Docker

Port	2376 if TLS, 2375 otherwise. Neither are open by default
Type	Service
Secure?	Decently

What is Docker

Docker is a new development tool made in 2013 and it is growing rapidly. In the olden days, software engineers had to struggle with making their apps work across different machines and setups due to different computers having different packages installed or even a different OS. Now you can create a Docker image that takes a snapshot of what your application requires to run and the current code. From that any system with docker installed can use that image to run the service with one command no setup.

This has some great advantages to it. The largest for cybersecurity is that it is an isolated environment. What is inside the docker container cannot see what is outside of it and can only access the command it needs to be able to run. So even if an app is compromised, little can be done by the hacker to compromise the actual system. Docker itself is pretty secure and as it is growing bigger, it is developing an ecosystem of tools that can help vuln-scan individual images.

Using Docker

More than likely docker is preinstalled if you're looking at this, if not, look farther down for setup instructions.

See docker images currently on the machine

```
docker images
```

Delete an image

```
docker rmi [IMAGE]
```

Run docker image.

```
docker run -p 8080:80  --name [NAME] -d [IMAGE]
```

`-p`: what port to run the docker image on. In this example, when the app listens to port 80, docker is forwarding that to real port 8080 (you must specify a port here. Docker by default won't let containers listen to ports.)

`--name`: give it an internal name (or else docker gives it a default name)

`-d`: runs it detached in the background

See images currently running

```
docker ps -a
```

The `-a` shows all processes that are supposed to be running, even if they died or were stopped. (But not if they were removed)

`docker ps` is also where you would find the name of your container if you didn't specify it in `docker run`.

With that name, you can start, stop, restart that process, or fully remove it.

```
docker start [NAME] / docker stop [NAME] / docker restart [NAME] / docker rm [NAME]
```

See the logs of a container

```
docker logs [NAME]
```

Something Nice

One of the really cool things about this is that whenever an image is created, you are also creating a backup. If your server goes down during a competition, don't debug it and try figuring out where the vulnerability is, kill that container and start a fresh one. Then go to wherever the code is stored and debug there. If you think you've fixed it, then make a new image off of that and run that one instead. But theoretically, if docker is properly configured to be isolated, then you can just keep playing the kill-create card over and over again with little the red team can do but keep running the same attack just to see it disappear a couple seconds later.

Docker Security

There is a `docker scan` utility but as far I can tell you have to pay for it.

As already said a couple of times, these things are decently secure. But there are some things you should be wary of.

1. Docker has root access. Anyone who is in the docker group also has root access.
2. Images can edit a host's filesystem in any way they want. No restriction. Docker specifies where in the filesystem images are run so make sure that is a place you don't mind getting destroyed. (Normally only a problem if you're downloading random containers off the internet)
3. `docker load` & `docker pull` both grab code off the internet, but Docker has been making it as hard as possible to pull malicious non-images using that.
4. Docker is highly reliant on the Linux kernel, which as always is crazy compromised. But why go for your docker images when you can just kill the system? Look up PAX kernel later.

CVEs

1. **CVE-2019-5736** runc vulnerability allows for remote code execution patched after runc 1.0-rc6 and Docker 18.09.2. The big one you have to look out for.
2. **CVE-2018-11756** Apache OpenWhisk can compromise docker if php runtime isn't properly configured. Make sure to update OpenWhisk
3. **Log4J** Affected Docker, but it's too new to really figure out how.

Use as a Security Tool?

I mean I might be going crazy but the backups and the isolation could really help harden services. But this would require making our own images which is a whole thing. Not difficult to start, but at scale it gets complicated.

Making your own images

Hopefully, you don't have to. That is not a cybersecurity person's job. But here are some quick things to know about making images.

```
docker build -t [NAME] PATH
exmaple: docker build -t apache:initial .
```

An image is made by a Dockerfile which is stored at the root of an app. It gives instructions on everything that is required to be stored in the image.

Hopefully, you don’t have to make a Dockerfile, if you do, search google for ‘[service] Dockerfile’ and be specific.

For instance, I searched ‘apache Dockerfile’ and an official docker website specifying how to make httpd work in docker came up. Just 4 commands!

Alternatively, if there is a docker-composed.yml file, you can run

```
docker-compose up -d
```

That will completely build a new image and start up the containers automatically.

```
docker-compose down
```

Will stop it.

Setting up Docker

For Windows and Mac, there are desktop applications for these things. Search them up. For Linux, it’s a bit more involved.

1. Install

Convenience Script

Docker has released a shell script that will theoretically install docker. In a competition situation where speed is important, this is probably the best path.

However, there are multiple downsides to this way as there is little control over what is going on under the hood. Not suggested for actual production. Also depending on how crazy your red team is, make sure the code downloaded from curl is what is on their GitHub.

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Docker should be installed properly and automatically running as a service on any Debian-based system. Test the setup with the following command.

```
sudo docker run hello-world
```

Setup Docker Repo

I have no clue what's going on. But here's Dockers page.

> Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Uninstall any old version of docker if you think you might have it.

```
sudo apt remove docker docker-engine docker.io containerd runc
```

Preferably be up to date.

```
sudo apt update
```

To install docker will require a gpg key, install required packages for that.

```
sudo apt install ca-certificates curl gnupg lsb-release
```

Add Docker's official GPG key (it's long)

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/s
hare/keyrings/docker-archive-keyring.gpg
```

Download the Repo

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-arch
ive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

```
sudo apt install docker-ce docker-ce-cli containerd.io
```

Check it works

```
sudo docker run hello-world
```

2. Post-Install

If there’s anything specific you need.

Configure where the Docker daemon listens for connections

By default, you can only connect to docker locally but you can change it up so that someone can ssh into a docker from a port. Unlike the default, this is extremely vulnerable and there are steps that have to be taken after this is set up or else privilege escalation is decently easy. Hopefully, we’re not asked to do this, but here’s the link.

https://docs.docker.com/engine/install/linux-postinstall/#configure-where-the-docker-daemon-listens-for-connections

TLS is not on by default. That’s bad. Here’s a link to fix that.

https://docs.docker.com/engine/security/protect-access/

Services Dockerstyle