Version : **2023.01**

Last update : 2023/12/19 09:15

DOF607 - Docker Security Management

Contents

LAB #1 - Using Docker Secrets

Docker secrets are a secure way of storing sensitive information such as user names, passwords and even files. They are only compatible with Docker in Swarm mode.

Consider the following example of a Docker stack file used to create a PostgreSQL container:

```
version: '3.1'

services:

  db:
    image: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: database

  adminer:
    image: adminer
    ports:
     - 8080:8080
```

You can see in this file the presence of non-secure sensitive information:

- POSTGRES_USER
- POSTGRES_PASSWORD
- POSTGRES_DB

In order to secure this information, start by creating the **postgres** context:

```
root@manager:~# mkdir postgres
```

Then create a Docker Secret called **pg_user** :

```
root@manager:~# cd postgres
root@manager:~/postgres# echo "postgres" | docker secret create pg_user -
lpk8eq80qvfiqw7z1686fmj5t
```

> **Important**: Note the use of the **-** character at the end of the line. This tells the **docker secret** command to read the contents of the pg_user secret from standard input.

To view the list of secrets, use the docker secrets **ls** command:

```
root@manager:~/postgres# docker secret ls
ID                          NAME                DRIVER              CREATED               UPDATED
lpk8eq80qvfiqw7z1686fmj5t   pg_user                                 About a minute ago    About a minute ago
```

> **Important**: Note that the **DRIVER** column is empty. This indicates that the secret is managed by Docker itself instead of being delegated to a third-party plugin.

Now create the **pg_password** and **pg_database** secrets:

```
root@manager:~/postgres# echo "postgres" | docker secret create pg_password -
h9tsfbfwz6o0sd35roklwpopi

root@manager:~/postgres# echo "database" | docker secret create pg_database -
5lx4zydpfocwgpdto0yy1jod9
```

Important: Note that a Docker secret is immutable.

Check that your commands are taken into account:

```
root@manager:~/postgres# docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
5lx4zydpfocwgpdto0yy1jod9   pg_database                             2 minutes ago       2 minutes ago
h9tsfbfwz6o0sd35roklwpopi   pg_password                             3 minutes ago       3 minutes ago
lpk8eq80qvfiqw7z1686fmj5t   pg_user                                 5 minutes ago       5 minutes ago
```

To obtain information about a secret, use the docker secret **inspect** command:

```
root@manager:~/postgres# docker secret inspect pg_database
[
    {
        "ID": "5lx4zydpfocwgpdto0yy1jod9",
        "Version": {
            "Index": 23
        },
        "CreatedAt": "2021-04-15T03:49:36.344367554Z",
        "UpdatedAt": "2021-04-15T03:49:36.344367554Z",
        "Spec": {
            "Name": "pg_database",
            "Labels": {}
        }
    }
]
```

Important: You can see in the output of this command the value

CreatedAt which corresponds to the date the secret was created as well as **UpdatedAt** which corresponds to the date the secret was modified.

The **–pretty** option of the command makes this information appear more clearly:

```
root@manager:~/postgres# docker secret inspect --pretty pg_database
ID:              5lx4zydpfocwgpdto0yy1jod9
Name:              pg_database
Driver:
Created at:        2021-04-15 03:49:36.344367554 +0000 utc
Updated at:        2021-04-15 03:49:36.344367554 +0000 utc
```

Now create the compose file **postgres-secrets.yaml** :

```
root@manager:~/postgres# vi postgres-secrets.yaml
root@manager:~/postgres# cat postgres-secrets.yaml
version: '3.1'

services:

    db:
        image: postgres
        restart: always
        environment:
            POSTGRES_USER_FILE: /run/secrets/pg_user
            POSTGRES_PASSWORD_FILE: /run/secrets/pg_password
            POSTGRES_DB_FILE: /run/secrets/pg_database
        secrets:
           - pg_password
           - pg_user
           - pg_database

    adminer:
```

```
        image: adminer
        ports:
         - 8080:8080

secrets:
  pg_user:
    external: true
  pg_password:
    external: true
  pg_database:
    external: true
```

Note that in this file the three variables **POSTGRES_USER**, **POSTGRES_PASSWORD** and **POSTGRES_DB** have a **_FILE** suffix because they reference **files** containing information rather than information itself. These files are located in the **/run/secrets** directory of the **container**.

Secondly the following section specifies the names of the secrets to be used with the service:

```
        secrets:
           - pg_password
           - pg_user
           - pg_database
```

The last section specifies that secrets are **external** :

```
secrets:
  pg_user:
    external: true
  pg_password:
    external: true
  pg_database:
    external: true
```

Important : The term **external** indicates that the secrets will not be stored in the image but **only** in the container.

Now deploy the service using the **docker stack** command:

```
root@manager:~/postgres# docker stack deploy -c postgres-secrets.yaml postgres
Ignoring unsupported options: restart

Creating network postgres_default
Creating service postgres_db
Creating service postgres_adminer
```

Important : Note a presence of the **Ignoring unsupported options: restart** error. This is due to the fact that the **restart** directive is compatible with the **docker-compose** command but not with the **docker stack** command. The directive that should have been used in the file is **restart_policy:**.

Now connect to Apache Guacamole and open a web browser in the **Debian11_10.0.2.46_VNC** virtual machine. Then navigate to the Manager address on port **8080** and fill in the secret values:

Validate the form and check that the secrets have been taken into account:

Lastly, delete the service:

```
root@manager:~/postgres# docker stack ls
NAME                SERVICES            ORCHESTRATOR
postgres            2                   Swarm

root@manager:~/postgres# docker stack rm postgres
Removing service postgres_adminer
Removing service postgres_db
Removing network postgres_default
```

LAB #2 - Creating a Trusted User to Control the Docker Daemon

Unlike traditional virtual machine management solutions where access is often conditional on the allocation of roles, Docker does not have this type of mechanism. As a result, anyone with access to the host either via **sudo** or by being a member of the **docker** group can access all containers and even stop, delete and create new ones.

```
root@manager:~# cat /etc/group | grep docker
docker:x:999:

root@manager:~# usermod -aG docker trainee

root@manager:~# exit
déconnexion

trainee@manager:~$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get
http://%2Fvar%2Frun%2Fdocker.sock/v1.40/containers/json: dial unix /var/run/docker.sock: connect: permission
denied

trainee@manager:~$ newgrp docker

trainee@manager:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
```

```
PORTS                 NAMES
d02c6115724c        alpine              "/bin/sh"                6 days ago          Exited (0) 6 days ago
alpine1

trainee@manager:~$ docker rm alpine1
alpine1

trainee@manager:~$ docker run -d --name alpine1 alpine sleep 99999
a214e2df0499c97e8da25a6c9ea751ac75344c9bcd7d238f8cb8d5c777510ab9

trainee@manager:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
a214e2df0499        alpine              "/bin/sh"           6 seconds ago       Up 5 seconds
alpine1
```

LAB #3 - The docker-bench-security.sh script

The **Center for Internet Security (CIS)** is an independent non-profit organisation that publishes best practices in many areas of IT. The guide for Docker can be downloaded from https://www.cisecurity.org/benchmark/docker/.

The guide is divided into several sections:

- Docker host configuration,
- Docker daemon configuration,
- Docker daemon configuration files,
- The images and the files used to build the images,
- The container runtime,
- Docker security operations,
- Docker Swarm configuration.

This guide is for use with the **Docker Benchmark Security** script.

Clone the **docker-bench-security.sh** script using **git** :

```
trainee@manager:~$ su -
Password: fenestros

root@manager:~# git clone https://github.com/docker/docker-bench-security.git
Cloning in 'docker-bench-security'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 1921 (delta 5), reused 6 (delta 2), pack-reused 1903
Receiving objects: 100% (1921/1921), 2.90 MiB | 908.00 KiB/s, done.
Delta resolution: 100% (1339/1339), done.
```

Now run the **Docker Benchmark Security** script :

```
root@manager:~# cd docker-bench-security/

root@manager:~/docker-bench-security# ./docker-bench-security.sh
# ------------------------------------------------------------------------------------------
# Docker Bench for Security v1.6.0
#
# Docker, Inc. (c) 2015-2023
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Based on the CIS Docker Benchmark 1.6.0.
# ------------------------------------------------------------------------------------------

Initializing 2023-12-17T14:22:08+01:00


Section A - Check results

[INFO] 1 - Host Configuration
```

```
[INFO] 1.1 - Linux Hosts Specific Configuration
[WARN] 1.1.1 - Ensure a separate partition for containers has been created (Automated)
[INFO] 1.1.2 - Ensure only trusted users are allowed to control Docker daemon (Automated)
[INFO]       * Users: trainee
[WARN] 1.1.3 - Ensure auditing is configured for the Docker daemon (Automated)
[WARN] 1.1.4 - Ensure auditing is configured for Docker files and directories -/run/containerd (Automated)
[WARN] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[WARN] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[WARN] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[INFO] 1.1.8 - Ensure auditing is configured for Docker files and directories - containerd.sock (Automated)
[INFO]       * File not found
[WARN] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[WARN] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[INFO] 1.1.11 - Ensure auditing is configured for Dockerfiles and directories - /etc/docker/daemon.json
(Automated)
[INFO]       * File not found
[WARN] 1.1.12 - 1.1.12 Ensure auditing is configured for Dockerfiles and directories -
/etc/containerd/config.toml (Automated)
[INFO] 1.1.13 - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
(Automated)
[INFO]       * File not found
[WARN] 1.1.14 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Automated)
[WARN] 1.1.15 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim
(Automated)
[INFO] 1.1.16 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v1
(Automated)
[INFO]       * File not found
[INFO] 1.1.17 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v2
(Automated)
[INFO]       * File not found
[WARN] 1.1.18 - Ensure auditing is configured for Docker files and directories - /usr/bin/runc (Automated)
[INFO] 1.2 - General Configuration
[NOTE] 1.2.1 - Ensure the container host has been Hardened (Manual)
[PASS] 1.2.2 - Ensure that the version of Docker is up to date (Manual)
```

```
[INFO]        * Using 19.03.4, verify is it up to date as deemed necessary

[INFO] 2 - Docker daemon configuration
[NOTE] 2.1 - Run the Docker daemon as a non-root user, if possible (Manual)
[WARN] 2.2 - Ensure network traffic is restricted between containers on the default bridge (Scored)
[PASS] 2.3 - Ensure the logging level is set to 'info' (Scored)
[PASS] 2.4 - Ensure Docker is allowed to make changes to iptables (Scored)
[PASS] 2.5 - Ensure insecure registries are not used (Scored)
[PASS] 2.6 - Ensure aufs storage driver is not used (Scored)
[INFO] 2.7 - Ensure TLS authentication for Docker daemon is configured (Scored)
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.8 - Ensure the default ulimit is configured appropriately (Manual)
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.9 - Enable user namespace support (Scored)
[PASS] 2.10 - Ensure the default cgroup usage has been confirmed (Scored)
[PASS] 2.11 - Ensure base device size is not changed until needed (Scored)
[WARN] 2.12 - Ensure that authorization for Docker client commands is enabled (Scored)
[WARN] 2.13 - Ensure centralized and remote logging is configured (Scored)
[WARN] 2.14 - Ensure containers are restricted from acquiring new privileges (Scored)
[WARN] 2.15 - Ensure live restore is enabled (Scored)
[WARN] 2.16 - Ensure Userland Proxy is Disabled (Scored)
[PASS] 2.17 - Ensure that a daemon-wide custom seccomp profile is applied if appropriate (Manual)
[PASS] 2.18 - Ensure that experimental features are not implemented in production (Scored)

[INFO] 3 - Docker daemon configuration files
[PASS] 3.1 - Ensure that the docker.service file ownership is set to root:root (Automated)
[PASS] 3.2 - Ensure that docker.service file permissions are appropriately set (Automated)
[PASS] 3.3 - Ensure that docker.socket file ownership is set to root:root (Automated)
[PASS] 3.4 - Ensure that docker.socket file permissions are set to 644 or more restrictive (Automated)
[PASS] 3.5 - Ensure that the /etc/docker directory ownership is set to root:root (Automated)
[PASS] 3.6 - Ensure that /etc/docker directory permissions are set to 755 or more restrictively (Automated)
[INFO] 3.7 - Ensure that registry certificate file ownership is set to root:root (Automated)
[INFO]      * Directory not found
[INFO] 3.8 - Ensure that registry certificate file permissions are set to 444 or more restrictively (Automated)
```

```
[INFO]      * Directory not found
[INFO] 3.9 - Ensure that TLS CA certificate file ownership is set to root:root (Automated)
[INFO]      * No TLS CA certificate found
[INFO] 3.10 - Ensure that TLS CA certificate file permissions are set to 444 or more restrictively (Automated)
[INFO]       * No TLS CA certificate found
[INFO] 3.11 - Ensure that Docker server certificate file ownership is set to root:root (Automated)
[INFO]       * No TLS Server certificate found
[INFO] 3.12 - Ensure that the Docker server certificate file permissions are set to 444 or more restrictively
(Automated)
[INFO]       * No TLS Server certificate found
[INFO] 3.13 - Ensure that the Docker server certificate key file ownership is set to root:root (Automated)
[INFO]       * No TLS Key found
[INFO] 3.14 - Ensure that the Docker server certificate key file permissions are set to 400 (Automated)
[INFO]       * No TLS Key found
[PASS] 3.15 - Ensure that the Docker socket file ownership is set to root:docker (Automated)
[PASS] 3.16 - Ensure that the Docker socket file permissions are set to 660 or more restrictively (Automated)
[INFO] 3.17 - Ensure that the daemon.json file ownership is set to root:root (Automated)
[INFO]       * File not found
[INFO] 3.18 - Ensure that daemon.json file permissions are set to 644 or more restrictive (Automated)
[INFO]       * File not found
[PASS] 3.19 - Ensure that the /etc/default/docker file ownership is set to root:root (Automated)
[PASS] 3.20 - Ensure that the /etc/default/docker file permissions are set to 644 or more restrictively
(Automated)
[INFO] 3.21 - Ensure that the /etc/sysconfig/docker file permissions are set to 644 or more restrictively
(Automated)
[INFO]       * File not found
[INFO] 3.22 - Ensure that the /etc/sysconfig/docker file ownership is set to root:root (Automated)
[INFO]       * File not found
[PASS] 3.23 - Ensure that the Containerd socket file ownership is set to root:root (Automated)
[PASS] 3.24 - Ensure that the Containerd socket file permissions are set to 660 or more restrictively (Automated)

[INFO] 4 - Container Images and Build File
[INFO] 4.1 - Ensure that a user for the container has been created (Automated)
[INFO]      * No containers running
```

```
[NOTE] 4.2 - Ensure that containers use only trusted base images (Manual)
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container (Manual)
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches (Manual)
[WARN] 4.5 - Ensure Content trust for Docker is Enabled (Automated)
[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images (Automated)
[WARN]      * No Healthcheck found: [nginx:latest]
[WARN]      * No Healthcheck found: [alpine:latest]
[WARN]      * No Healthcheck found: [ubuntu:latest]
[WARN]      * No Healthcheck found: [centos:latest]
[PASS] 4.7 - Ensure update instructions are not used alone in the Dockerfile (Manual)
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed (Manual)
[PASS] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles (Manual)
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles (Manual)
[NOTE] 4.11 - Ensure only verified packages are installed (Manual)
[NOTE] 4.12 - Ensure all signed artifacts are validated (Manual)

[INFO] 5 - Container Runtime
[INFO]   * No containers running, skipping Section 5

[INFO] 6 - Docker Security Operations
[INFO] 6.1 - Ensure that image sprawl is avoided (Manual)
[INFO]      * There are currently: 4 images
[INFO]      * Only 0 out of 4 are in use
[INFO] 6.2 - Ensure that container sprawl is avoided (Manual)
[INFO]      * There are currently a total of 0 containers, with 0 of them currently running

[INFO] 7 - Docker Swarm Configuration
[WARN] 7.1 - Ensure swarm mode is not Enabled, if not needed (Automated)
[PASS] 7.2 - Ensure that the minimum number of manager nodes have been created in a swarm (Automated) (Swarm mode
not enabled)
[PASS] 7.3 - Ensure that swarm services are bound to a specific host interface (Automated) (Swarm mode not
enabled)
[PASS] 7.4 - Ensure that all Docker swarm overlay networks are encrypted (Automated)
[PASS] 7.5 - Ensure that Docker's secret management commands are used for managing secrets in a swarm cluster
```

```
(Manual) (Swarm mode not enabled)
[PASS] 7.6 - Ensure that swarm manager is run in auto-lock mode (Automated) (Swarm mode not enabled)
[PASS] 7.7 - Ensure that the swarm manager auto-lock key is rotated periodically (Manual) (Swarm mode not
enabled)
[PASS] 7.8 - Ensure that node certificates are rotated as appropriate (Manual) (Swarm mode not enabled)
[PASS] 7.9 - Ensure that CA certificates are rotated as appropriate (Manual) (Swarm mode not enabled)
[PASS] 7.10 - Ensure that management plane traffic is separated from data plane traffic (Manual) (Swarm mode not
enabled)


Section C - Score

[INFO] Checks: 86
[INFO] Score: 1
```

This script is used to automate the checking of the points previously mentioned and produces a report containing annotations:

- **[PASS]**: Concerns points that do not need to be modified,
- **[WARN]**: Refers to items that **need** to be modified,
- **[INFO]**: Refers to points that need to be reviewed according to the needs of your configuration,
- **[NOTE]**: Informs you of **best practice**.

LAB #4 - Securing the Docker Host Configuration

When running the script, you should get a result similar to this with regards to Securing the Docker Host Configuration:

```
...
[INFO] 1 - Host Configuration
[INFO] 1.1 - Linux Hosts Specific Configuration
[WARN] 1.1.1 - Ensure a separate partition for containers has been created (Automated)
[INFO] 1.1.2 - Ensure only trusted users are allowed to control Docker daemon (Automated)
[INFO]       * Users: trainee
```

```
[WARN] 1.1.3 - Ensure auditing is configured for the Docker daemon (Automated)
[WARN] 1.1.4 - Ensure auditing is configured for Docker files and directories -/run/containerd (Automated)
[WARN] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[WARN] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[WARN] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[INFO] 1.1.8 - Ensure auditing is configured for Docker files and directories - containerd.sock (Automated)
[INFO]        * File not found
[WARN] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[WARN] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[INFO] 1.1.11 - Ensure auditing is configured for Dockerfiles and directories - /etc/docker/daemon.json
(Automated)
[INFO]        * File not found
[WARN] 1.1.12 - 1.1.12 Ensure auditing is configured for Dockerfiles and directories -
/etc/containerd/config.toml (Automated)
[INFO] 1.1.13 - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
(Automated)
[INFO]        * File not found
[WARN] 1.1.14 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Automated)
[WARN] 1.1.15 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim
(Automated)
[INFO] 1.1.16 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v1
(Automated)
[INFO]         * File not found
[INFO] 1.1.17 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v2
(Automated)
[INFO]         * File not found
[WARN] 1.1.18 - Ensure auditing is configured for Docker files and directories - /usr/bin/runc (Automated)
[INFO] 1.2 - General Configuration
[NOTE] 1.2.1 - Ensure the container host has been Hardened (Manual)
[PASS] 1.2.2 - Ensure that the version of Docker is up to date (Manual)
[INFO]        * Using 19.03.4, verify is it up to date as deemed necessary
...
```

Security issues that should be addressed are indicated by **[WARN]** annotations.

[WARN] 1.1.1 - Ensure a separate partition for containers has been created (Automated)

By default, all Docker files are stored in the **/var/lib/docker** directory, including all images, all containers and all volumes. On a host system with only one partition there is a risk, just like the risk associated with the **/var/log/** directory, that the disk will become saturated.

[WARN] 1.1.3 - Ensure auditing is configured for the Docker daemon (Automated)

```
[WARN] 1.1.4 - Ensure auditing is configured for Docker files and directories -/run/containerd (Automated)
[WARN] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[WARN] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[WARN] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[WARN] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[WARN] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[WARN] 1.1.12 - Ensure auditing is configured for Dockerfiles and directories - /etc/containerd/config.toml
(Automated)
[WARN] 1.1.14 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Automated)
[WARN] 1.1.15 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim
(Automated)
[WARN] 1.1.18 - Ensure auditing is configured for Docker files and directories - /usr/bin/runc (Automated)
```

These warnings are present because **auditd** is not installed and because there are no rules specific to the Docker daemon and its associated directories and files.

Edit the **/etc/apt/sources.list** file as follows:

```
root@manager:~/docker-bench-security# vi /etc/apt/sources.list

root@manager:~/docker-bench-security# cat /etc/apt/sources.list
deb http://archive.debian.org/debian/ stretch main
deb-src http://archive.debian.org/debian/ stretch main
deb http://archive.debian.org/debian-security stretch/updates main
deb-src http://archive.debian.org/debian-security stretch/updates main
```

```
deb [arch=amd64] https://download.docker.com/linux/debian stretch stable
```

Exécute the **apt-update** command:

```
root@manager:~/docker-bench-security# apt update
Ign:1 http://archive.debian.org/debian stretch InRelease
Atteint:2 http://archive.debian.org/debian-security stretch/updates InRelease
Atteint:3 http://archive.debian.org/debian stretch Release
Réception de:4 https://download.docker.com/linux/debian stretch InRelease [44,8 kB]
44,8 ko réceptionnés en 0s (107 ko/s)
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
254 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

To install auditd, use **apt-get** :

```
root@manager:~/docker-bench-security# apt-get install auditd -y
```

Then modify the **/etc/audit/rules.d/audit.rules** file:

```
root@manager:~/docker-bench-security# vi /etc/audit/rules.d/audit.rules
root@manager:~/docker-bench-security# cat /etc/audit/rules.d/audit.rules
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 0

## Set failure mode to syslog
```

```
-f 1

##Docker
-w /usr/bin/docker -p wa
-w /var/lib/docker -p wa
-w /etc/docker -p wa
-w /lib/systemd/system/docker.service -p wa
-w /lib/systemd/system/docker.socket -p wa
-w /etc/default/docker -p wa
-w /etc/docker/daemon.json -p wa
-w /usr/bin/docker-containerd -p wa
-w /usr/bin/docker-runc -p wa
-w /usr/bin/containerd -p wa
-w /run/containerd -p wa
-w /etc/containerd/config.toml -p wa
-w /usr/bin/containerd-shim -p wa
-w /usr/bin/runc -p wa
```

Important: The **-w** option indicates **watch** and concerns the file that follows. The **-p** option logs any changes.

Then restart auditd:

```
root@manager:~/docker-bench-security# systemctl restart auditd
```

Then check for rule support:

```
root@manager:~/docker-bench-security# cat /etc/audit/audit.rules
## This file is automatically generated from /etc/audit/rules.d
-D
-b 8192
```

```
-f 1
--backlog_wait_time 0
-w /usr/bin/docker -p wa
-w /var/lib/docker -p wa
-w /etc/docker -p wa
-w /lib/systemd/system/docker.service -p wa
-w /lib/systemd/system/docker.socket -p wa
-w /etc/default/docker -p wa
-w /etc/docker/daemon.json -p wa
-w /usr/bin/docker-containerd -p wa
-w /usr/bin/docker-runc -p wa
-w /usr/bin/containerd -p wa
-w /run/containerd -p wa
-w /etc/containerd/config.toml -p wa
-w /usr/bin/containerd-shim -p wa
-w /usr/bin/runc -p wa
```

Important - For more information about creating custom rules with auditd, see this **page**.

Re-run the **Docker Benchmark Security** script:

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[PASS] 1.1.4 - Ensure auditing is configured for Docker files and directories -/run/containerd (Automated)
[PASS] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[PASS] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[PASS] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[PASS] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[PASS] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[PASS] 1.1.12 - Ensure auditing is configured for Dockerfiles and directories - /etc/containerd/config.toml
```

```
(Automated)
[PASS] 1.1.14 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Automated)
[PASS] 1.1.15 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim
(Automated)
[PASS] 1.1.18 - Ensure auditing is configured for Docker files and directories - /usr/bin/runc (Automated)
...
```

LAB #5 - Securing the Docker daemon configuration

Run the **docker-bench-security.sh** script again. You should get a result similar to this regarding the security of the Docker daemon configuration:

```
...
[INFO] 2 - Docker daemon configuration
[NOTE] 2.1 - Run the Docker daemon as a non-root user, if possible (Manual)
[WARN] 2.2 - Ensure network traffic is restricted between containers on the default bridge (Scored)
[PASS] 2.3 - Ensure the logging level is set to 'info' (Scored)
[PASS] 2.4 - Ensure Docker is allowed to make changes to iptables (Scored)
[PASS] 2.5 - Ensure insecure registries are not used (Scored)
[PASS] 2.6 - Ensure aufs storage driver is not used (Scored)
[INFO] 2.7 - Ensure TLS authentication for Docker daemon is configured (Scored)
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.8 - Ensure the default ulimit is configured appropriately (Manual)
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.9 - Enable user namespace support (Scored)
[PASS] 2.10 - Ensure the default cgroup usage has been confirmed (Scored)
[PASS] 2.11 - Ensure base device size is not changed until needed (Scored)
[WARN] 2.12 - Ensure that authorization for Docker client commands is enabled (Scored)
[WARN] 2.13 - Ensure centralized and remote logging is configured (Scored)
[WARN] 2.14 - Ensure containers are restricted from acquiring new privileges (Scored)
[WARN] 2.15 - Ensure live restore is enabled (Scored)
[WARN] 2.16 - Ensure Userland Proxy is Disabled (Scored)
[PASS] 2.17 - Ensure that a daemon-wide custom seccomp profile is applied if appropriate (Manual)
```

```
[PASS] 2.18 - Ensure that experimental features are not implemented in production (Scored)
...
```

Security issues that should be addressed are indicated by the **[WARN]** annotations.

[WARN] 2.2 - Ensure network traffic is restricted between containers on the default bridge (Scored)

By default, Docker allows unrestricted network traffic between containers on the same host. However, it is possible to change the default configuration. To prevent this, set the value of **icc** to **false**. In this way, docker creates containers that can communicate with each other **only** if there is a link.

For more information, see this **page**.

[WARN] 2.9 - Enable user namespace support (Scored)

This warning indicates that the use of **user namespaces** is not enabled. The Linux kernel's **user namespaces** support allows a unique range of UIDs and GIDs to be assigned to a process and therefore to a container, outside the traditional range used by the Docker host. The advantage here is that processes with the root UID in the container will be mapped to an unprivileged UID in the Docker host. To use user namespace, you need to set the value of **userns-remap** to **default**. In this case Docker creates a user called **dockremap**. Note that it is also possible to set your own values with **“userns-remap”: “user:group “**.

For more information, see this **page**.

[WARN] 2.12 - Ensure that authorization for Docker client commands is enabled (Scored)

By default, Docker allows unrestricted access to Docker daemons. It is possible to restrict access to authenticated users using a plug-in. This line is not important because access to the local Docker socket is restricted to members of the **docker** group.

For more information, see this **page**.

[WARN] 2.13 - Ensure centralized and remote logging is configured (Scored)

This warning indicates that the rsyslog configuration does not allow traces to be sent to a remote logging server. It also indicates that the **log-driver** value has not been specified. To enable this configuration, you need to set the **log-driver** value to **syslog** and then configure **syslog** and the **log-opts** value correctly.

For more information, see this **page**.

[WARN] 2.14 - Ensure containers are restricted from acquiring new privileges (Scored)

By default, a container can escalate privileges using the setuid or setgid binaries. To prevent this, set the **no-new-privileges** value to **true**.

For more information, see this **page**.

[WARN] 2.15 - Ensure live restore is enabled (Scored)

The –live-restore option enables full support of daemon-less containers within Docker. It ensures that Docker does not stop containers on shutdown or restore and that it properly reconnects to the container when restarted.

[WARN] 2.16 - Ensure Userland Proxy is Disabled (Scored)

There are two ways for a container to route to the outside world:

- **Hairpin NAT** mode,
- **Userland Proxy**.

It is preferable to use Hairpin NAT mode, which can use iptables and has better performance. Most modern operating systems can use Hairpin NAT mode. To disable Userland Proxy, set **userland-proxy** to **false**.

For more information, see this **page**.

5.1 - The /etc/docker/daemon.json file

Create the **/etc/docker/daemon.json** file:

```
root@manager:~/docker-bench-security# vi /etc/docker/daemon.json
root@manager:~/docker-bench-security# cat /etc/docker/daemon.json
{
    "icc": false,
    "userns-remap": "default",
    "log-driver": "syslog",
    "live-restore": true,
    "userland-proxy": false,
    "no-new-privileges": true
}
```

Restart the Docker service:

```
root@manager:~/docker-bench-security# systemctl restart docker
```

Check for the presence of the user named **dockremap**:

```
root@manager:~/docker-bench-security# id dockremap
uid=116(dockremap) gid=121(dockremap) groups=121(dockremap)
```

Re-run the **Docker Benchmark Security** script:

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[PASS] 2.2 - Ensure network traffic is restricted between containers on the default bridge (Scored)
[PASS] 2.3 - Ensure the logging level is set to 'info' (Scored)
[PASS] 2.4 - Ensure Docker is allowed to make changes to iptables (Scored)
[PASS] 2.5 - Ensure insecure registries are not used (Scored)
[PASS] 2.6 - Ensure aufs storage driver is not used (Scored)
```

```
[PASS] 2.9 - Enable user namespace support (Scored)
[PASS] 2.10 - Ensure the default cgroup usage has been confirmed (Scored)
[PASS] 2.11 - Ensure base device size is not changed until needed (Scored)
[PASS] 2.13 - Ensure centralized and remote logging is configured (Scored)
[PASS] 2.14 - Ensure containers are restricted from acquiring new privileges (Scored)
[PASS] 2.15 - Ensure live restore is enabled (Scored)
[PASS] 2.16 - Ensure Userland Proxy is Disabled (Scored)
[PASS] 2.17 - Ensure that a daemon-wide custom seccomp profile is applied if appropriate (Manual)
[PASS] 2.18 - Ensure that experimental features are not implemented in production (Scored)
...
```

For more information, see this **page**.

LAB #6 - Securing Images and Construction Files

Create a mysql container:

```
root@manager:~/docker-bench-security# apt install  --only-upgrade docker-ce
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Le paquet suivant a été installé automatiquement et n'est plus nécessaire :
  libsasl2-modules
Veuillez utiliser « apt autoremove » pour le supprimer.
Les paquets suivants seront mis à jour :
  docker-ce
1 mis à jour, 0 nouvellement installés, 0 à enlever et 252 non mis à jour.
Il est nécessaire de prendre 22,7 Mo dans les archives.
Après cette opération, 497 ko d'espace disque supplémentaires seront utilisés.
Réception de:1 https://download.docker.com/linux/debian stretch/stable amd64 docker-ce amd64
5:19.03.15~3-0~debian-stretch [22,7 MB]
22,7 Mo réceptionnés en 0s (26,0 Mo/s)
```

```
Lecture des fichiers de modifications (« changelog »)... Terminé
(Lecture de la base de données... 112865 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../docker-ce_5%3a19.03.15~3-0~debian-stretch_amd64.deb ...
Dépaquetage de docker-ce (5:19.03.15~3-0~debian-stretch) sur (5:19.03.4~3-0~debian-stretch) ...
Paramétrage de docker-ce (5:19.03.15~3-0~debian-stretch) ...
Traitement des actions différées (« triggers ») pour systemd (232-25+deb9u14) ...

root@manager:~/docker-bench-security# docker container run -d --name mysql -e MYSQL_ROOT_PASSWORD=password mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
e9f2695d7e5b: Pull complete
80c6055edb33: Pull complete
c646ab461d8b: Pull complete
012006c6a591: Pull complete
929d5fa34b95: Pull complete
17e0243877fa: Pull complete
1850b459cd2f: Pull complete
8dceaed53baf: Pull complete
197b834ea1cd: Pull complete
8df78c25b227: Pull complete
Digest: sha256:ceb98918916bd5261b3e9866ac8271d75d276b8a4db56f1dc190770342a77a9b
Status: Downloaded newer image for mysql:latest
e503dd98f88992ae6ed5ec4dcaa2e18982ade8ec74966869515a120763418f74

root@manager:~/docker-bench-security# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
e503dd98f889        mysql               "docker-entrypoint.s…"   2 minutes ago       Up 2 minutes
3306/tcp, 33060/tcp   mysql
```

Run the **docker-bench-security.sh** script again. You should get a result similar to this regarding the security of the images and their build files:

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
```

```
[INFO] 4 - Container Images and Build File
[WARN] 4.1 - Ensure that a user for the container has been created (Automated)
[WARN]      * Running as root: mysql
[NOTE] 4.2 - Ensure that containers use only trusted base images (Manual)
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container (Manual)
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches (Manual)
[WARN] 4.5 - Ensure Content trust for Docker is Enabled (Automated)
[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images (Automated)
[WARN]      * No Healthcheck found: [mysql:latest]
[PASS] 4.7 - Ensure update instructions are not used alone in the Dockerfile (Manual)
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed (Manual)
[PASS] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles (Manual)
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles (Manual)
[NOTE] 4.11 - Ensure only verified packages are installed (Manual)
[NOTE] 4.12 - Ensure all signed artifacts are validated (Manual)
...
```

[WARN] 4.1 - Ensure that a user for the container has been created (Automated)

The processes in the **mysql** container run under the root UID. This is the default Docker action.

For more information, see this **page**.

[WARN] 4.5 - Ensure Content trust for Docker is Enabled (Automated)

This line indicates that Content trust support has not been enabled. Content trust ensures that the images used are signed.

To enable Content trust, the value of the **DOCKER_CONTENT_TRUST** variable must be set to **1** :

```
root@manager:~/docker-bench-security# echo "DOCKER_CONTENT_TRUST=1" | sudo tee -a /etc/environment
DOCKER_CONTENT_TRUST=1
```

```
root@manager:~/docker-bench-security# source /etc/environment
```

Restart the **Manager** virtual machine and start the **mysql** container:

```
root@manager:~/docker-bench-security# reboot
Connection to 10.0.2.62 closed by remote host.
Connection to 10.0.2.62 closed.

root@debian11:~# ssh -l trainee 10.0.2.62
trainee@10.0.2.62's password: trainee
Linux manager.i2tch.loc 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec 17 18:39:07 2023 from 10.0.2.46

trainee@manager:~$ su -
Mot de passe : fenestros

root@manager:~# cd docker-bench-security/

root@manager:~/docker-bench-security#

root@manager:~/docker-bench-security# docker start mysql
mysql
```

Run the script again and note the contents of section 4:

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
```

```
[INFO] 4 - Container Images and Build File
[WARN] 4.1 - Ensure that a user for the container has been created (Automated)
[WARN]      * Running as root: mysql
[NOTE] 4.2 - Ensure that containers use only trusted base images (Manual)
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container (Manual)
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches (Manual)
[PASS] 4.5 - Ensure Content trust for Docker is Enabled (Automated)
[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images (Automated)
[WARN]      * No Healthcheck found: [mysql:latest]
[PASS] 4.7 - Ensure update instructions are not used alone in the Dockerfile (Manual)
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed (Manual)
[PASS] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles (Manual)
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles (Manual)
[NOTE] 4.11 - Ensure only verified packages are installed (Manual)
[NOTE] 4.12 - Ensure all signed artifacts are validated (Manual)
...
```

For more information, see this **page**.

[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images

When an image is built, it is possible to add a **HEALTHCHECK** whose status can be checked by Docker in order to restart the container if necessary.

To set up a HEALTHCHECK, it would be appropriate, for example, to include the following line in the DOCKERFILE file used to build the image:

```
HEALTHCHECK --interval=20s --timeout=3s CMD curl -f http://localhost:8000/ || exit 1
```

This test checks that the container can reach the specified URL every 20 seconds and produces an error after 3 seconds.

For more information, see this **page**.

LAB #7 - Securing the Container Runtime

Run the **docker-bench-security.sh** script again, you should get a result similar to this regarding Container Runtime security:

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[INFO] 5 - Container Runtime
[WARN] 5.1 - Ensure that, if applicable, an AppArmor Profile is enabled (Automated)
[WARN]      * No AppArmorProfile Found: mysql
[WARN] 5.2 - Ensure that, if applicable, SELinux security options are set (Automated)
[WARN]      * No SecurityOptions Found: mysql
[PASS] 5.3 - Ensure that Linux kernel capabilities are restricted within containers (Automated)
[PASS] 5.4 - Ensure that privileged containers are not used (Automated)
[PASS] 5.5 - Ensure sensitive host system directories are not mounted on containers (Automated)
[PASS] 5.6 - Ensure sshd is not run within containers (Automated)
[PASS] 5.7 - Ensure privileged ports are not mapped within containers (Automated)
[PASS] 5.8 - Ensure that only needed ports are open on the container (Manual)
[PASS] 5.9 - Ensure that the host's network namespace is not shared (Automated)
[WARN] 5.10 - Ensure that the memory usage for containers is limited (Automated)
[WARN]       * Container running without memory restrictions: mysql
[WARN] 5.11 - Ensure that CPU priority is set appropriately on containers (Automated)
[WARN]       * Container running without CPU restrictions: mysql
[WARN] 5.12 - Ensure that the container's root filesystem is mounted as read only (Automated)
[WARN]       * Container running with root FS mounted R/W: mysql
[PASS] 5.13 - Ensure that incoming container traffic is bound to a specific host interface (Automated)
[WARN] 5.14 - Ensure that the 'on-failure' container restart policy is set to '5' (Automated)
[WARN]       * MaximumRetryCount is not set to 5: mysql
[PASS] 5.15 - Ensure that the host's process namespace is not shared (Automated)
[PASS] 5.16 - Ensure that the host's IPC namespace is not shared (Automated)
[PASS] 5.17 - Ensure that host devices are not directly exposed to containers (Manual)
[INFO] 5.18 - Ensure that the default ulimit is overwritten at runtime if needed (Manual)
[INFO]       * Container no default ulimit override: mysql
[PASS] 5.19 - Ensure mount propagation mode is not set to shared (Automated)
```

```
[PASS] 5.20 - Ensure that the host's UTS namespace is not shared (Automated)
[PASS] 5.21 - Ensure the default seccomp profile is not Disabled (Automated)
[NOTE] 5.22 - Ensure that docker exec commands are not used with the privileged option (Automated)
[NOTE] 5.23 - Ensure that docker exec commands are not used with the user=root option (Manual)
[PASS] 5.24 - Ensure that cgroup usage is confirmed (Automated)
[PASS] 5.25 - Ensure that the container is restricted from acquiring additional privileges (Automated)
[WARN] 5.26 - Ensure that container health is checked at runtime (Automated)
[WARN]       * Health check not set: mysql
[INFO] 5.27 - Ensure that Docker commands always make use of the latest version of their image (Manual)
[WARN] 5.28 - Ensure that the PIDs cgroup limit is used (Automated)
[WARN]       * PIDs limit not set: mysql
[INFO] 5.29 - Ensure that Docker's default bridge 'docker0' is not used (Manual)
[INFO]       * Container in docker0 network: mysql
[PASS] 5.30 - Ensure that the host's user namespaces are not shared (Automated)
[PASS] 5.31 - Ensure that the Docker socket is not mounted inside any containers (Automated)
...
```

Security issues that should be addressed are indicated by **[WARN]** annotations.

[WARN] 5.1 - Ensure that, if applicable, an AppArmor Profile is enabled (Automated)

This warning is present because the container does not use AppArmor.

For more information, see this **page**.

[WARN] 5.2 - Ensure that, if applicable, SELinux security options are set (Automated)

This warning is present because the container does not use SELinux.

For more information, see this **page**.

[WARN] 5.10 - Ensure that the memory usage for containers is limited (Automated)

This warning is due to the fact that containers automatically have access to the entire RAM of the Docker host :

```
root@manager:~/docker-bench-security# docker run -d -p 8081:80 nginx
Unable to find image 'nginx:latest' locally
sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee: Pulling from library/nginx
1f7ce2fa46ab: Pull complete
9b16c94bb686: Pull complete
9a59d19f9c5b: Pull complete
9ea27b074f71: Pull complete
c6edf33e2524: Pull complete
84b1ff10387b: Pull complete
517357831967: Pull complete
Digest: sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee
Status: Downloaded newer image for nginx@sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee
Tagging nginx@sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee as nginx:latest
e14d5112c2feb71e6f37252bcf99d03603d6b7a3e200bff0d55611a0e9a25e2b

root@manager:~/docker-bench-security# docker stats
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
e14d5112c2fe        strange_bassi       0.00%               2.215MiB / 1.957GiB   0.11%               2.16kB / 0B
0B / 0B             2
e503dd98f889        mysql               0.51%               351.3MiB / 1.957GiB   17.53%              5.54kB / 0B
167MB / 118MB       37
^C
```

Delete the container and re-create it with a memory limit:

```
root@manager:~/docker-bench-security# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
```

```
e14d5112c2fe        nginx               "/docker-entrypoint.…"   About a minute ago   Up About a minute
0.0.0.0:8081->80/tcp   strange_bassi
e503dd98f889        mysql               "docker-entrypoint.s…"   18 minutes ago       Up 7 minutes
3306/tcp, 33060/tcp    mysql

root@manager:~/docker-bench-security# docker rm -f e14
e14

root@manager:~/docker-bench-security# docker run -d -p 8081:80 --memory="256m" nginx
38e91e096c83f7cbe78089617a4d70110bd273f53339f8fed8df2503d3cd65ca

root@manager:~/docker-bench-security# docker stats
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
38e91e096c83        sweet_vaughan       0.00%               2.223MiB / 256MiB     0.87%               2.16kB / 0B
0B / 0B             2
e503dd98f889        mysql               0.49%               351.3MiB / 1.957GiB   17.53%              5.61kB / 0B
167MB / 118MB       37
^C
```

For more information, see this **page**.

[WARN] 5.11 - Ensure that CPU priority is set appropriately on containers (Automated)

This warning is due to the fact that containers automatically have access to all CPUs on the Docker host. To limit this access, several options are available, the most commonly used of which is **–cpu-shares**.

The value of cpu-shares is relative to the default value of **1024**. A value of 512 allows the container to access 50% of CPU cycles but only when cycles are limited. When CPU cycles are not restricted, each container uses as much as it needs.

For more information, see this **page**.

[WARN] 5.12 - Ensure that the container's root filesystem is mounted as read only (Automated)

To minimise the risk of a container being compromised by the presence of malicious code, it is advisable to start containers as read-only, except for volumes that require read/write access.

Create the file **write_a_file** in the **mysql** container:

```
root@manager:~/docker-bench-security# docker exec mysql touch /write_a_file
```

The **docker container diff** Command indicates the differences made to the container compared to the image from which it is derived:

```
root@manager:~/docker-bench-security# docker diff mysql
C /run
C /run/mysqld
A /run/mysqld/mysqld.pid
A /run/mysqld/mysqld.sock
A /run/mysqld/mysqld.sock.lock
A /run/mysqld/mysqlx.sock
A /run/mysqld/mysqlx.sock.lock
A /write_a_file
```

Important: Note that the output shows changes made to the container.

Stop and delete the container:

```
root@manager:~/docker-bench-security# docker stop mysql
mysql

root@manager:~/docker-bench-security# docker rm  mysql
mysql
```

Launch a read-only mysql container:

```
root@manager:~/docker-bench-security# docker run -d --name mysql --read-only -v /var/lib/mysql -v /tmp -v
/var/run/mysqld -e MYSQL_ROOT_PASSWORD=password mysql
711ab28bdfb41220c84246c1658bcde398681a78291bbbe7d3bbfd9bc317d41b
```

Create the **write_a_file** file in the **mysql** container:

```
root@manager:~/docker-bench-security# docker exec mysql touch /write_a_file
touch: cannot touch '/write_a_file': Read-only file system
```

> **Important** : Note the error **touch: cannot touch '/write_a_file': Read-only file system**.

Run the **docker diff** command:

```
root@manager:~/docker-bench-security# docker diff mysql
root@manager:~/docker-bench-security#
```

> **Important**: Note that the command returns no output. This is because the container is read-only, so no changes can be made.

[WARN] 5.14 - Ensure that the 'on-failure' container restart policy is set to '5' (Automated)

This warning concerns the container restart policy. The **on-failure[:max-retries]** policy implies that the container is restarted in the event of a shutdown due to an error that manifests itself as a non-zero return code. The value of **max-retries** is the number of times Docker will try to restart the container. This policy can be set at container startup, **for example**:

```
# docker run -d --name mysql --read-only --restart on-failure:5 -v /var/lib/mysql -v /tmp -v /var/run/mysqld -e
MYSQL_ROOT_PASSWORD=password mysql
```

For more information, see this **page**.

[WARN] 5.26 - Ensure that container health is checked at runtime (Automated)

See Warning 4.6.

[WARN] 5.28 - Ensure that the PIDs cgroup limit is used (Automated)

Without using the **–pids-limit** option, a container could fall victim to a **Fork Bomb** attack, a specific type of denial of service. This type of attack can cause the Docker host to crash and the only remedy is to restart the host. Here's **an example** of a Fork Bomb (do **NOT** execute the following command):

```
# docker run -u 1000 ubuntu bash -c ":() { : | : & }; :; while [[ true ]]; do sleep 1; done"
```

The **manager** Docker host crashes.

To avoid this, create a container using the **–pids-limit** option :

```
root@manager:~/docker-bench-security# docker run -u 1000 --pids-limit 100 ubuntu bash -c ":() { : | : & }; :;
while [[ true ]]; do sleep 1; done"
Unable to find image 'ubuntu:latest' locally
sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b: Pulling from library/ubuntu
a48641193673: Pull complete
Digest: sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Status: Downloaded newer image for ubuntu@sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Tagging ubuntu@sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b as ubuntu:latest
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
```

```
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
^P^Q
```

For more information, see this **page**.

Now delete all containers that have been created :

```
root@manager:~/docker-bench-security# docker rm -f `docker ps -aq`
db5ae43c3e55
f3b2528fbac0
711ab28bdfb4

root@manager:~/docker-bench-security# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
```

Re-create the mysql container incorporating the points seen above:

```
root@manager:~/docker-bench-security# docker run -d --name mysql --read-only --restart on-failure:5 --security-
opt="no-new-privileges:true" --pids-limit 100 --memory="256m" --cpu-shares 512 -v /var/lib/mysql -v /tmp -v
/var/run/mysqld -e MYSQL_ROOT_PASSWORD=password mysql
f49d1ffdeae2e83435e8cc3a2e03fb2e0b33e5609d266e5a3403ff8859e5d122

root@manager:~/docker-bench-security# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
```

```
NAMES
f49d1ffdeae2        mysql               "docker-entrypoint.s…"   16 seconds ago      Up 15 seconds
3306/tcp, 33060/tcp   mysql
```

Run the **docker-bench-security.sh** script again, you should get a result similar to this regarding Container Runtime security:

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[PASS] 5.2 - Ensure that, if applicable, SELinux security options are set (Automated)
[PASS] 5.10 - Ensure that the memory usage for containers is limited (Automated)
[PASS] 5.11 - Ensure that CPU priority is set appropriately on containers (Automated)
[PASS] 5.12 - Ensure that the container's root filesystem is mounted as read only (Automated)
[PASS] 5.14 - Ensure that the 'on-failure' container restart policy is set to '5' (Automated)
[PASS] 5.28 - Ensure that the PIDs cgroup limit is used (Automated)
...
```

LAB #8 - Securing Images with Docker Content Trust

Docker Content Trust (DCT) was introduced with Docker Engine 1.8 and Docker CS Engine 1.9.0. DCT verifies the authenticity, integrity and publication date of a Docker image in a registry. By default, DCT is **disabled**.

DCT is used by the **Docker Hub Registry** but can also be implemented in private registries, in particular through the implementation of the **Docker Container Registry** which is included with **Docker Enterprise**.

DCT is based on the use of the **Docker Notary** tool for publishing and managing content and the **The Update Framework (TUF)**.

For more information about DCT, see this **page**.

8.1 - DOCKER_CONTENT_TRUST

To use **Docker Content Trust (DCT)**, you need to check that the value of the **DOCKER_CONTENT_TRUST** variable is **1** :

```
root@manager:~# echo $DOCKER_CONTENT_TRUST
1
```

Otherwise, the value of the variable must be set to 1 :

```
root@manager:~# export DOCKER_CONTENT_TRUST=1
root@manager:~# echo $DOCKER_CONTENT_TRUST
1
```

8.2 - DCT and the docker pullcommand

In order to use your own registry of the Docker Hub, it is necessary to connect to docker.io:

```
root@manager:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username: <your_account>
Password: <your_password>
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

To see the impact of using DCT, simply perform a **pull** of an unsigned image:

```
root@manager:~# docker image pull i2tch/docker:unsigned
Error: remote trust data does not exist for docker.io/i2tch/docker: notary.docker.io does not have trust data for
docker.io/i2tch/docker
```

Important: Note the error **Error: remote trust data does not exist for**

docker.io/i2tch/docker …. This is because Docker Trust prevents the use of unsigned images.

On the other hand, all **official** images are signed:

```
root@manager:~# docker image pull centos
Using default tag: latest
Pull (1 of 1): centos:latest@sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos@sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Tagging centos@sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177 as centos:latest
docker.io/library/centos:latest
```

This image is now present on **manager.i2tch.loc** :

```
root@manager:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mysql               latest              380f0456d1c1        6 days ago          619MB
ubuntu              latest              174c8c134b2a        6 days ago          77.9MB
alpine              latest              f8c20f8bbcb6        11 days ago         7.38MB
nginx               latest              a6bd71f48f68        3 weeks ago         187MB
centos              latest              5d0da3dc9764        2 years ago         231MB
```

The disable-content-trust option

It is also possible to enable or disable the use of DCT with the **–disable-content-trust=false/true** options when using the **docker build**, **docker push** and **docker pull**, **docker create** and **docker run** commands :

```
root@manager:~# docker image pull --disable-content-trust=true i2tch/docker:unsigned
```

```
unsigned: Pulling from i2tch/docker
10d70a43a9f9: Pull complete
4f4fb700ef54: Pull complete
8951e3a91277: Pull complete
d1814ff35b8b: Pull complete
ff2a2bbf6141: Pull complete
b7205da5c3c9: Pull complete
458ea241cc75: Pull complete
74d1c0702786: Pull complete
c66f3692932d: Pull complete
9224bd1b9757: Pull complete
Digest: sha256:885fc831cb853700ded04029b4fa70ed502947042f6f154e432395cb35619d11
Status: Downloaded newer image for i2tch/docker:unsigned
docker.io/i2tch/docker:unsigned

root@manager:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mysql               latest              380f0456d1c1        6 days ago          619MB
ubuntu              latest              174c8c134b2a        6 days ago          77.9MB
alpine              latest              f8c20f8bbcb6        11 days ago         7.38MB
nginx               latest              a6bd71f48f68        3 weeks ago         187MB
centos              latest              5d0da3dc9764        2 years ago         231MB
i2tch/docker        unsigned            9b915a241e29        7 years ago         212MB

root@manager:~# docker rmi i2tch/docker:unsigned
Untagged: i2tch/docker:unsigned
Untagged: i2tch/docker@sha256:885fc831cb853700ded04029b4fa70ed502947042f6f154e432395cb35619d11
Deleted: sha256:9b915a241e29dc2767980445e3109412b1905b6f1617aea7098e7ac1e5837ae2
Deleted: sha256:27eb08aec7b41dbfa2fd49bc2b3fad9b020fe40b0bc8289af7f53770f0843e7d
Deleted: sha256:7ad0aff4b88909fcff6372fdd26c24d688803b06845426b5a90bcd2f2cae93f4
Deleted: sha256:b93bcd594116ac8886f2daa0fc8d75a59da00161731dab24ababea853d031908
Deleted: sha256:54eda0a22e4b2a1b166cf996eb0651a4f53dec7e9dfad3549bbfe6078f2238a4
Deleted: sha256:36575f1e2764d54fdb92b5296cf4e993499836d6dd9a006f32e173865835070e
Deleted: sha256:27074774f844bdeba18e786585604c8b6352e925a7bd560deb66252bc8ccb861
```

```
Deleted: sha256:0da68695f8bc66fcea8f09004b5cb078861f5d99748f8b7ed035690e02c41477
Deleted: sha256:5dbda9873cdda8ff912b0ae5c34790ee06d7117fa27b193610fa2f7063bf55ff
Deleted: sha256:149690c37bdc8680ec66b0e2cc138f6d63caad74b091acf86a2a18111b90ea79
Deleted: sha256:2caf8a80130d6e9f4ed22e1ec1c3abd2c3f4330d2df9ec62f3b751300190b9e4
Deleted: sha256:1445a9131f2b28a12ff6396faebd6b4beb2cccd7af8eae28d5ff659d65de03ad
Deleted: sha256:4d9799a0754804f5cd623ab744757d16ec81862ee6e5d6986d9d1b0c5e5d5637
Deleted: sha256:dd833146402e8e6e67c48a6ae79a3c86101123e3d6ab1fc7999685eeea06ccba
Deleted: sha256:08d8e6ed6c3a5ac1bfee00f7b11f0a870d6bdc4af6d34169fa1e032c241a63a6
Deleted: sha256:0f3637356bb908638dda037c9c6aa4a2be8a19dbcf452a00cd733a8a456077ac
Deleted: sha256:aedb1b3b3b6e70ae4a342dfdcea874495b9d095ed6ba8eb4bc08f90ad9e83125
Deleted: sha256:05903cd969529ea56beec880bbeb7e90f1bdc281882f1cf3755760e41b181409
Deleted: sha256:d124781fc06a73b05a8644958397994bae668aba2f06f397fe1387c676b0d86f
```

8.3 - DCT and the docker push command

To send the image whose IMAGE ID is **f8c20f8bbcb6** to the private registry, the image tag must be modified:

```
root@manager:~# docker image tag alpine:latest <your_account>/docker:alpine
```

The image with IMAGE ID **f8c20f8bbcb6** now has two tags **alpine:latest** and **<your_account>/docker:alpine** :

```
root@manager:~# docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
mysql                   latest              380f0456d1c1        6 days ago          619MB
ubuntu                  latest              174c8c134b2a        6 days ago          77.9MB
<votre_compte>/docker   alpine              f8c20f8bbcb6        11 days ago         7.38MB
alpine                  latest              f8c20f8bbcb6        11 days ago         7.38MB
nginx                   latest              a6bd71f48f68        3 weeks ago         187MB
centos                  latest              5d0da3dc9764        2 years ago         231MB
```

When pushing to the private registry, passphrases must be created for **two** keys:

- the **root** key also known as the **offline** key (ID 192fc7e), which is only requested the **first** time after DCT is set up when a **repositry** is created,

- the **repository** key, also known as the **tagging** key (ID 168c754), used to sign the image by affixing a **tag**. The signature is specific to the **repositry**.

```
root@manager:~# docker push <your_account>/docker:alpine
The push refers to repository [docker.io/<your_account>/docker]
77cae8ab23bf: Mounted from library/alpine
alpine: digest: sha256:e4355b66995c96b4b468159fc5c7e3540fcef961189ca13fee877798649f531a size: 528
Signing and pushing trust metadata
You are about to create a new root signing key passphrase. This passphrase
will be used to protect the most sensitive key in your signing system. Please
choose a long, complex passphrase and be careful to keep the password and the
key file itself secure and backed up. It is highly recommended that you use a
password manager to generate the passphrase and keep it safe. There will be no
way to recover this key. You can find the key in your config directory.
Enter passphrase for new root key with ID 192fc7e: fenestros
Repeat passphrase for new root key with ID 192fc7e: fenestros
Enter passphrase for new repository key with ID 168c754: fenestros
Repeat passphrase for new repository key with ID 168c754: fenestros
Finished initializing "docker.io/<your_account>/docker"
Successfully signed docker.io/<your_account>/docker:alpine
```

The keys are stored in the **~/.docker/trust/private** directory :

```
root@manager:~# ls -l ~/.docker/trust
total 8
drwx------ 2 root root 4096 nov 10 14:49 private
drwx------ 3 root root 4096 nov. 8 13:48 tuf

root@manager:~# ls -l ~/.docker/trust/private
total 8
-rw------- 1 root root 447 nov. 10 14:49 168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac.key
-rw------- 1 root root 416 nov. 10 14:49 192fc7ed9543ad4bceec58886ab1d605b7433c35f7462d7343d0780d8fddf1db.key
root@manager:~# cat ~/.docker/trust/private/168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
```

```
gun: docker.io/i2tch/docker
role: targets

MIHuMEkGCSqGSIb3DQEFDTA8MBsGCSqGSIb3DQEFDDAOBAhm7HwR0y8FFAICCAAw
HQYJYIZIAWUDBAEqBBC729tU73wKHFQSbmZ1EVZaBIGgmGiFSs4lM5tElSGukl1B
HrELT9aFooFgW7oSXNLM8aFfF/vJ+BSjsgfqWLdvuH+DUXXdUidxcoGMEWnVZNIC
3m40g3MywHilW4rUcjoHVTTUXABGXUQ3f7h+nI15CXcZ11qRLyWbf2uywE9yYH9O
M7GLUcE+pTENJKfZAhRGBEL+LgXNfGI1aAVqaEbBDcDnKKf4Uj1Xu4oLJ7je8+nT
dg==
-----END ENCRYPTED PRIVATE KEY-----

root@manager:~# cat ~/.docker/trust/private/192fc7ed9543ad4bceec58886ab1d605b7433c35f7462d7343d0780d8fddf1db.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
role: root

MIHuMEkGCSqGSIb3DQEFDTA8MBsGCSqGSIb3DQEFDDAOBAiAtCzEar3AhgICCAAw
HQYJYIZIAWUDBAEqBBAO7hHWVoqOo6xcETQQDXRdBIGgPUoLzTz07Ajx8K3D8+Vv
2NUiflMYhH/0I9PL6iA2JJCmD0l+8Ueljy+vHRCu7UAIyWXyIHFN5Aab40mk9/Pg
V2BwSlXp7t1Cnqp/ah7g0T40+OnT64JkTS+l3cS0CaCf2E4l6nY8g4cl40hZIFJz
KREO8uEq3v7HcSBBqFm0+TU+92d7hVuDApPaj0lZYP+3f7H6AjUOqu6hUoK8Ck/Y
Ig==
-----END ENCRYPTED PRIVATE KEY-----
```

====8.4 - DCT and the docker build==== command

The following example demonstrates a Dockerfile that references an unsigned parent image:

```
root@manager:~# mkdir nottrusted
root@manager:~# cd nottrusted/
root@manager:~/nottrusted# vi Dockerfile
root@manager:~/nottrusted# cat Dockerfile
FROM docker/trusttest:latest
RUN echo
```

When building the **<your_account>/docker:nottrusted** image that uses this Dockerfile, an error is returned because its creation does not comply with the use of DCT:

```
root@manager:~/nottrusted# docker build -t <your_account>/docker:nottrusted .
Sending build context to Docker daemon

error during connect: Post
http://%2Fvar%2Frun%2Fdocker.sock/v1.40/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquot
a=0&cpusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=defaul
t&rm=1&shmsize=0&t=i2tch%2Fdocker%3Anottrusted&target=&ulimits=null&version=1: Error: remote trust data does not
exist for docker.io/docker/trusttest: notary.docker.io does not have trust data for docker.io/docker/trusttest
```

Using the **-disable-content-trust** option builds the **<your_account>/docker:nottrusted** image :

```
root@manager:~/nottrusted# docker build --disable-content-trust -t <your_account>/docker:nottrusted .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM docker/trusttest:latest
latest: Pulling from docker/trusttest
Image docker.io/docker/trusttest:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image
for better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/
aac0c133338d: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:50c0cdd0577cc7ab7c78e73a0a89650b222f6ce2b87d10130ecff055981b702f
Status: Downloaded newer image for docker/trusttest:latest
 ---> cc7629d1331a
Step 2/2 : RUN echo
 ---> Running in 694e79d3cd88

Removing intermediate container 694e79d3cd88
 ---> 686e85ee76b8
Successfully built 686e85ee76b8
Successfully tagged <votre_compte>/docker:nottrusted
```

When pushing the **<your_account>/docker:nottrusted** image to the remote repositry, it is **signed**:

```
root@manager:~/nottrusted# docker push <your_account>/docker:nottrusted
The push refers to repository [docker.io/<your_account>/docker]
5f70bf18a086: Layer already exists
c22f7bc058a9: Mounted from docker/trusttest
nottrusted: digest: sha256:1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc size: 734
Signing and pushing trust metadata
Enter passphrase for repository key with ID 168c754: fenestros
Successfully signed docker.io/<your_account>/docker:nottrusted
```

Important : Note the use of the same root key as when pushing the **<your_account>/docker:alpine** image as this is the same repositry.

Create a second repositry

However, by changing the image tag **<your_account>/docker:nottrusted** to **<your_account>/otherimage:latest**, another repositry will be created when the renamed image is pushed:

```
root@manager:~/nottrusted# docker tag <votre_compte>/docker:nottrusted <votre_compte>/otherimage:latest

root@manager:~/nottrusted# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
<votre_compte>/docker         nottrusted          686e85ee76b8        9 minutes ago       5.03MB
<votre_compte>/otherimage     latest              686e85ee76b8        9 minutes ago       5.03MB
mysql                         latest              380f0456d1c1        6 days ago          619MB
ubuntu                        latest              174c8c134b2a        6 days ago          77.9MB
alpine                        latest              f8c20f8bbcb6        11 days ago         7.38MB
<votre_compte>/docker         alpine              f8c20f8bbcb6        11 days ago         7.38MB
nginx                         latest              a6bd71f48f68        3 weeks ago         187MB
centos                        latest              5d0da3dc9764        2 years ago         231MB
```

```
docker/trusttest             latest              cc7629d1331a        4 years ago         5.03MB

root@manager:~/nottrusted# docker push docker.io/<your_account>/otherimage:latest
The push refers to repository [docker.io/<your_account>/otherimage]
5f70bf18a086: Mounted from <your_account>/docker
c22f7bc058a9: Mounted from <your_account>/docker
latest: digest: sha256:1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc size: 734
Signing and pushing trust metadata
Enter passphrase for root key with ID 192fc7e: fenestros
Enter passphrase for new repository key with ID 7b13d02: fenestros
Repeat passphrase for new repository key with ID 7b13d02: fenestros
Finished initializing "docker.io/<your_account>/otherimage"
Successfully signed docker.io/<your_account>/otherimage:latest
```

Important: Note the creation of a second repositry key (ID 7b13d02 instead of ID 168c754) when pushing the **<your_account>/otherimage:latest** image because it is another repositry.

The presence of this second repositry key (**7b13d02d74264624fb201e7ae13ae694286b9f761aa86adddefd0408c7234a58.key**) can be seen in the **~/.docker/trust/private** directory:

```
root@manager:~/nottrusted# ls -l ~/.docker/trust/private
total 12
-rw------- 1 root root 447 nov. 10 14:49 168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac.key
-rw------- 1 root root 416 nov. 10 14:49 192fc7ed9543ad4bceec58886ab1d605b7433c35f7462d7343d0780d8fddf1db.key
-rw------- 1 root root 451 nov. 10 17:37 7b13d02d74264624fb201e7ae13ae694286b9f761aa86adddefd0408c7234a58.key
```

By inspecting the keys of the images created, the use of the different keys is demonstrated very clearly:

```
root@manager:~/nottrusted# docker trust inspect <your_account>/docker:alpine
```

```
[
    {
        "Name": "<your_account>/docker:alpine",
        "SignedTags": [
            {
                "SignedTag": "alpine",
                "Digest": "e4355b66995c96b4b468159fc5c7e3540fcef961189ca13fee877798649f531a",
                "Signers": [
                    "Repo Admin"
                ]
            }
        ],
        "Signers": [],
        "AdministrativeKeys": [
            {
                "Name": "Root",
                "Keys": [
                    {
                        "ID": "d4074334a4ff5a9a43ebd1320ad77c2df88c990ec812f90eb045c603c01ab698"
                    }
                ]
            },
            {
                "Name": "Repository",
                "Keys": [
                    {
                        "ID": "168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac"
                    }
                ]
            }
        ]
    }
]
root@manager:~/nottrusted# docker trust inspect <your_account>/docker:nottrusted
```

```
[
    {
        "Name": "<your_account>/docker:nottrusted",
        "SignedTags": [
            {
                "SignedTag": "nottrusted",
                "Digest": "1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc",
                "Signers": [
                    "Repo Admin"
                ]
            }
        ],
        "Signers": [],
        "AdministrativeKeys": [
            {
                "Name": "Root",
                "Keys": [
                    {
                        "ID": "d4074334a4ff5a9a43ebd1320ad77c2df88c990ec812f90eb045c603c01ab698"
                    }
                ]
            },
            {
                "Name": "Repository",
                "Keys": [
                    {
                        "ID": "168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac"
                    }
                ]
            }
        ]
    }
]
```

Important: Note that the keys used are the same for both images.

```
root@manager:~/nottrusted# docker trust inspect <your_account>/otherimage:latest
[
    {
        "Name":"<your_account>/otherimage:latest",
        "SignedTags": [
            {
                "SignedTag": "latest",
                "Digest": "1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc",
                "Signers": [
                    "Repo Admin"
                ]
            }
        ],
        "Signers": [],
        "AdministrativeKeys": [
            {
                "Name": "Root",
                "Keys": [
                    {
                        "ID": "26f00698f51be2824c6fe85a14722c279bbd487125fe8fa18c0fc8f76dd6280d"
                    }
                ]
            },
            {
                "Name": "Repository",
                "Keys": [
                    {
                        "ID": "7b13d02d74264624fb201e7ae13ae694286b9f761aa86adddefd0408c7234a58"
                    }
```

```
                ]
            }
        ]
    }
]
```

Important : Note that the keys used are different.

Deleting a Signature

Lately it is possible to delete the signature of an image with the **docker trust revoke** command:

```
root@manager:~# docker trust revoke <your_account>/docker:alpine
Enter passphrase for repository key with ID 168c754:
Successfully deleted signature for <your_account>/docker:alpine
root@manager:~# docker trust inspect <your_account>/docker:alpine
[
    {
        "Name":"<your_account>/docker:alpine",
        "SignedTags": [],
        "Signers": [],
        "AdministrativeKeys": [
            {
                "Name": "Root",
                "Keys": [
                    {
                        "ID": "d4074334a4ff5a9a43ebd1320ad77c2df88c990ec812f90eb045c603c01ab698"
                    }
                ]
            },
```

```
            {
                "Name": "Repository",
                "Keys": [
                    {
                        "ID": "168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac"
                    }
                ]
            }
        ]
    }
]
```

> **Important**: There is another cryptographic signature mechanism that certifies the content of images made available on a Registry. Called **Notary**, this system was developed by the Docker community and incorporates part of the **The Update Framework** (TUF) specification. For more information, see this **article**.

LAB #9 - Securing the Docker daemon socket

By default the Docker daemon can be contacted using a local Unix socket which implies that an SSH connection to the Docker host is required. Docker can however use an http socket.

To contact the Docker daemon securely over the network, you need to install, configure and activate TLS support using the **tlsverify** and **tlscacert** options.

The configuration implies that :

- for the Docker daemon, only connections from clients authenticated by a certificate signed by the server's **certification authority** (CA) will be accepted,
- clients can only connect to servers with a certificate signed by the server's CA.

Implementation requires **openssl** :

```
root@manager:~# which openssl
/usr/bin/openssl
```

9.1 - Creating the Certificate Authority Certificate

Start by creating a **ca-key.pem** private key for the CA:

```
root@manager:~# openssl genrsa -aes256 -out ca-key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....................................................................................................
.....++++
.....................................................................................................
.....................................................++++
e is 65537 (0x010001)
Enter pass phrase for ca-key.pem:fenestros
Verifying - Enter pass phrase for ca-key.pem:fenestros
```

Next, create the **ca.pem** certificate for the CA:

```
root@manager:~# openssl req -new -x509 -days 365 -key ca-key.pem -sha256 -out ca.pem
Enter pass phrase for ca-key.pem:fenestros
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:GB
State or Province Name (full name) [Some-State]:SURREY
```

```
Locality Name (eg, city) []:ADDLESTONE
Organization Name (eg, company) [Internet Widgits Pty Ltd]:I2TCH LIMITED
Organizational Unit Name (eg, section) []:TRAINING
Common Name (e.g. server FQDN or YOUR name) []:manager.i2tch.loc
Email Address []:infos@i2tch.co.uk
```

9.2 - Creating the Docker Daemon Host Server Certificate

With the CA keys created, create a **server-key.pem** key for the Docker daemon host server:

```
root@manager:~# openssl genrsa -out server-key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....................................................................................................
..............++++
.........................++++
e is 65537 (0x010001)
```

Next, create a **Certificate Signing Request** (CSR) **server.csr** :

```
root@manager:~# echo $HOSTNAME
manager.i2tch.loc
root@manager:~# openssl req -subj "/CN=`echo $HOSTNAME`" -sha256 -new -key server-key.pem -out server.csr
```

A TLS connection can be made using an FQDN or an IP address. For this reason, create the **extfile.cnf** file:

```
root@manager:~# echo subjectAltName = DNS:`echo $HOSTNAME`,IP:10.0.2.62,IP:127.0.0.1 >> extfile.cnf
```

Set the daemon key usage extended attribute to **serverAuth** :

```
root@manager:~# echo extendedKeyUsage = serverAuth >> extfile.cnf
```

Check that your file has been created correctly:

```
root@manager:~# cat extfile.cnf
subjectAltName = DNS:manager.i2tch.loc,IP:10.0.2.62,IP:127.0.0.1
extendedKeyUsage = serverAuth
```

Now sign the server CSR **server.csr** with the CA private key **ca-key.pem** to produce the server certificate **server-cert.pem** :

```
root@manager:~# openssl x509 -req -days 365 -sha256 -in server.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -
out server-cert.pem -extfile extfile.cnf
Signature ok
subject=CN = manager.i2tch.loc
Getting CA Private Key
Enter pass phrase for ca-key.pem:fenestros
```

9.3 - Creating the Client Certificate

Next, create the **key.pem** private key for the client that will connect to the daemon from the network:

```
root@manager:~# openssl genrsa -out key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....................................................................................................
..................++++
..............++++
e is 65537 (0x010001)
```

Modify the entry for address 10.0.2.61 in the manager's **/etc/hosts** file:

```
root@manager:~# vi /etc/hosts
root@manager:~# cat /etc/hosts
127.0.0.1 localhost
10.0.2.60 debian9.i2tch.loc debian9
10.0.2.61 myregistry.i2tch.loc myregistry
10.0.2.62 manager.i2tch.loc manager
```

```
10.0.2.63 worker1.i2tch.loc worker1
10.0.2.64 worker2.i2tch.loc worker2

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Next, create the client CSR **client.csr** :

```
root@manager:~# openssl req -subj '/CN=myregistry.i2tch.loc' -new -key key.pem -out client.csr
```

Set the client key usage extended attribute to **clientAuth** :

```
root@manager:~# echo extendedKeyUsage = clientAuth > extfile-client.cnf
```

Sign the client CSR **client.csr** with the CA private key **ca-key.pem** to create the client certificate **cert.pem** :

```
root@manager:~# openssl x509 -req -days 365 -sha256 -in client.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -
out cert.pem -extfile extfile-client.cnf
Signature ok
subject=CN = myregistry.i2tch.loc
Getting CA Private Key
Enter pass phrase for ca-key.pem:fenestros
```

Check that the generated files are present:

```
root@manager:~# ls -l
total 60
-rw------- 1 root root 3326 nov. 11 10:53 ca-key.pem
-rw-r--r-- 1 root root 2163 nov. 11 10:57 ca.pem
-rw-r--r-- 1 root root 17 Nov. 11 11:15 ca.srl
-rw-r--r-- 1 root root 1907 Nov. 11 11:15 cert.pem
-rw-r--r-- 1 root root 1594 nov. 11 11:12 client.csr
```

```
drwxr-xr-x 5 root root 4096 nov. 8 12:58 docker-bench-security
-rw-r--r-- 1 root root 1707 Nov. 8 12:35 docker-stack.yml
-rw-r--r-- 1 root root 30 Nov. 11 11:13 extfile-client.cnf
-rw-r--r-- 1 root root 95 nov. 11 11:06 extfile.cnf
-rw------- 1 root root 3243 nov. 11 11:10 key.pem
drwxr-xr-x 2 root root 4096 nov. 10 17:21 nottrusted
-rw-r--r-- 1 root root 1964 nov. 11 11:08 server-cert.pem
-rw-r--r-- 1 root root 1594 nov. 11 11:01 server.csr
-rw------- 1 root root 3243 nov. 11 10:59 server-key.pem
-rw-r--r-- 1 root root 882 oct. 27 15:46 stats
```

Delete the files that have already been used, namely the two CSRs and the two files with the extensions :

```
root@manager:~# rm -v client.csr server.csr extfile.cnf extfile-client.cnf
client.csr' deleted
server.csr' deleted
deleted 'extfile.cnf
'extfile-client.cnf' deleted
```

Change the permissions of the private keys:

```
root@manager:~# chmod -v 0400 ca-key.pem key.pem server-key.pem
the mode of 'ca-key.pem' has been changed from 0600 (rw-------) to 0400 (r--------)
the mode of 'key.pem' has been changed from 0600 (rw-------) to 0400 (r--------)
the mode of 'server-key.pem' has been changed from 0600 (rw-------) to 0400 (r--------)
```

As well as the certificate permissions:

```
root@manager:~# chmod -v 0444 ca.pem server-cert.pem cert.pem
the mode of 'ca.pem' has been changed from 0644 (rw-r--r--) to 0444 (r--r--r--)
the mode of 'server-cert.pem' has been changed from 0644 (rw-r--r--) to 0444 (r--r--r--)
the mode of 'cert.pem' has been changed from 0644 (rw-r--r--) to 0444 (r--r--r--)
```

Stop and delete the **mysql** container:

```
root@manager:~# docker stop mysql
mysql
root@manager:~# docker rm mysql
mysql
```

9.4 - Starting the Docker Daemon with a Direct Invocation

Stop and disable the Docker service:

```
root@manager:~# systemctl stop docker
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
root@manager:~# systemctl disable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable docker
```

Run a direct invocation of **dockerd** so that the daemon only accepts connections from clients providing a certificate signed by the CA :

```
root@manager:~# dockerd --tlsverify --tlscacert=ca.pem --tlscert=server-cert.pem --tlskey=server-key.pem -
H=0.0.0.0:2376 &
[1] 1868
root@manager:~# INFO[2019-11-11T12:01:43.638540628+01:00] Starting up
INFO[2019-11-11T12:01:43.639187816+01:00] User namespaces: ID ranges will be mapped to subuid/subgid ranges of:
dockremap:dockremap
INFO[2019-11-11T12:01:43.641133642+01:00] User namespaces: ID ranges will be mapped to subuid/subgid ranges of:
dockremap:dockremap
INFO[2019-11-11T12:01:43.646949782+01:00] parsed scheme: "unix" module=grpc
INFO[2019-11-11T12:01:43.648299396+01:00] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2019-11-11T12:01:43.648633123+01:00] ccResolverWrapper: sending update to cc:
{[{unix:///run/containerd/containerd.sock 0 <nil>}] <nil>} module=grpc
INFO[2019-11-11T12:01:43.650756512+01:00] ClientConn switching balancer to "pick_first" module=grpc
INFO[2019-11-11T12:01:43.656738368+01:00] parsed scheme: "unix" module=grpc
```

```
INFO[2019-11-11T12:01:43.657165991+01:00] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2019-11-11T12:01:43.660938142+01:00] ccResolverWrapper: sending update to cc:
{[{unix:///run/containerd/containerd.sock 0 <nil>}] <nil>} module=grpc
INFO[2019-11-11T12:01:43.661309281+01:00] ClientConn switching balancer to "pick_first" module=grpc
INFO[2019-11-11T12:01:43.663242717+01:00] [graphdriver] using prior storage driver: overlay2
WARN[2019-11-11T12:01:43.681131788+01:00] Your kernel does not support cgroup rt period
WARN[2019-11-11T12:01:43.681397622+01:00] Your kernel does not support cgroup rt runtime
INFO[2019-11-11T12:01:43.681845166+01:00] Loading containers: start.
INFO[2019-11-11T12:01:43.824330430+01:00] Default bridge (docker0) is assigned with an IP address 172.17.0.0/16.
Daemon option --bip can be used to set a preferred IP address
INFO[2019-11-11T12:01:43.887849374+01:00] Loading containers: done.
INFO[2019-11-11T12:01:43.908567890+01:00] Docker daemon commit=9013bf583a graphdriver(s)=overlay2 version=19.03.4
INFO[2019-11-11T12:01:43.908851991+01:00] Daemon has completed initialization
INFO[2019-11-11T12:01:43.969272646+01:00] API listen on [::]:2376
[Enter]
root@manager:~#
```

Check that the process is running :

```
root@manager:~# ps aux | grep docker
root 1868 0.2 4.0 421876 82236 pts/0 Sl 12:01 0:00 dockerd --tlsverify --tlscacert=ca.pem --tlscert=server-
cert.pem --tlskey=server-key.pem -H=0.0.0.0:2376
root 1995 0.0 0.0 12780 964 pts/0 S+ 12:02 0:00 grep docker
```

Install the **net-tools** package which contains the **netstat** binary:

```
root@manager:~# apt install -y net-tools
```

Check that the **2376** port is listening:

```
root@manager:~# netstat -anp | grep 2376
tcp6 0 0 :::2376 :::* LISTEN 1868/dockerd
```

9.5 - Client configuration

Then transfer the CA certificate and the client certificate and private key to the **debian91** VM :

```
root@manager:~# scp ca.pem key.pem cert.pem trainee@10.0.2.61:/home/trainee/
The authenticity of host '10.0.2.61 (10.0.2.61)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjqF/aJgUc9jg56slNaZQdAUcvBOvE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.61' (ECDSA) to the list of known hosts.
trainee@10.0.2.61's password:
ca.pem 100% 2163 917.8KB/s 00:00
key.pem 100% 3243 3.0MB/s 00:00
cert.pem 100% 1907 921.7KB/s 00:00
```

Run the **docker version** command on the **debian91** VM:

```
trainee@myregistry:~$ docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem --tlskey=key.pem -
H=manager.i2tch.loc:2376 version
Client: Docker Engine - Community
 Version: 19.03.4
 API version: 1.40
 Go version: go1.12.10
 Git commit:        9013bf583a
 Built:             Fri Oct 18 15:52:34 2019
 OS/Arch: linux/amd64
 Experimental: false

Server: Docker Engine - Community
 Engine:
  Version: 19.03.4
  API version: 1.40 (minimum version 1.12)
  Go version: go1.12.10
  Git commit:       9013bf583a
```

```
 Built:            Fri Oct 18 15:51:05 2019
 OS/Arch: linux/amd64
 Experimental: false
containerd:
 Version: 1.2.10
 GitCommit: b34a5c8af56e510852c35414db4c1f4fa6172339
runc:
 Version: 1.0.0-rc8+dev
 GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
 Version: 0.18.0
 GitCommit: fec3683
```

To make it easier to use commands on the server from myregistry, create the **~/.docker** directory :

```
trainee@myregistry:~$ mkdir -pv ~/.docker
mkdir: create directory '/home/trainee/.docker'
```

Then copy the *.pem files into the **~/.docker** directory :

```
trainee@myregistry:~$ cp -v {ca,cert,key}.pem ~/.docker
ca.pem' -> '/home/trainee/.docker/ca.pem
cert.pem' -> '/home/trainee/.docker/cert.pem
'key.pem' -> '/home/trainee/.docker/key.pem'
```

Create the two **DOCKER_HOST** and **DOCKER_TLS_VERIFY** variables:

```
trainee@myregistry:~$ export DOCKER_HOST=tcp://manager.i2tch.loc:2376 DOCKER_TLS_VERIFY=1
```

Now the connection is secure by default:

```
trainee@myregistry:~$ docker image ls
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
<votre_compte>/docker       nottrusted          686e85ee76b8        9 minutes ago       5.03MB
```

```
<votre_compte>/otherimage    latest              686e85ee76b8        9 minutes ago       5.03MB
mysql                        latest              380f0456d1c1        6 days ago          619MB
ubuntu                       latest              174c8c134b2a        6 days ago          77.9MB
alpine                       latest              f8c20f8bbcb6        11 days ago         7.38MB
<votre_compte>/docker        alpine              f8c20f8bbcb6        11 days ago         7.38MB
nginx                        latest              a6bd71f48f68        3 weeks ago         187MB
centos                       latest              5d0da3dc9764        2 years ago         231MB
docker/trusttest             latest              cc7629d1331a        4 years ago         5.03MB
```
