ATTACK DEFENSE

by PentesterAcademy

Name	Hadolint
URL	https://attackdefense.com/challengedetails?cid=2164
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.

Haskell Dockerfile Linter aka Hadolint is a Haskell-based linter to help developers stick to best practices while creating Dockerfiles.

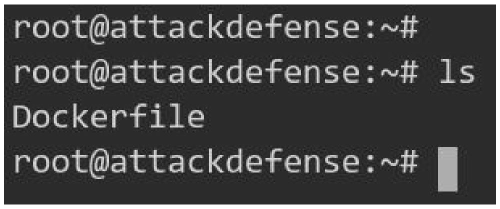
A Dockerfile is provided in the home directory of the root user (i.e. /root). Hadolint is installed on the machine.

Objective: Analyze the Dockerfile with Hadolint. Edit Dockerfile to remove all issues detected by Hadoint!

Solution:

Step 1: List files present in current directory.

Command: ls

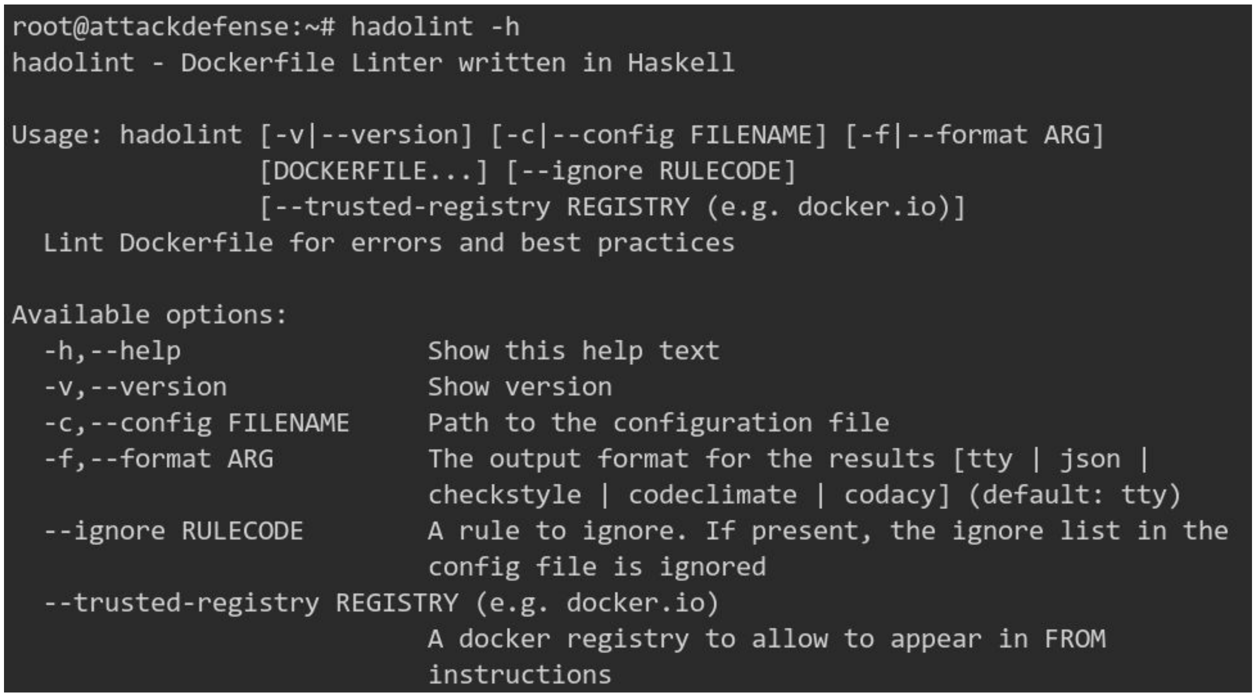


```
root@attackdefense:~#
root@attackdefense:~# ls
Dockerfile
root@attackdefense:~#
```

There is a Dockerfile present in the current directory.

Step 2: Check help menu for hadolint.

Command: hadolint -h



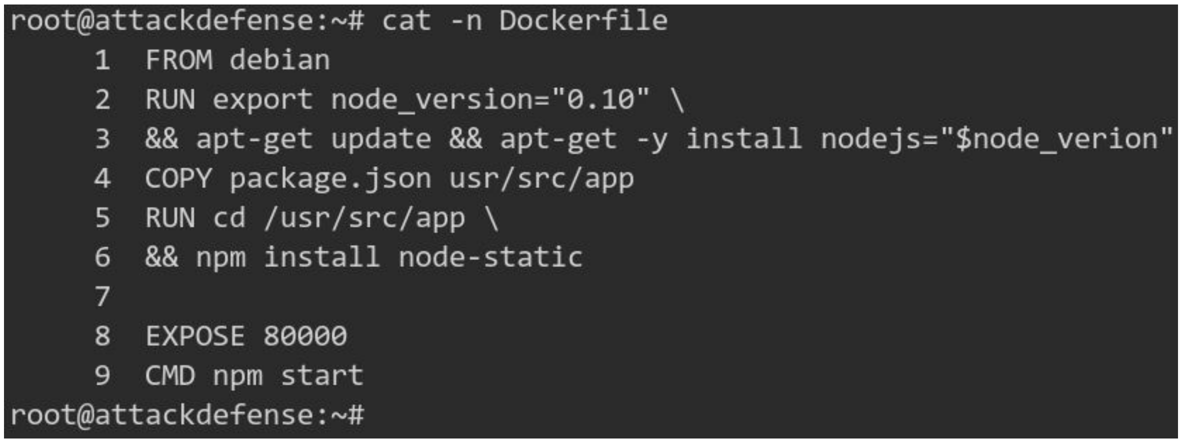
```
root@attackdefense:~# hadolint -h
hadolint - Dockerfile Linter written in Haskell

Usage: hadolint [-v|--version] [-c|--config FILENAME] [-f|--format ARG]
                [DOCKERFILE...] [--ignore RULECODE]
                [--trusted-registry REGISTRY (e.g. docker.io)]
  Lint Dockerfile for errors and best practices

Available options:
  -h,--help                Show this help text
  -v,--version             Show version
  -c,--config FILENAME     Path to the configuration file
  -f,--format ARG          The output format for the results [tty | json |
                           checkstyle | codeclimate | codacy] (default: tty)
  --ignore RULECODE        A rule to ignore. If present, the ignore list in the
                           config file is ignored
  --trusted-registry REGISTRY (e.g. docker.io)
                           A docker registry to allow to appear in FROM
                           instructions
```

Step 3: Read the contents of Dockerfile.

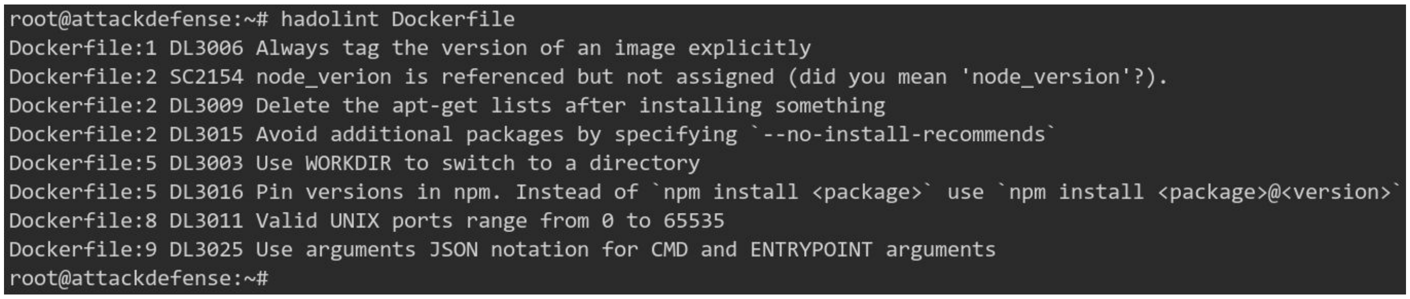
Command: cat -n Dockerfile



```
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_verion"
     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
root@attackdefense:~#
```

Step 4: Run hadolint on Dockerfile.

Command: hadolint Dockerfile



```
root@attackdefense:~# hadolint Dockerfile
Dockerfile:1 DL3006 Always tag the version of an image explicitly
Dockerfile:2 SC2154 node_verion is referenced but not assigned (did you mean 'node_version'?).
Dockerfile:2 DL3009 Delete the apt-get lists after installing something
Dockerfile:2 DL3015 Avoid additional packages by specifying `--no-install-recommends`
Dockerfile:5 DL3003 Use WORKDIR to switch to a directory
Dockerfile:5 DL3016 Pin versions in npm. Instead of `npm install <package>` use `npm install <package>@<version>`
Dockerfile:8 DL3011 Valid UNIX ports range from 0 to 65535
Dockerfile:9 DL3025 Use arguments JSON notation for CMD and ENTRYPOINT arguments
root@attackdefense:~#
```

Explanation

DL3006: Never rely on the 'latest' tag to get a specific version. Hadolint recommends that you always use a specific tagged image, for example ubuntu:12.04.

SC2154: There is a typo in the line 3 of Dockerfile. Spelling of node_version is incorrect.

DL3009: Cleaning up the apt cache and removing /var/lib/apt/lists helps keep the image size down. Since the RUN statement starts with apt-get update, the package cache will always be refreshed prior to apt-get install.

DL3015: Avoid installing additional packages that are not required. This helps in keeping the image size down.

DL3003: Only use `cd` in a subshell. Most commands can work with absolute paths and it is in most cases not necessary to change directories. Docker provides the `WORKDIR` instruction to change the current working directory.

DL3016: Version pinning forces the build to retrieve a particular version regardless of what's in the cache. This technique can also reduce failures due to unanticipated changes in required packages.

DL3011: A port outside the 0-65535 is used. Valid UNIX ports range from 0 to 65535.

DL3025: The arguments can't be passed like this to the program. These should almost always be used in the form of CMD ["executable", "param1", "param2"…]

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: Correct the spelling of 'version'.
Before Modification: && apt-get update && apt-get -y install nodejs="$node_verion"
After Modification: && apt-get update && apt-get -y install nodejs="$node_version"

Line 3: Add command for cleaning apt cache. Add one more command at the end of RUN statement present in Line 2 using '\' and '&&'.
Before Modification: && apt-get update && apt-get -y install nodejs="$node_version"
After Modification: && apt-get update && apt-get -y install nodejs="$node_version" \
&& rm /var/lib/apt/lists/*

Line 3: Add --no-install-recommends flag to apt-get command.
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 4: Add WORKDIR before COPY command and modify COPY Command.
Before Modification: COPY package.json usr/src/app
After Modification: WORKDIR /usr/src/app
COPY package.json .

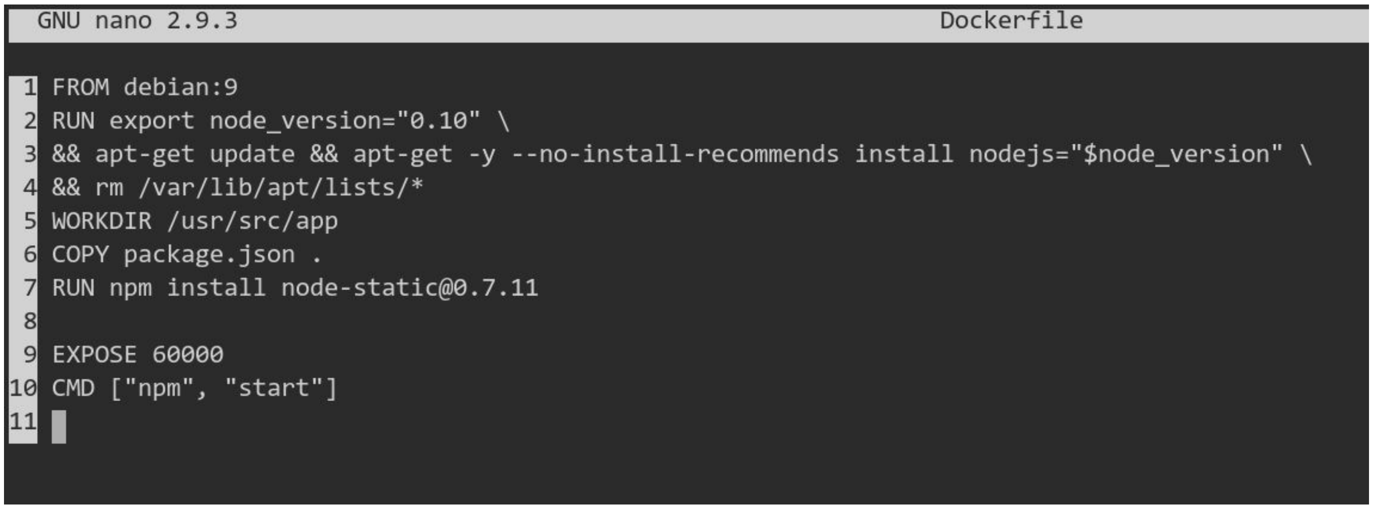
Line 5: Modify RUN command at line 5 to accommodate 5th modification and pin the version of npm package.
Before Modification: RUN cd /usr/src/app \
&& npm install node-static
After Modification: RUN npm install node-static@0.7.11

Line 8: Change port number.
Before Modification: EXPOSE 80000
After Modification: EXPOSE 60000

Line 9: Change CMD command.
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



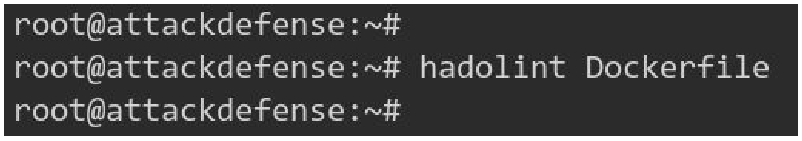
```
  GNU nano 2.9.3                                              Dockerfile

 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 hadolint again on the modified Dockerfile.

Command: hadolint Dockerfile



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

No output means that we have successfully addressed all the issues mentioned by hadolint.

References:

- Hadolint (https://github.com/hadolint/hadolint)