**HP SaaS - Docker Hardening Guide**

## 1. Integrity and loading the image & container

#### 1.1 Image repository

* Verify that the FROM argument points at a local repository from which to pull the image file (e.g. by using the docker PULL or docker FROM arguments).27
* It is highly recommended to create a local repository from which the image files can be pulled by Daemons on the different hosts.
* Do not use the *--insecure-registry* flag on a private repository. An Insecure private repository is either not having a valid registry certificate or is not using TLS.

Avoid using the following command on the Docker daemon:

*Docker –d --insecure-registry <ip address of the private repository>*

* Setup a local registry mirror to avoid the need of going out to the internet and fetch images from public or private Docker registry, thus allows to manage and securely store images and also reduce the threat vector.

Use the following command on the Docker daemon to configure local registry mirror: docker –registry-mirror=<registry path> -d

#### 1.2 Dockerfile

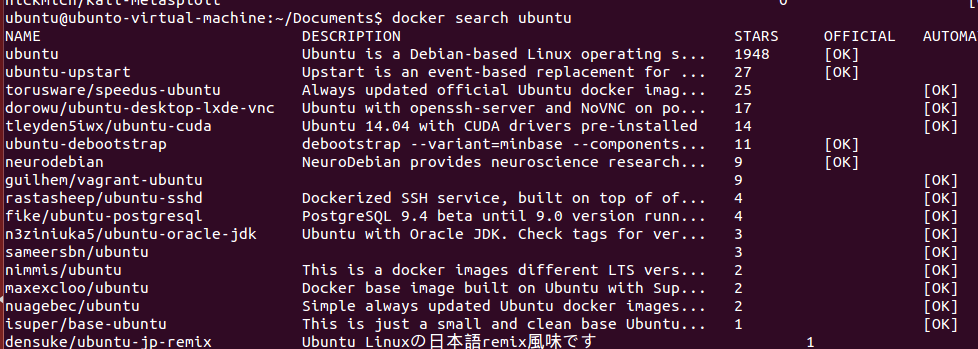
* Make sure to create an unprivileged user using the command:

*RUN useradd –ms /bin/bash <username> USER <username>*

* Ensure that the container image is from a source that can be trusted either from the official repository supported by Docker or from the vendor itself. In addition, make sure the communication is made over a secure channel, since the Docker image signing and verification feature is not ready yet and it is still a work in progress.

As described in the paper, the integrity of the image file can be checked either online by using the docker search command. The official image files are marked “OFFICIAL” with an [OK] as below:

27 <https://blog.codecentric.de/en/2014/02/docker-registry-run-private-docker-image-repository/>



* Verify that other unnecessary services/packages are not installed on the container – this, in order to avoid increasing the attack surface of the container.

The following command will list the installed packages on the container, make sure to review the list and uninstall everything that does not justify the purpose of the container:

*docker exec <container ID> rpm -qa*

* Rebuild images in order to include new security patches using the command:

*docker build –no-cache <dockerfile name>*

* Do not use the aufs storage driver. It is the oldest storage drive and it is known to cause some Kernel crashes28.

Avoid using the following command on the Docker daemon:

*docker –s aufs -d*

## 2. Container

* Run services within the container as non-root whenever possible. Treat root within a container as if it is root outside of the container.
* Verify that other unnecessary services do not run within the context of the container, as mentioned in the Dockerfile installation section.
* Verify that the *Run* command includes the *–u* flag with the user created in the corresponding dokcerfile to ensure that the container is not running under a root user.
* Verify that containers run only a single main process.

Since Docker, by design, watches one single process within the container, running multiple applications within single container breaks the concept of ‘one container, one process’. This also increases the attack surface of the container.

The following command will list the process running within a container, make sure there is only one process running and it is the intended one:

*docker exec <container ID> ps -e*

28 <https://github.com/docker/docker/issues/6047>

* Restrict Linux Kernel Capabilities within containers using the following command:

*docker run --cap-drop <capability> <container image name>*

* Do not use privileged containers with *--privileged* flag which overwrites the *–cap-add* and *–cap-drop*

flags.

* Do not mount sensitive host system directories on containers such as: /, /boot, /dev, /etc, /lib, /proc,

/sys, /usr.

* Do not run SSH within containers, SSH should only run inside the Docker host. The correct approach to interact with a container instance is using the Docker EXEC command or Docker ATTACH:

*docker exec -i -t <container ID> <sh command>* OR *docker attach <container ID>*

The following command will list the running process, make sure there is no process for SSH server:

*docker exec <container ID> ps -el*

* Expose only the required ports on the container in order to decrees the attack surface on the container.

By using the following command make sure the list does not contain unnecessary mapped ports:

*docker port <container ID>*

* Limit memory usage per container using the *–m* flag, for example:

*docker run –m <memory size> <container image name>*

* Limit CPU share between containers using *–c* flag:

*docker run –c <CPU share> <container image name>*

* Set the value of the flag –*restart=on-failure* to 5 attempts in order to prevent denial of service on the host. This is especially relevant when there are many containers on the same host that are restarting simultaneously.
* Do not use the LXC execution driver, it is the old execution driver and it is mainly for legacy support. The following command shouldn’t return any result if the LXC is not set:

*ps –ef | grep docker | grep lxc*

In addition, avoid using the following command on the Docker daemon:

*docker -d --exec-driver=lxc*

## 3. Permissions

* Verify that the *RUN* command with the flag *–v* for mounting files is used with the flag “*–read- only=true”.* This ensures that process within the container cannot write anything to the root file system.
* Do not share the host's process namespace by using the flag *--pid=host*.
* Do not share the host's IPC namespace by using the flag *--ipc=host*.
* Verify that the ownership of the following files and directories is set to root:root and their permission to 664 or more restricted:
  + */usr/lib/systemd/system/docker.service*
  + */usr/lib/systemd/system/docker-registry.service*
  + */usr/lib/systemd/system/docker.socket*
  + */etc/sysconfig/docker* or someteims at */etc/default/docker*
  + */etc/sysconfig/docker-network* o */etc/sysconfig/docker-registry* o */etc/sysconfig/docker-storage* o */etc/docker*
  + */etc/docker/certs.d/<registry-name>*
  + *<path to TLS CA certificate file>*
  + *<path to Docker server certificate file>*
  + *<path to Docker server certificate key file>*
  + */var/run/docker.sock*

## 4. Permissions

* Verify that the *RUN* command with the flag *–v* for mounting files is used with the flag *“ –read-*

*only=true” .* This ensures that process within the container cannot write anything to the root file system.

* Do not share the host's process namespace by using the flag *--pid=host*.
* Do not share the host's IPC namespace by using the flag *--ipc=host*.
* Verify that the ownership of the following files and directories is set to root:root and their permission to 664 or more restricted:
  + */usr/lib/systemd/system/docker.service*
  + */usr/lib/systemd/system/docker-registry.service*
  + */usr/lib/systemd/system/docker.socket*
  + */etc/sysconfig/docker* or someteims at */etc/default/docker*
  + */etc/sysconfig/docker-network* o */etc/sysconfig/docker-registry* o */etc/sysconfig/docker-storage* o */etc/docker*
  + */etc/docker/certs.d/<registry-name>*
  + *<path to TLS CA certificate file>*
  + *<path to Docker server certificate file>*
  + *<path to Docker server certificate key file>*
  + */var/run/docker.sock*

## 5. Network

* If remote communication to Docker daemon is enabled through the Remote API, configure it to use HTTPS according to Linux Best Practices (Configure TLS authentication in the Linux host).

If you need Docker to be reachable via the network in a safe manner, you can enable TLS by specifying the *tlsverify* flag and pointing Docker’s container *tlscacert* flag to a trusted CA certificate:

* Bind incoming container traffic to a specific host interface in order to avoid accepting connections from all exposed ports on any network interfaces.

By default, Docker daemon binds to a non-networked UNIX socket and runs with 'root' privileges. If you change the default docker daemon binding to a TCP port or any other UNIX socket, anyone with access to that port or socket can have full access to Docker daemon and in turn to the host system. Hence, you should not bind the Docker daemon to another IP/Port or a UNIX socket.

For example, avoid using the following command:

*docker run –p 0.0.0.0:38333:80 <container image name>*

The above command will let the container accept connections on any host interface on the specific port *38333*.

* Do not bind Docker to another IP/Port or a UNIX socket, unless it is for a secure TLS connection.

Changing the default Docker daemon binding to a TCP port or UNIX *docker* user group will increase your security risks by allowing non-root users to gain *root* access on the host. Make sure you control access to Docker. If you are binding to a TCP port, anyone with access to that port has full Docker access; so it is not advisable on an open network. However, binding the Socket is necessary for remote connection to the Docker API. You can audit this using the following command:

*docker ps -ef | grep docker*

Ensuring that the –‘H’ parameter is not present, for example:

*docker -H tcp://10.1.2.3:2375 -H unix:///var/run/example.sock -d*

* Do not use the host network mode on container when starting it, by using the *--net=host* flag*.* It causes the container to skip the separate network stack and tells Docker to not containerize the container’s

networking, which allows the container to have full access to the host's network services and interfaces.

* Restrict network traffic between containers and link specific containers together that require inter communication by using the following command on the Docker daemon:

*docker -d --icc=false*

* Allow Docker daemon to make changes to the *iptables* automatically and make the needed changes based on how the networking options were chosen for the containers.

## 6. Linux Host

* Verify that SELinux security options are applied on all the containers.

##### SELinux is an effective and easy-to-use Linux application security system. It is available on quite a few Linux distributions by default such as Red Hat and Fedora.

SELinux provides a Mandatory Access Control (MAC) system that greatly augments the default Discretionary Access Control (DAC) model. You can thus add an extra layer of safety by enabling SELinux on your Linux host. The interaction between SELinux policy and Docker is focused on two concerns: protection of the host, and protection of containers from one another.

The following command should return all the security options currently configured for the containers:

*docker ps -q | xargs docker inspect –format='{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}*

If SELinux is not enabled and your OS supports it, use it following these steps:

1. Set the SELinux Policy.
2. Create or import a SELinux policy template for Docker containers.
3. Start Docker is daemon mode with SELinux enabled, for example:

*Docker –d –selinux-enabled*

1. Start the container using the security options, for example:

*Docker run –I –t –security-opt label:level:TopSecret <host-os>*

##### The container process should then have set of restrictions as defined in SELinux policy.

* Do not directly expose host devices to containers by using the flag *–device*.
* Verify that the containers are mounted on a separate partition:

##### Docker depends on */var/lib/docker* as the default directory where all docker related files, including the images, are stored. This directory might fill up fast and soon Docker and the host could become unusable. So, it is advisable to create a separate partition (logical volume) for storing Docker files.

To return the partition details for */var/lib/docker* mount point, use the following command:

*grep /var/lib/docker/etc/fstab*

* Use the most updated version of the Linux Kernel, using the following command:

*uname –r*

##### The latest version can be found here:

<https://www.kernel.org/finger_banner>

* Do not use development tools in production.
* Remove all non-essential services from the host.
* Use the most updated version of Docker. The latest version can be found here: <https://docs.docker.com/release-notes/>
* Only allow trusted users to control the Docker Daemon.