THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Scrutinizing Docker Security and Applications

Pawandeep Kaur
Department of computer science and engineering, Lovely Professional University, Phagwara, Punjab,India
Pawandeep.23808@lpu.co.in

Bhanu Pratap Choudhary
Department of computer science and engineering, Lovely Professional University, Phagwara, Punjab,India
bhanupratap.8047@gmail.com

Gunseerat Kaur
Department of computer science and engineering, Lovely Professional University, Phagwara, Punjab,India
Gunseerat.22157@lpu.co.in

***Abstract*—In recent times the scenario of virtualization has changed, virtualization of resources is becoming an active trend in the corporate sectors. Virtualization was limited to some boundaries and they were generating some serious issues, and people were unable to utilize the most of the resources. So a new concept emerged namely " CLOUD ", to extend those boundaries and utilize the most of the resources provided by the vendors. Applications needed to run in the most powerful, optimised and fully functional environment, which lead to the development of " Containers". Container unlike VM's run user applications on the top of the operating system's kernel, often called as OS level virtualization. It allows number of remote user program instances to run on a single hosted platform or a server. And due to the rapid industrialization around the globe the security of the platform becomes a major issue which can lead to many mishappenings to the latest tech world. This paper describes about the docker container technology and the analysis of its security features and pros and cons which helps to deploy powerful applications and keep them secured over the networks from the threats .**

***Keywords*—** *Docker, Container, Cloud Computing , Virtualization, Security, Cyber Security, Threats,*

I. INTRODUCTION

Docker is a promising technology that can deliver software in the form of packages named as containers using operating system level virtualization, which permits virtualization layer to run as an application in an OS [5]. Containers are solitary packed with all necessary files, libraries, directories, thus, every container is different from one another , having their own software. Therefore, they could converse with each other following organized pathways. Docker provides isolation to its users by providing high level API's in low capacity containers[7]. Below fig 1, shows container based and virtual machine based implementation.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

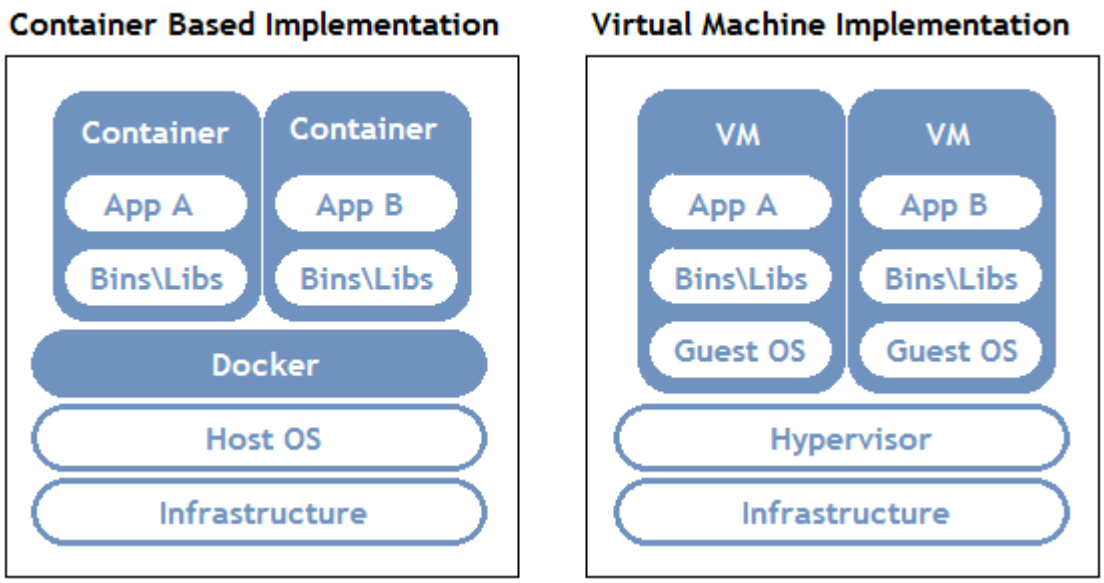


Fig 1: Container and Virtual machine based implementation.

Table1: Virtual Machine vs Docker container

METRICS	VIRTUAL MACHINE	CONTAINER
Definition	Operating system images that are installed and deployed with full capacity and working environment using hypervisor.	Lightweight virtual package applications with isolated environments to develop and operate professional applications.
Technology	Virtualization using user space in addition with Kernel space of an OS	Containerization using user space of an operating system
Specification	Heavyweight	Lightweight
Memory Space requirement	More	Less
Start up time	Minutes	Milliseconds
Virtualization	Hardware level	Operating System level
Security	Fully isolated so it provides more security	Offers process level isolation only, so it is less secure
Example	VMware vSphere, Virtual Box, Hyper-V	Docker, LXD,LXC, Kubernetes, CG Manager
Provisioning	Slow	Real Time
Performance	Limited	Native

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Types of Operating System	Multiple isolated operating systems	Same Kernel but different distribution
Size	in GB	in MB
Lifecycle	Long lived because it has a boot process	Short lived because it has no boot processing as it runs directly on Kernel
Mode of communication	Through ethernet devices	Through IPC (Isolates Interprocess Communication)mechanism
Business benefits	Recovery , Automation, easy to maintenance and availability	Agile development speedup go to market of products
Cost benefits	Less hardware required for managing and upgrading ; energy saving from fewer server, lower operating cost	Efficiently using resources, Increased use of infrastructure , reducing overall total cost of ownership
Shift to Cloud	Fast duplication, scaling and workload mobility makes it easy to host virtual machines in the cloud.	Easy scale and provide elastic cloud services.

II. DOCKER CONTAINER ARCHITECTURE

Docker changes the approach of abstracting the resources to the OS (operating system) level from the hardware level where application is to be self contained. For further understanding of the Docker , a brief explanation of Docker architecture is given below, which uses a client-server model and it is composed of the following units .

- **DAEMON**: responsibilities of daemon is to provide container associated actions and it accepts commands through CLI or REST API.
- **CLIENT:** communication between users and Docker . It can be on the same host machine acting as a daemon or it may reside on another remote host.
- **OBJECTS:** assembling of an application is done by an objects . Excepting from the services, networks, volumes and other objects , essential objects are :-

  -**IMAGES:** for building containers , images are read-only templates. These are used for shipment and storage of an application.

  -**CONTAINERS :** applications are running in an encapsulated environment of Containers. It is defined by configuration options and images. For initiating and supervising containers performance , a core container runtime is established at a lower level .

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

- **REGISTRIES:** Registries are areas from where images can be downloaded and stored .[6]

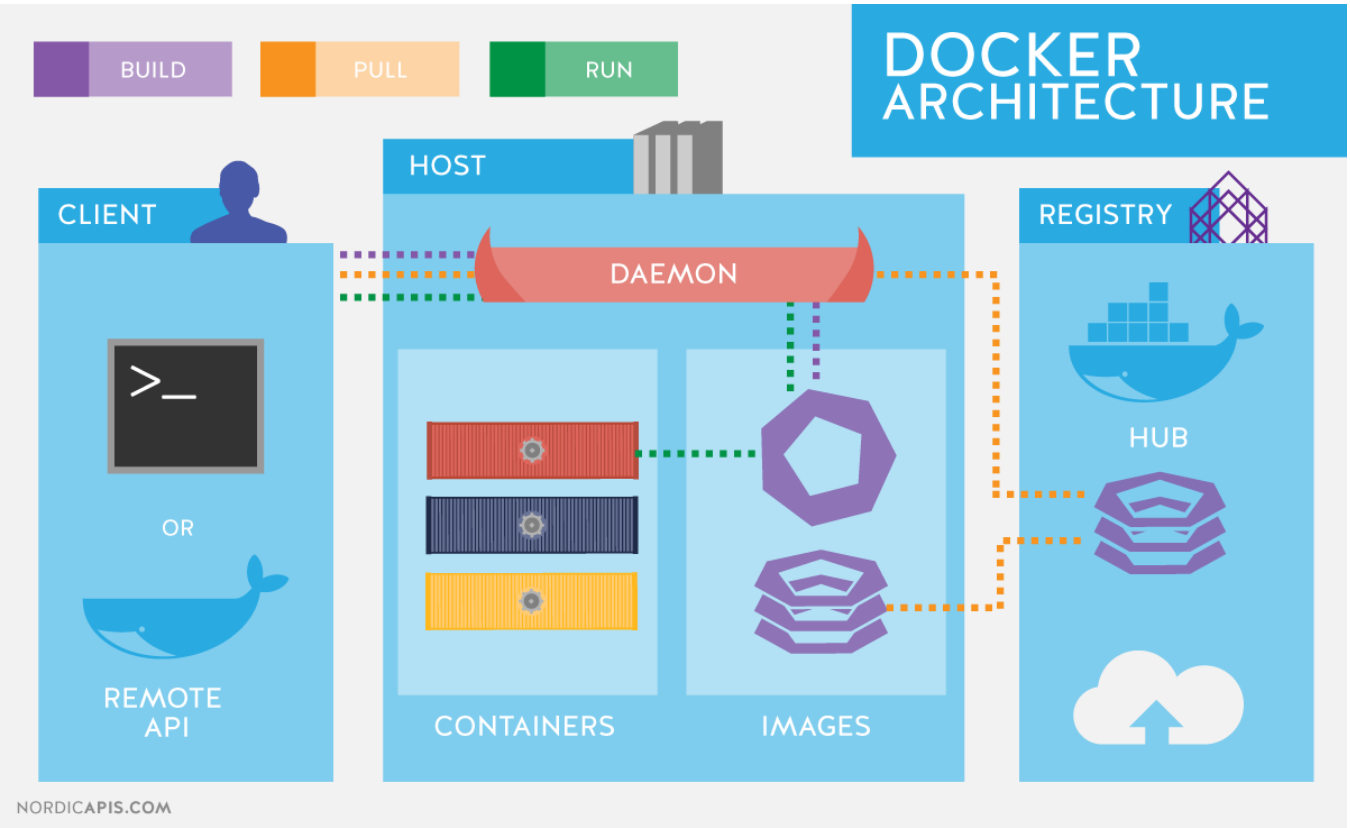


Fig 2 : Docker Container Architecture

III. DOCKER SECURITY

Most prominent factors to be taken under consideration while analysing Docker security are as follows:-

1. The Kernel's inconsistent security and its assistance to namespaces and cgroups.
2. The strike area of the Docker daemon.
3. By default gaps in the container configuration account or it may be generated when personalized by the users.
4. The toughened security features of the Kernel and how they interrelate with containers.

Further description of the above factors are as follows :-

1. Namespaces of Kernel

When a command *docker run* is initiated , then Docker generate a set of namespaces and container's control groups. Namespaces provides isolation, processes running in a container can not see other processes running in host machine or in other containers . Every container has its own network stack where containers does not have authority to access interfaces or sockets of other containers. Sometimes at the time of configuration settings are done in such a way that containers can communicate with others via particular network interfaces , therefore IP traffic is allowed between containers. If required containers can ping one another , begin TCP connections and can send or receive UDP packets. Namespaces are defined as merging the features of mainstream Kernel with help of reimplementing OpenVZ.[9]

2. CONTROL GROUPS

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Essential elements of Linux Containers are Control Groups. They limit and account implementation of resources. They ensure resource allocation such as memory , disk I/O, CPU etc. They are necessary to repel denial-of-service attacks, as they are not responsible for preventing container access and effect of data processes running within another containers. Specifically these are for multi-tenant platforms, such as private and public PAAS, to ensure a consonant performance and uptime while some application does not perform well.

3. Attack surfaces of Docker daemon

Running Docker Daemon is running the applications in containers, with the requirement of *root* privilege and client should know all necessary details . Especially, authorized users must process Docker Daemon. Docker permits sharing of directories with Docker host and a guest container. It is easy to start a container with *host* directory and then container can alter host filesystem without any limitation. The Docker CLI communicate with the Docker Daemon using REST API endpoints which is changed in Docker 0.5.2 and latest uses a UNIX socket in lieu of TCP socket bound on 127.0.0.1. It is necessary to protect API endpoints with HTTPS and certificates. It is only accessed from an authorized VPN.

4. Capabilities of Linux Kernel

Initially Docker starts container with limited capabilities. There are many certain fields where root privileges are needed. By default docker drops all capabilities and accept only needed capabilities. There is an issue with running Docker container that the primary capabilities may offer incomplete isolation, either independently or when used with Kernel loopholes. Docker assist the adding up and deleting capabilities, permit the use of non-default profile. This feature of Docker makes it more secure by removing capabilities and less secure by adding capabilities.

5. Trust Signature verification of Docker Content

Only signed images can be run on Docker engine. In the *dockerd* binary by default Docker Content Trust Signature Verification feature is built in . To sanction this feature , in *daemon.json* trustpinning can be configured, as repositories signed can be run with a user-specific root key. Docker Content Trust (DCT) offers the provision of using digital signatures for information received or sent from any remote docker registries. These digital signatures provide runtime verification or client -side verification of the integrity and of particular image tags. Docker security provides an interesting look into securing systems that can diverge from traditional data center controls. There's a lot of finger pointing that can be done in this case, and while there are some efforts Docker can make towards better security, it's still only a part of security as a whole. First of all understanding of security boundaries are there, a process that often analyzes the shared responsibility model. When stating the shared responsibility model, it is the responsibility shared between parties.A person having knowledge of cloud computing in any capacity it's a very core concept to ensuring overall security and compliance. AWS, for example, lays out their shared responsibility model on their compliance site. They have responsibility in keeping the data centers in availability zones secure. On the other hand, they are not responsible for ensuring that your IAM policies are suitable for your business and compliance needs. This is because such needs are very business centric, and it would be infeasible for AWS to set guidelines for every single compliance

 Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

case. Docker shares a similar shared responsibility model. They provide Docker Store and Docker Hub as methods to register and download docker images. It's important to note that the Docker Store will be more vetted out than the Docker Hub. In terms of shared responsibility, Docker maintains the infrastructure for Docker Store and Docker Hub. They also have some degree of responsibility for many of the official images. Your containers being properly firewalled, however, would not be something Docker would assume responsibility for.

A. Usage Of Official Images

Official images are vetted through a process described on the official Docker website. Notable features include:

1) *Signed images*
2) *Best practices Dockerfile*
3) *GitHub repository as source of truth*

In cases where you need customization, you may want to consider utilizing official images as your base and building off of that using a custom Dockerfile.

B. Docker Trusted Registry

If you're in an enterprise environment in an on-prem/cloud/hybrid type setup, Docker Trusted Registry can help with the business side of things (I tried to write this in a way that didn't sound like a blatant ad but here we are). This is beneficial if you want to use Docker for in house IP solution. It also handles trusted content signing, security scanning, and can even integrate with a company LDAP/AD setup (guess I'll start freelancing marketing copy). That said, you're really going to want to utilize this if you're a business entity and is probably a bit much for the average user/open source projects.

C. Dockerd With User Namespaces

This utilizes Linux namespace capabilities with regards to user IDs. Normally a container will run in a very privileged environment. This could potentially have an impact on the host system depending on how the container is configured. To avoid this attack surface, you can utilize user namespaces. This will map container bound users to unprivileged users on the host. Given the way users are mapped you can avoid potential attack vectors involving volumes bound to top level host directories (/ for example). The official Docker site has information on how to setup user namespaces with dockerd.

D. Hyper-V Isolation

Containers running on a Windows host can use Hyper-V isolation for their containers. This will essentially isolate the container from the host system reducing the attack vector. It primarily does this by wrapping it in a virtual machine. Running a container on a Windows host with Hyper-V isolation can be achieved through a simple command line parameter:

Docker run -it --isolation=hyperv microsoft/nanoserver cmd

E. Basic Virus Scan

A virus scan can be nice for identifying basic viruses and malware. ClamAV will be used for this. While it does detect some Linux variants it really shines for Windows based container guests. From a security point of view however, such scans are best run when the system is not live to avoid malicious pre-existing malicious software from interfering. Essentially we need a snapshot of the container's filesystem and run a scan against it. Using docker save you could get a snapshot of the image. Unfortunately, it uses docker's incremental storage called "layers" as a result. This means you have to use various JSON metadata files to figure out the exact order to apply layers in.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

F. ***Static Code Analysis***

This is completely out of bounds for container security, but just like package dependencies the container bound application code written can be vulnerable as well. Static code analysis essentially looks at the code and attempts to locate poor coding practices that could lead to a security vulnerability. If program is running in a PHP app for example, there are a number of tools available to check code.

The main approach to the Docker technology is virtualization techniques, which leads to a well secured platform and infrastructure to compute our necessity on the internet.

G. ***Docker Security Analysis***

Utmost challenge is security while running services in virtual environments, particularly in a multi-tenant cloud architecture. It is claimed that virtual machines enabled with hypervisor-based virtualization techniques are more secure because of an additional isolation layer added between host and applications. A program running within a VM is capable of communicating only with VMKernel and not with the host machine kernel. As a result of which application escalating must be out of a VM, bypassing the VMKernel and the hypervisor for attacking the host kernel.

H. ***Docker Hub***

Docker HUB [3] is a depository of public and private images through which customers can share their customized images. Customers can download images by searching their published images with Docker client. Moreover, users could check the integrity and authenticity of the downloaded images as Docker verifies and signs the images while submission to the hub.

I. ***Escaping technology in Docker***

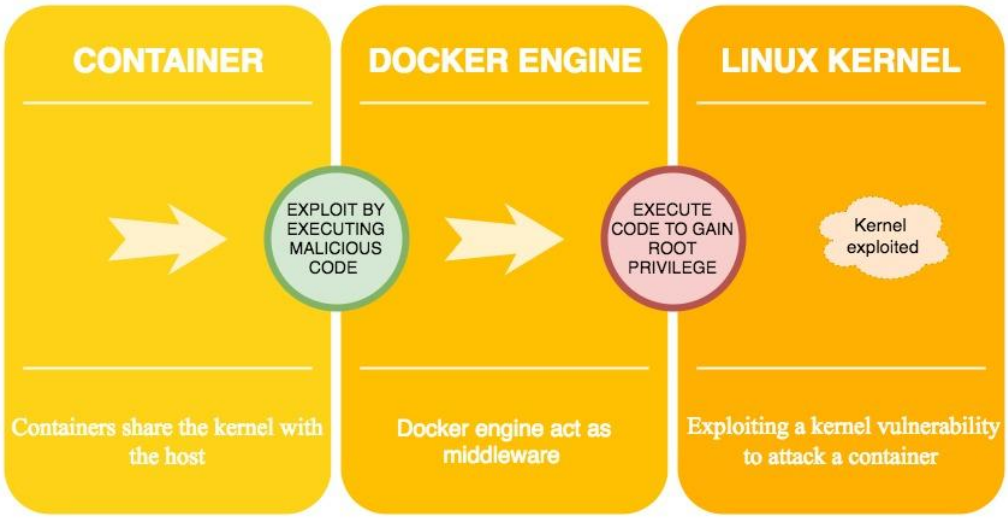


Figure 2: Kernel explosion by malicious code attack

Sharing of kernel with the host is done by Container , it is isolated from each other via CGroup mechanism and Namespace mechanism given by Linux kernel[10] . When malicious code(for exploiting kernel) is running inside the container , ultimately it will crash the entire host , hence escalating user privileges to root privileges for controlling the whole host machine. This process of exploiting the Linux kernel through running malicious code is shown in the above figure 2.

IV. CONCLUSION

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

By default docker containers are very secure, especially if someone is running their code as unauthorized user within the container. It can provide more security by adding extra layer through GRSEC, SELinux, AppArmor or many such hardening systems. containerization technique automates the applications of Docker. The host operating system is added with an extra layer which is of docker engine. The execution enhances in case of dockers as it has less resource overhead when compared with virtual machines. Increasing advent of uses of containers due to its lightweight implementation has lead to security issues, ultimately a solution is to be established to tackle this problem.

REFERENCES

[1] Combe, T., Martin, A., & Pietro, R. D. (2016). To Docker or Not to Docker: A Security Perspective.*IEEE Cloud Computing*,*3*(5), 54–62. doi: 10.1109/mcc.2016.100

[2]Freeman, A. (2017). Docker Images and Containers.*Essential Docker for ASP.NET Core MVC*, 39–67. doi: 10.1007/978-1-4842-2778-7_4

[3]Cook, J. (2017). Docker Hub.*Docker for Data Science*, 103–118. doi: 10.1007/978-1-4842-3012-1_6

[4]Jangla, K. (2018). Docker Basics.*Accelerating Development Velocity Using Docker*, 27–53. doi: 10.1007/978-1-4842-3936-0_4

[5] Bernstein, D.: Containers and cloud: "From lxc to docker to kubernetes". IEEE Cloud Computing (3), 81-84 (2014).

[6] https://www.aquasec.com/wiki/display/containers/Docker+Containers

[7] https://en.wikipedia.org/wiki/Docker_(software)

[8]https://docs.docker.com/engine/security/security/

[9]https://docs.docker.com/engine/security/trust/content_trust/

[10] Shu, R., Gu, X., & Enck, W. (2017). A Study of Security Vulnerabilities on Docker Hub.*Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy - CODASPY 17*. doi: 10.1145/3029806.3029832

[11] A Defense Method against Docker Escape Attack. (n.d.). Retrieved from https://dl.acm.org/citation.cfm?id=3058085.

[12] 5 security concerns when using Docker - OReilly Media. (n.d.). Retrieved from https://www.oreilly.com/ideas/five-security-concerns-when-using-docker.

[13] Docker Security. (n.d.). Retrieved from https://success.docker.com/training/courses/docker-security.

[14] Docker Reference Architecture: Securing Docker Enterprise and Security Best Practices. (n.d.). Retrieved from https://success.docker.com/article/security-best-practices.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

[15] Docker. (2019, November 5). docker/docker-bench-security. Retrieved from https://github.com/docker/docker-bench-security.

[16] Zeltser, L., & AuthorLenny Zeltser. (2015, December 1). Security Risks and Benefits of Docker Application Containers. Retrieved from https://zeltser.com/security-risks-and-benefits-of-docker-application/.

[17] Docker Security - 6 Ways to Secure Your Docker Containers. (n.d.). Retrieved from https://www.sumologic.com/blog/securing-docker-containers/.

[18] Enterprise Container Platform. (n.d.). Retrieved from https://www.docker.com/.

Copyright © 2019Authors