National College of Ireland

Securing Docker Image against Privilege Escalation and Misconfiguration

MSc Research Project
MSc in Cloud Computing

Student ID: 19189257

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Ameya Patil
Student ID:	19189257
Programme:	MSc in Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	17/12/2020
Project Title:	Securing Docker Image against Privilege Escalation and Misconfiguration
Word Count:	7077
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	15th December 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Securing Docker Image against Privilege Escalation and Misconfiguration

Ameya Patil
19189257
MSc in Cloud Computing

Abstract

Containerization technology is one of the key pillars of information technology (IT) infrastructure. Docker is widely used due to its mass popularity. Docker as a technology is vulnerable in terms of IT security. This research provides evidence of the same and also provides importance of few best practise's mandatory for implementation. Further there are two experiments performed to which provides a glimpse on dependency of official images imported/PULLED from Dockerhub. An architecture is proposed in this research which provides an extra layer of security pre deployments of any images. Anchore Engine is the backbone of the proposed architecture. A complete analysis on the Docker images on Dockerhub is provided along with evidence. Hundred random images are tested on the proposed architecture where fifty of them are official images from Dockerhub and rest fifty are the non-verified ones. A live privilege escalation attack has been performed where an Ubuntu local host is exploited on the basis of misconfigurations. Towards the end of the research, security efficiency of docker as a technology is provided on the basis of the experiments conducted.

1 Introduction

The requirement of containerization has been noticed over the last decade with upgrading technology over virtualization. Container based containerization has gained popularity due to the benefits it provides over traditional virtual machines via hypervisors. Containers are extremely light-weighted and possess a minimum boot-up time requirement in comparison to virtual machines Aparna Tomar and Bisht (2020)MSollfrank et al. (2019)Loukidis-Andreou et al. (2018). Being light weighted, containers leverage the efficiency requirement and enable developers to smoothly build and deploy applications in Continuous Integration/Continuous Delivery model (CI/CD). Docker is among the market leaders in the domain of containerization technologies. It manages the optimization of resources and adds-on to the tasks like rapid design/development as well as testing and deployments Kelly Brady and Coffman (2020). Being an elite microservice technology, it inherits multiple perks but comes along with cyber security glitches. Isolation is among the major concerns in all technological areas and sectors. In the cases of containerization, it is not as promising as in the cases of traditional virtualization based on hypervisors. A virtual machine gathers its required application related resources from the VM kernel whereas in case of containers the communication related to resources and its and dependencies is held between the container and the host kernel. This describes the security

concerns by itself as the number of barriers attacker needs to breach are more in terms of virtual machines whereas containers are directly communicating with host kernel which makes host system vulnerable Aparna Tomar and Bisht (2020)Mattetti et al. (2015).

A parent Docker image can be accessed (PULL and PUSH) via public repositories. The most common one is known as Dockerhub. It is not guaranteed that the sources images stored on Dockerhub are completely legitimate and safe to use or vulnerability-free Katrine Wist and Gligoroski (2020). In a consciously motivated state, a personal may create an image with vulnerabilities or may append a misconfigured/malicious code on purpose. Extraction of such an image from the repository can grant easy backdoor access to the attacker via Trojan or similar viruses Handong Cui and Huang (2019). Presence of any such backdoors within running containers may grant multi-level/layer access of the container to the attacker leading to a promising case of privilege escalation. If the root of any machine is compromised via one of the containers, the attacker can potentially breach the docker engine server responsible to manage the cluster of existing containers TianshuoYang et al. (2019). This research aims towards a study regarding cases of Privilege Escalations and detection of misconfigured (vulnerable) images leading to vulnerable linux environment.

Docker official/verified images on Docker Hub repository have large number of vulnerabilities, details of which are present in the evaluation (6) section of this research. There is a wide presence of static vulnerability assessment tools in the market. Few of the popular ones are Clair and Anchore Engine. They assists in the assessment of docker image with regards to known CVE vulnerabilities. Anchore Engine is the central system used in this research which is capable of providing a static vulnerability assessment results. It can also provide extremely efficient policy enforcement engine which can be customised as per the compliance requirement.

Privilege escalation attacks can be successfully carried out by either exploiting vulnerabilities at application level or kernel level of any Linux server/system LifengWei et al. (2016). If a container deployed on a linux host is compromised by elevated privileges, that may lead to the entire docker engine server and the complete host system being compromised. This research proves the presence of such an exploits which is a prominent case of privilege escalation.

At the end of the research, we would be able to filter out the unhealthy images prior to their deployments as running containers. Also this research introduces a presence of a misconfiguration schema which potentially leads to a privilege escalation attack on host system while deployment of containers.

1.1 Research Question

- How can the docker images be more secure and dependable in terms of security threats like privilege escalations caused due to misconfiguration?
- How to avoid unhealthy docker images containing vulnerabilities from public repositories like DockerHub?

Main motives beneath this research lies within the area of container security. Primal aim being securing docker containers against privilege escalation attacks majorly caused due to misconfigurations of docker images. Docker Images contain high vulnerability potential if not designed using appropriate protocols. A misconfigured image may grant any external identity elevated privileges and permissions. This research focuses on upgrading

docker securing using the proposed Automatic Image Analyzer (AIA). Static vulnerability assessment would be conducted prior to deployment of images into the production environment. AIA functions majorly based on Anchore Engine. Anchore Engine is a static vulnerability assessment tool which is based on CVE (Common Vulnerabilities and Exposures) scores. Post successful completion of this research deployments into production environments would completely base on the CVE (Common Vulnerabilities and Exposures) scores.

2 Related Work

This chapter of related work has been bifurcated into multiple sections. Research has been conducted completely in the domain of containerisation. The first section of the literature review contains overall study on architecture of Docker. Main emphasis is towards C-groups and Namespaces which are responsible for overall security concerns and isolation in terms of docker. Second section contains information related the research conducted with respect to Linux privilege escalation attacks and the last section is about overall summary of the research along with the limitations and research gaps.

2.1 Docker Namespaces and Cgroups

Containerisation technologies have been available via Linux since a long time. Docker as a technology came into existence by 2013 after which it dominated the global markets in terms of containerization and deployments. In practical scenarios, use case of containers over virtual machines is dependent on multiple factors related to requirements. Docker always function on the terms of a shared responsibility model with its host, where it shares its resources with the kernel responsible for hosting it. This phenomena leads to a security risk where securing host and docker as a platform is equally important from the standpoint of cyber security Aparna Tomar and Bisht (2020)Combe et al. (2016). There are two features of kernel which are responsible for overall the security and isolation of container technologies, namely Cgroups and Namespaces.

The first feature is known as control groups also known as Cgroups. Cgroups are responsible for limitation of the resources for containers allocated. These resources are inclusive of system memory, CPU run-time, network bandwidth and input/output. Cgroups are the barriers to the overuse of resources by a single container. Sultan et al. (2019) have conducted prominent research with regards to Cgroups as well as namespaces. Control groups can hold various system data.

Second Functionality of kernel can be referred as Namespaces. Isolation of resources over the containers are among the responsibilities of Namespaces. All the functionalities are isolated via namespaces, beginning with file systems to processes and hostnames. Container binding is among the responsibilities of namespaces which includes isolation of processes related to containers from adjacent containers or host OS. Namespaces are the functionalities responsible for granting private isolated root directories for deployed containers. Various types of namespaces available in Linux are listed in the Table 1 Sultan et al. (2019)Gao et al. (2017).

Situations where namespaces are compromised commonly arise due to elevated permissions or privilege escalations whereas denial of service attacks are caused due to breaches in Cgroups Chelladhurai et al. (2016)Aparna Tomar and Bisht (2020).

Table 1: Types of Namespaces

Namespaces	**Isolates**
Mount Namespace	Container mounting points
Network Namespace	Network devices information, Stacks and Ports
UTS Namespace	Hostname and NIS (Network Information Service)
	domain name
User Namespace	User and Group IDs
Process ID Namespace	Processes
IPC Namespace	Access to IPC resources

2.1.1 Docker Images

Docker images are templates which contain configuration information required to deploy a container. Docker engine plays an important role in management of the containers over a host server as well as it is also responsible for deployment of containers via docker images. Docker images consist of data required by middleware, network configuration, application, OS files and libraries. Authors of research Kwon and Lee (2020) and Bui (2015) have shared complete overview on docker images as well as docker engine. There are several ways of building up an docker image, two of the most used methods include interactive method and docker file method. Interactive method grants the administrator to manually make changes in the configurations of readily available docker images from repositories. In Docker file method, the administrator has the flexibility to create a file (Docker image file) from the scratch. Table 2 contains the list of important commands/instructions used in the docker file for the purpose of container configuration.

Table 2: A table caption.

Commands	**Responsibility**
EXPOSE	Port configurations
FROM	Pulling of images
LABEL	Metadata
CMD	Entry point related arguments
COPY	Copy jobs which are location specific
RUN	Installation of add-ons
WORKDIR	Configuring working directory
ADD	Similar to copy but enables the option of extracting a .zip or a .tar file from source (local host) to destination(container).
ENV	Setting up the environment variable
STOPSIGNAL	Force the container to exit()
USER	User access and privilege (Extremely important instruction as by default the deployed container will run in root group)
VOLUME	Mounting of a volume

2.1.2 Docker Engine

As specified in the official documentation of docker, docker engine can be defined as a lightweight and portable packaging tool which relies on container-based virtualization. Docker engine is a client server application which consist of three major components. It is also known as a daemon process which is generally used by dockerd command. It is also responsible for the command line interface which is responsible for management of entire docker client OTunde-Onadele et al. (2019). The third component is REST API which is used for communication purposes with regards to daemon client. Responsibilities of Docker engine extend to sort docker images into multiple layers like base layer and container layer which must have controlled permissions of read only and read/write mode.

2.2 Linux Privilege Escalation

Privilege escalation attacks are often caused due to breaches in namespaces which being primal Linux feature to safeguard the operating system against such attacks. These types of attacks aim either at application level segment or kernel level segment of the operating system. Process space technologies could be potentially utilised to prevent application-level privilege escalation attacks. Defence against kernel level privilege escalation attacks is rather complex as it involves bifurcation of attacks into two separate groups. One being privilege escalations with control-data attacks and other being privilege escalations with non-control groups attacks. Research WQiang et al. (2018) states that the kernel control data attack is dependant on memory corruption of kernels data flow, where corrupted memory is redirected in accords to the malicious code deployed by the attacker also known as code injection attack in past. Mitigation of such attacks were possible with the implementation of Supervisor Mode Execution Prevention (SMEP) or Privileged Execute Never (PXN). Next type of attack utilizes memory corruption vulnerability leading to tampering of security critical data without even disturbing the control flow, such categories of attacks exploit the kernel by non-control data. As these types of attack tamper sensitivity kernel data structures and policies, they are highly dangerous. There are various types of mitigation techniques which are used to eliminate the risks of both above categories of attacks but one of the unique approaches is proposed by WQiang et al. (2018). Key emphasis of the approach is based on log monitoring and analysis. A log file contains complete information of the systems state (hardware, software, and network-based activities). Research F. Jaafar (2016) stands different as it speaks about efficiency via an approach which is based on automated pattern recognition techniques. It eradicates the need of manual examination of the log files while looking out for prints of privilege escalation activities in the logs. The approach proposed in research LifengWei et al. (2016) which leads to hardening of kernel security layer of SElinux. The approach leads to overhead of 1% which is far efficient as other security applications causes more than that. Process space protection technology could be possibly used to defend application level privilege escalation attacks in Linux.

2.3 Summary

2.3.1 Academic references related to Docker Security

There are several variations of attacks which are possible on Docker platforms, which proves it being vulnerable via multiple sources. Aparna Tomar and Bisht (2020) have

performed detailed analysis on possibilities of attacks on the platform and its vulnerabilities. The proposed model in thier research covers host to container and vica-versa case scenarios, focusing the application security post deployment. Authors have listed 15 possible vulnerabilities which lead to various versions of attacks and an overview of mitigation. Commonly witnessed limitations of static vulnerability assessment and various add-on dynamic security assessment offerings are described by Kelly Brady and Coffman (2020). These benefits are explained and proved with the help of experimentation based on anchore engine and clair deployed on the platform of AWS. The authors have explained the entire CI/CD process validating the security concerns related to Docker throughout SDLC (Software development life cycle). A Docker security frameworks has been proposed by Handong Cui and Huang (2019). This framework is capable of scanning images for static vulnerabilities. Authors have introduced malicious prediction module which is based on machine learning. This module can predict and mitigating unknown attacks. Provisions are set up for monitoring IP/DNS requests and resources usage for fluctuations and immediate response. Research conducted in TianshuoYang et al. (2019) explains the importance of Cgroups and offers an approach to reduce the influence of DDOS via control groups. Linux security features are not utilized in order to reduce the overhead. Total dependency seeks around cgroup mechanism leading to improvisation in efficiency. One of the four methods used in the research are based on Java exploits and others are based on consumption of CPU loads. An innovative solution of a diagnostic system has been proposed by Kwon and Lee (2020). The proposed system by the authors can examine the docker images pre and post pulling from public repository of Docker-Hub. The core of the system is based on Clair which is a static vulnerability detection utility/tool. There are several limitations in case of the proposed system, one of the key mains being use-case of static testing tools over dynamic ones. One the other end, Clair is used as the heart of the system which does not have commercial support and does not assist with strict compliance or policy management or enforcement.

2.3.2 Academic references related to Linux security against privilege escalation attacks

Researchers have researched the domain of cyber security over the year, specifically around the area of privilege escalations and elevated permissions. Research LifengWei et al. (2016), WQiang et al. (2018) and F. Jaafar (2016) have conducted analysis on privilege escalation attacks and their mitigation methods. Protection against elevated permissions related to Linux kernels can be achieved via security identifier randomization method proposed in LifengWei et al. (2016). For increasing kernel security, WQiang et al. (2018) have proposed a lightweight protection mechanism for mitigation of non-control data attacks. The approach is based on hooking system calls, for monitoring changes in sensitive kernel data and verification of its integrity during run time of the system. Also, it leverages stack canary for protection of duplicated kernel data. Automated pattern recognition techniques are used as an approach by researchers of F. Jaafar (2016) for mitigation of privilege escalation attacks.

2.3.3 Academic references related to generic container security

Research references in the sector of generic container security have been referred via research Sultan et al. (2019). Container-host protection via its authors are bifurcated in form of 4 use-cases or scenarios. Their research can be compared with scenarios of

Aparna Tomar and Bisht (2020) which have a lot in common. Scenarios proposed are container to application security and vica-versa as well as host to container security and vica-versa. Towards the end it also includes protection of semi- honest Linux kernel features from malicious activities. Evaluation of present container security against exploits have been done by authors of XinLin et al. (2018). They have used a dataset of 223 exploits and categorised them into two-dimensional taxonomy for testing the Linux container mechanisms security. At the end of experimentation authors have provided evidence of containers being compromised with regards to exploits escaping containers and affecting the hosts. Authors have provided a defence mechanism capable of encountering such phenomena against critical exploits. Authors of EBacis et al. (2015) and Bui (2015) have provided detailed analysis over the topics like Cgroup and Namespaces as well as explaining Linux mechanisms of Apparmor and SElinux respectively.

The below Table 4 contains cumulative study of related work in the particular domain of docker security or container security. Each research mentioned above have proposed a model or a design to mitigate certain attacks and have lead to hardening of container security as a whole. Certainly there are limitations to the approaches used in which are mentioned in the above Table 4. Table 3 is an extension to table 4 for the purpose of referencing.

2.3.4 Research Gap

Containers deployed in the production environments are extremely critical in nature. Administrators responsible for deployments might deploy Docker verified or official images from DockerHub. This research proves the presence of vulnerabilities and exploits/anomalies in the official images. There are different approaches proposed for partially similar study over images like Handong Cui and Huang (2019) and Kwon and Lee (2020) but this research adds a reliable approach and also enforcement of policies for compliance. The central mechanism used in this research for detection and analysis of vulnerabilities is anchore engine. Various researches like Kelly Brady and Coffman (2020) Kwon and Lee (2020)have used a similar static vulnerability detection system called Clair. Clair does not have commercial support and also cannot be used for assignment of certain policies with regards to deployments making it fairly unreliable. This research also introduces the presence of a misconfiguration schema which may be used in order to perform privilege escalation attack on the host linux machine.

Table 3: References

Sr. No	**References**
1	2020 Aparna Tomar and Bisht (2020)
2	2020 Kelly Brady and Coffman (2020)
3	2020 Kwon and Lee (2020)
4	2019 Handong Cui and Huang (2019)
5	2019 TianshuoYang et al. (2019)
6	2019 Sultan et al. (2019)
7	2018 XinLin et al. (2018)
8	2015 Bui (2015)

Table 4: Limitations over the approaches used

Sr. No	Year	Limitation over the approach used
1	2020	Experimentation performed over a single host. Swarm/Kubernetes clusters are tricky to be attacked using the same techniques under standard security configuration
2	2020	Possible implementation of policy and compliance management inorder to resolve the issue of deployments and verification in the production environment.
3	2020	DIVD system proposed by the authors could be improvised by including dynamic analysis and policy enforcement in order to meet compliance requirements. Also the back-end for the system (clair) is open source utility without commercial support available.
4	2019	Could possibly include verified images for the experimentation as it may have been more realistic way to evaluate the results with relation to the production deployment scenario.
5	2019	The researchers have covered all the possible scenarios related to analysis with regards to one particular attack: DDOS. All the 5 methods/parameters stated are excellent. While authors could specify additional attacks which may be tracked using similar parameters like container worm attack.
6	2019	Authors have conducted deep research in a single container environment with a controlled host but can possibly work on multi container clusters as well (swarm).
7	2018	Statistical study over types of attacks have been conducted with wide coverage of a vulnerable(CVE) data-set. A defence mechanism is also proposed which is limited to a particular type of attack (Priv Esc). A comprehensive model can be proposed which covers other attacks like container escape.
8	2015	The research is outdated as it specifies that default docker configuration is safe and secure. Where as above other researches disapprove this theory. But on the contradictory side, significant research have been conducted over docker internal security features.

3 Methodology

Docker image is a base to any container deployed. The image must be configured correctly and it must comply with the essential policies. Some of the generic best practices related to Docker image compliance include:

- Not using huge (in size) and popular dockerhub images unless specifically required: Reason being, one does not require all the libraries and system utilities provided by large official images present on DockerHub.
- Using a user tag in the dockerfile to avoid potential privilege attacks: By default system executes the container using root privileges, where as it is not necessary all

the time. This creates an unnecessary risk with regards to possibility of privilege escalation.

- Always enable content trust in order to avoid pulling of non verified images: This reduces but does not mitigate the risk of pulling vulnerable images.
- Avoiding minor errors like:
  - Using COPY in the DOCKERFILE in place of ADD.
  - Not using metadata labels.
  - Failing to utilise the container firewall.

Integrity of any image must be determined prior to deployments. This research assist in the determining the integrity of the image pre-deployment. The Automatic Image Analyzer(AIA) proposed in this research is responsible for rectification of vulnerabilities or exploits present in the image. This eliminates the risk of vulnerable image deployment in the production environment. AIA is based on anchore engine which is capable of inspecting and analysing the images. These images can be present either in the online repositories(DockerHub) or local ones(local machine). There are multiple prerequisites for setting up the AIA which are explained in the next chapter that is design specifications. AIA further communicates with the public CVE database present online. It contains set of entries containing details related to updated vulnerabilities and exploits information. These entries are used across the globe by many cyber security products. CVE entries are also used in U.S. National Vulnerability Database (NVD). AIA contain two more minor modules which helps with the analysis of the output received. These modules include result management module and policy enforcement module. There are several ways of performing result management, one of the most user-friendly ways include the use of spreadsheet (MS Excel or Google Sheets). Whereas Policy enforcement module grants the users to specify the requirement needed for compliance. There are several generic tools used in for policy management in any organization. Anchore engine can also be used to do so with value added configurations whereas YAML scripts can also be used.

AIA is capable of analyzing any Docker image but for this research, images from DockerHub are used. DockerHub is an online repository/library used widely across the world for management of Docker images. On DockerHub images are sub-divided into three primal categories:

- Official Images: Images published by Docker
- Verified Images: Images published by verified publishers.
- Generic Images: Neither official nor verified, published by normal users with DockerHub accounts.

Vulnerabilities and exploits can be possibly present in any of the images stored on DockerHub. In this research, all the categories of images from DockerHub are being tested using the proposed AIA.

Lastly this research provides an insight on: The impact of Docker Image misconfiguration on a linux based system. Details about the specifications and the entire experiment are provided in the evaluation section of this research. Briefly, the attack would be conducted on a locally created Dockerfile which would be misconfigured and using a parent

image from DockerHub. This would result in the root access being compromised of the host machine/server leading to a case of privilege escalation. The repercussion of such a scenarios are clearly mentioned by Sultan et al. (2019) and might lead to multiple possibilities which include:

- Remote code execution: Attacker may run a malicious code on the host as it has the highest level of privilege (root).
- Tampering: May tamper the file system or other running container configuration.
- DOS: Attack on any of the particular container and allowing it to consume excessive host resources may lead to starving of host or starving of other containers running services.

4 Design Specification

This chapter is divided into multiple sections, each based on the major components of the solution architecture. Each section will grant an insight on the respective module(component) with a brief overview and will also state its perks over similar options available. The architectural overview of the proposed solution is illustrated in the Figure 1. Key components of the solutions include:

- Publicly Available Resources
  - DockerHub and CVE Database
- Design on the Local Host
  - Docker Engine
  - Automatic Image Analyzer (Result Management Module & Policy Enforcement Module)

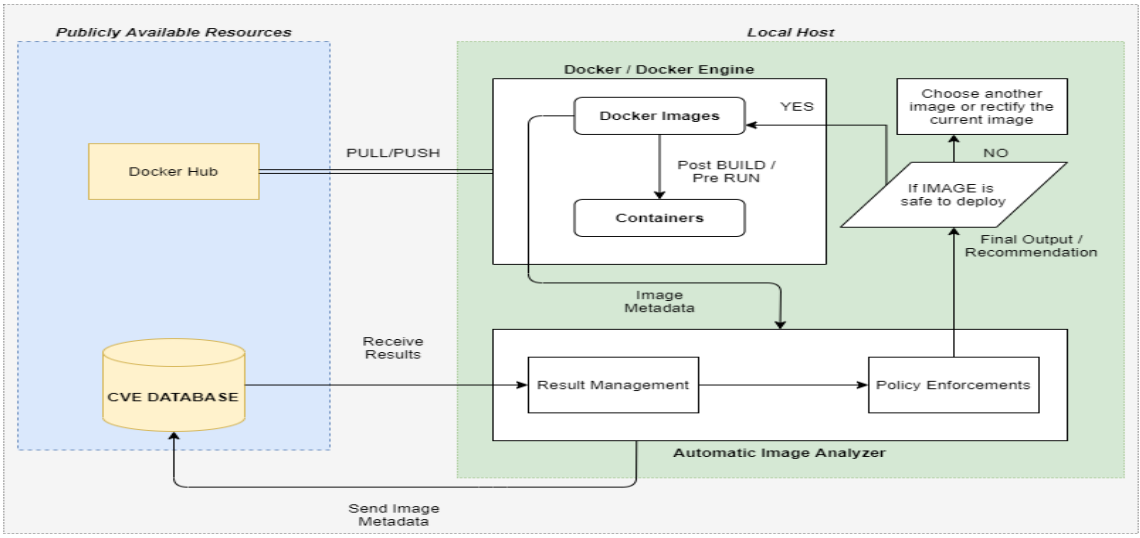

Figure 1: Architectural View

4.1 Publicly available resources

Two of the primal resources which are a part of the solution are DockerHub and CVE Database.

DockerHub, as mentioned earlier in this report: is the largest online repository which manages the global presence of container images. There are other competitors in the market providing similar services to DockerHub like Red hat Quay, Sandboxie, Amazon ECR(Elastic Container Service), JFrog Artifactory, etc. The primal reason of using DockerHub in this research is its vast popularity and the advantage of being the biggest public repository for container images.

Common Vulnerability Exposure can be defined as a documentation by MITRE Corporation containing vulnerability information in open-source or publicly released software's. Mass usage of CVE entries could be found in US National Vulnerability Database and also in the Security Content Automation Protocol(SCAP). CVE was launched in 1999 and operates with the funding of United States Department of Homeland Security.

With regards to this research, DockerHub account is associated with the Docker Engine installed and configured on the local host. DockerHub is used to PUSH (Export locally created images) and PULL (Import parent/public images to the repository) images to & from the local host. Whereas CVE identities/entries are used by the AIA based on anchore engine to determine the integrity of those images. Metadata related to the inspected images are used for the analysis and results are exported to the local host.

4.2 Design of the Local Host

Local host contains a default configuration of Docker and Docker Compose. Docker Engine (Daemon) is used for management of images as well as containers post deployment. This works as a client-server approach where daemon process acts like a server and Docker CLI works as a client. Diagrammatic representation of Docker Daemon architecture is illustrated in Figure 2.

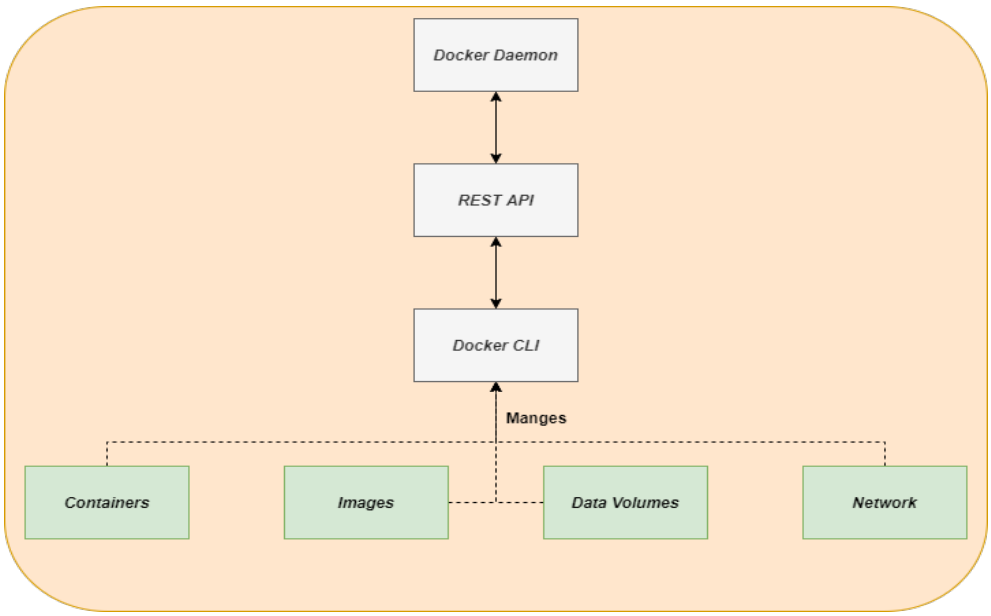

Figure 2: Docker Engine Architecture

Image metadata from Docker Engine is analysed by Automatic Image Analyzer(AIA). AIA is based on Anchore Engine as discussed in the last chapter. One of the primal prerequisite for successful configuration of anchore engine is Docker-Compose. Docker Compose enables the users to integrate multiple containers to run/support a particular application. Two containers used for the configurations contain Anchore Engine and PostGRE SQL used by anchore. Depending on the Docker-compose configuration file, appending of other value added services are also possible. A Flow-chart of the AIA module is illustrated in Figure 3. Python Scripts are used to automate the entire AIA module which assists the users with complete image analysis process. The module allow the users to check the list of present images on the local host, add or delete new/existing images for the analysis. The module also allows the users to check the analysed result for vulnerabilities. Complete analysis of the results obtained by various tests conducted are briefed in the evaluation section of this research. Lastly, policy enforcement's are possible with respect to the compliance. An image must comply by the set of rules set in the policy enforcement module in order to be eligible for deployment. This enables the users to create a barrier and prevents them from deployment of misconfigured images which contain anomalies and vulnerabilities.

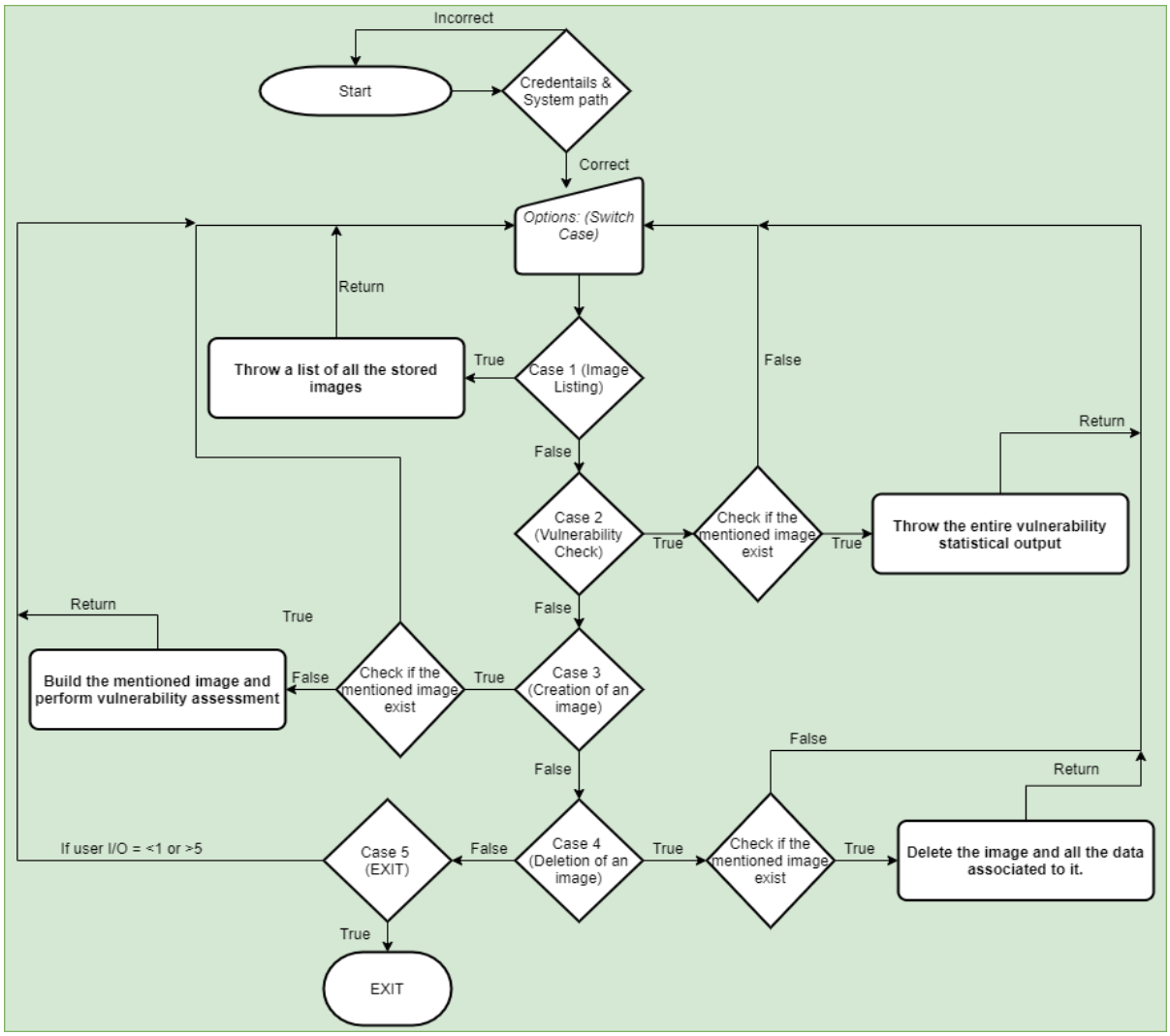

Figure 3: AIA Module

5 Implementation

This chapter emphasises on the implementation of two experiments conducted in this research. The first experiment states the results of tests conducted over 100 Docker images over the proposed architecture. These 100 Docker Images are pulled out from DockerHub and include 50 official as well as 50 non-verified images. Whereas, the second experiment contain demonstration of privilege escalation attack on host machine via misconfigured docker container. The next chapter of evaluation contains all the results and complete summary of both the experiments. Table 5 contains all the hardware and software specification used for both the experiments. The state of the local machine and the virtual machine has been statically maintained over the period of the timeline required for both the experiments. Python 3.6 have been used for automation purposes (related to AIA). Microsoft office scripts are used in excel for analysis purposes(related to result management module).

Table 5: Hardware and Software Specification

Utilities used	**Versions/Specification**
Operating system(Hardware)	Windows 10, 64-bit (Build 18363) 10.0.18363
Total Physical Memory(Hardware)	7,999 MB
Total Storage(Hardware)	1 TB
Processor(Hardware)	Intel Core i7-9750H
Hypervisor(Type 2)	VMware Workstation 15 Pro
	(15.5.0 build-14665864)
Operating System(Hypervisor)	Linux Ubuntu 18.04.5 LTS
Total Physical Memory(Hypervisor)	4,000 MB
Total Storage(Hypervisor)	40 GB
Docker	Version 19.03.13
Docker-compose	Version 1.17.1
Docker-py	Version 2.5.1
Anchor-cli	Version 0.8.2
python3.6	3.6.9-1 18.0

6 Evaluation

This chapter contains detailed analysis over the outputs retrieved from both the experiments performed. This chapter is further divided into two sections where experiment 1 is about analysis performed via AIA over 100 docker images. Whereas the second section contains details on the privilege escalation attack performed on the local machine (ubuntu 18.04).

6.1 Analysis over verified and non verified images

The proposed architecture is tested in this section with respect to 100 Docker images. All the 100 images considered for testing are pulled from the DockerHub. 50 images out of the 100 are official Docker images which are published by Docker Inc and the rest are non-verified generic Images. AIA is completely responsible for all the 100 tests. Result

management module assist in the analysis of the output with respect to 5 categories of vulnerabilities. These categories are based on NVD severity:

- Unknown/Negligible
- Low
- Medium
- High

Figure 6, contains analysis of top 40 images out of 100 which are tested. Figure 4 and Figure 5 contains graphical representation of verified (50 images) and non verified (50 images).

6.1.1 Analysis of Official Images

Figure 4 contains detailed analysis of the tests performed over 50 verified docker images. These are random official images selected from DockerHub. Figure 6 contains sample test results (yellow) of all the verified images starting from owncloud image with highest number of vulnerabilities detected (856). Average number of vulnerabilities in all the 50 images are 226 with the lowest being 1. The image of owncloud has highest number of 'High severity' vulnerabilities(129). Average of 'High severity' risks among 50 images is 5 vulnerabilities. Whereas the average of unknown, negligible, low and medium risks for 50 images are 16, 166, 8 and 29 respectively.

6.1.2 Analysis of Non-verified Images

Figure 5 contains detailed analysis of the tests performed over 50 non-verified docker images. These are random official images selected from DockerHub. Figure 6 contains sample test results (green) of all the verified images starting from pasnsxt/python-tasks image with highest number of vulnerabilities detected (2092). Average number of vulnerabilities in all the 50 images are 266 with the lowest being 0. The image of pasnsxt/python-tasks has highest number of 'High severity' vulnerabilities(2092). Average of 'High severity' risks among 50 images is 10 vulnerabilities. Whereas the average of unknown, negligible, low and medium risks for 50 images are 23, 122, 42 and 67 respectively.

6.2 Privilege escalation attack on Docker

In this experiment a live privilege escalation attack has been performed over/via Docker container services. Misconfigurations lead to such a phenomena with the main focus over gaining undesignated privileges. In this section of evaluation, the entire attack flow is discussed from a administrator and attackers point of view.

6.2.1 Administrator's Role

In the cases where a new team member needs access to any container services it is mandatory for administrator to add the user in the Docker group. Being a part of that group grants the access for usage of any docker services with certain predefined privileges. Once the new_employee(attacker) which in this case is Paul is added to the

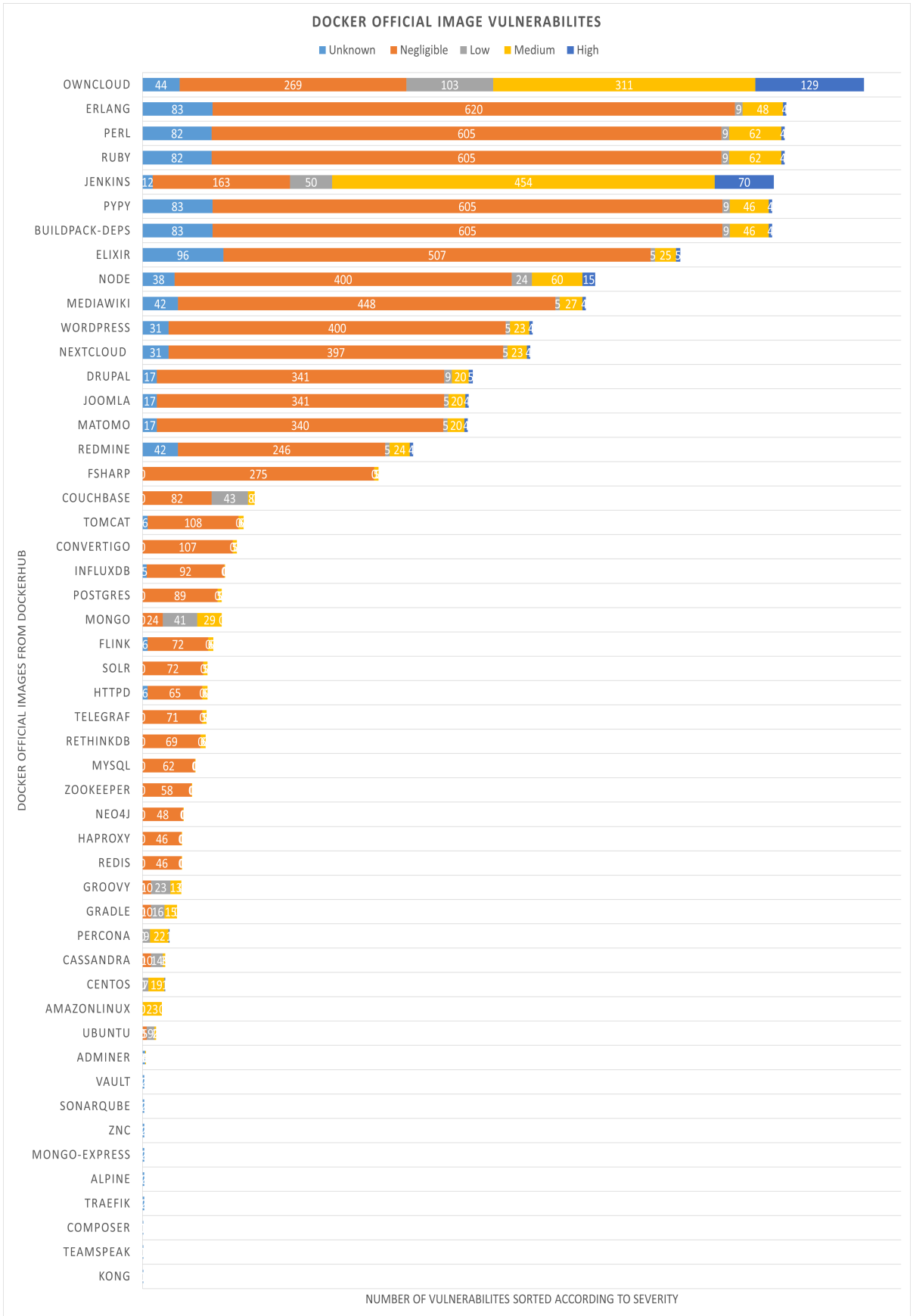

Figure 4: Analysis of Official/Verified Images

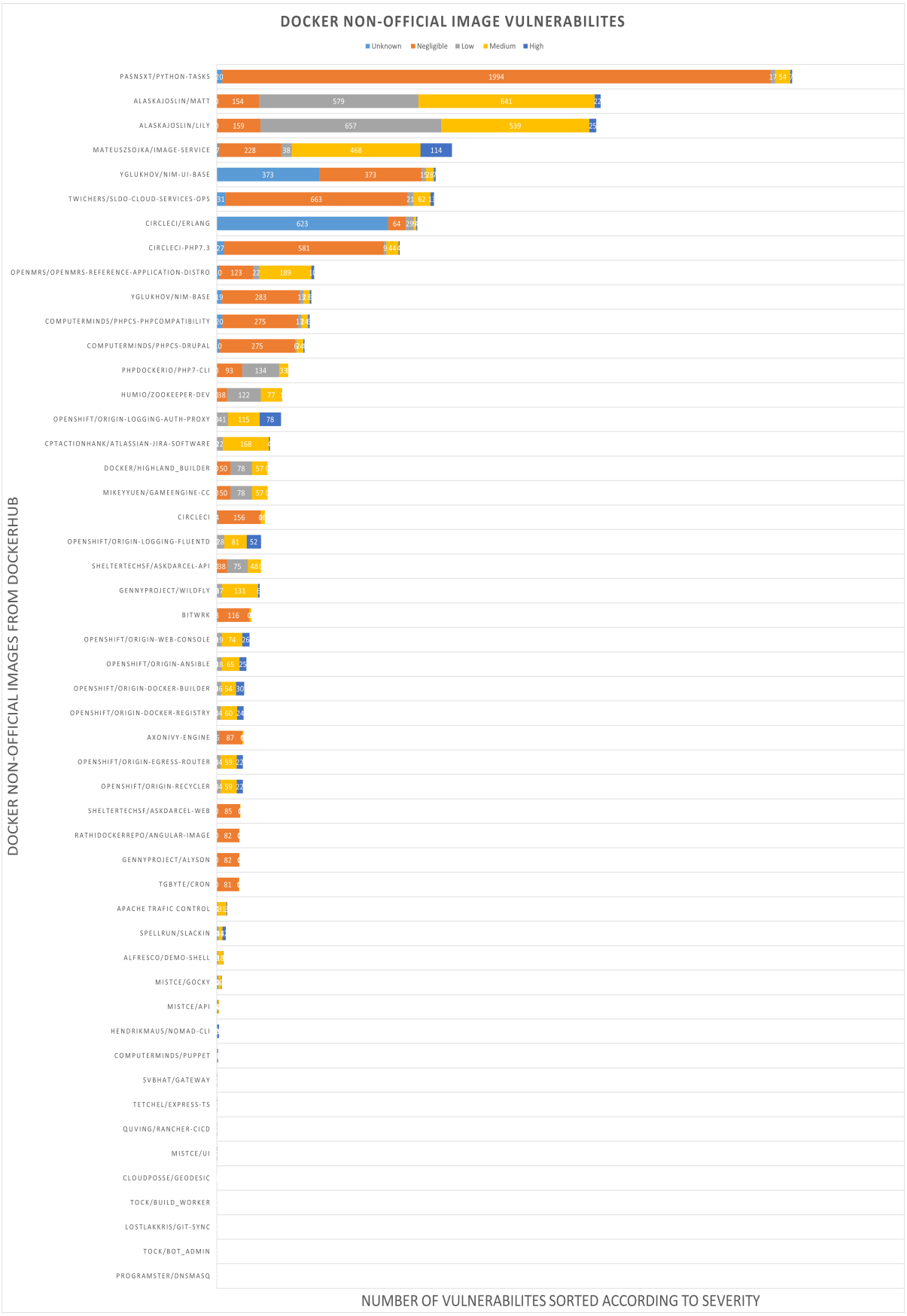

Figure 5: Analysis of Non-Verified Images

Image Name	Unknown	Negligible	Low	Medium	High	Total
pasnsxt/python-tasks	20	1994	17	54	7	2092
alaskajoslin/matt	0	154	579	641	22	1396
alaskajoslin/lily	0	159	657	539	25	1380
mateuszsojka/image-service	7	228	38	468	114	855
yglukhov/nim-ui-base	373	373	15	28	7	796
twichers/sldo-cloud-services-ops	31	663	21	62	13	790
circleci/erlang	623	64	29	9	4	729
circleci-php7.3	27	581	9	44	4	665
openmrs/openmrs-reference-application-distro	10	123	22	189	10	354
yglukhov/nim-base	19	283	13	23	6	344
computerminds/phpcs-phpcompatibility	20	275	13	24	6	338
computerminds/phpcs-drupal	10	275	6	24	4	319
phpdockerio/php7-cli	0	93	134	33	0	260
humio/zookeeper-dev	0	38	122	77	1	238
openshift/origin-logging-auth-proxy	0	0	41	115	78	234
cptactionhank/atlassian-jira-software	0	0	22	168	4	194
mikeyyuen/gameengine-cc	0	50	78	57	0	185
docker/highland_builder	0	50	78	57	0	185
CircleCI	4	156	0	15	0	175
sheltertechsf/askdarcel-api	0	38	75	48	0	161

Image Name	Unknown	Negligible	Low	Medium	High	Total
owncloud	44	269	103	311	129	856
erlang	83	620	9	48	4	764
ruby	82	605	9	62	4	762
perl	82	605	9	62	4	762
jenkins	12	163	50	454	70	749
buildpack-deps	83	605	9	46	4	747
pypy	83	605	9	46	4	747
elixir	96	507	5	25	5	638
Node	38	400	24	60	15	537
mediawiki	42	448	5	27	4	526
wordpress	31	400	5	23	4	463
nextcloud	31	397	5	23	4	460
drupal	17	341	9	20	5	392
joomla	17	341	5	20	4	387
matomo	17	340	5	20	4	386
redmine	42	246	5	24	4	321
fsharp	0	275	0	5	0	280
couchbase	0	82	43	8	0	133
Tomcat	6	108	0	6	0	120

Figure 6: Test results of top-20 Images(Verified and Non-verified)

group of Docker, Paul gains basic deployment and access privileges. Important thing to note is that, Paul does not have sudo access at his local machine.

6.2.2 Attacker's Role

The moment Paul receives access to docker services. Paul makes an empty directory at its /home/paul/ location. This directory plays an important role in the entire plot. Next step is Paul making a Dockerfile (custom Docker image), with worst docker practices and misconfigurations. Layers of the docker files are given below:

- FROM httpd : This is the first layer with importing an vulnerable apache image without any modifications. (Vulnerabilities of the particular image are proved in the first experiment).
- ENV WORKDIR /name_of_the_directory : This is the second layer which sets the environment variable called as WORKDIR. (Paul calls the name of the directory created at /home/Paul location).
- RUN mkdir -p $WORKDIR : This layer enables Paul to run the particular directory using the -p i.e. the parent tag.
- Volume [$WORKDIR] : The next layer is used to mount a volume with the variable declared in layer 2.
- WORKDIR $WORKDIR : This is the final layer 5 which states that working directory for container is aligned to the variable assigned in layer 2 and volume mounted in the layer 4.

There are many misconfiguration in the above mentioned image by the attacker. None of the best practices have been followed with the regulation of docker compliance. Methodology(3) section describes the generic practices one must follow prior to Docker deployments. Post setting up a misconfigured image, Paul runs the image with the same directory created earlier (/home/Paul/Name_Directory) with the following command: **docker build -t exploit .** (exploit is the tag used) , this enables Paul to build the misconfigured container present in the same directory. Post building up the container Paul runs the container by mounting the volume at the present targeted directory which leads to mounting of sample root system of host to the container: **docker run -v /:/Name_Directory -it exploit /bin/bash**.

The outcomes of the experiment are phenomenon. Part one of using this technique grants Paul to directly access containers root. It is not a best practice to grant root privileges of the containers to users. To avoid this a USER layer must be mentioned in the Dockerfile.

Part two leads to creation of a replica root directory of host machine into the container. As Paul is the admin of the container, he is able to manipulate the sudoers files in the replica as root. Later post exiting the container Paul has sudo access in the host machine and can log in as root (sudo bash). The entire details of this experiment are provided in configuration manual for better understanding.

6.3 Discussion

There are multiple learning outcomes from both the experiments performed in section 6.1 and 6.2. Firstly section 6.1 proves the presences of vast number of vulnerabilities in the Docker images present on Dockerhub. This even includes random 50 official Docker images along with 50 non-verified images. Average number of vulnerabilities in all the 100 images are approximately 250 vulnerabilities per image. The proposed architecture based on AIA module is able to detect vulnerabilities registered in CVE databases. On the contradictory side it lacks the capability of dynamic analysis where it cannot detect vulnerabilities which are new and not present in the CVE. Lastly in the second experiment present in 6.2, evidence has been provided based on importance of proper configuration requirement. In the absence of proper configuration and few best practises, Docker environment is extremely vulnerable.

7 Conclusion and Future Work

The proposed research questions have been addressed in the above research. Results have been obtained irrespective of any limitations with regards to efficiency of the proposed architecture. Though there are few limitations to the research conducted, one of which being the fundamental disadvantage of static tests over dynamic tests/analysis. As far as future work is concerned, the AIA module used in this research can potentially detect vulnerabilities based on misconfiguration patterns or abnormal behaviours using machine learning trainers. Productivity and reliability of AIA module would increase drastically if dynamic analysis is possible.

References

Aparna Tomar, Preeti Mishra, D. J. and Bisht, R. (2020). Docker security: A threat model, attack taxonomy and real-time attack scenario of dos, *IEEE* pp. 150–155.

Bui, T. (2015). Analysis of docker security, *CoRR* **abs/1501.02967**.
URL: *http://arxiv.org/abs/1501.02967*

Chelladhurai, J., Chelliah, P. R. and Kumar, S. A. (2016). Securing docker containers from denial of service (dos) attacks, *2016 IEEE International Conference on Services Computing (SCC)* pp. 856–859.

Combe, T., Martin, A. and Di Pietro, R. (2016). To docker or not to docker: A security perspective, *IEEE Cloud Computing* **3**(5): 54–62.

EBacis, SMutti, SCapelli and SParaboschi (2015). Docker policy modules: Mandatory access control for docker containers, *2015 IEEE Conference on Communications and Network Security (CNS)* pp. 749–750.

F. Jaafar, G.Nicolescu, C. R. (2016). A systematic approach for privilege escalation prevention, *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* pp. 101–108.

Gao, X., Gu, Z., Kayaalp, M., Pendarakis, D. and Wang, H. (2017). Containerleaks: Emerging security threats of information leakages in container clouds, *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* pp. 237–248.

Handong Cui, Shihao Wen, D. H. and Huang, C. (2019). Security analysis and threats detection techniques on docker container, *IEEE* pp. 1214–1220.

Katrine Wist, M. H. and Gligoroski, D. (2020). Vulnerability analysis of 2500 docker hub images, *arxiv* .

Kelly Brady, Seung Moon, T. N. and Coffman, J. (2020). Docker container security in cloud computing, *IEEE* pp. 0975–0980.

Kwon, S. and Lee, J.-H. (2020). Divds: Docker image vulnerability diagnostic system, *IEEE* **8**: 42666–42673.

LifengWei, YudanZuo, YanDing, PanDong, ChenlinHuang and Gao, Y. (2016). Security identifier randomization: A method to prevent kernel privilege-escalation attacks, *IEEE* pp. 838–842.

Loukidis-Andreou, F., Giannakopoulos, I., Doka, K. and Koziris, N. (2018). Docker-sec: A fully automated container security enhancement mechanism, *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)* pp. 1561–1564.

Mattetti, M., Shulman-Peleg, A., Allouche, Y., Corradi, A., Dolev, S. and Foschini, L. (2015). Securing the infrastructure and the workloads of linux containers, *2015 IEEE Conference on Communications and Network Security (CNS)* pp. 559–567.

MSollfrank, FLoch and BVogel-Heuser (2019). Exploring docker containers for time-sensitive applications in networked control systems, *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)* **1**: 1760–1765.

OTunde-Onadele, JHe, TDai and XGu (2019). A study on container vulnerability exploit detection, *2019 IEEE International Conference on Cloud Engineering (IC2E)* pp. 121–127.

Sultan, S., Ahmad, I. and Dimitriou, T. (2019). Container security: Issues, challenges, and the road ahead, *IEEE Access* **7**: 52976–52996.

TianshuoYang, ZhongxuanLuo, ZheliangShen, YicanZhong and XinHuang (2019). Docker's security analysis of using control group to enhance container resistance to pressure, *IEEE* pp. 655–660.

WQiang, JYang, HJin and XShi (2018). Privguard: Protecting sensitive kernel data from privilege escalation attacks, *IEEE Access* **6**: 46584–46594.

XinLin, LingguangLei, YuewuWang, JiwuJing, KunSun and Zhou, Q. (2018). A measurement study on linux container security: Attacks and countermeasures, *the 34th Annual Computer Security Applications Conference* .