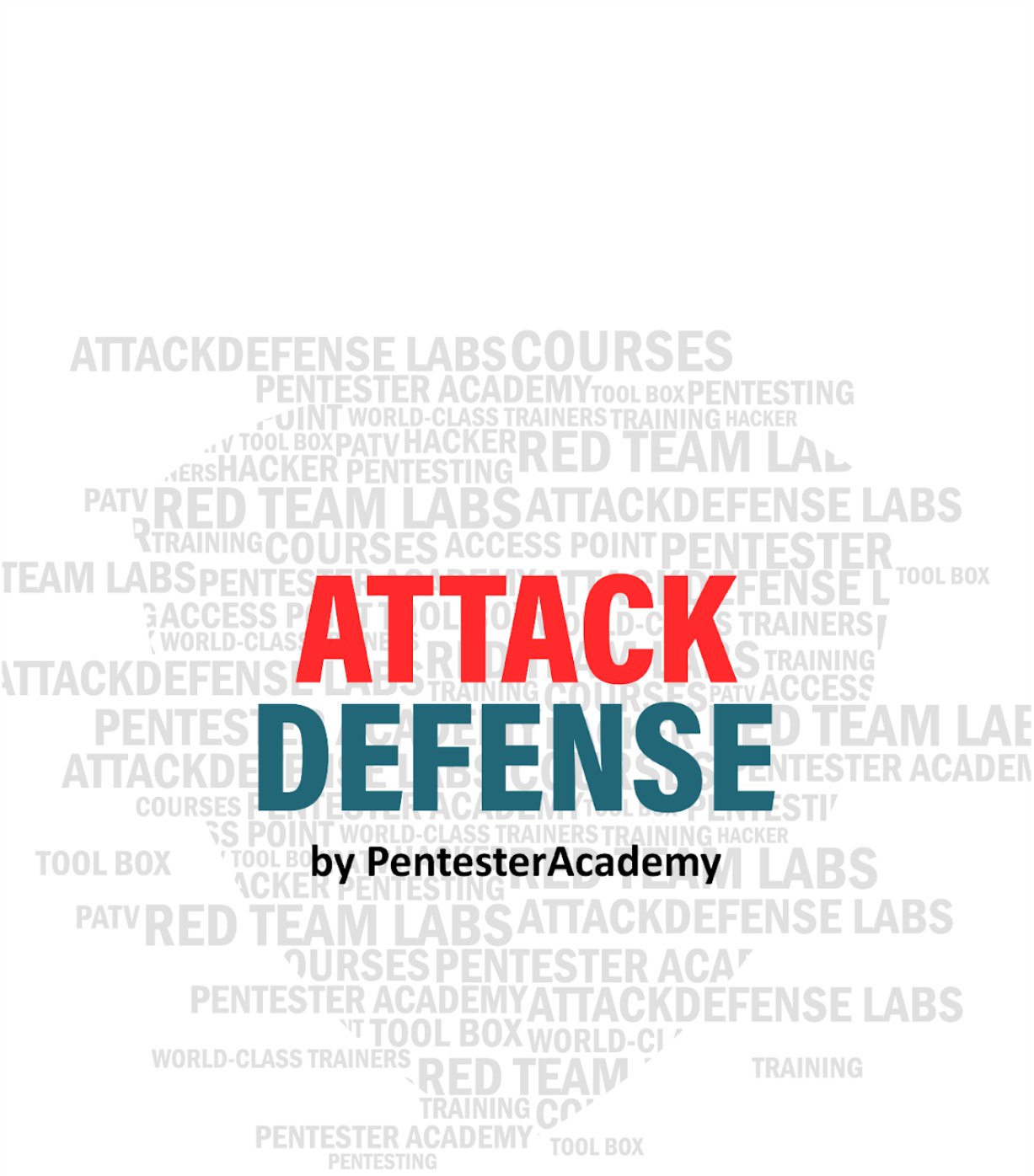
ATTACK
DEFENSE
by PentesterAcademy

| Name | Dockerfile-linter |
|---|---|
| URL | https://attackdefense.com/challengedetails?cid=2161 |
| Type | Docker Security: Dockerfile Linting |

**Important Note:** This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Dockerfile Linting is a process to check and modify the Dockerfile as per the industry's best practices.

Dockerfile-linter is a Dockerfile linter that you can use to quickly check if your Dockerfile follows the best practices for building efficient Docker images.

A Dockerfile is provided in the home directory of the root user (i.e. /root). Dockerfile-linter is installed on the machine.

**Objective:** Analyze the Dockerfile with Dockerfile-linter. Edit Dockerfile to remove all issues detected by Dockerfile-linter!

**Solution:**

**Step 1:** List files present in current directory.

**Command:** ls

```
root@attackdefense:~#
root@attackdefense:~# ls
Dockerfile
root@attackdefense:~#
```

There is a Dockerfile present in the current directory.

**Step 2:** Check help menu for dockerfile-linter.

**Command:** dockerfilelinter -h

```
root@attackdefense:~# dockerfilelinter -h
Usage: dockerfilelinter [options]

Options:
  -V, --version              output the version number
  -f, --file [filepath]      file to lint
  -i, --ignore [rules,...]   comma separated list of ignored rules
  -s, --shellcheck [string]  use rules for specific shell[sh,dash,bash,ksh]
  -j, --json                 return JSON array
  -e, --error                return error code 1 if there is any errors
  -y, --yaml [filepath]      YAML file with ignored rules
  -h, --help                 output usage information
root@attackdefense:~#
```

**Step 3:** Read the contents of Dockerfile.

**Command:** cat -n Dockerfile

```
root@attackdefense:~# cat -n Dockerfile
     1  FROM debian
     2  RUN export node_version="0.10" \
     3  && apt-get update && apt-get -y install nodejs="$node_version"
     4  COPY package.json usr/src/app
     5  RUN cd /usr/src/app \
     6  && npm install node-static
     7
     8  EXPOSE 80000
     9  CMD npm start
```

**Step 4:** Run dockerfile-linter on Dockerfile.

**Command:** dockerfilelinter -f Dockerfile

```
root@attackdefense:~# dockerfilelinter -f Dockerfile
Line 1:
   /EF0004/ Always tag the version of an image explicitly
Line 3:
   /ER0002/ Delete the apt-get lists after installing something
   /ER0010/ Avoid additional packages by specifying --no-install-recommends
Line 5:
   /ER0003/ Use WORKDIR to switch to a directory
Line 6:
   /ER0014/ Pin versions in npm install
Line 8:
   /EE0001/ Valid UNIX ports range from 0 to 65535
Line 9:
   /EJ0002/ CMD and ENTRYPOINT should be written in JSON form
root@attackdefense:~#
```

**Explanation**

**EF0004:** Using the latest tag does not mean it is assigned to a specific image version. Always tag the version of an image explicitly.

**ER0002:** Removing 'apt-get lists' will decrease the size of the image. Note: the operation must be performed in the same RUN instruction as the installation.

**ER0010:** Avoid installing additional packages that you did not explicitly want. This helps in decreasing the size of the image.

**ER0003:** Most commands run with an absolute path and don't require changing the working directory. If you do need to change the directory, you should use the 'WORKDIR' instruction.

**ER0014:** Version pinning forces the build to retrieve a particular version regardless of the cache contents. It also reduces failures due to unanticipated changes in required packages.

**EE0001:** Valid UNIX ports range from 0 to 65535. The provided port was not in this range.

**EJ0002:** If plain text is used for passing arguments, signals from the OS are not correctly passed to the executables, which is in the majority of the cases what you would expect.

Modify the Dockerfile to address these issues.
Please note that line numbers below are respective to unmodified Dockerfile.

**Modifications:**

**Line 1:** Specify tag for the base image used.
**Before Modification:** FROM debian
**After Modification:** FROM debian:9

**Line 3:** Add --no-install-recommends flag to apt-get statement.
**Before Modification:** && apt-get update && apt-get -y install nodejs="$node_version"
**After Modification:** && apt-get update && apt-get -y --no-install-recommends install nodejs="$node_version"

**Line 3:** Remove apt cache after installing packages.
**Before Modification:** && apt-get update && apt-get -y --no-install-recommends install nodejs="$node_version"
**After Modification:** && apt-get update && apt-get -y --no-install-recommends install nodejs="$node_version" \
&& rm -rf /var/lib/apt/lists/*

**Line 4:** Add WORKDIR statement.
**Before Modification:** COPY package.json usr/src/app
**After Modification:** WORKDIR /usr/src/app
COPY package.json .

**Line 5:** Modify RUN statement to adjust for WORKDIR statement.
**Before Modification:** RUN cd /usr/src/app \
&& npm install node-static
**After Modification:** RUN npm install node-static

**Line 6:** Pin the version of npm package.
**Before Modification:** RUN npm install node-static
**After Modification:** RUN npm install node-static@0.7.11

**Line 8:** Specify a valid port.
**Before Modification:** EXPOSE 80000
**After Modification:** EXPOSE 60000

**Line 9:** Modify CMD statement.
**Before Modification:** CMD npm start
**After Modification:** CMD ["npm", "start"]

**Step 5:** Check the file in nano after applying the above mentioned modifications.

**Command:** nano -l Dockerfile

```
 GNU nano 2.9.3                                                  Dockerfile

 1 FROM debian:9
 2 RUN export node_version="0.10" \
 3 && apt-get update && apt-get -y --no-install-recommends install nodejs="$node_version" \
 4 && rm /var/lib/apt/lists/*
 5 WORKDIR /usr/src/app
 6 COPY package.json .
 7 RUN npm install node-static@0.7.11
 8
 9 EXPOSE 60000
10 CMD ["npm", "start"]
11
```

Save the file and exit nano. Press 'Ctrl + X' followed by 'Y' and Enter to exit and save changes.

**Step 6:** Run dockerfile-linter again on the modified Dockerfile.

**Command:** dockerfilelinter -f Dockerfile

```
root@attackdefense:~#
root@attackdefense:~# dockerfilelinter -f Dockerfile
root@attackdefense:~#
```

No output means that we have successfully addressed all the issues mentioned previously by dockerfile-linter.

**References:**

- dockerfile-linter (https://github.com/buddy-works/dockerfile-linter)