Security of Docker containers

Report for the Computer Security exam at the Politecnico di Torino

Carmine D'Amico (239540)
tutor: Antonio Lioy

July 2018

Contents

1 Introduction

Explain here why the XYZ protocol is important and what was the purpose of the present work.

If you want to reference a web site you can do like this: http://www.polito.it.

2 State of the art

In computer science, the term *virtualisation* is referred to the creation of virtual computational resources. These resources, normally supplied as hardware, are instead provided to the user by the operating system through the creation of a new abstraction layer. OSs, storage devices or network resources could all be virtualised. Virtualisation can be obtained at different levels and using different techniques.

Virtual machines have represented for many years the state of the art of virtualisation, being used in both consumer and enterprise contexts. In the last years a new technology, based on *containers*, has started to gain more attention, thanks to its benefits. Docker is an open source container technology, stepped into the limelight thanks to its simple interface, which allows to create and control containers.

2.1 From virtual machines...

With the term virtual machines it is often intended an **hypervisor-based virtualisation**, that is a type of virtualisation that acts at hardware level. Virtual machines (VMs) are established on top of the host operating system, providing applications with their dependencies, but also an entire guest OS and a separate kernel. One or more virtual machines can be run on the same machine. Hypervisors are distinguished in two different types, the one that works directly on top of the host's hardware (**bare metal hypervisor**) and the one that is on top of the host's OS (**hosted hypervisor**).

Bare metal hypervisor provides better performances, not having the overhead of the extra layer of the host's operating system. It manages directly hardware and the guest's operating system. On the contrary hosted hypervisor can be manged in an easier way, running as a normal computer program on the user's operating system.

As said before, the hypervisor needs to run on the user's computer, which is defined as *host machine*, while each virtual machines is called *guest machine*. It is important to remember this terminology, because it will be used also in the following, referring to containers.

2.2 ...to containers

Container-based virtualisation represents another approach to virtualisation, mainly spread in the last years. Compared to hypervisor-based virtualisation it results more lightweight, using the host's kernel to run multiple virtual environments. It virtualises at operating system level (it is also known as **OS-level virtualisation**), allowing other applications to run without installing their own kernel on the host. Containers look like separated process that just share host's kernel.

Resources are provided by the host's OS, together with the container engine. A container engine is the technology in charge of create and manage containers. **Docker** represents one of the most important and most used container engine. A computer program running into a container can only see the resources allocated to that particular container. On the same host there could be more than one container, each one with its personal set of dedicated resources. Although it could be possible to run more than one computer program inside the same container, it is always suggested to run only one program per container, in order to separate areas of concern. It is better to connect multiple containers using user-defined networks and shared volumes.

There are many representatives of this container-based virtualisation approach, like *Linux-VServer*, *OpenVZ* and *LinuX Container (LXC)*. This last will be better described in the next section, allowing to better understand the behaviour of containers.

2.3 LXC

LinuX Containers, better known as LXC, are an OS-level virtualisation technique created around 2008. They allow to run multiple isolated Linux instances (the *containers*), on top of a single host *LXC host*, which shares its Linux kernel. Each container *sees* its own CPU, memory, network interface, I/O, ecc...

The isolation among containers is obtained thanks to some Linux kernel's tools: namespace and cgroups. In the following subsections these two components will be analysed, both for their importance for containerisation in general and for the fact that they are also the basic components in Docker.

2.3.1 Kernel namespace

Namespaces allow to create isolated environments, in which each process that belongs to that particular environment can see global host's resources as personal isolated resources. In other words they allow to create pool of processes that think to be the only ones of the system. In this way groups of processes, that are part of different namespaces, can see different set of resources. Namespaces work by assigning to different resources the same name in different namespaces. In the Linux kernel six different type of environments are implemented:

1. PID Namespace
2. IPC Namespace
3. UTS Namespace
4. Network Namespace
5. User Namespace
6. Mount Namespace

2.3.2 Cgroups

Cgroups are a kernel tool used to manage processes' resources. They gather, track and limit processes' usage of resources. It is possible to create and manage *cgroups* using high level code, assigning PID to a specific *cgroup*. They represent the fundamental tool to obtain resource isolation, playing an important role also for the CPU and I/O's scheduling.

2.4 Docker

2.4.1 History

2.4.2 Implementation

3 Hardening Docker

References

[1] P. Papadimitratos, G. Calandriello, J.-P. Hubaux, A. Lioy, "Impact of vehicular communication security on transportation safety", MOVE 2008: IEEE INFOCOM-2008 workshop on Mobile Networking for Vehicular Environments, Phoenix (AZ, USA), April 13-18, 2008, pp. 1-6

[2] W. Diffie, M.E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976, pp. 644–654

[3] R. Shirey, RFC-4949 "Internet Security Glossary, Version 2", August 2007