Nicolas Gomez
Professor: Kevin Gallagher
Application Security
November 17, 2019

Unit 4

The next section of the project will allow us to securely deploy our application with the help of Docker. Docker is a platform that allows "OS-Level virtualization to deliver software in packages called containers"(Wikipedia). In order to use this tool, I followed the instructions on https://docs.docker.com/install to install the program on my machine.

Docker and Secrets

With Docker installed, I proceeded to create a Dockerfile that will contain the necessary instructions to assemble an image. I wrote the following Dockerfile https://github.com/ng1968/Application-Security/blob/master/Dockerfile which I uploaded to my GitHub repository https://github.com/ng1968/Application-Security. In this file I specified the need for Python 3.7, copying all of the files to a folder named app. Then I ran "make" to build the spell_check binary and copied it to the webroot folder. After that I installed all of the necessary requirements and started the service on the expose port 8080. When writing this file I followed the same instructions as another GitHub user Lvthillo https://github.com/lvthillo/python-flask-docker and modified it to do what I needed to do. I ran the command below to create an image for my application:

docker build -t application-security .

With the image created, I proceeded to test it by running the command below and opening a web browser in my laptop and going to 0.0.0.0:8080 to make sure everything worked properly.

docker run --name test -d -p 8080:8080 application-security

In order to make it easier for any user to deploy this service by running the "docker-compose up -d" command, I needed to run a docker-compose.yml file. I read about this file in https://docs.docker.com/compose/reference/up/ and what components need to be in place for the command to work. I wrote the docker-compose.yml file and created a service named web that uses the image created before by the build command and uses the 8080 ports. Before

running the command I ran “docker stop test” to stop the container I started before and then ran the “docker-compose up -d” command and went to 0.0.0.0:8080 to ensure everything worked properly. In order to stop the container I ran the “docker-compose stop” command. In order to protect sensitive data like the secret keys for the JWT cookies and databases, I used the Docker Secrets to store items that should be protected. I added a secrets section to the docker-compose.yml file and added two secrets. In the terminal I ran “echo "something random" | docker secret create name_of_secret -” commands to create a secret and changed the way the service accessed these secrets by adding the “open("/run/secrets/name_of_secretts", "r").read().strip()” where ever the secret is needed.

Docker Content Trust

In order to deliver updates to the service after it has been deployed a system needs to be setup to ensure the secure and trusted transfer of data. The Content Trust in Docker can be used to accomplish this task. The Content Trust allows us to push or pull images and verify the integrity of both the publisher and the data received over any channel. To do this a Notary key will need to be setup so that it can sign all of the images and other information sent.

Docker Swarm

In order to be able to deploy our service with multiple replicas I used the docker swarm tool to do so. I followed the instructions on the Docker Deploying to Swarm page and ran “docker swarm init“ to start the service. After that, I ran the command below to add the service to the swarm.

docker stack deploy -c docker-compose.yml demo

With this, I was only able to start one replica of the service which I was able to see with the “docker service ls” command. In order to start multiple replicas I needed to make some changes to the docker-compose.yml file. I added the deploy section under the web section of the file and added the replicas option which I set to 4. In addition to this, I added the resources section to limit the CPU and Memory usage of each replica. I set the CPU to 0.5 or half a core and the memory to 350M with reservations of 0.1 CPU and 100M or Memory.

Resources

- Assignment Github: https://github.com/ng1968/Application-Security/tree/master/your/webroot
- Assignment Guidelines: https://drive.google.com/file/d/1Sv4v-4exLVOmJyetYcprK0udtS_YEFaJ/view?usp=sharing
- Docker Compose: https://www.tutorialspoint.com/docker/docker_compose.htm
- Docker Compose with Flask Apps: https://runnable.com/docker/python/docker-compose-with-flask-apps
- Docker Dockerfile Best Practices: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- Docker Install: https://docs.docker.com/install/
- Docker Dockerfile Reference: https://docs.docker.com/engine/reference/builder/
- Dockerize Simple Flask App: http://containertutorials.com/docker-compose/flask-simple-app.html
- Docker Secrets: https://testdriven.io/blog/running-flask-on-docker-swarm/#docker-secrets
- Installing Docker in Kali Linux (Updated for 2019.2): https://medium.com/@airman604/installing-docker-in-kali-linux-2017-1-fbaa4d1447fe
- GitHub Lvthillo: https://github.com/lvthillo/python-flask-docker
- Set hard CPU limits in a Docker Swarm: https://forums.docker.com/t/set-hard-cpu-limits-in-a-docker-swarm/70198