**Exercise1.**

We have a golang container that serves a webpage. It is built using the attached dockerfile. Security audit has revealed that the container contains source code for the application which is insecure. How will you modify the dockerfile to ensure that the final container contains only the binaries? Suggest 2 ways

**Solution1 :** Since this is a go application, binaries need not be exported when we create the container. We can remove api.go in the Dockerfile and after we build

FROM ubuntu:18.04

RUN apt-get update -y \

&& apt-get install -y git gcc make golang-1.10

ENV GOROOT /usr/lib/go-1.10

ENV PATH $GOROOT/bin:$PATH

ENV GOPATH /root/go

ENV APIPATH /root/go/src/api

WORKDIR $APIPATH

COPY . .

RUN \

go get -d -v \

&& go install -v \

&& go build

RUN rm api.go

EXPOSE 3000

CMD ["./api"]

**Solution2 :** We can see that the resulting image still has a layer of binaries and the image size is huge as well. The best solution would be to create multi-stage to remove the binaries as well reduce the image size

FROM golang:latest AS multistage

WORKDIR /go/src/api

COPY . .

RUN go env -w GO111MODULE=off

RUN go get -d -v \

&& go install -v \

&& go build

##

FROM alpine:3.8

COPY --from=multistage /go/bin/api /go/bin/

EXPOSE 3000

CMD ["/go/bin/api"]

**Exercise2.**

Create a container using the attached Dockerfile. Build the image and note down the image size. Study the commands in the file and identify what can be done to reduce the image size. What is the image size of the optimised container?

**Solution:** the solution is similar to Solution 2 where we can create multi-stage images to reduce the size of the image. Please try other Dockerfiles available in options folder for reference.

**Exercise3.**

You have a containerized application that renames the file present in the user's home directory. Modify the below docker file to allow the code to work using the ~/demo-files directory as the source.Hint: Create a volume using docker create. Run a container and mount the volume.

**Exercise4.**

In the previous case, how will you modify the same container such that it will copy the source files to ~/demo-result. You should ensure that your container has only read permissions on ~/demo-files as it is a security requirement. Hint: Mount the volume as read-only.

**Exercise5.**

This container runs as the root user. Reviewer find that your set-up is not following the principle of least privileges. How will you modify the container to run it with a lower privileged user named "customer”? Hint: Add a new user to the container using adduser in Dockerfile and set the user as default user

**Solution:**

FROM ubuntu:18.04

WORKDIR /root

COPY . .

RUN adduser --shell /bin/bash --disabled-password --gecos "" --uid 1001 customer

USER 'customer'

RUN whoami

**To verify the layers:**

docker build --progress=plain --no-cache -t <image\_name> .

**Exercise6.**

You find that a new image is added to your container registry. You do not have the Dockerfile used to build the image. How will you verify the commands used to build this image without executing it? How will you find the start-up command for this container?

**Command to pull the Image:**

docker pull dollopofsunshine/handsonexerciseamd64 --platform=linux/amd64

Solution:

docker image inspect *dollopofsunshine/handsonexerciseamd64:latest* | grep -B5 -A5 Entrypoint