JW4158-PROG8280-22F-Portfolio2

John White

6714518

PROG8280

Lab <1> - <LDAP and Active Directory>

Description

The purpose of this lab is to teach us how to use Apache Active Directory to retrieve LDAP information.

Preparation

Download Apache Active Directory and Install it. Also install JRE 19.

Observations

Enter the information into Apache Active Directory as shown in the screenshot section

below.

Screenshots

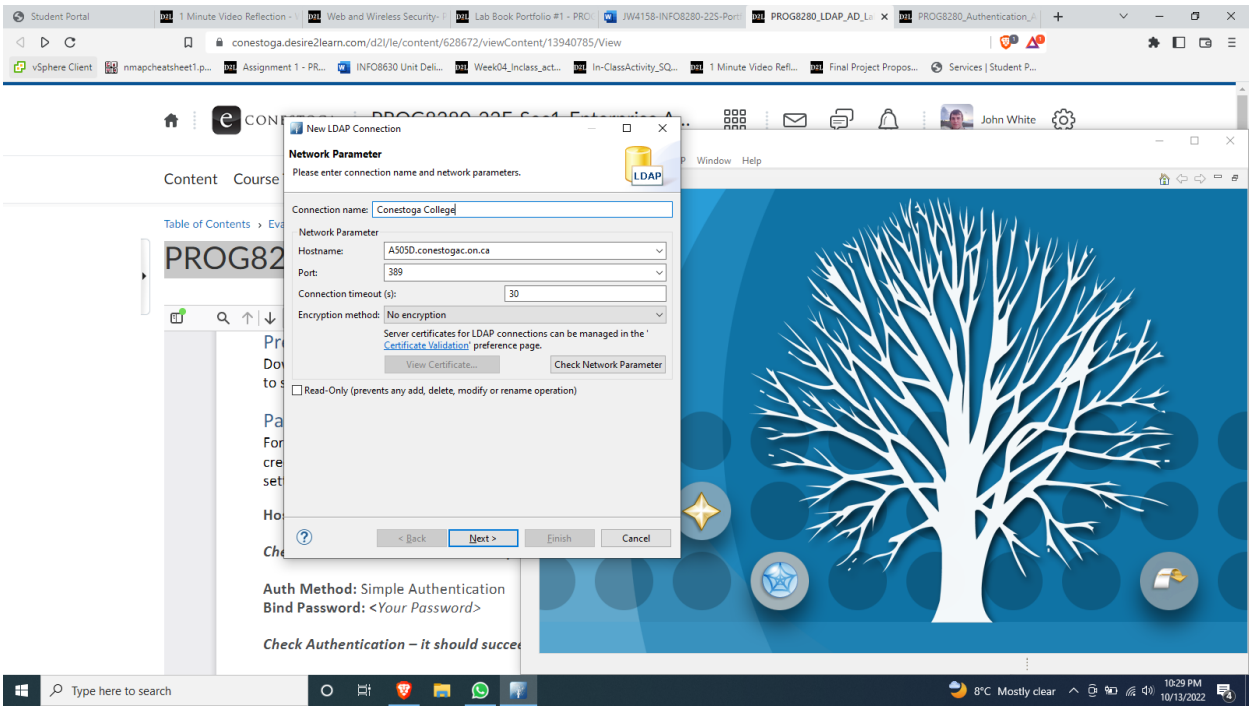

Figure 1 – Entering the hostname

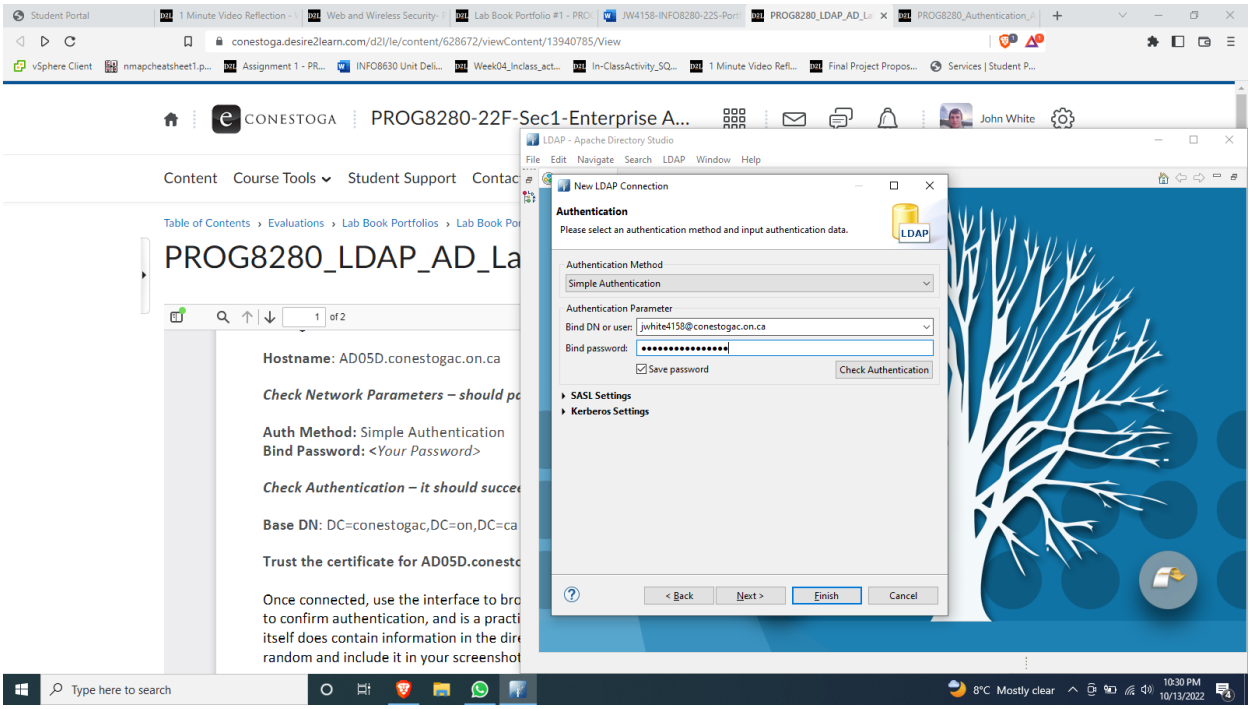

Figure 2 – Entering the authentication information

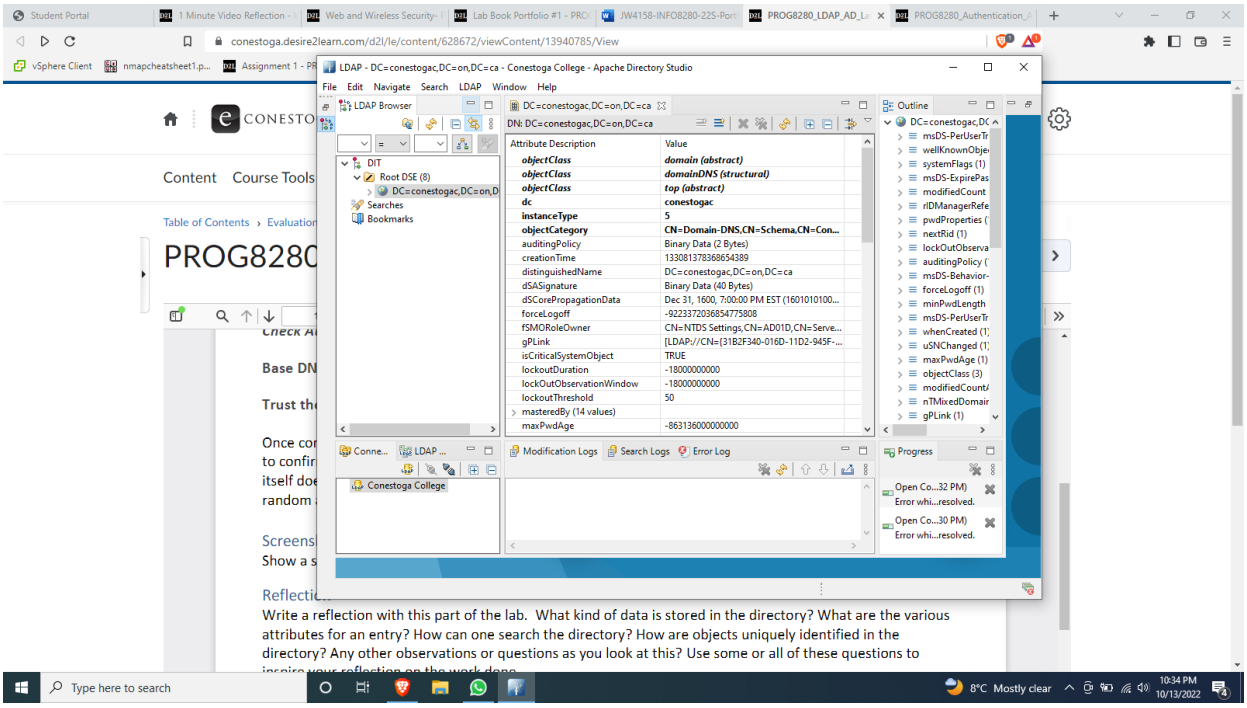

Figure 3 – LDAP information retrieved

Reflections

This lab was straightforward. The only issue I had was that Apache Active Directory kept trying to use the wrong version of Java so I uninstalled that version so it would be forced to use Java 19.

Lab <2> - <Docker Getting Started>

Description

The purpose of this lab is to teach us the basics of using docker which includes running a .js file, building images, and running them.

Preparation

To prepare for this lab, I installed Node.js and Docker on my Kali VM using the following commands:

sudo apt install nodejs

sudo apt install -y docker.io
sudo systemctl enable docker --now

Observations

All Docker and Node commands needed for this lab are shown in **Figure 3.1** and **Figure 3.2**.

The app.js file contains the following code:

console.log("Hello Docker!!! (revised)");

The dockerfile contains the following code:

FROM node:alpine
COPY . ./app
CMD node app/app.js

Reflections

The components of this application are the .js file and the dockerfile, both in the hello-docker folder. To containerize this application, we simply need to build the image using the 'build' command in Docker. Because this application is so simple, there is not a lot of benefit to containerizing it. However, many applications can become extremely complex with a lot of different dependencies. The more dependencies an application has, the more benefit there is to containerization because it eliminates many of the problems that come with depending on libraries to run.

Screenshots

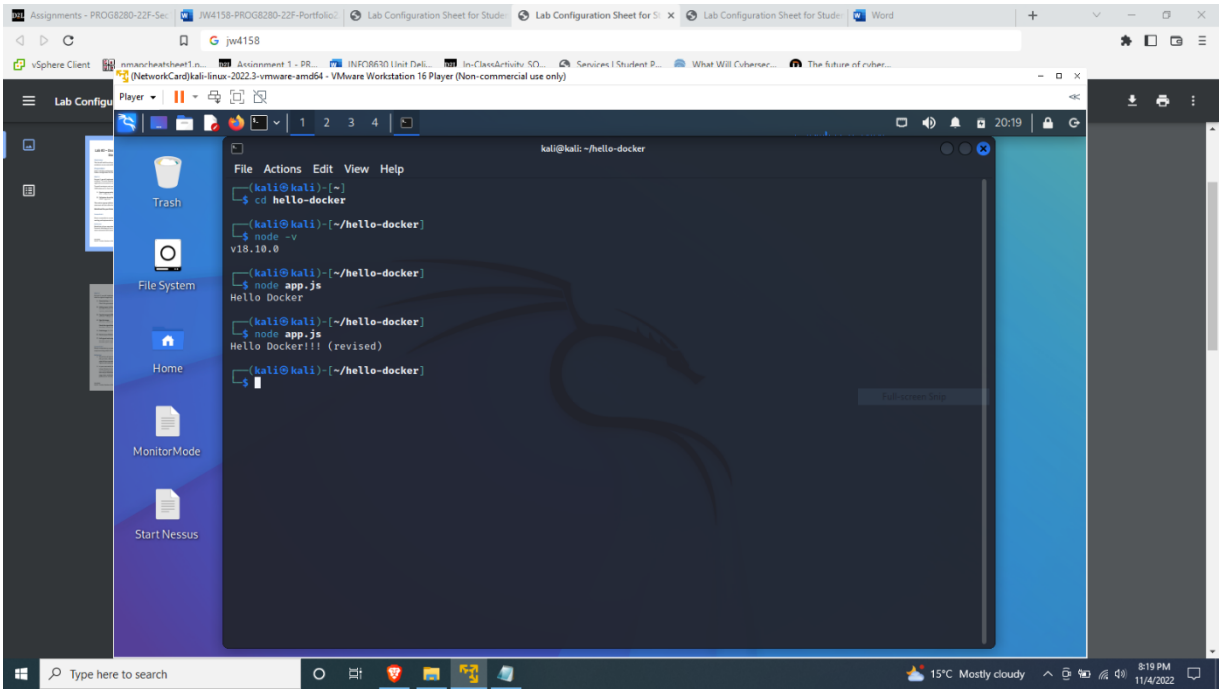

Figure 3.1 - Running a .js file, updating the file, then running it again with different results.

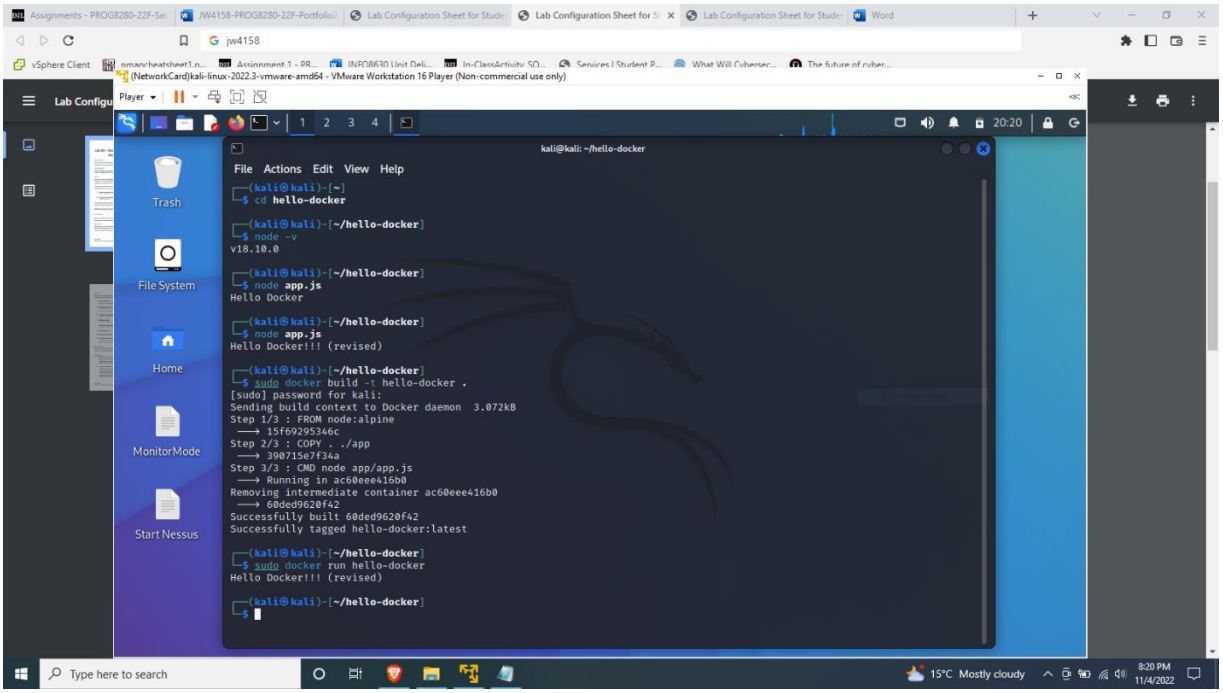

Figure 3.2 - Building the docker image and running it.

Lab <3> - <Docker Container as Non-Root User and Docker Content Security (DCT)>

Description

The purpose of this lab is to teach us how to run a docker image as a user instead of always running docker images as root.

Preparation

No preparation for this lab was necessary beyond the preparation for lab 3.

Observations

All Docker commands needed for this lab are shown in Figure 4.1 and Figure 4.2.

The dockerfile was modified to contain the following code:

```
FROM node:alpine
#Create a group and user
RUN addgroup -S appgroup && adduser -S appuser -G appgroup
# Tell docker that all future commands should run as the appuser user
USER appuser
COPY . ./app
CMD node app/app.js
```

Reflections

I can't think of any non-security advantages to running as a user as opposed to running as root. Running as root is more convenient in most cases but it has the downside of allowing the program to do ANYTHING. I guess there could be a bug in the program that causes damage rather than specifically being related to security so that would be my answer. For docker to run certain commands with elevated privledges, I think the privledges of the user or the group would need to be elevated.

Screenshots

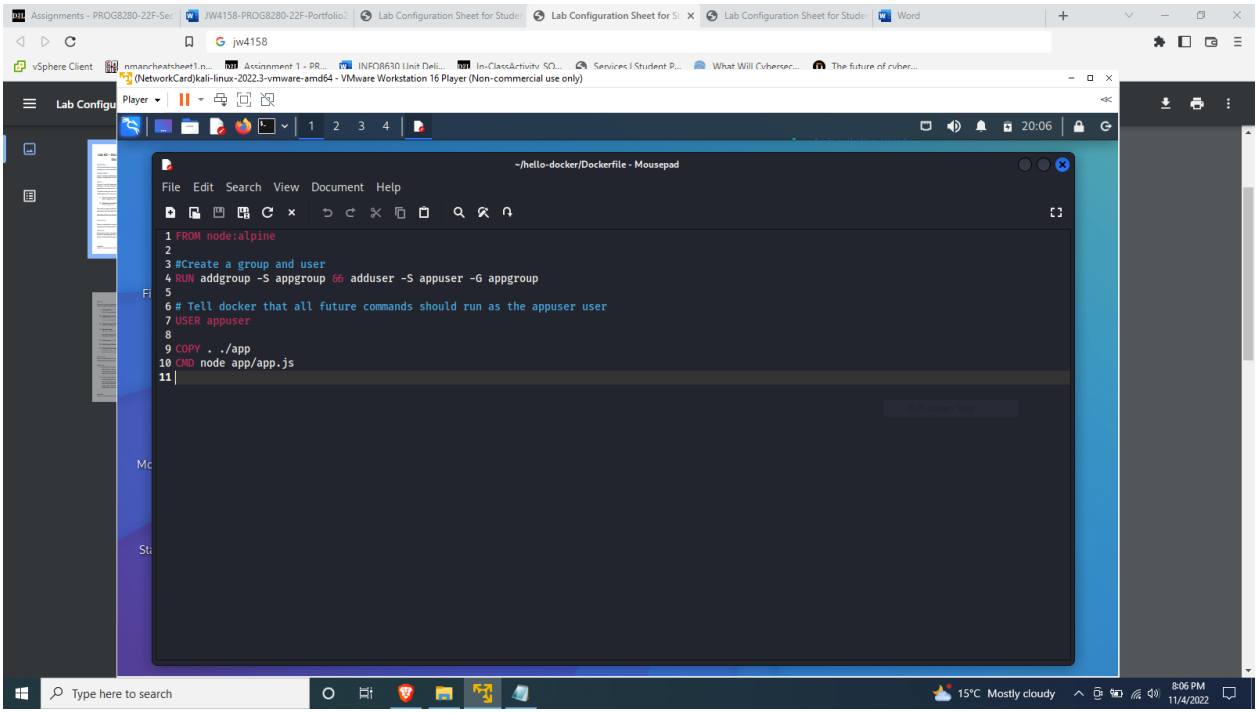

Figure 4.1 - Changing the docker file to run as a user rather than as root.

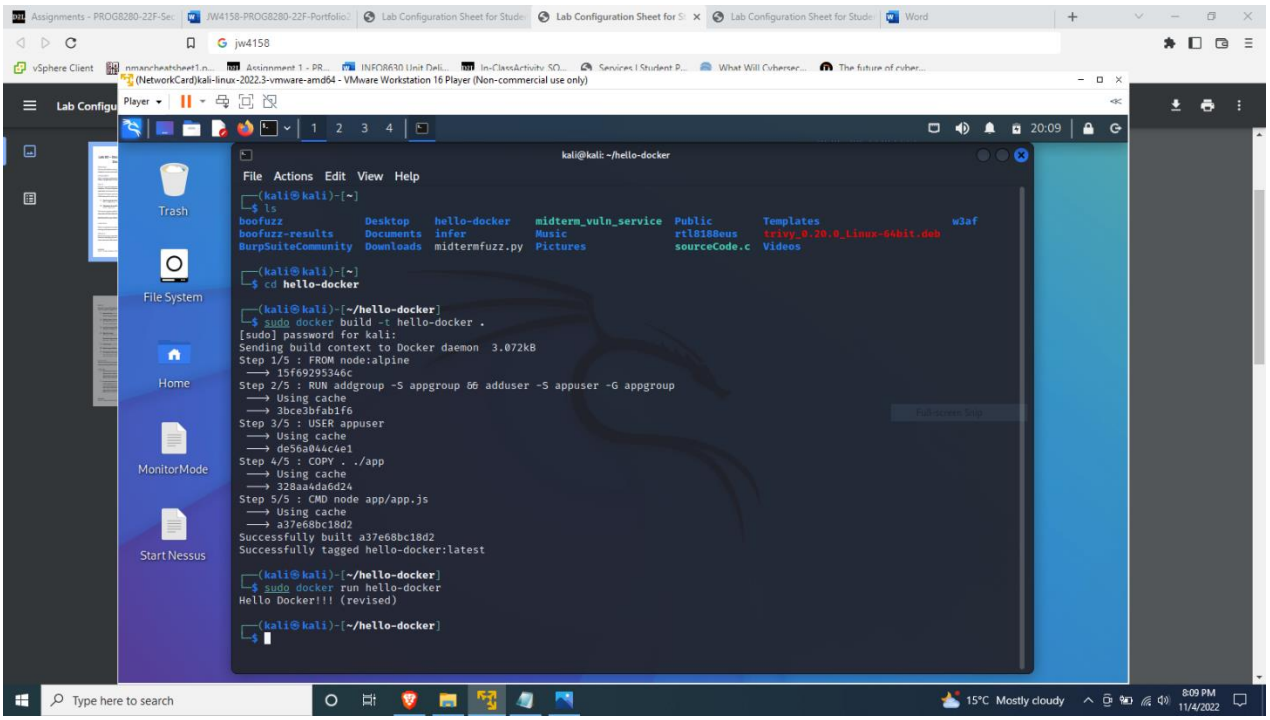

Figure 4.2 - Rebuilding the image and running it after the changes in Figure 4.1.

Lab <4> - <Docker Image Security: Static Analysis Tool>

Description

The purpose of this lab is to teach us how to scan a docker image using Trivy to check how safe it is from outside attack and also to confirm that the image itself does not contain any malicious code.

Preparation

To prepare for this lab, I followed the preparation steps in lab 3 and also installed Trivy on my Kali VM using the following commands:

wget https://github.com/aquasecurity/trivy/releases/download/v0.20.0/trivy_0.20.0_Linux-64bit.deb
sudo dpkg -i trivy_0.20.0_Linux-64bit.deb

Observations

During this lab, I changed the first line of the dockerfile to use this:

FROM node:lts-alpine3.9

This changed the version of Alpine that was being used from the most recent version to an older version that has some vulnerablities that could be found by Trivy. I then rebuilt the image then scanned it with Trivy using the command shown in **Figure 5.2**.

Reflections

1. This lab was very easy, I had no problems with it. I'd like to research more into the vulnerabilities that were found as I do not understand the security implications of a lot of them.
2. I think the best places to implement image scanning would be at the beginning and at the end of the development process. If a new image needs to be downloaded before development can begin, it should be scanned right away to make sure that it is safe and so that any vulnerabilities can be taken into account. At the end of the development process, if an image has been modified or newly developed, it should also be scanned before redeployment so that any vulnerabilities that have been introduced can be caught and dealt with in an appropriate manner.

Screenshots

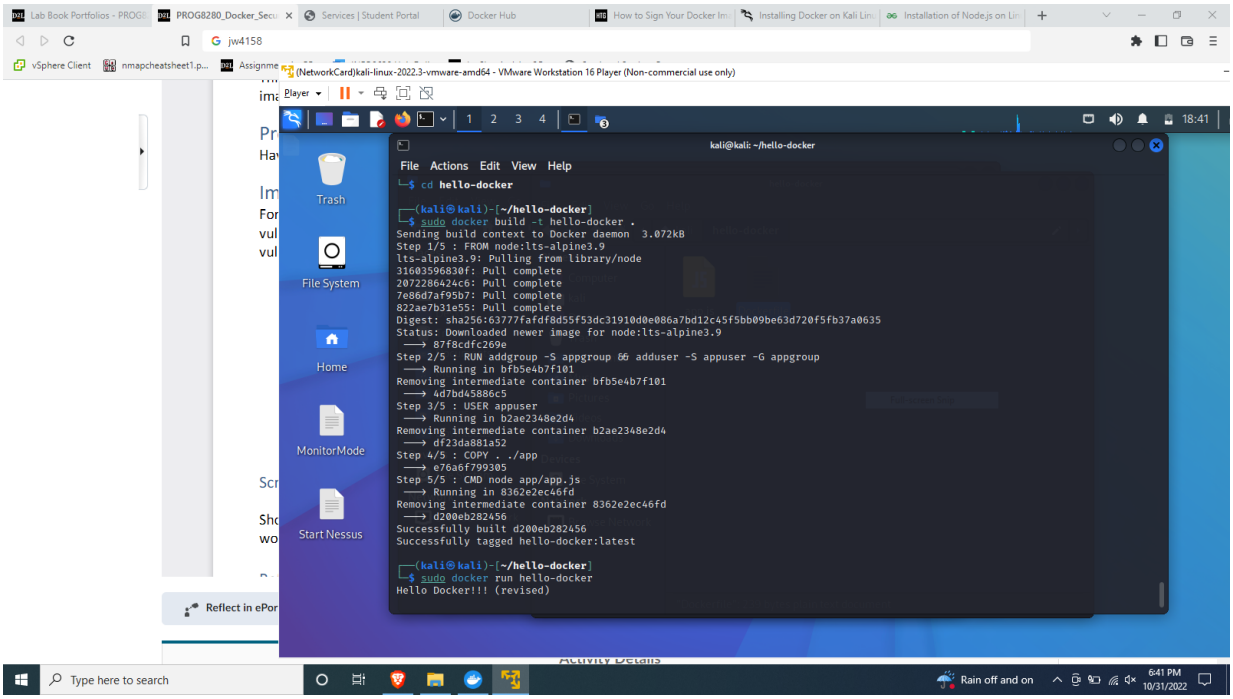

Figure 5.1 - Rebuilding the docker image after modifying it to use lts-alpine3.9 and then running it.

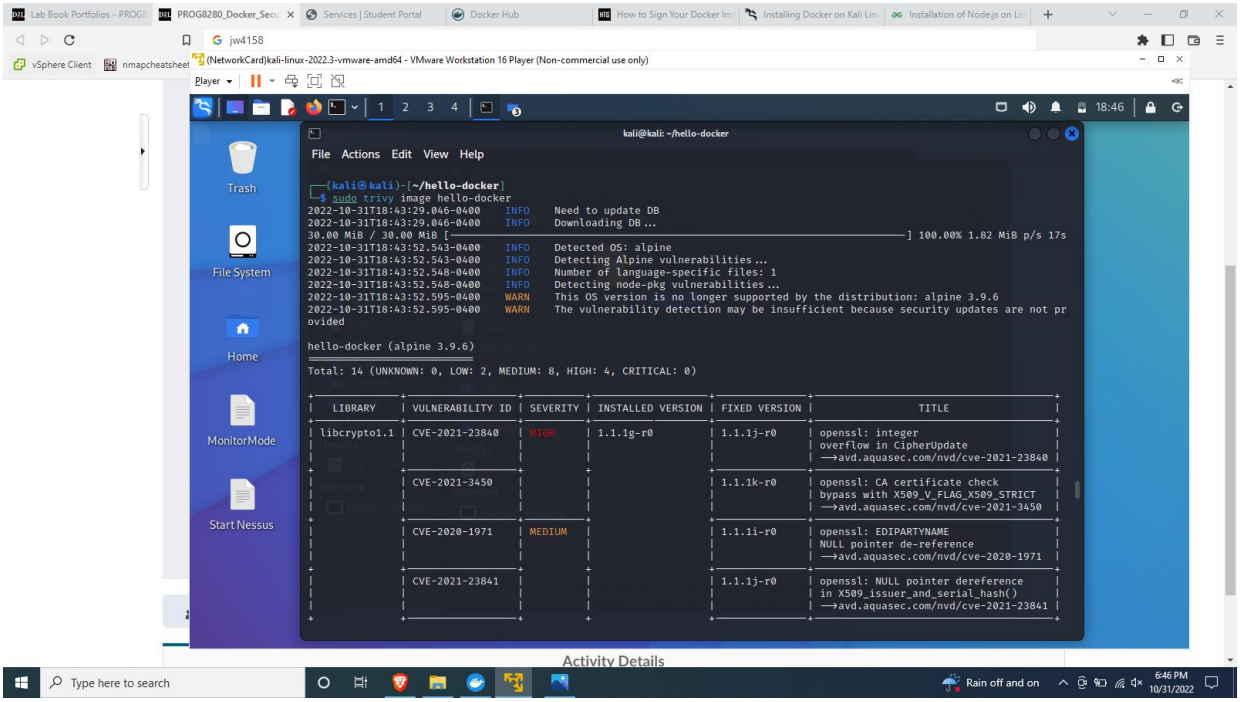

Figure 5.2 - Scanning the rebuilt image using Trivy.

References

1. eConestoga, 2022 (PROG8280_LDAP_AD_Lab_1, retrieved from https://conestoga.desire2learn.com/d2l/le/content/628672/viewContent/13940785/View on October 13, 2022)
2. eConestoga, 2022 (PROG8280_Authentication_Authorization_Lab_2, retrieved from https://conestoga.desire2learn.com/d2l/le/content/628672/viewContent/13940784/View on October 13, 2022)