Docker Hands-On Lab 10 — Securing Docker Images and Containers

Trainer: Sandeep Kumar Sharma

Duration: 90–120 minutes

Goal: In this lab, you'll learn to identify, mitigate, and prevent security risks in Docker images and containers. You'll build and analyze a simple Apache image, then apply best practices for image hardening, vulnerability scanning, and secure container runtime configurations.

---

1) Introduction — Why Security Matters in Docker

Containers provide speed, portability, and consistency — but they also bring unique **security challenges**. Since Docker images often use third-party base images, unpatched vulnerabilities or misconfigurations can expose your system.

Common Security Risks:

- Using outdated base images
- Running as root inside containers
- Including unnecessary packages
- Weak file permissions or exposed credentials
- Not scanning for vulnerabilities

In this lab, we'll:

- Build a sample Apache image (from Ubuntu)
- Identify security flaws
- Apply secure configurations
- Scan for vulnerabilities using Trivy
- Implement runtime security best practices

---

2) Sample Dockerfile (Insecure Version)

We'll start with an intentionally insecure Dockerfile:

```
FROM ubuntu:18.04
MAINTAINER sandeep
RUN apt-get update -y
```

```
RUN apt-get install apache2 -y
RUN apt-get install apache2-utils -y
RUN echo "This is Web application Running inside Container" > /var/www/html/
index.html
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Security Problems in This Dockerfile

1. Using an outdated base image (`ubuntu:18.04`).
2. Running everything as the **root user** (default behavior).
3. Using multiple `RUN` layers — increases attack surface.
4. No verification of packages or updates.
5. No non-root user defined for runtime.

---

3) Building and Testing the Insecure Image

Step 1: Build the Image

```
mkdir ~/docker-security-lab
cd ~/docker-security-lab
nano Dockerfile  # Paste the above content

docker build -t sandeep/insecure-apache:v1 .
```

Step 2: Run the Container

```
docker run -d --name sandeep-insecure -p 8080:80 sandeep/insecure-apache:v1
```

Visit `http://localhost:8080` to verify that it works.

Step 3: Check Running User

```
docker exec -it sandeep-insecure whoami
```

Output: `root` → This is a security risk.

---

4) Scan the Image for Vulnerabilities

We'll use **Trivy** — an open-source vulnerability scanner.

Install Trivy (if not installed)

```
sudo apt install -y wget
wget https://github.com/aquasecurity/trivy/releases/latest/download/
trivy_0.50.1_Linux-64bit.deb
sudo dpkg -i trivy_0.50.1_Linux-64bit.deb
```

Scan the Image

```
trivy image sandeep/insecure-apache:v1
```

You'll see a report with **Critical**, **High**, and **Medium** vulnerabilities.

---

5) Secure the Dockerfile (Hardened Version)

Let's fix the issues identified above.

```
# Use a smaller, updated base image
FROM ubuntu:22.04

# Add metadata
LABEL maintainer="Sandeep Kumar Sharma"

# Avoid caching and minimize layers
RUN apt-get update &&
    apt-get install -y apache2 apache2-utils &&
    echo "Secure Apache Container by Sandeep" > /var/www/html/index.html &&
    rm -rf /var/lib/apt/lists/*

# Create a non-root user
RUN useradd -m webuser && chown -R webuser:webuser /var/www/html

# Switch to non-root user
USER webuser

# Expose port 80
EXPOSE 80

# Run Apache in the foreground
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Security Improvements

✅Updated base image (Ubuntu 22.04) ✅Fewer `RUN` layers (less attack surface) ✅Cleaned up package lists (reduces image size and risk) ✅Created a non-root user (`webuser`) ✅Ensured proper file permissions ✅Added metadata for tracking

---

6) Build and Test the Secure Image

```
docker build -t sandeep/secure-apache:v1 .
docker run -d --name sandeep-secure -p 8081:80 sandeep/secure-apache:v1
```

Check user inside the container:

```
docker exec -it sandeep-secure whoami
```

Output: `webuser` ✅(Not root)

---

7) Rescan the Image

```
trivy image sandeep/secure-apache:v1
```

Compare results — you should see significantly fewer vulnerabilities.

---

8) Apply Runtime Security Best Practices

- **Limit container privileges:**

  ```
  docker run -d --name sandeep-secure-restricted --p 8082:80 --security-opt
  no-new-privileges sandeep/secure-apache:v1
  ```

- **Read-only filesystem:**

  ```
  docker run -d --name sandeep-secure-ro -p 8083:80 --read-only sandeep/
  secure-apache:v1
  ```

- **Drop unnecessary Linux capabilities:**

```
docker run -d --name sandeep-secure-limited --cap-drop ALL -p 8084:80
sandeep/secure-apache:v1
```

- **Use user namespace remapping:** (configure in Docker daemon for extra isolation)

---

9) Practice Tasks

1. Modify the Dockerfile to add SSL/TLS configuration.
2. Enable Apache basic authentication and test user access.
3. Create a cron job to rescan your image weekly using Trivy.
4. Experiment with running containers using `--read-only` and observe behavior.

---

10) Summary

- Image security starts with using trusted base images.
- Avoid running containers as root.
- Keep images minimal and up to date.
- Regularly scan images for vulnerabilities.
- Apply runtime security features like **no-new-privileges**, **cap-drop**, and **read-only filesystems**.

✅**Checkpoint:** You've successfully secured a Docker image and container, from building and scanning to enforcing runtime hardening!