International Journal of Innovative Engineering Applications

Journal homepage: https://dergipark.org.tr/ijiea

SECURING VULNERABILITIES IN DOCKER IMAGES

Ahmet Efe[1], Ulaş Aslan[2], Aytekin Mutlu Kara[2]*

[1] *Internal Auditor at Ankara Development Agency, Turkey*
[2] *Yıldırım Beyazıt University, Department of Computer Science, Turkey*

Abstract

Review Paper

Docker is an alternative application development and publishing infrastructure tool to various virtualization environments such as Virtual box and the like. The most popular containerization platform is Docker which is the area where Docker images are run. Container is a lightweight contrasting option to full machine virtualization that includes exemplifying an application in a container with its own working condition. These two concepts, virtualization and containerization are competing in the cloud-based environments. When virtualization became the mainstream, VM security concerns was common. IT Security experts are discussing the potential weaknesses of a virtualized environment for a long time. In this paper, focusing on Docker container, its vulnerabilities and possible measurements against security concerns, we have provided information about assessment of risks and vulnerabilities of containerization and the main differences between these two concepts via vulnerability analysis.

Keywords: *Technology, Vulnerabilities, Dockers, Containers, Cloud computing security*

1 Introduction

Docker is a program that performs working framework level virtualization otherwise called containerization. Docker is essentially developed for Linux, where it utilizes the asset confinement properties of the Linux, for example, cgroups and portion namespaces, and an association skilled record framework to permit free "containers" keeping in mind the end goal to keep running inside a solitary Linux occurrence, staying away from the overhead of beginning and keeping up virtual machines (VMs). Container-based virtualization uses single kernel to run multiple instances on an operating system and virtualization layer runs as an application within the operating system. It is also called operating system virtualization and in this approach, the kernel of operating system runs on the hardware node with different isolated guest virtual machines (VMs) called containers. [1]

Docker is an open source virtualization platform for software developers and system builders. With Docker you can run Linux and Windows virtual containers (machines) on Linux, Windows and MacOSX. With this platform, you can easily install, test and deploy web systems. The Docker takes the image of the software's installed state (such as an .iso DVD image) and makes it available again. If you wish, you can create this image once and send it to the servers you want, if you do not, then you will create an image from scratch on each server. Each server can reconstruct the same image by looking at the instruction files called the Dockerfile. There is no need for manual intervention.

As is depicted in the Fig 1, the structure development at Docker systems, depends on the Moby Project upon which ConteinerD, LinuxKit and İnfraKit have been used. Therefore, the Docker is developed by the Moby Project.

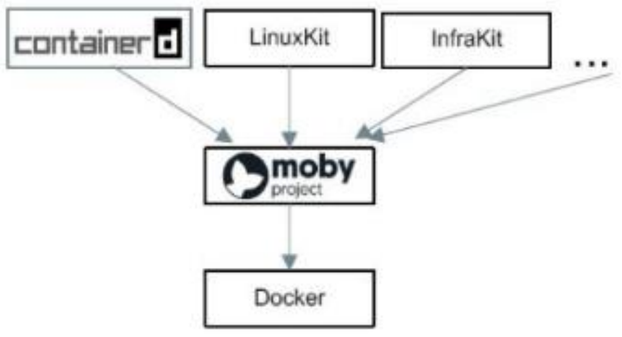

Figure 1 Development of Docker [2]

Docker is an apparatus that makes it simpler to make and run applications by utilizing container. Docker enables an engineer to bundle up an application with the greater part of the prerequisites, for example, libraries conditions and do everything as one bundle. By doing this, the engineer can rest guaranteed that the application will take a shot at some other Linux machine without depending of any altered settings that machine may have that could contrast from the machine utilized for composing and testing the code [3]. As is depicted in the Fig. 2, the usage of container technology will likely looks very different in ten years than they do now. Currently, containers are predominantly being used within the software development arena. Platforms such as Docker (the most mature of the container environments) are growing at exponential rates and taking nearly all market shares. Fig. 2 shows the rapid growth rate of Docker [4].

⁎ Corresponding author.
E-mail address: icsiacag@gmail.com (A. Efe) | ORCID Number : 0000-0002-2691-7517
Received 9 September 2019; Received in revised form 2 December 2019; Accepted 30 April 2020
2587-1943 | © 2020 IJIEA. All rights reserved. Doi: https://doi.org/10.46460/ijiea.617181

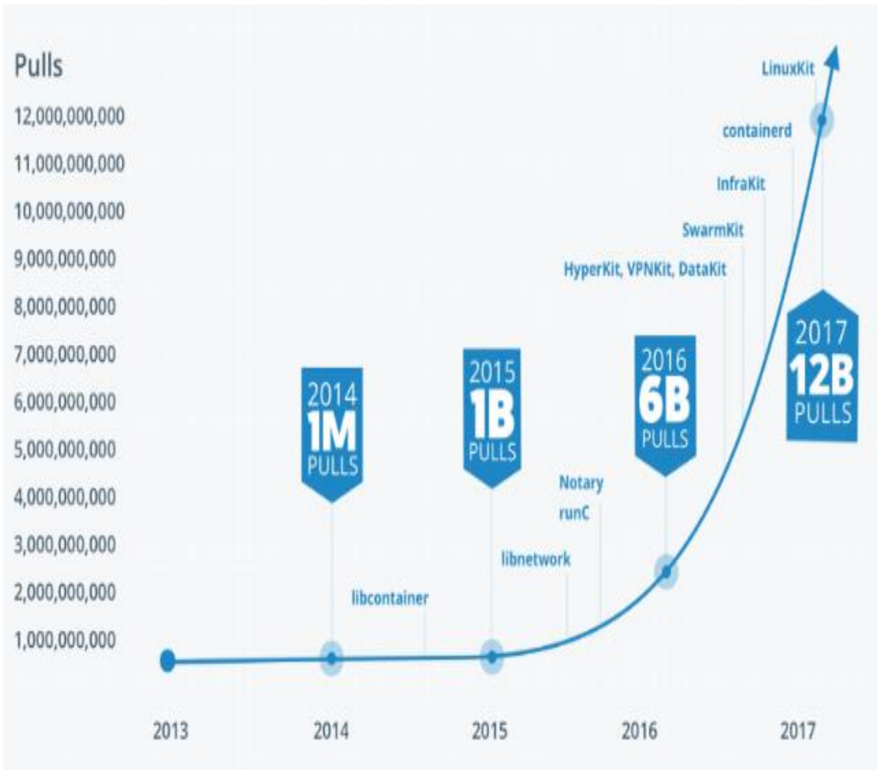

Figure 2 Growth rate of Docker [4]

2 Security Problems in the Docker Literature

Security studies on Docker platforms are still developing. In a study conducted by Bui, container-based virtualization is considered to be able to provide a more lightweight and efficient virtual environment, but not without security concerns. He analyzed the security level of Docker, a well-known representative of container-based approaches considering just two areas: the internal security of Docker, and how Docker interacts with the security features of the Linux kernel, such as SELinux and AppArmor, in order to harden the host system. He proposed that the security level of Docker containers could be increased if the operator runs them as "*non-privileged*" and enables additional hardening solutions in Linux kernel, such as AppArmor or SELinux [5].

In an analysis done by Combe *et al,* [6] it is revealed that Docker usages have security implications for both containers and their hosts, and repositories. Basics at al [7] have proposed that in order to increase Docker security and flexibility, an extension to the Dockerfile format to let image maintainers ship a specific SELinux policy for the processes that run in a Docker image, enhancing the security of containers is required.

Shu *et al*. [8], who have studied 356,218 docker images, made the following findings: (1) both official and community images contain more than 180 vulnerabilities on average when considering all versions; (2) many images have not been updated for hundreds of days; and (3) vulnerabilities commonly propagate from parent images to child images. They have proposed a scalable Docker Image Vulnerability Analysis (DIVA) framework for automatically discovering, downloading, and analyzing vulnerabilities in images from Docker Hub.

Manu *et al*, have provided unified security and privacy multilateral security architecture for cloud services stack, using key latest technology via LxC in general and Docker containers in specific to help assess the security design and architecture quality using multilateral security framework for Docker container. [9]. They have made a deep dive of the PAAS security and also try to analyze, compare and contrast the PAAS Docker container security, with other container technologies, and also with Virtual machine security with and without Hypervisor, and current security level of Dockers container[10]. Chelladhurai *at al*, have proposed security algorithms and methods to address DoS attacks related issues in the Docker container technology in their study [11].

Gao *et al*, discussed the root causes of the containers' information leakages and propose a two-stage defense approach [12]. According to Jian, who has searched defense methods against escape attack, Docker is faced with the risk of attacks that exploit kernel vulnerability by malicious users, once the exploit program in the container launches an effective escape attack can gain root privilege of the host, which will affect the reliability of other containers and the entire system [13].

3 Problem Definition and Methodology

There are many advantages of Docker (or container) technologies. However as Cyber security is a top concern in nearly every major industry, in Docker technology it is also the most controversial topic. Millions of web applications which run on Docker platform are open for cyber-attack. Since the time of Docker’s release in 2013, several vulnerabilities have been discovered.

We used Docker and related technologies not only development environment but also production for many projects. We have faced with some problems and search for suitable solutions. All problems and solutions were documented and published in the company for the future use. The recommendations in this paper were collected from these experiments.

This paper focuses on the security threats of web applications. We explore current situation and several vulnerabilities. We suggest a new method to help increase security for the most common vulnerabilities. In order to research and suggest new methods, firstly we need to clarify the problems so that we can work on a much more specific area. In the below you can see some problems that we need to find a way to improve;

- How to analyze the security vulnerabilities of Docker system?
- What are the most commonly found vulnerabilities in Docker systems?
- How to detect vulnerabilities of a Docker system?
- How to detect and prevent DoS attacks on Docker?
- How to detect vulnerabilities in Docker images?

In this research, there are many methodologies combined to give more clear and whole explanations about the subject. In this paper, Literature review is the most crucial methodology. Because we have tried to see as much as many papers about this subject to observe what is done and what can be done about Docker Security. We try to give descriptions about docker and docker security. This paper is also a qualitative research paper due to the fact that we used the knowledge based on experience and observation that is obtained in the past researches that we review in order to make assumptions.

4 Working Structure of Docker Dynamics and Its Benefits

Developers can bundle their development environments into these containers with the necessary configurations and transfer them to the desired environment. This structure inspired by sea transport removes many problems from the point of view of both developers and system administrators. Docker leverages the resource isolation features of Linux to create a segmented, virtual environment that applications can operate on it. This is conceptually like using a tool like "*chroot*" to create an isolated, protected filesystem, although the segmentation under Docker extends beyond filesystem isolation alone to circumscribe resources. Below, there are the three main functions of the Docker platform:

- *Build*—Docker allows you to compose your application from microservices, without worrying about inconsistencies between development and production environments, and without locking into any platform or language.
- *Ship*—Docker provides developers to setup the whole software development steps such as development, testing, distribution and with a consistent user interface.
- *Run*—Docker provides developers the ability to deploy scalable services securely and reliably on a wide variety of platforms.

Docker has three main parts:

- *Docker Engine*; Open-source containerization platform
- *Docker Cloud;* including Docker Hub; Software as a Service (SaaS) platform for sharing and managing Docker containers
- *Docker Datacenter*; On-premise solution for sharing and managing Docker containers and Docker-containerized applications.

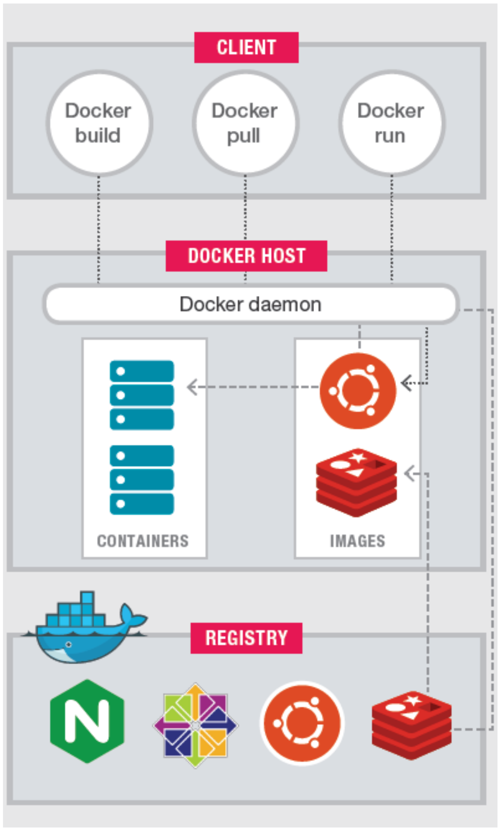

Figure 3. Docker Architecture [15]

Docker uses client-server architecture. The Docker client tells the Docker daemon, which is on the host operating system and builds, runs and distributes containers. Docker users communicate with the daemon through the client, which is the Docker binary. In the figure, Docker architecture is shown. [16]

Users are always expecting applications to available, elastic, scalable and interoperable at any time. Docker provides these features to users. Main benefits of Dockers are;[17]

Docker containers are negligible: Containers has one or only a couple of running procedures. Less programming implies littler likelihood of being influenced by helplessness.

Docker containers are task-particular: There is a pre-meaning of what precisely ought to keep running in the containers, way of the information indexes, required open ports, daemon arrangements, mount focuses, and so forth. Any security-related oddity is less demanding to identify than in other multi-reason frameworks.

Docker containers are segregated: It is disengaged both from the facilitating framework and from different containers.

Docker containers are reproducible: Due to their explanatory form frameworks any administrator can without much of a stretch assess how the holder is manufactured and completely see each progression. Based on various evaluation and experiments, Docker provides fast deployment, small footprint and good performance which make it potentially a viable Edge Computing platform [18].

Let us express the difference between other virtualization technologies and Docker in a simple example. For example; An Ubuntu operating system will be installed. To perform this operation on other virtualization technologies, an Ubuntu version must have an ISO file. In the Docker structure, it is enough to write the command:

docker pull ubuntu

This command will install all versions of Ubuntu on the Docker repository. In other virtualization technologies, it is necessary to download and install the ISO file for each version, which causes more time loss. Docker operations are carried out via the terminal. Some interfaces have been developed to make our operations even easier with Docker. Using these interfaces we can manage our containers even more easily [19].

5 Comparison with VM Structures

Containers are less adaptable contrasted with VM's, MS windows, can't be keep running on a Linux OS, Containers are less secure contrast with VM, because of containers keep running in tight coupling with have OS and keeps running over it, if the container is traded off then hacker/assailant can get finish access to have OS and assets, additionally it is perplexing to introduce, setup the earth, oversee, direct, and automate the framework utilizing Container innovation. Container base virtualization utilizes have OS level virtualization, so framework security is of most extreme significance to screen utilizing the multilateral adjusted security Containers are executed in client space on peak of an OS piece.

- *Virtual Machines*

  Each virtual machine includes the application, the necessary binaries, libraries, and an entire guest operating system — all of which may be tens of GBs
- *Containers*

  Containers include the application and all of its dependencies but share the kernel with other containers. They run as an isolated process in user space on the host operating system. They are also not tied to any specific infrastructure — Docker containers run on any computer, on any infrastructure and in any cloud [16].

6 Docker Vulnerabilities

Since Docker's release in 2013, there has been discovered several vulnerabilities that could lead to privilege escalation code execution. Here in the below there are top five vulnerabilities that found different versions of Docker.

1. *CVE-2014-9357*

This vulnerability was identified in Docker Version 1.3.2 which allowed the execution of arbitrary code with root privilege. During the decompression of LZMA (.xz) archives, there was privilege escalation vulnerability. This vulnerability was patched by version 1.3.3.*26* **(CVSS Score: 10.0)** [23]

2. *CVE-2014-6407*

This vulnerability was identified in Docker Version 1.3.1, which allowed privilege escalation through symlink, and hard link traversals found in the Docker's image extraction. This vulnerability was patched by version 1.3.2.*27* **(CVSS Score: 7.5)** [24].

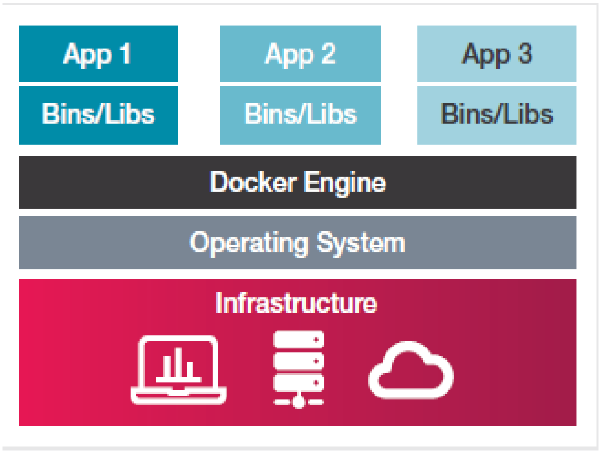

Figure 4 Container Architecture [17]

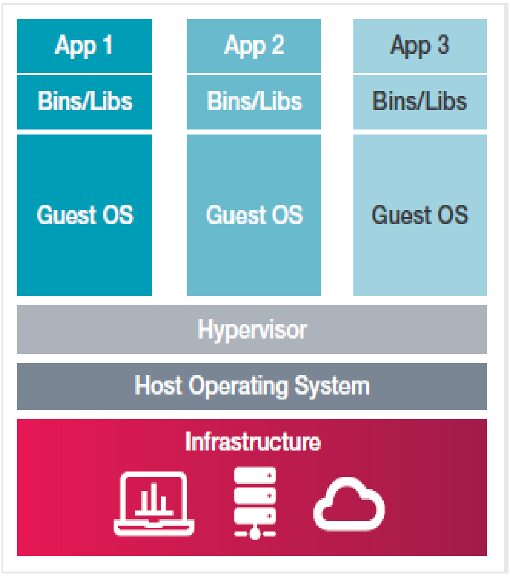

Figure 5 VM Architecture [15]

3. *CVE-2014-3630*

This vulnerability was identified in Docker Version 1.6.0 and allowed attackers to bypass security due to weak permissions on the /proc paths. Attackers could access sensitive information and perform unauthorized actions due to the security bypass. This vulnerability was patched by version 1.6.1.*28* **(CVSS Score: 7.2)** [25]

4. *CVE-2014-3499*

This vulnerability was identified in Docker 1.0.0 which indicated that Docker was using "world-readable" and "world-writable" permissions on the management socket. This vulnerability allowed local users to gain root privileges to the local machine. This vulnerability was patched by version 1.0.1.*29* (CVSS Score: 7.2) [26]

5. CVE-2015-3627

This vulnerability was identified in Docker 1.6 and the Libcontainer version 1.6.0 that allowed a “mount namespace breakout” when a container was respawned. This function created an exploit to allow codes to escape the container. Through this exploit, attackers can create a privilege escalation. This vulnerability was patched in Docker 1.6.1. (CVSS Score 7.2) [27]

There are some specific parts of Docker hub contains both official and community images. Official Docker based architectures, which are more prone to attacks. Some of them are related to the images. Docker distributes applications in the form of images. Each image contains the target application software as well as its supporting libraries and configuration files. Repositories contain public, certified images from vendors. In contrast, any user or organization can create community repositories. [15] Some further vulnerability is related to the Docker architecture. [17]

If any attacker compromises host system, the container isolation and security safeguards will not make much of a difference. Besides, containers run on top of the host kernel by design. The kernel is shared among all containers and the host, magnifying the importance of any vulnerabilities present in the kernel. Should a container cause a kernel panic, it will take down the whole host [14].

An attacker who gains access to a container should not be able to gain access to other containers or the host [14.] The “container breakout” term is used to denote that the Docker container has bypassed isolation checks, accessing sensitive information from the host or gaining additional privileges.

Containers are much more numerous than virtual machines on average, they are lightweight, and you can spawn big clusters of them on modest hardware, but it implies that many software entities are competing for the host resources. If one container can monopolize access to certain resources, it can starve out other containers on the host, resulting in a denial-of-service (DoS), whereby legitimate users are unable to access part or all of the system. [14]

Docker’s popularity is since, with containers, anybody can bundle code and conditions into a picture and distribute effortlessly to a registry whenever anywhere. From there, anybody can download the picture and run containers from the picture. This has brought convey ability of code crosswise over groups and stimulated the application lifecycle.

However, containers can incidentally uncover vulnerabilities, if the required precautions are not taken. This is particularly obvious when working with container images that are shared amongst clients and associations. Container images are downloaded from registries like Docker Hub or outsider registries. There are official and unauthorized repositories. These registries contain a large library of images. These registries are mainly user created and uncontrolled, this leads to low frequency of updates which results in vulnerabilities in the images.

A few examinations demonstrate that over 30% of authority repositories contain images that are insecure to an assortment of security assaults [21].

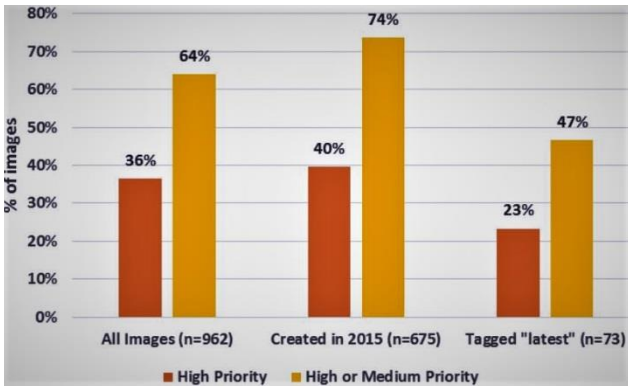

Figure 6 Official Image Vulnerabilities [21]

Fig 6 demonstrates the primary outcomes got by breaking down every official image from Docker Hub. In excess of 33% of all images have high need vulnerabilities and near 66% have high or medium need vulnerabilities [21].

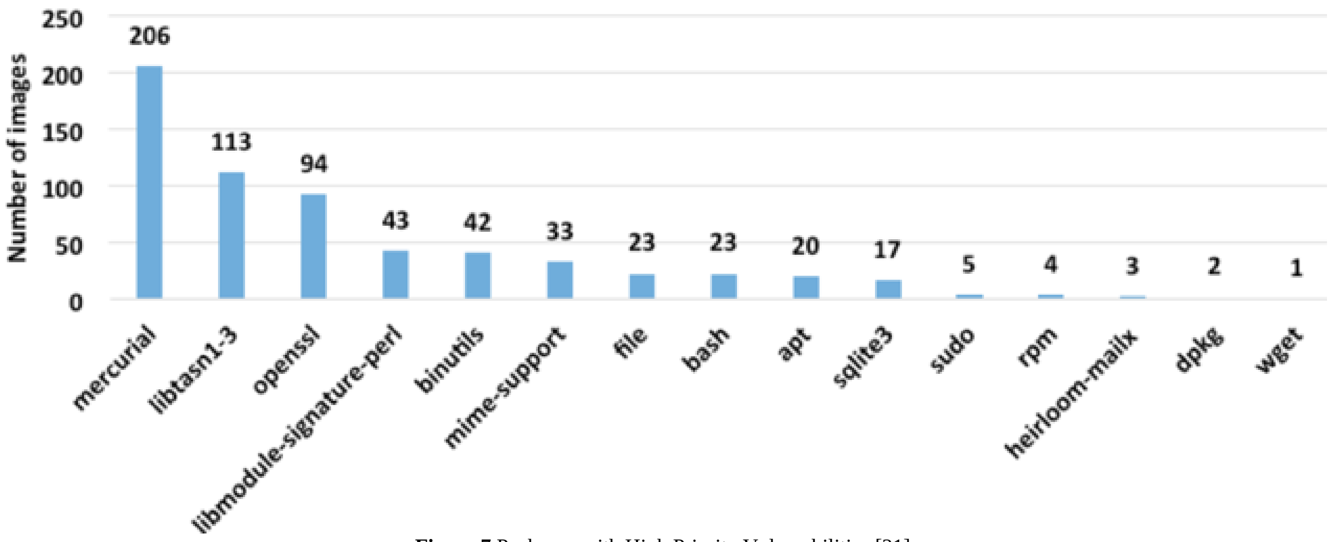

Figure 7 Packages with High Priority Vulnerabilities [21]

The as of late discharged powerlessness in inconsistent is available in a huge part of images (~20%). Prominent OpenSSL vulnerabilities, for example, Heartbleed and Poodle are available in near 10% of authority Docker Hub images [21].

Fig. 8 demonstrates the fundamental outcomes after breaking down general images. Generally speaking, the level of vulnerabilities is essentially higher than that of Official images [21].

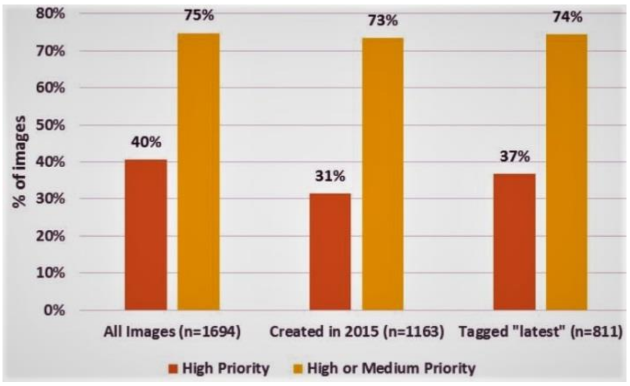

Figure 8 General Image Vulnerabilities [21]

According to a security analysis report released by a researcher from Federacy, 24% of images in public repositories were found to have significant vulnerabilities, with around 11% among them rated high, 13% as moderate and the rest as potentially vulnerable. [22]

Docker Hub, one of the cloud-based Docker container libraries, has been compromised by data spoofing by an unknown attacker by accessing the company's only Hub database in 2019. Docker Hub is an online repository service where users and organizations can create, test, store and distribute Docker container images both publicly and privately. It was reported that sensitive data were generated for approximately 190,000 Hub users, including the Github and Bitbucket markers for a small percentage of the affected users, as well as user names and hash codes for Docker repositories. Docker Hub has sent information e-mails to affected users, providing information about the security event and suggesting that they change their passwords to their online accounts using the same password for Docker Hub. Docker Hub said that the company will continue to investigate security breaches and will share more information whenever possible. The company is also working to improve its overall security processes and review its policies following violations. [30]

7 Possible Measures and Precautions Against Risks and Threats

Some security analysis tools can be used to inspect public Docker images. Security audit, container image verification, runtime protection, automated policy learning or intrusion prevention capabilities can be tested using these tools. Anchore Navigator, AppArmor, AquaSec, BlackDuck Docker, Cavirin, Cilium are some tools that can be used to analyze container security.

7.1. Checking the source of images

Container images are downloaded from registries like Docker Hub or outsider registries like Quay. These registries have Container images from associations and people alike. The same number of as official stores from IT merchants, there are numerous unapproved repositories too. Over the application lifecycle, designers, QA and IT will download numerous images for various sorts of purposes. It's vital to screen these images and play out a few controls previously they are introduced.

In order to do this, Docker Content Trust can be enabled, which integrates with third-party registries to verify digital signatures for container images that are downloaded from them. This helps developers to make whitelist of official repositories from authorized, trusted sources.

In the event that it is expected to work with unsubstantiated images from accomplices and sellers, for instance, it could be considered updating your images checking to heartier container security devices like Twist bolt. It filters images, as well as gives you a chance to set up custom cautions at whatever point anybody endeavors to introduce suspicious images.

7.2. Implementing powerful access controls

The wellspring of container images and use of official images can leave images traded off. Along these lines, get to control is to a great degree critical for holder pictures.

Normally, all clients are doled out root benefits inside a container. In any case, this is not the best practice from the security perspective. At whatever point a client is made, you have to change their entrance level to non-root. Certain clients will in the long run require root access to finish certain undertakings, however these special cases ought to be made just inside those compartments that play out the assignment, and just on the off chance that it is important. This undertaking driven access control guarantees that regardless of whether one client account is traded off, assailants cannot make much harm on whatever is left of the framework. It is unrealistic to physically change the status of clients for each container unfailingly. This errand should be robotized. A stage like Twistlock empowers role-based access control (RBAC) for pictures, and gives you a chance to obtain benefits to clients in light of their activity work. You can arrange RBAC in view of complex decides and guarantee that all clients have the vital benefits to do their undertakings nothing less, and nothing more.

7.3. Keeping containers lightweight

Engineers are attracted in to containers because of the lighter structure contrasted with virtual machines (VMs). When running containers, it's conceivable to stack an excessive number of bundles on a container with the goal that it progresses toward becoming enlarged to in excess of 100 MB. The perfect container size ought to be only several MBs.

While choosing an OS for your image's base layer, search for a moderate choice. There are two or three great choices like BusyBox, Alpine Linux, and RancherOS. Furthermore, introduce just the bundles that are required for a container to play out its undertaking. This enhances the execution of containers, and critically, diminishes the assault surface territory.

7.4. Keeping images solid

Once you have taken after every single best practice to set up your images the correct way, it is essential to screen their wellbeing amid runtime. This requires routine "wellbeing checks" on the holders. On the off chance that Docker Engine discovers containers that are not working, it can naturally supplant them. Along these lines, you can keep the framework sound regardless of whether singular containers are observed to be helpless.

A critical practice to guarantee the great strength of your containers is to keep container images refreshed with the most recent form and apply security patches to them as often as possible. You should have the capacity to filter the images amid runtime to discover vulnerabilities and fix them expeditiously.

Identifying vulnerabilities is not a simple undertaking, as your framework could run a huge number of containers. At this scale, you require a risk discovery instrument like Twistlock that screens container runtime with the assistance of machine learning calculations. It can spot concerning examples and alarm you on their effect. This activity of finding the needle in the pile is not conceivable through manual poring over of log information and measurements—it takes wise calculations and a cutting-edge risk discovery stage like Twistlock.

7.5. Handling secret information with mind

In spite of doling out read-just access to clients, despite everything you have to watch what information stored in your containers. For instance, you ought to never store confidential information like passwords, tokens, keys, and private client data inside docker documents. Regardless of whether erased later, this information can be recovered from the image's history. Rather, you should utilize the privileged insights administration highlight that accompanies both Kubernetes and Docker Swarm. Every one of them have solid defaults to guarantee security credentials are appropriately scrambled, put away in an encoded design, and when recovered, can be decoded just by approved clients.

Holder images are likely the best time some portion of the Docker encounter. However, they can likewise be the most perilous from a security outlook. By understanding the different subtleties to Docker image security, you can guarantee your cloud-local applications are considerably more secure than your heritage applications ever were. [20]

8 Conclusions

Development environments can vary from project to project. Even the installations of the same project with different customers contain different components. It may take a lot of time to install the same environment in the developer environment in order to solve the problems of the customers. Some software houses maintain developer environments on virtual servers in order to repeat installations in developer environments in different versions with different customers. Considering that a virtual machine is around 25 GB on average, if a single project is installed on 5 customers, 125 GB of space will be required on each developer's computer. Even if we ignore the problem created by the provision of virtual servers in resource usage, in every projects that are actively developing, web server changes, database schema or reference data changes, etc., the distribution of the master virtual machine in binary format to all developers, and if any, it requires the developers to apply their own customizations to these machines again and again. Another development environment problem is the time spent in setting up the computers of newcomers with the necessary tools in development environments where no virtual server is used. This is usually done by an experienced senior member of the team, in which case a period of time that both the experienced member of the team and the new member can effectively use is lost.

Cloud computing makes extensive use of virtual machines because they permit workloads to be isolated from one another and for the resource usage to be somewhat easily controlled [28]. Docker is an open platform for developers and system administrators to build, ship, and run distributed applications using Docker Engine, which is a portable, lightweight runtime and packaging tool, and Docker Hub, which is a cloud service for sharing applications and automating workflows. The main advantage is that, Docker can get code tested and deployed into production as fast as possible [29].

Since the Docker was introduced to the market, a series of updates and changes have been made to improve its functionality and security. However, as any programmer knows, there is no safe platform and for the Docker there is also no exception. Also, most of the time, security is affected by how a user interacts with the Docker, which faces several problems with human errors. In a study of more than 700 companies from many developed countries conducted by Cloud Foundry in 2016, half of the companies used the container technology. Of these, 64% plan to expand the use of container technology. When we look at this rapid increase in container usage, new security problems are constantly emerging. The more a system is used by users, the more it becomes a strategic target for attackers. The Docker seems to be the most widely used container technology today, and as with many other similar platforms, many have protocol-specific security issues. A list of weaknesses previously found in the Docker can be viewed on CVE-List and information about these security issues can be found.

Cyber Security is at the forefront of today's data centers and for good reason. Even a very small-scale violation can lead to very large damage. For this reason, organizations need to be particularly sensitive to security implications and new security vulnerabilities. When virtualization became the mainstream, VM security was common. IT Security experts are discussing the potential weaknesses of a virtualized environment for a long time.

Perhaps the worst case scenario is "*VM Escape*". This term is used to describe a situation in which an attacker can endanger a guest virtual machine and be "escaped" from within the virtual machine and access other main operations on the hypervisor. Virtualization leaders have made great efforts to ensure that such security issues are addressed, and the VM escape is rejected as a real threat. A weakness called "*CEN-2015-*

3456" and "*VENOM*", and by exploiting some of the vulnerable codes in QEMU's virtual floppy drive, the attackers potentially had the advantage to exploit other virtual machines running on the host computer. For this reason, the Docker can help with container security, even if it is not a virtualization service.

From Docker 1.8 version, a new security feature called "*Docker Content Trust*" has been introduced. This feature allows verifying the authenticity, integrity and releasing date of all Docker images in the "Docker Hub" repository. This content assurance is not enabled by default. When enabled, Docker cannot download any of the unsigned images and therefore becomes more secure.

To enable this feature:

sudo export DOCKER_CONTENT_TRUST = 1.

The Docker will inform when trying to download an image that is not already signed. By default, there is no resource constraint in a container, and the host can use most of the given resource to the extent allowed. The Docker provides a way to control the amount of memory, CPU, or block that the container run command can use. It is possible that when a container runs into trouble and starts consuming all of the host's resources, bad results may occur on a scenario. The resource limits for containers can be set from the "*docker run*" command for further security considerations.

References

[1] Singh, S., Singh, N. "Containers & Docker: Emerging roles & future of Cloud technology", 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 16850813 DOI: 10.1109/ICATCCT.2016.7912109, IEEE, 21-23 July 2016.

[2] Morgan, T. P. "Docker Completes Its Platform Wıth Dıy Lınux" 2017.

[3] What is Docker? (n.d.). Retrieved from URL: https://opensource.com/resources/what-docker

[4] Docker. (n.d.). Retrieved from Docker Website URL: https://www.docker.com/

[5] Bui, T., "Analysis of Docker Security" Aalto University T-110.5291 Seminar on Network Security, 2014.

[6] Combe, T., Martin, A., Pietro, R.D. "To Docker or not to Docker: a security perspective" IEEE Cloud Computing, 2016.

[7] Bacis E., Mutti, S. Capelli, S. Paraboschi, S. "DockerPolicyModules: Mandatory Access Control for Docker containers" IEEE Publishing, 2015.

[8] Shu, R., Gu X., Enck, W., "A Study of Security Vulnerabilities on Docker Hub" CODASPY '17 Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017.

[9] Manu, A R, Patel, J. K., Akhtar S., Agrawal, V. K., Murthy K N, "Docker container security via heuristics-based multilateral security-conceptual and pragmatic study", IEEE Publishing, 2016.

[10] Manu, A R, Patel, J. K., Akhtar S., Agrawal, V. K., Murthy K N, "A study, analysis and deep dive on cloud PAAS security in terms of Docker container security", International Conference on Circuit, Power and Computing Technologies (ICCPCT), 2016.

[11] Chelladhurai, J., Chelliah, P., Kumar, S. A., "Securing Docker Containers from Denial of Service (DoS) Attacks" International Conference on Services Computing (SCC), 2016.

[12] Gao X., Gu, Z. Kayaalp, M., Pendarakis, D., Wang, H., "ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds" 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2017.

[13] Jian, Z., Chen, L.,"A Defense Method against Docker Escape Attack" ICCSP '17 Proceedings of the 2017 International Conference on Cryptography, Security and Privacy, 2016.

[14] Five Security concerns when using docker. (n.d.). Retrieved from Oreilly URL: https://www.oreilly.com/ideas/five-security-concerns-when-using-docker

[15] Rui, S., Xiaohui, G., & William, E. A Study of Security Vulnerabilities on Docker Hub. CODASPY '17 Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (pp. 269-280). Scottsdale, Arizona, USA: ACM. March 22 - 24, 2017.

[16] ISACA, Understanding the Enterprise Advantages of Application Containerization. (n.d.). USA. 2016.

[17] Docker Security Vulnerabilities. (n.d.). Retrieved from Sysdig URL: https://sysdig.com/blog/7-docker-security-vulnerabilities/

[18] Ismail B. I. et al., "Evaluation of Docker as Edge computing platform," 2015 IEEE Conference on Open Systems (ICOS), Bandar Melaka, 2015, pp. 130-135.doi: 10.1109/ICOS.2015.7377291

[19] Ayaz Ö., Aydın G., "Uygulama Sanallaştırmada Yeni Bir Yaklaşım: Docker", URL: https://ab.org.tr/ab15/bildiri/312.pdf

[20] Twistlock. (n.d.). Retrieved from 5 Best Practices to Container Image Security: https://www.twistlock.com/2017/08/31/container-image-security-best-practices/

[21] Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities. (n.d.). Retrieved from banyanops URL: https://blog.banyansecurity.io/blog/over-30-of-official-images-in-docker-hub-contain-high-priority-security-vulnerabilities

[22] Federacy. (n.d.). Retrieved from Container Scanning Specification URL: https://www.federacy.org/docker_image_vulnerabilities

[23] Docker 1.3.3 Security Advisor. Retrieved from Security focus Website URL: https://www.securityfocus.com/archive/1/archive/1/534215/100/0/threaded

[24] Bug 1167505 - (CVE-2014-6407) CVE-2014-6407 docker: symbolic and hardlink issues leading to privilege escalation. Retrieved from bugzilla.redhat Website URL: https://bugzilla.redhat.com/show_bug.cgi?id=1167505

[25] SUSE-SU-2015:0984-1: moderate: Security update for docker.Retrieved from Suse Website: http://lists.suse.com/pipermail/sle-security-updates/2015-June/001419.html

[26] RHSA-2014:0820 - Security Advisory. Retrieved from redhat Website URL: https://access.redhat.com/errata/RHSA-2014:0820

[27] Docker 1.6.1 - Security Advisory [150507]. Retrieved from Seclists Website URL: http://seclists.org/fulldisclosure/2015/May/28https://www.cvedetails.com/vulnerability-list/vendor_id-13534/product_id- 28125/Docker-Docker.html

[28] Felter, W. Ferreira, Rajamony A. R. and Rubio, J. "An updated performance comparison of virtual machines and Linux containers," 2015 IEEE International Symposium on Performance Analysis of Systems and

Software (ISPASS), Philadelphia, PA, 2015, pp. 171-172. doi: 10.1109/ISPASS.2015.7095802

[29] Preeth E N, F. J. P. Mulerickal, B. Paul and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," 2015 International Conference on Control Communication & Computing India (ICCC), Trivandrum, pp. 697-700.doi: 10.1109/ICCC.2015.7432984, 2015.

[30] Wei, W. “Docker Hub Suffers a Data Breach, Asks Users to Reset Password” 2019.