Lab #2 – Docker Container as Non-Root User and Docker Content Security (DCT)

Overview

This lab will build from the previous Docker lab and follow best practices in Docker Security by running containers as non-root and by implementing Docker Content Trust (DCT),

Preparation

Have a working containerized application with a built Docker image that runs as a container successfully. The Docker configuration file (Dockerfile) is not configured to run with a non-root user.

Part 1

For part 1, we will implement the *principle of lease privilege* by not running applications as root inside the container. The most important advantage to running your containers as non-root is to ensure that your application environment is secure.

To avoid running as root, your Dockerfiles should always create a non-privileged user and switch to it with a USER statement or from an entrypoint script. For example:

1. **Create a group and user:**
   ```
   RUN addgroup -S appgroup && adduser -S appuser -G appgroup
   ```

2. **Tell docker that all future commands should run as the appuser user**
   ```
   USER appuser
   ```

This creates a group called user_grp and a new user called user who belongs to that group. The USER statement will take effect for all following instructions.

Rebuild and Run your Docker image – the process should complete successfully.

Screenshots

Show a screenshot or screenshots that are identifiable with your account name/environment showing this is working and implemented as specified.

Reflection

Record any of your own observations, solutions, or comments about the work you did. While security is the foremost advantage of non-root containers, are there any others? How would you allow the container to run some commands with elevated privileges? Explain and make clear your configuration choices.

Part 2

For part 2, you will implement Docker Content Trust (DCT) by signing your image and also restricting your client to signed images only.

1. **Generate key:** `docker trust key generate <key label>`
   Check the generated key by observing the contents of the <key label>.pub file

2. **Adding signer to the repository**
   ```
   docker trust signer add --key <key label>.pub <key label>
   <your_username>/hello-docker
   ```

3. **Tag the image to differentiate**
   ```
   docker image tag IMAGEID <your_username>/hello-docker:signed
   ```

4. **Sign the image**
   ```
   docker trust sign <your_username>/hello-docker:signed
   ```

   Check the signed image by inspecting the Trust
   ```
   docker trust inspect --pretty <your_username>/hello-docker:signed
   ```

5. **Push image:** `docker push <your_username>/hello-docker:signed`

6. **Restrict your Docker client to use signed images only:** `export DOCKER_CONTENT_TRUST=1`

7. **Pull signed and unsigned images**
   ```
   docker pull goludocker/hello-docker
   docker pull goludocker/hello-docker:signed
   ```

Screenshots

Show a screenshot or screenshots that are identifiable with your account name/environment showing your script executing and/or the result.

Reflection

1. Record any of your own observations, solutions, or comments about the work you did. What problems did you have, what was not clear, what did you take away that you value? If everything went great, what did you see while you were doing it that you are looking to investigate more – or did investigate? Explain and make clear your configuration choices. This is mandatory.

2. In your own words, with any websites you gain information from listed as references, and without just using a thesaurus to change words that are copy and pasted from such a website, tell us about how one could implement Docker Content Security within an organization for a private register? (note: the best way to avoid academic issues with writing this sort of thing is ready on the topic, put that web page away, and write it from your knowledge; if you have difficulty, just read it, talk about it with your professor or a friend, and then write again in your own understanding).