Docker Security

An analysis of security threats and recommended practices for building a secure Docker infrastructure

Andreas Pfefferle

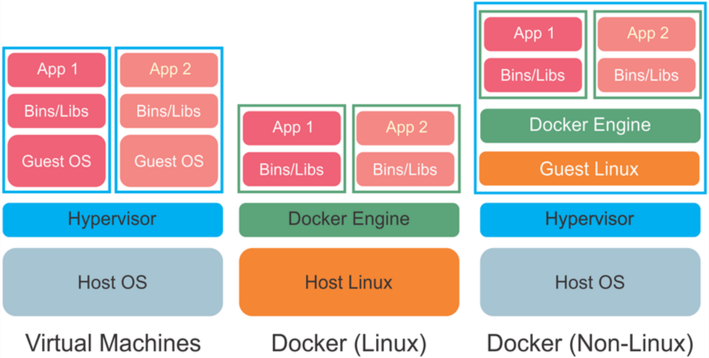
August 24, 2018

Overview

1. Introduction to Docker
2. Built-in Security Features of Docker
3. Docker Security Threats
4. Recommendations for Security Improvements

Introduction to Docker

Comparison with Hypervisor-based Virtualization

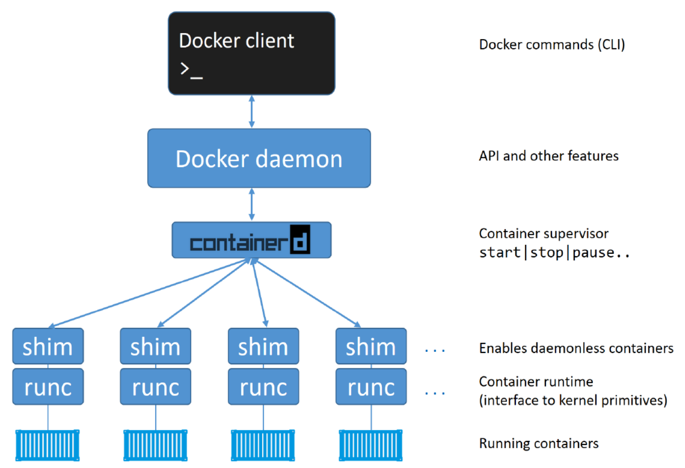


(Docker Inc. 2013)

History

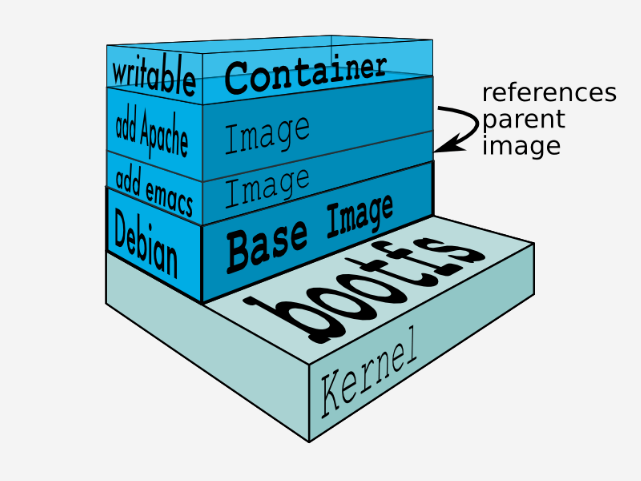
- 2000: FreeBSD *jails*
- 2002-2008: namespaces, capabilities & control groups were added to the Linux kernel
- 2004: Solaris *Zones*
- 2005: Open VZ
- 2008: Linux Containers (LXC)
- 2013: Docker (with LXC)
- 2014: Docker's self-developed *libcontainer* replaced LXC in Docker
- 2015: Open Container Initiative
- 2015: Docker *runC*
- 2016: Docker *containerd*
- 2017: Moby

Architecture



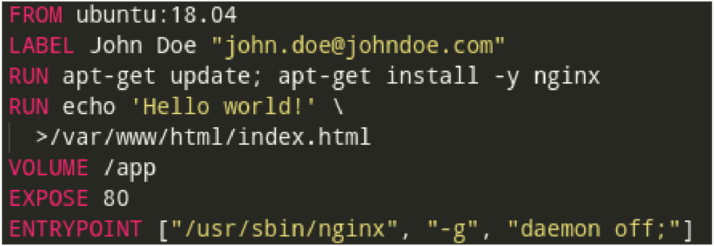
Docker Engine (Poulton 2016)

Images



Docker filesystem layers

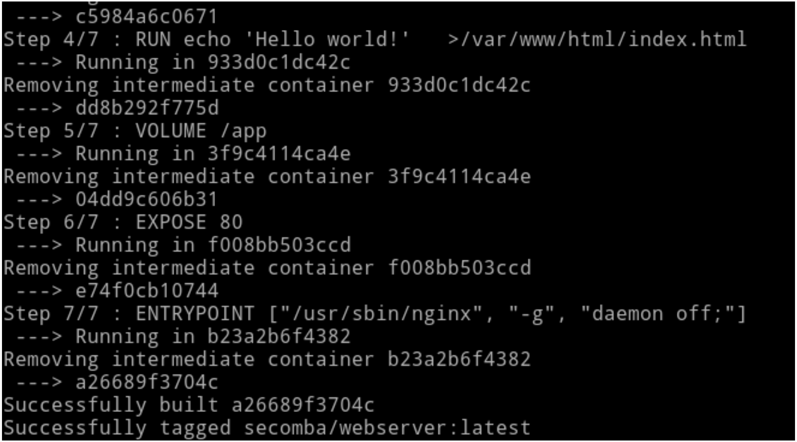
Dockerfile



```
FROM ubuntu:18.04
LABEL John Doe "john.doe@johndoe.com"
RUN apt-get update; apt-get install -y nginx
RUN echo 'Hello world!' \
  >/var/www/html/index.html
VOLUME /app
EXPOSE 80
ENTRYPOINT ["/usr/sbin/nginx", "-g", "daemon off;"]
```

Dockerfile for a simple web server

Image Build and Container Start



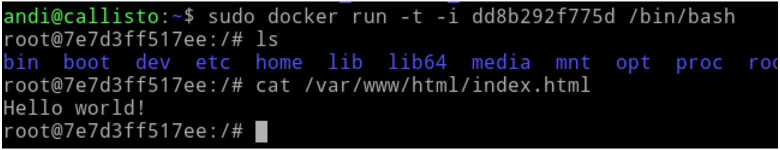
```
 ---> c5984a6c0671
Step 4/7 : RUN echo 'Hello world!'   >/var/www/html/index.html
 ---> Running in 933d0c1dc42c
Removing intermediate container 933d0c1dc42c
 ---> dd8b292f775d
Step 5/7 : VOLUME /app
 ---> Running in 3f9c4114ca4e
Removing intermediate container 3f9c4114ca4e
 ---> 04dd9c606b31
Step 6/7 : EXPOSE 80
 ---> Running in f008bb503ccd
Removing intermediate container f008bb503ccd
 ---> e74f0cb10744
Step 7/7 : ENTRYPOINT ["/usr/sbin/nginx", "-g", "daemon off;"]
 ---> Running in b23a2b6f4382
Removing intermediate container b23a2b6f4382
 ---> a26689f3704c
Successfully built a26689f3704c
Successfully tagged secomba/webserver:latest
```

Running $ `sudo docker build -t="secomba/webserver:latest" .`

Image Build and Container Start

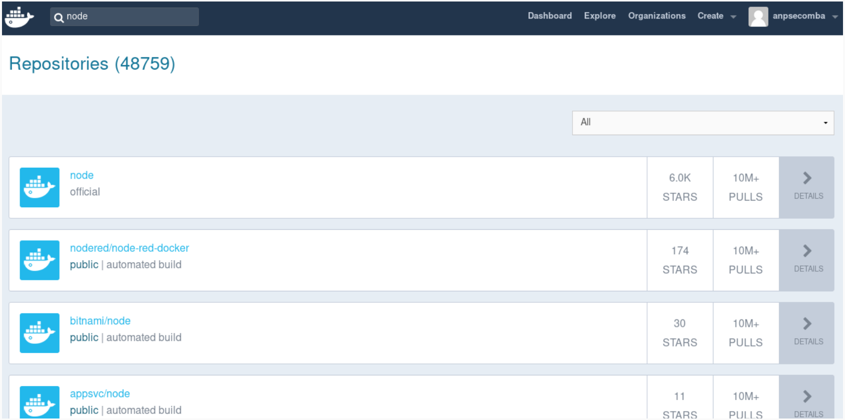
```
$ sudo docker run -t -i secomba/webserver:latest
```

Image Build and Container Start



```
andi@callisto:~$ sudo docker run -t -i dd8b292f775d /bin/bash
root@7e7d3ff517ee:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  roo
root@7e7d3ff517ee:/# cat /var/www/html/index.html
Hello world!
root@7e7d3ff517ee:/#
```

Docker Registries



Repositories (48759)

node official	6.0K STARS	10M+ PULLS	DETAILS
nodered/node-red-docker public | automated build	174 STARS	10M+ PULLS	DETAILS
bitnami/node public | automated build	30 STARS	10M+ PULLS	DETAILS
appsvc/node public | automated build	11 STARS	10M+ PULLS	DETAILS

Docker Hub

Built-in Security Features of Docker

Linux Namespaces for Container Isolation

- Process isolation through the PID namespace
- Network Isolation through the network namespace
- File system isolation through the mount namespace
- Isolation of inter-process communication through the IPC namespace
- UTS isolation through the UTS namespace
- (User isolation through the user namespace)
- ~~Control Group isolation through the cgroup namespace~~

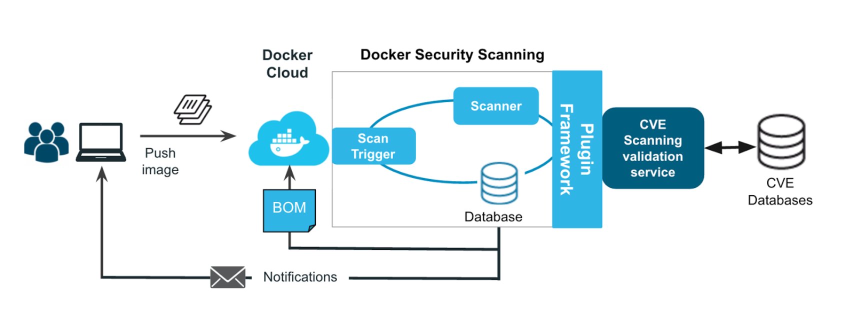
Linux Capabilities to restrict Access Rights

- Access Control in Unix: Discretionary Access Control (DAC)
- Linux capabilities: Dividing root privileges into more than 30 individual capabilities
- Docker: currently 14 capabilities
- Docker run command allows `cap-add` and `cap-drop` parameters

Resource Management through Control Groups

- Control groups manage the host's resources
- Docker: individually allocate resources to each container
- Control groups try to prevent Denial of Service attacks

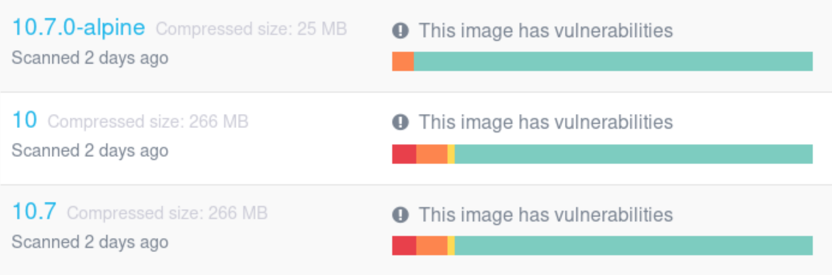
Docker Security Scanning for Image Inspection



Docker Security Scanning

(Toli Kuznets 2016)

Docker Security Scanning for Image Inspection



10.7.0-alpine Compressed size: 25 MB

Scanned 2 days ago

This image has vulnerabilities

10 Compressed size: 266 MB

Scanned 2 days ago

This image has vulnerabilities

10.7 Compressed size: 266 MB

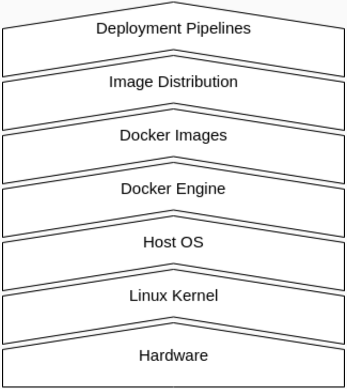
Scanned 2 days ago

This image has vulnerabilities

Multiple images in the official Node repository

Docker Security Threats

Overview and Approach



Overview of the investigated layers. Each layer will be analyzed with STRIDE to identify the threats.

Hardware

Identified Threats:

Spoofing -

Tampering -

Repudiation -

Information Disclosure Unauthorized processes can read arbitrary memory

Denial of Service -

Elevation of Privilege -

Hardware

Exploitation:

- Meltdown (Kernel patches mitigated the risk)
- Spectre (multiple variants disclosed regularly)
- Prime+Probe in Intel SGX (worked in a Docker environment)

Linux Kernel

Identified Threats:

Spoofing -

Tampering Using a memory corruption vulnerability in the kernel, arbitrary code can be executed. Thus, an attacker could modify files, which violates the integrity of the files.

Repudiation -

Information Disclosure -

Denial of Service -

Elevation of Privilege -

Linux Kernel

Exploitation:

- Missing pointer checks lead to memory corruption
  - Attacker uses a kernel function and passes a pointer value
  - Kernel should use the `access_ok` function
  - Kernel does not check whether the value of the provided pointer points to only user-space memory
  - Attacker can read and write to arbitrary kernel locations and execute arbitrary code

Docker Host

Identified Threats:

Spoofing -

Tampering -

Repudiation -

Information Disclosure -

Denial of Service -

Elevation of Privilege Mounting the host's root filesystem with write permissions enables the execution of highly privileged executable program files.

Docker Host

Exploitation:

- Mounting `/var/run/docker.sock` into a container: e.g., `nginx-proxy`

Docker Engine

Identified Threats:

Spoofing -

Tampering As in CVE-2014-6408, containers can bypass the isolation. It has been shown that containers obtain read and write access on the entire host file system in the event of an outbreak.

Repudiation -

Information Disclosure -

Denial of Service -

Elevation of Privilege -

Docker Engine

Exploitation:

- 3 CVE vulnerabilities have not yet been mitigated

Docker Images

Identified Threats:

Spoofing -

Tampering -

Repudiation -

Information Disclosure -

Denial of Service A maliciously prepared image from a public repository uses the computing power of the machine to mine cryptocurrencies.

Elevation of Privilege -

Docker Images

Exploitation:

- Docker Hub repository ”docker123321”: 5 million downloads, around 90,000$ Monero coins

Image Distribution

Identified Threats:

Spoofing -

Tampering -

Repudiation -

Information Disclosure -

Denial of Service Unpacking untrusted zipped archives may cost a lot of computing power and, eventually, lead to full memory.

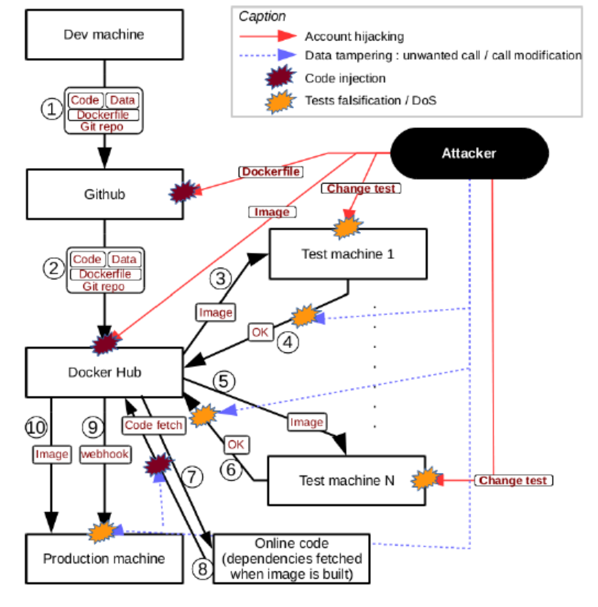
Elevation of Privilege -

Image Distribution

Exploitation:

- Zip Bomb: 42.zip

Deployment Pipelines



Deployment Pipelines

Exploitation:

- 5 minutes and 30 seconds after the push to Github: container with malicious image runs in production machine
- July 2018: 22,000 management interfaces publicly accessible, 300 without a password

Recommendations for Security Improvements

Hardware

- Intel Cascade Lake processors in the second half of 2018

Linux Kernel

- Updating the kernel regularly
- Unikernels?

Docker Host

- Using a host operating system that supports Mandatory Access Control (MAC) such as AppArmor or SELinux
- Seccomp: filter incoming system calls using a whitelist
- Removing all capabilities except those that are explicitly required
- Reducing mount operations

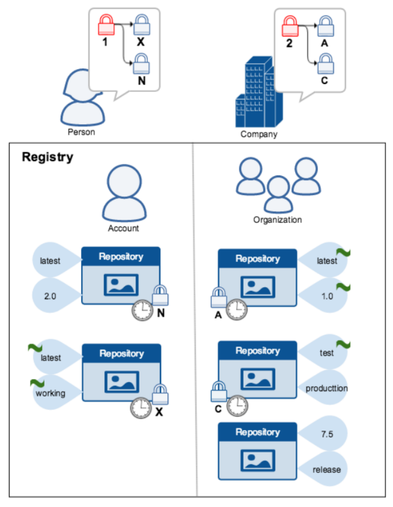
Docker Engine

- User namespace: root privileges inside the container, but is effectively mapped to an unprivileged UID on the host system

Docker Images

- If a service does not need root privileges inside the container: Dockerfile `USER` instruction
- Smaller base images = smaller attack surface
- Remove package installer
- Local security scans with Anchore or Clair
- Avoid vulnerability-prone technologies

Image Distribution



Docker Content Trust

Deployment Pipelines

- 2FA, U2F (e.g., Google: no successful phishing attack in over a year and a half)
- fewer automation in automated pipelines

Conclusion

- complexity overload, especially with Kubernetes, Docker Swarm, etc.
- only on dedicated hardware
- only with hardening measures