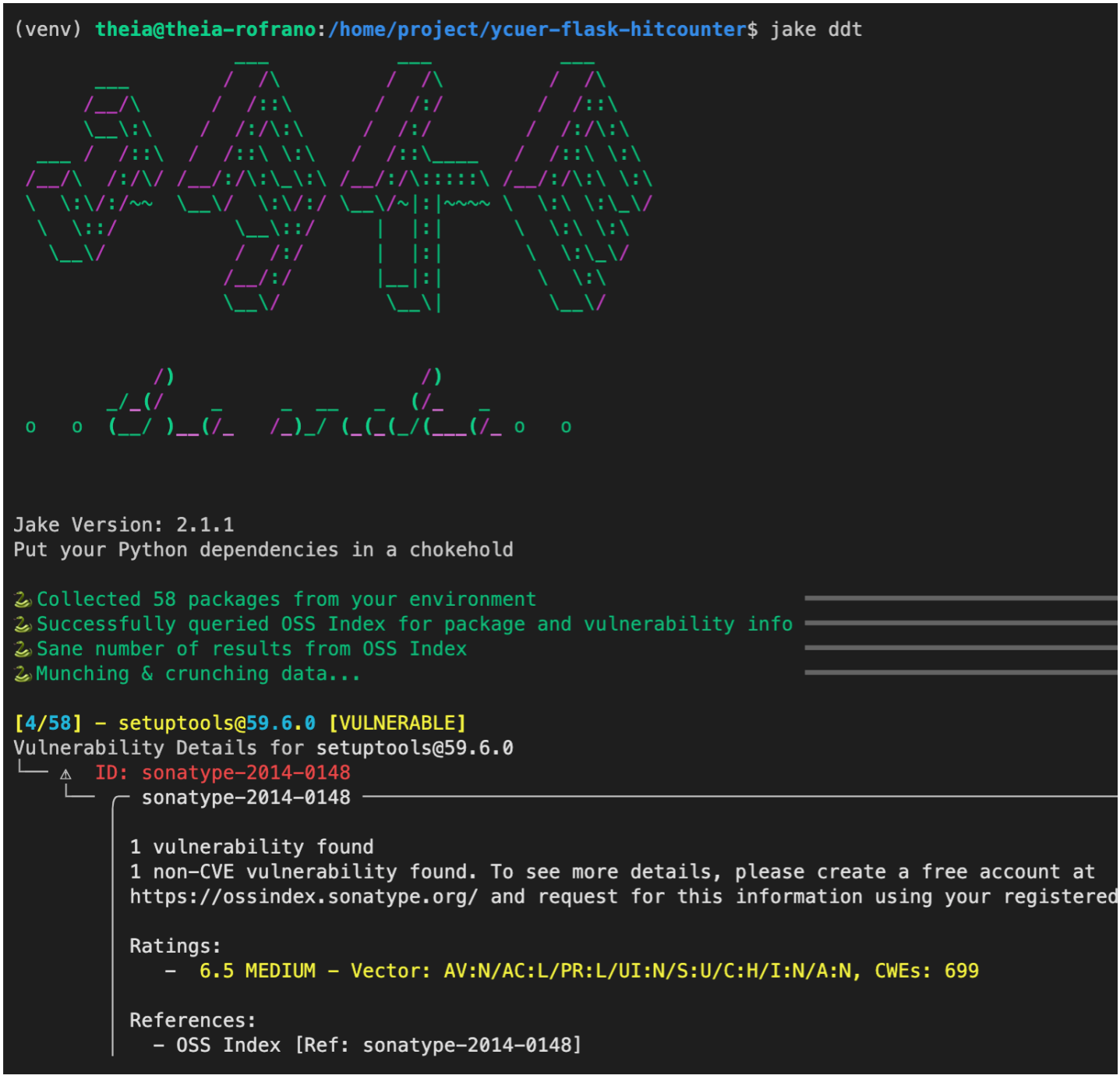
Your Task

1. Run the `jake` command from the `ycuer-flask-hitcounter` folder:

```
jake ddt
```

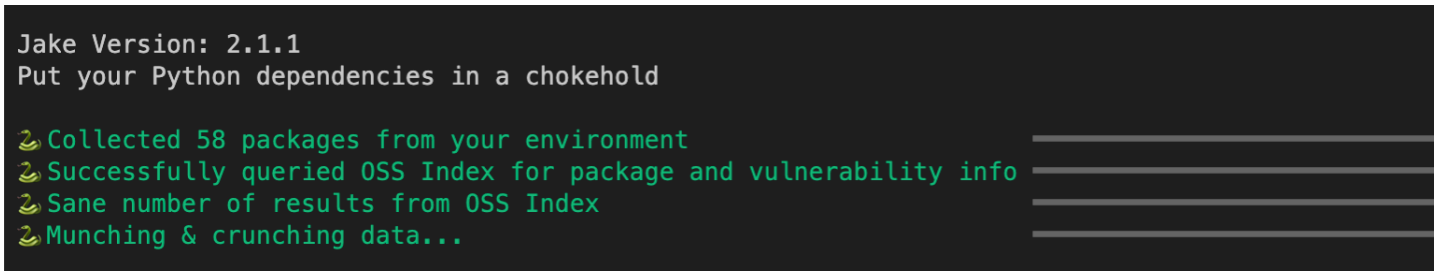
Results

The output should look similar to the following image. We say similar because, over time, it may find more vulnerabilities. Depending on when you run this lab, you might get different results. This is just the top of the output:

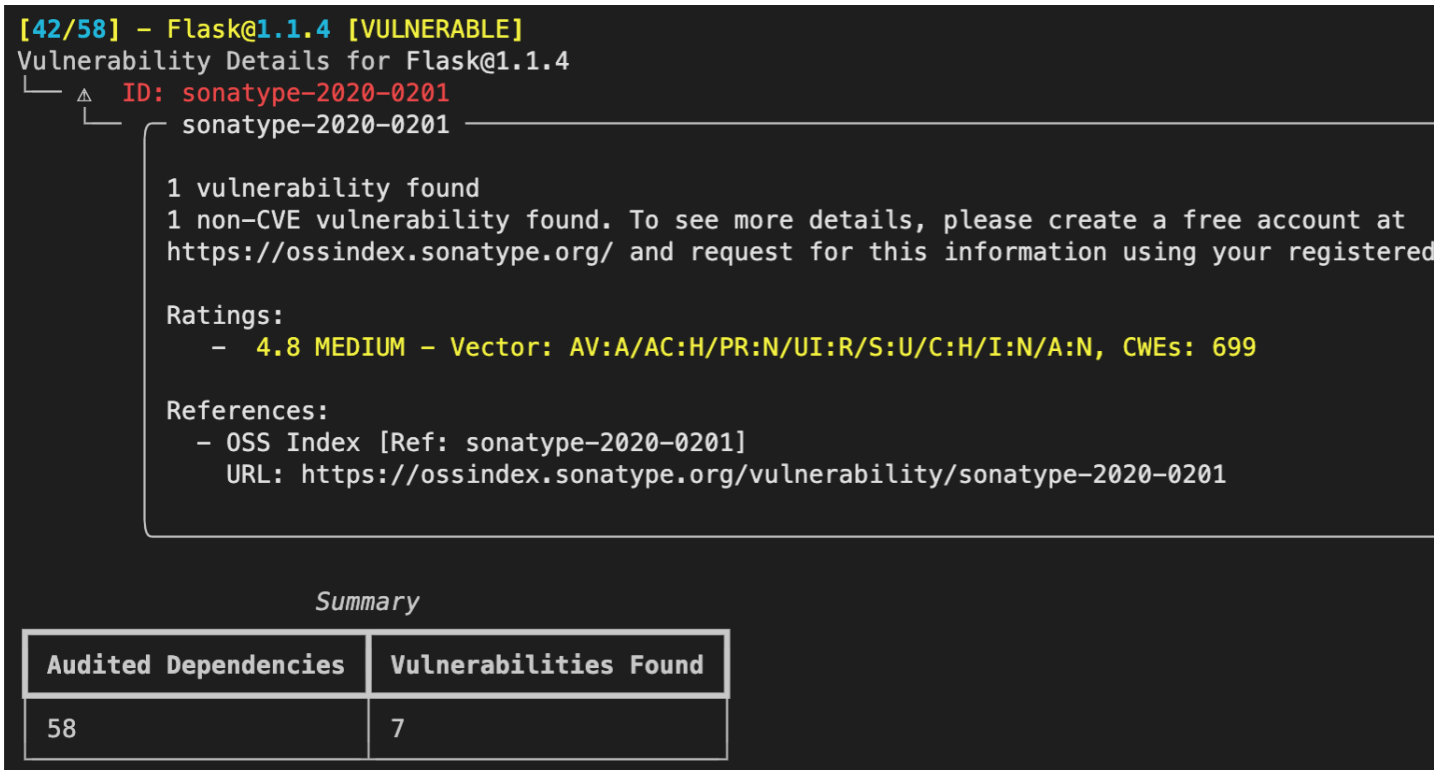


Step 4: Interpreting the Results

The scan collected information on 58 packages that this application uses either directly or indirectly and uncovered several vulnerabilities.



It then provided details on the 7 vulnerabilities with the name of the package that is vulnerable, details about the vulnerability, including URLs to further investigate what the vulnerability is, and how it can be remediated. This is an example of the end of the output:



As you can see, Flask 1.1.4 has a medium vulnerability, and so it should probably be upgraded to a newer version that does not have that vulnerability. You could do that by updating the version in the `requirements.txt` file. More details can be found at the URL in the details. Also, at the end of the list, there is a summary showing that out of 58 packages 7 were vulnerable.

Conclusion

Congratulations! You have completed your first attempt at running a vulnerability scan on a web application. Scanning for vulnerabilities is an essential skill to have for developing secure applications.

In this lab, you learned how to perform a vulnerability analysis, which is a crucial component of the security scanning process. It is an iterative process that involves identifying the vulnerability, analyzing it, assessing its risk, and remedying it before repeating the procedure.

Next Steps

Now it is time to put this new knowledge into practice. Run `jake` on your Python applications to check for security vulnerabilities in your dependencies and perform additional research on fixing the vulnerabilities you have uncovered. Most of the time, you just need to update your `requirements.txt` file with newer versions that are not vulnerable.

Author

John J. Rofrano
David Pasternak

Other Contributor(s)

Sam Propochuk

Michelle R. Sanchez, Instructional Designer at Skill-up Technologies with over 25 years of enterprise-level technical training and support experience.