National College of Ireland

Eliminating Misconfiguration and Privilege Escalation in Docker Images

MSc Research Project
Cloud Computing

Adarsh Sharma
Student ID: 20140207

School of Computing
National College of Ireland

Supervisor: Divyaa Elango

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Adarsh Sharma
Student ID:	20140207
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Divyaa Elango
Submission Due Date:	16th December 2021
Project Title:	Eliminating Misconfiguration and Privilege Escalation in Docker Images
Word Count:	6835
Page Count:	18

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.

Signature:	
Date:	16th December 2021

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):	

Eliminating Misconfiguration and Privilege Escalation in Docker Images

Adarsh Sharma
20140207

Abstract

Containerization is indeed a type of virtualization of the operating system in which programs operate although sharing the equivalent operating system in separated user zones classified as containers. Docker as a technology is very efficient and effective in the field of information technology security. In this research project we have analysed that This study gives proof of the same, as well as the significance of few more benchmarks that must be implemented. Furthermore, two tests are carried out to offer insight into the reliance on certified images which gets imported or pulled using Dockerhub.This project recommends an architecture that adds an explicit level of protection before image distribution. Anchore Engine is the recommended architecture's instead of Clair and other tools to analyse the Docker images. Alongside proof, a comprehensive examination of the Docker images on the Dockerhub is presented. On the suggested methodology, hundred images collected randomly are tested, half of which are legitimate image through Dockerhub whereas the remaining fifty are unverified. For the privilege escalation attack part we carried out in which local host which contains an Ubuntu operating system was attacked due to misconfigurations. Using the grounds of the trials, the security effectiveness of docker as a technologies is delivered approaching the end of the research project.

1 Introduction

Considering the advancement of technologies in virtualisation throughout the last decade, the need for containerization has become apparent. Containerization based on containers has increased in acceptance as a result of the advantages it offers against traditional VM's (virtual machines) through hypervisors. In compared to virtual computers, container technology provides incredibly light-weight and need very little boot-up time interval. Liu and Zhao (2014) and Sultan et al. (2019).Containers, due to their minimal weight, take advantage of the performance need and allows developers to construct and deliver programs in a timely manner in CI/CD model (Continuous Integration/Continuous Delivery).Docker is one of the market experts in container orchestration technology. This handles resource optimization including add-ons to activities such as fast development, testing, and deployment Brady et al. (2020). As an exceptional microservice technology, it inherited several benefits but also has some cyber security flaws. Isolation is a key problem across most technological fields and industries. Containerization was nowhere so promising than classical virtualization built on hypervisors. Each virtual machine obtains its essential application-related services through the VM kernel, meanwhile containers communicate about resources as well as their requirements among the container

as well as the host kernel. This highlights the security problems on its own since the amount of hurdles an attacker must overcome in regards of VM is greater, however containers communicate straightforwardly with the host kernel, making the host computer susceptible. Sultan et al. (2019). Public repositories could be used to obtain a parent Docker image (push and pull). Dockerhub being the most widely used. It cannot be assured that those source images posted on Dockerhub seem to be totally legal, secure to use, plus free of vulnerabilities. Liu and Zhao (2014). In a deliberately motivated condition, a person would purposefully generate an image containing vulnerabilities or insert a misconfigured and malicious program. Import of that kind of an image form the repository might provide the attacker with simple loophole access using Trojan or equivalent infections Oya et al. (2016).The existence of these backdoors within operating containers might provide the intruder multi-level/layer accessibility to the container, resulting to a potential example of privilege escalation. Whereas if root of any computer is penetrated through any of those containers, then attacker may be able to corrupt the server of docker engine , which is in charge of managing the clusters of established containers Liu and Zhao (2014). This investigation will look at examples of Privilege Escalation including the identification for misconfigured (vulnerable) images that leads to a susceptible linux environment.

Docker's official and verified images on the Docker Hub repositories contain a huge variety of vulnerabilities, those are detailed in the evaluation portion of this study. Static vulnerability assessment technologies are widely available on the marketplace. Clair and Anchore Engine being two well-known examples. These aid in the evaluation of docker images for existing CVE vulnerabilities. Our primary system employed in this project, Anchore Engine, is competent of generating static vulnerability evaluation findings. It would also deliver an exceptionally efficient policies enforcement engine that can be tailored to meet specific compliance requirements. Privilege escalation threats could be performed out effectively by leveraging vulnerabilities somewhere at applications or kernel levels of either Linux server Zhong and Liu (2020). When a container installed on a linux host is penetrated by enhanced privileges, the whole docker engine server including host computer may be affected. This study demonstrates the prevalence of such exploitation, which is a famous example of privilege escalation.

We would just be enabled to filter out all the hazardous pictures subsequent to their deployment as operational containers somewhere at conclusion of the study. This study also exposes the potential of a misconfiguration scheme, which might contribute to a privilege escalation threat on the host systems during container deployments.

2 Research Questions

• What actions may be done to safeguard the Docker image against attacks such as Privilege Escalation?

• How can unsafe Docker images containing security problems can be prevented from being published into public repositories such as Docker Hub?

This study aims to improve Docker security through the usage of the suggested Automatic Image Analyzer (AIA). Even before of deploying images within the operational environment, a static vulnerability evaluation must be performed. AIA is mostly powered

by the Anchore Engine. The tool used Anchore Engine is indeed a static vulnerability evaluation tool that uses CVE (Common Vulnerabilities Exposures) ratings to determine vulnerabilities. Following the satisfactory conclusion of this research, installations into production settings would be entirely based on CVE (Common Vulnerabilities and Exposures) assessments.

3 Related Work

This related work part is structured into many pieces. The majority of the analysis has been done in the containerisation sector. The very first element of literature review is an overview of Docker's architecture design. The main focus is on "C-groups" and "Namespaces", who are in charge of general security issues isolating in docker. The later part offers information regarding the study done on Linux privilege escalation threats, and the final chapter is an overall assessment of the studies, as well as drawbacks and areas for further research.

3.1 Docker C-groups and Namespaces

Containerization methods have been accessible for a long time through Linux. Docker as a technologies first appeared in 2013, whereby it dominating the worldwide containerization and deployments industries. In practice, the use cases of containers against virtual machines is determined by a variety of requirements-related criteria. Docker constantly operates on the basis of a shared responsibilities approach with its hosts, in which it distributes its resources to the kernel that hosts it.This phenomenon creates a security concern, as safeguarding the hosts plus docker as just a platform is mutually critical in terms of cyber security. Sultan et al. (2019) Shameem Ahamed et al. (2021). Cgroups Namespaces were indeed two kernel features that are accountable for the entire security as well as isolation of container techniques.

Cgroups are also known as Cgroups. They are the initial characteristic. Cgroups seem to be in charge of limiting the resources available to containers. This includes system storage, CPU run-time, network connectivity, and input/output operations. Cgroups provide as a check against a singular container's excessive consumption of resources.Sultan et al. (2019) had also published significant research on Control groups and namespaces. C-groups typically store a variety of system information.

Namespaces are the kernel's second possible functionality. They were responsible for the isolation of assets over containers. Most functionality, from file systems to procedures and domain names, are separated by namespaces.Binding of Containersis one of the tasks of namespaces, that also include the separation of container-related activities from neighbouring containers or the hosts Operating System. Namespaces are the functionality in charge of offering deployment containers unique isolated root directories. Table 1 lists the many types of namespaces accessible in Linux. Amontamavut et al. (2012) Seo et al. (2014).

3.2 Docker Engine

Docker engine, as stated in the canonical literature, is a lightweight as well as transportable packaging technology that depends on container-based virtualization. It is indeed a client-server program made up of three primary components. This is also referred as

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

Figure 1: Type of Namespaces

a daemon process, and this is commonly utilized by the docker operation. It is also in charge of a command line interfaces, which is in charge of managing the whole Docker clients Liu and Zhao (2014). The third element is the REST API, that is utilized for communicating from the daemon client. Docker engine objectives include sorting docker images onto many layers such as container layer and base layer, that must possess regulated rights of read/write and read only privilege.

3.3 Docker Image

Docker images are the templates that include the configuration data needed to install a container. Docker engine is required for container installation via docker images along with container management across a host system. Docker images contain data needed by middleware, network setup, applications, operating system files, and library. The authors of the study, Sultan et al. (2019), provided a comprehensive analysis of docker images and indeed the docker engine.These are various techniques for creating a Docker image, and the most common of which are both interactive approach as well as the Docker file approach. The interactive technique allows the admins to manually modify a configuration of widely accessible docker images through repositories. The Docker file approach allows the administrator to generate a file (Docker image file) from beginning. Table 2 offers a list of the most significant commands/instructions required in the docker file during container setting.

3.4 Privilege Escalation

Privilege escalation threats are frequently caused by namespace incursions, which are a fundamental Linux characteristic designed to protect the operating system from such assaults. Such attacks target either the application domain or the kernel layer of the os. Processes space technologies might be used to protect against application-layer privilege escalation assaults. The defense versus kernel-layer privilege escalation assaults is fairly difficult since it requires the attack to be divided into two distinct groups. The first involves privilege escalation through control-data threats, while the second involves privilege escalation using non-control groups assaults. According to Sinha et al. (2020), this assault on kernel controlling data is based on memory degradation of the kernel's flow of data, wherein tainted memory is redirected in accordance with the malicious code introduced by the attackers, which was previously characterized as a code injection

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

Figure 2: Commands

assault.These attacks might be mitigated by using Supervisory Mode Execution Prevention (SMEP) as well as Privileged Execute Never (PXN). The following sort of attack exploits a memory corruption vulnerabilities, allowing tampering with security-critical content without disrupting the control flow; similar attacker exploits the kernel by using non-control data. Such attacks are extremely risky since they mess with sensitive kernel data structures including directives. There are several sorts of mitigation strategies utilized to remove the dangers of each of the aforementioned categories of attacks, however Sinha et al. (2020) provide one of the distinct ways. The approach's main focus is on log monitoring and assessment. A log file includes detailed information on the health of the machine (software, hardware, and network-based jobs). Tasci et al. (2018) investigation is unique in that it discusses efficiency using a method centered on automatic pattern recognition algorithms. It eliminates the requirement for manual analysis of log records to seek for evidence of privilege escalation actions. The method provided in the study Teplyuk et al. (2020), ultimately contributes to the strengthening of SElinux's kernel security protocol. The technique results in a 1 percent overhead, making it significantly more efficient compared to existing security applications. Using Linux, task space protection technologies might be used to guard against application-layer privilege escalation threats.

3.5 Summary

In the research of Sultan et al. (2019),suggested model spans host to containers and vice-versa case situations, with an emphasis on applications security after deployment. The authors have identified 15 potential vulnerabilities that might lead to several types of attacks, as well as an outline of mitigating. Brady et al. (2020) explain the common constraints of static vulnerability analysis and different add-on dynamic security evaluation products. These advantages are presented and shown by testing using the tools like Anchore Engine and Clair implemented on the AWS infrastructure. The researchers have

described the whole CI/CD process, confirming Docker security issues across the Software development life cycle also known as SDLC. Rangnau et al. (2020) have presented a Docker security architecture.This framework could scan pictures for static vulnerabilities. The authors have developed a malevolent prediction module built on machine learning techniques. This modules is capable of predicting and mitigating unpredictable threats. Provisions are in place to monitor IP/DNS inquiries as well as resource use for changes and fast reaction. **?** study highlights the relevance of Cgroups and proposes a method to mitigate the impact of DDOS using control groups. Therefore order to decrease cost, Linux security mechanisms are not used. Total reliance tries to revolve around the cgroup mechanism, resulting in increased efficiency. Several of the four approaches employed in the study is focused on Java vulnerabilities, while the others are centered on CPU load usage. N et al. (2015) have presented a unique diagnostics system solution. This authors' suggested approach may inspect docker images before and after they are pulled from DockerHub's open repo. Clair, a static vulnerabilities detecting utility/tool, serves as the system's foundation. There really are various drawbacks in the proposed approach, one of which being the preference for static testing methods over dynamic options. On the other hand, Clair is employed as the system's backbone, despite the fact that it lacks commercial sponsorship and therefore does not aid in tight compliance, policy administration, or implementation.

3.6 Research Void

Containers used in operational contexts are incredibly important. Administrators in charge of deployments may use Docker certified or authorized images provided on DockerHub. This investigation demonstrates the prevalence of vulnerabilities including exploits within official images. Gholami et al. (2021) and Rad et al. (2015) offered alternative ways for partially comparable image studies, however this research includes a dependable methodology as well as policy enforcement for conformance.The anchore engine tool is the key mechanism employed in this research for vulnerability identification and analysis. Several studies, notably Brady et al. (2020) and Fadil et al. (2020), have employed a static vulnerability detection approach named Clair. This tool Clair does not provide commercial assistance and could not be utilized to apply specific deployment procedures, making it rather unstable. This study also presents the existence of a misconfiguration schema that might be utilized to launch a privilege escalation hack on the hosted linux computer.

4 Methodology

The Docker image serves as the foundation for each container that is deployed. This image needs be appropriately configured and adhere to the necessary regulations. Some of the general recommended practices for Docker image compliance are as follows:

• To mitigate potential privilege assaults, include a user tag with in dockerfile: Through default, the system implements the container with root privileges, which is not always necessary. This introduces an unneeded risk in terms of privilege escalation.

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.

Figure 3: Limitations over approach

• Unless absolutely necessary, avoid using large (in size) and prominent Dockerhub images: Another reason for this is that not all of the library and system functions given by the official docker images on DockerHub are required.

• Constantly enable contents trust to prevent pulling unverified photos: This minimizes, but does not eliminate, the danger of pulling susceptible images.

Ahead to deployment, the integrity of each picture must be confirmed. This analysis aids in evaluating the image's integrity before to dissemination. This AIA (Automatic Image Analyzer) suggested in this study is in charge of correction of image weaknesses or exploits. This reduces the danger of deploying a vulnerable image throughout the operational environment. AIA is built on the anchore engine, that can inspect and analyze docker images. Those images may be found in either online repositories example DockerHub or local machine also known as local repository. There seem to be several conditions for establishing the AIA, those are discussed in the next section, design requirements. AIA also interfaces with the public CVE database, which is available online. It provides a collection of entries including facts on current vulnerabilities and hacks. Several cyber security solutions use these entries all around the world. CVE entries are also utilized in the National Vulnerability Database of the United States (NVD). AIA has two more small modules that aid in the evaluation of the output collected. These sections also include policy enforcement component and the outcome management component. There are numerous methods for managing results, among the most user-friendly being the usage of spreadsheets (Google Sheets or Microsoft Excel). In contrast, this Policy enforcement modules allows users to determine the requirements for compliance. In each business, there are various general instruments used for policy administration. Anchore engine may likewise be utilized to do so using value - adding customizations, as can YAML scripting. AIA may analyze any Docker image, however for this study, images from open repo of DockerHub are being used. It is a popular web repository/library for managing Docker images throughout the Globe. The docker images can be classified into below mentioned category on Docker hub.

• Verified Images- These are the images which are published by the verified source.

• Official Images- These are the images which are official and published by Docker itself.

• Generic Images- These are the images which are neither published by Docker nor by any verified source. These are uploaded by any user who have Dockerhub account.

Vulnerabilities as well as hacks might be found in many of the DockerHub images. Throughout this study, the suggested AIA is used to assess all sorts of images from open repo of DockerHub. Finally, this study sheds light on the implications of Docker Image misconfiguration on some kind of Linux-based infrastructure. The assessment portion of this research contains information regarding the requirements and the complete experiment. In a nutshell, the exploit would be carried out on a locally produced Dockerfile that was misconfigured but also used a parent image through DockerHub. This would undermine the host machine's/root server's access, resulting in an instance of privilege escalation.

5 Design Specification

This part is organized into several sections, each depending on the solution architecture's primary components. Every subsection will provide a summary of the particular module or component as well as its advantages over comparable choices accessible. Figure 5 depicts an architectural layout of the suggested solution. The following are key aspects of the approaches.

• Open resources available online.
- Docker Hub Repository and CVE Database

• Implementation on Local System.
- Docker Engine - AIA (Automatic Image Anylyzer comprising of result management and policy making modules.)

Please find the Architectural diagram of the proposed study which is implemented.

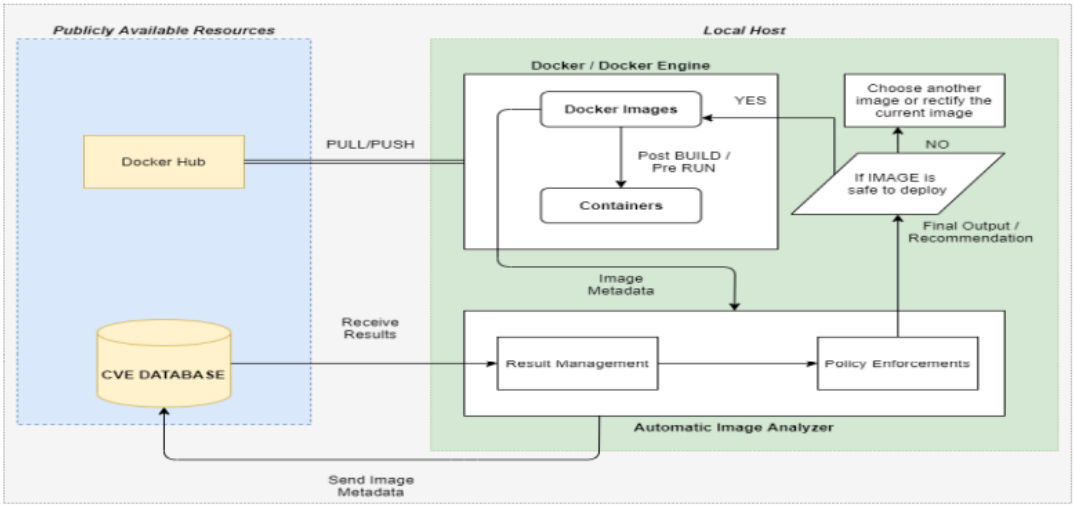

Figure 4: Architecture Diagram

5.1 Open resources available online

In this section there are two main components which are available online and they are open for everybody. One is Docker Hub and the other one is CVE database.

As previously stated in this project, DockerHub is indeed the leading internet repository that handles the worldwide availability of container images. Similarly rivals in market are offering comparable products to DockerHub include Red Hat Quay, Amazon ECR which is a prominent Container Service offered by Amazon, JFrog Artifactory, and others. The primary rationale for adopting DockerHub throughout this study is its widespread appeal and the fact that it is the largest public source providing container images.

CVE database which is also known as Common Vulnerability Exposure is indeed an MITRE Corporation publication presenting vulnerability details in open-source or commercially available software. CVE identifiers were widely used in the United States

National Vulnerability Database as well as the SCAP (Security Conttent Automation Protocol).

5.2 Implementation on Local System

In the local machine user needs to install Docker and Docker compose before configure Docker Engine. Docker Engine (Daemon) primarily designed to handle images and containers once they have been deployed. This is indeed a client-server strategy in which the daemon process operates as a server and the Docker CLI serves as a client. Architectural diagram of Docker Engine is given below in figure 6.

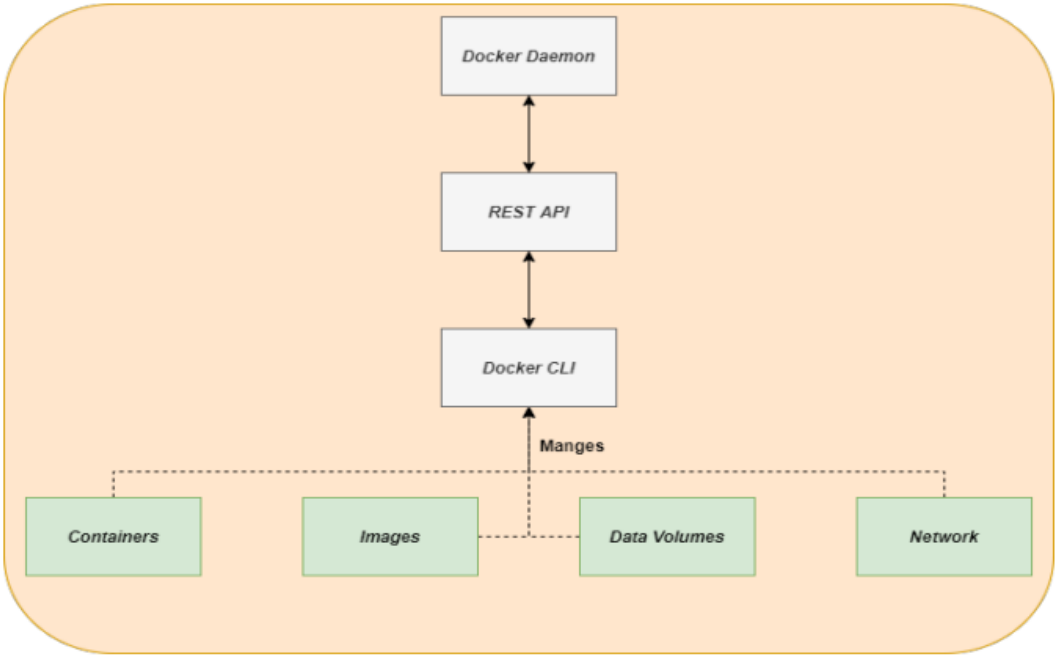

Figure 5: Docker Engine Diagram

Automatic Image Analyser examines image metadata through Docker Engine. As mentioned in the last section, AIA is built on the Anchore Engine. We need to install Docker-Compose as it is a critical requirement for effective anchore engine deployment. Docker Compose allows users to combine many containers to operate a certain application. Anchore Engine and database PostGRE SQL, which anchore uses, are both contained in two configurable containers. Manually inputting other value-added solutions is also conceivable, based on their Docker-compose setup file. Figure 7 depicts a flow diagram of the AIA modules. For Scripting Python is used which is automating the module of AIA, assisting users through the full image analysis phase.This module allows users to view the list of images currently available on the host machine, as well as add or remove new or any existing images for evaluation. The program also permits users to examine the results of the analysis for vulnerabilities. An assessment portion of this research provides a comprehensive review of the data received from the numerous tests performed. Finally, policy enforcement is conceivable in terms of compliance. To just be authorized for deployment, each image must follow the set of criteria defined in the policy enforcement modules. This allows users to construct a barrier that prohibits them from deploying

malformed images that include abnormalities and vulnerabilities.

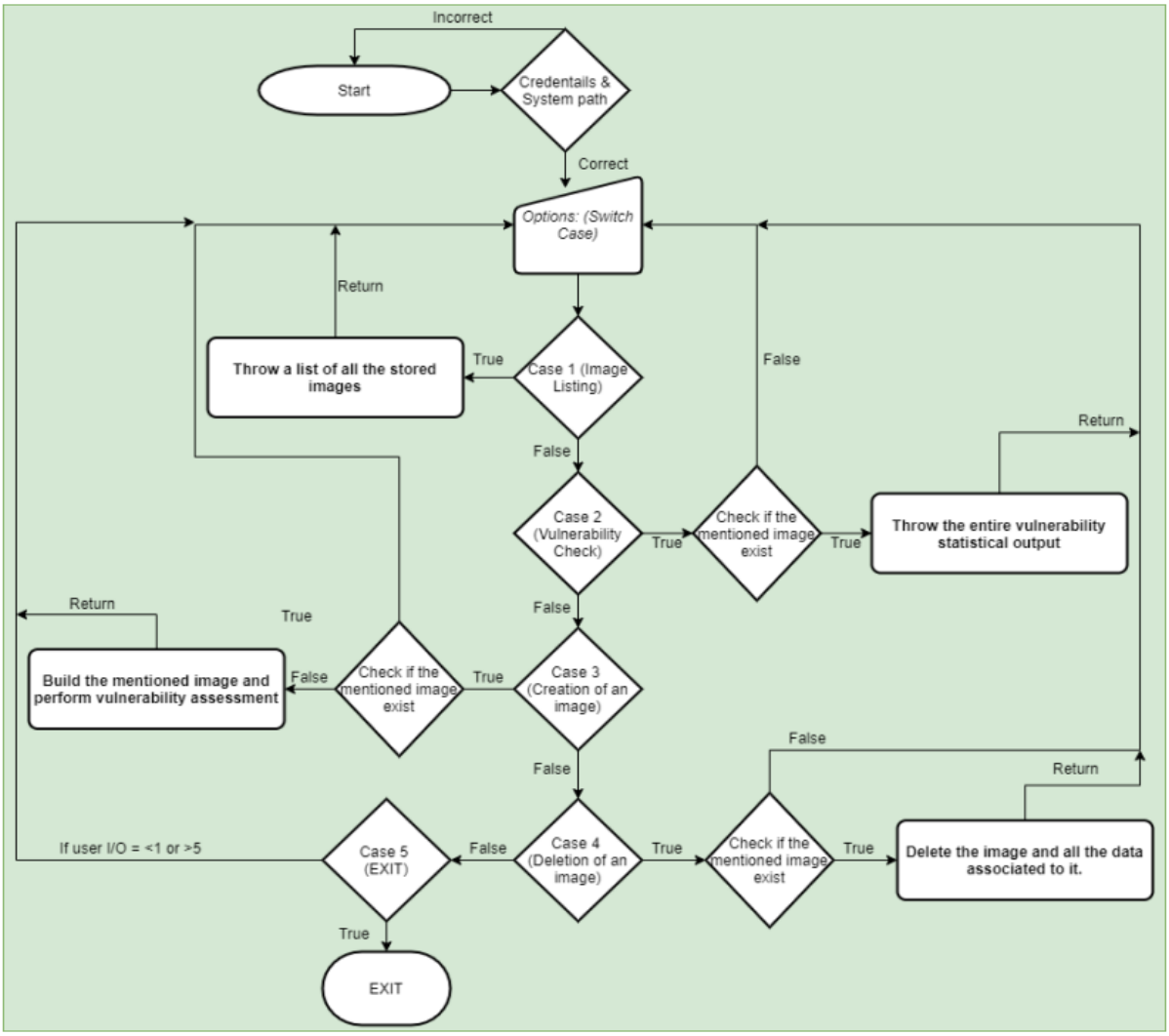

Figure 6: AIA Diagram

6 Implementation

This part focuses on overall implementation of 2 operations carried out in this study. The very first experiment reports on the outcomes of testing performed on hundred Docker images using the suggested framework. Mentioned hundred Docker Images are taken from DockerHub and comprise 50 recognized and 50 unverified images. The other experiment, on the other hand, includes an illustration of a privilege escalation assault on the host system through a misconfigured Docker containers. The next chapter of assessment includes all of the findings and a detailed overview of both studies. Table below lists all of the specifications related to hardware and software that were used in both research. The status of the local computer and the virtual machine was therefore statically maintained throughout the duration of the investigations. Python 3.6 has been applied for automating AIA. Excel analyzes data using MS Office scripts (connected to the result management component).

Utilities Used	Versions/Specification
Operating system(Hardware)	Windows 10, 64-bit
Total Physical Memory(Hardware)	8 GB
Total Storage(Hardware)	1 TB
Processor(Hardware)	Intel Core i5-10210U
Hypervisor(Type 2)	VMware Workstation 16.2.x
Operating System(Hypervisor)	Linux Ubuntu 20.04.1 LTS
Total Physical Memory(Hypervisor)	4 GB
Total Storage(Hypervisor)	100 GB
Docker	Version 20.10.11
Docker-compose	Version 1.17.1
Anchor-cli	Version 0.9.3
Python3	Version 3.8.10

Figure 7: Specifications

7 Evaluation

Each chapter offers a comprehensive examination of the results obtained through both studies. This part is further separated into two segments, with experiment 1 focusing on AIA examination of 100 Docker hub images. The next section provides information on the privilege escalation exploit carried out on the host system (ubuntu 20.04)

7.1 Evaluation over validated and non-validated images

Under this part, the suggested infrastructure is tested against hundred Docker images. Apparently DockerHub is used to download all the hundred images eligible for evaluation. Half of the total images are authorized Docker images provided by Docker Inc, with the other being non-verified common images. AIA is entirely accountable for all 100 evaluations. The result managing module aids in the study of the outcome in relation to the five kinds of vulnerabilities. These classifications are dependent on the intensity of the NVD:

- High
- Medium
- Low
- Negligible or Unknown

Figure 11 shows an investigation of the top 20 images each from 100 examined. Figures 9 and 10 provide graphical representations of authenticated 50 Docker images and unauthenticated rest half Docker images.

7.1.1 Verified Image

Figure 9 shows a thorough breakdown of the tests run on 50 validated images. Those are Docker-Hub certified images chosen at random. Figure 11 shows exemplary test results in yellow colour for all confirmed photos, beginning with image "owncloud" having the

largest proportion of vulnerabilities discovered (856). The average count of vulnerability in all fifty docker images is 226; the least is 1. This owncloud image contains the most 'High severity' risks (129). The average number of 'High severity' dangers amongst 50 images is 5. Considering 50 images, the average of unknowns, insignificant, low, and considerable dangers is 16, 166, 8, as well as 29, respectively.

7.1.2 Non-Verified Image

Figure 10 shows a thorough breakdown of the testing run across fifty non-verified images. All those are Docker-Hub images chosen at random. Figure 11 shows sample test outcomes in green colour for all confirmed images, beginning with image named pasnsxt/pythontasks containing the largest amount of vulnerabilities discovered (2092). The average security vulnerabilities throughout all 50 images is 266, having 0 being the minimum. The images of pasnsxt/python-tasks contain the most 'High severity' vulnerability counted (2092). The average number of 'High severity' dangers among 50 docker images is ten. Considering 50 images, the aggregate of unknown, insignificant, low, and medium dangers is 23, 122, 42, as well as 67, respectively.

7.2 Docker Privilege Escalation Exploit

A realtime privilege escalation exploit was carried out over/via Docker container facilities in this research. Misconfigurations cause such a phenomenon, with the major focus being on getting unspecified rights. The whole attack flow is detailed from both an administrator as well as an attacker's perspective in this portion of the report.

7.2.1 Role of Admin

When a fresh team worker requires access to specific container services, an administrator must enroll the individual to Docker group. Becoming a member of such group permits you to use every docker service with specific predefined permissions. When a new worker we are considering him as attacker, in this example Ankit, is enrolled to the Docker group, Ankit obtains basic deploying and access capabilities. It is crucial to remember that Ankit does not have sudo privileges on his local system.

7.2.2 Role of Attacker

Ankit's accessibility to Docker services is granted. Immediately Ankit creates a blank directory at /home/ankit/. One such directory is critical to the overall plot. Ankit will then create a Dockerfile (customized Docker image) using the worst Docker techniques and misconfigurations. The following are the levels of the docker container files:

- FROM httpd layer is utilized to import the image of apache with faults.

- ENV (dollar sign)WORKDIR layer is used to set the variable for the environment.

- RUN mkdir -p WORKDIR using this ankit can run the directory

- Using Volume with the directory to mount the layer with volume on variable declared.

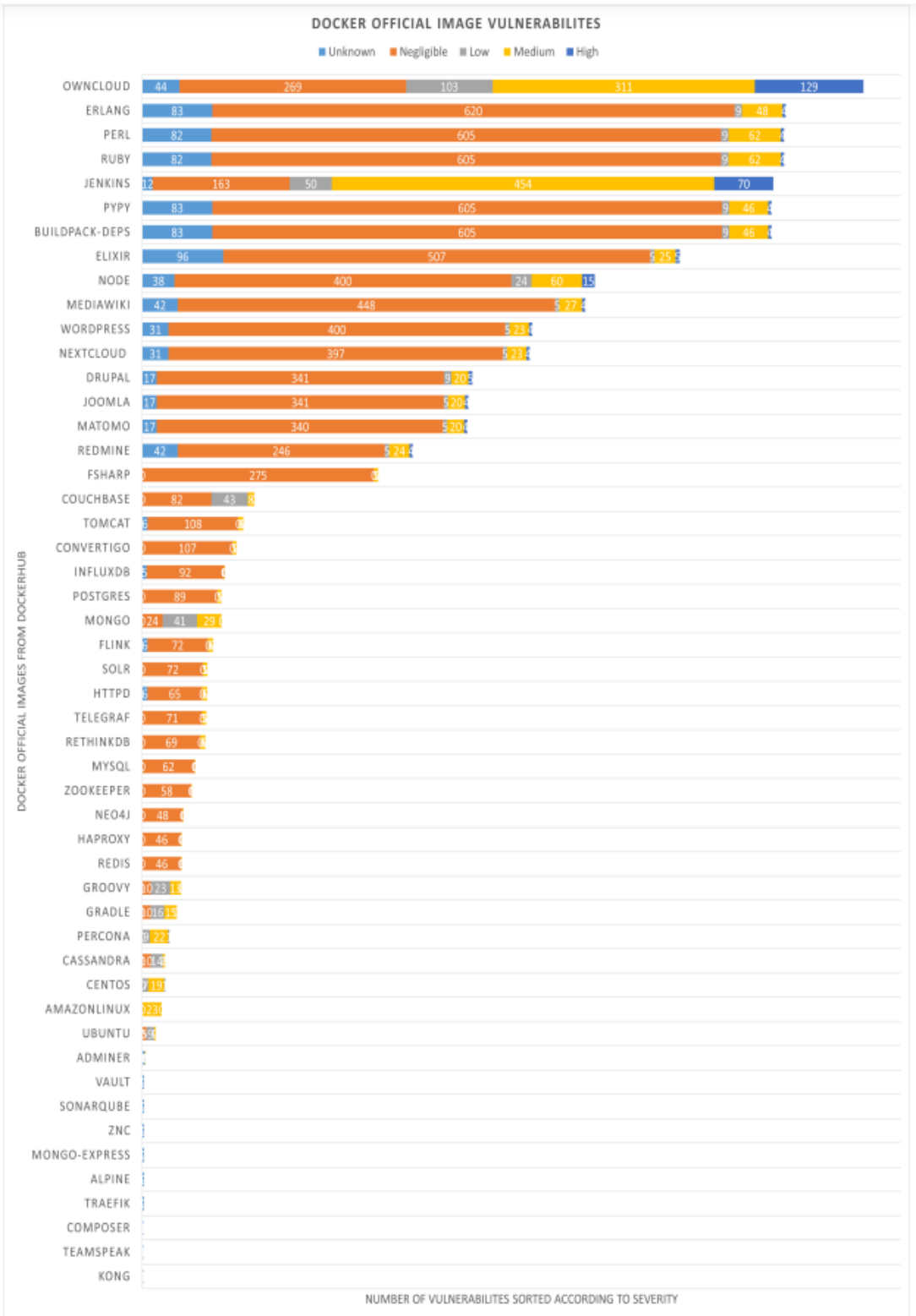

Figure 8: Docker Verified Image

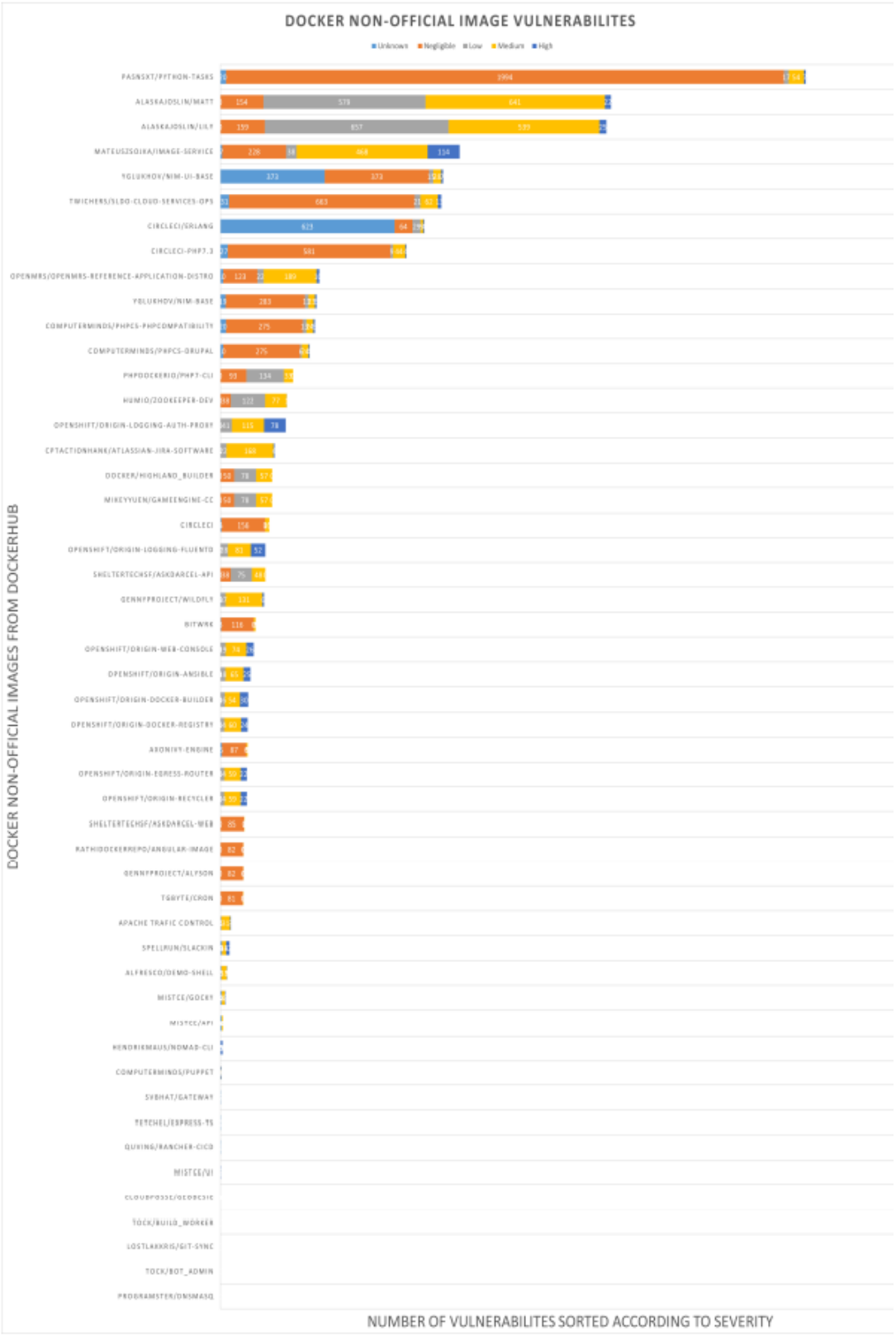

Figure 9: Docker Non-Verified Image

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 10: Study results for the top 20 Verified and Unverified Images

-WORKDIR (dollar sign)WORKDIR with this the directory is alligned.

This attacker has made several irregularities in the above created image. With regard to the enforcement of docker compliance, neither of the best procedures have now been implemented. The Methodology segment explains the general principles that must be followed prior to Docker installations. Ankit runs the imaged using the same directory generated before (/home/ankit/Name Directory) after establishing a misconfigured images with the following instruction: " Docker build -t exploit" is an abbreviation for Exploit tag. Which allows Ankit to create the misconfigured container that is located in the same directory. Following the construction of the container, Ankit launches it by staging the volume at the current desired directory, that results in the transferring of the experimental root systems of the host towards the container using : docker run-v /:/Directory -it exploit /bin/bash.

The study's results are unexpected. Phase one of this strategy provides Ankit immediate access to the container's root. Granting users container root capabilities is not a great practice. The USER layer barrier should be specified with in Dockerfile to prevent this.

Phase two involves replicating the host computer's root directory inside the container. Because Ankit has the container's administrative rights, he has access to all sudoers files in the clone as root. Ankit now has sudo privilege on the host system and therefore can sign in as root after quitting the container.

8 Conclusion and Future Work

The aforementioned study addressed the suggested research questions. Regardless of any limits in the suggested architecture's efficiency, outcomes were produced. Although this research has certain drawbacks, including one that being the inherent disadvantage of static testing versus dynamic analysis. In terms of future work, the AIA modules employed in this study may be able to discover vulnerabilities predicated on misconfiguration tendencies or anomalous behaviors utilizing machine learning algorithms. If dynamic analysis is feasible, AIA module efficiency and dependability would rise.

References

Amontamavut, P., Nakagawa, Y. and Hayakawa, E. (2012). Separated linux process logging mechanism for embedded systems, *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 411–414.

Brady, K., Moon, S., Nguyen, T. and Coffman, J. (2020). Docker container security in cloud computing, *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0975–0980.

Fadil, I., Saeppani, A., Guntara, A. and Mahardika, F. (2020). Distributing parallel virtual image application using continuous integrity/continuous delivery based on cloud infrastructure, *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, pp. 1–4.

Gholami, S., Khazaei, H. and Bezemer, C.-P. (2021). Should you upgrade official docker hub images in production environments?, *2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 101–105.

Liu, D. and Zhao, L. (2014). The research and implementation of cloud computing platform based on docker, *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP)*, pp. 475–478.

N, P. E., Mulerickal, F. J. P., Paul, B. and Sastri, Y. (2015). Evaluation of docker containers based on hardware utilization, *2015 International Conference on Control Communication Computing India (ICCC)*, pp. 697–700.

Oya, M., Shi, Y., Yanagisawa, M. and Togawa, N. (2016). In-situ trojan authentication for invalidating hardware-trojan functions, *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pp. 152–157.

Rad, P., Muppidi, M., Agaian, S. S. and Jamshidi, M. (2015). Secure image processing inside cloud file sharing environment using lightweight containers, *2015 IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1–6.

Rangnau, T., Buijtenen, R. v., Fransen, F. and Turkmen, F. (2020). Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines, *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pp. 145–154.

Seo, K.-T., Hwang, H.-S., Moon, I.-Y., Kwon, O.-Y. and Kim, B.-J. (2014). Performance comparison analysis of linux container and virtual machine for building cloud, pp. 105–111.

Shameem Ahamed, W. S., Zavarsky, P. and Swar, B. (2021). Security audit of docker container images in cloud architecture, *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pp. 202–207.

Sinha, N., Sundaram, M. and Sinha, A. (2020). Authorization secured dynamic privileged escalation, *2020 International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT)*, pp. 110–117.

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

Tasci, T., Melcher, J. and Verl, A. (2018). A container-based architecture for real-time control applications, *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pp. 1–9.

Teplyuk, P., Yakunin, A. and Sharlaev, E. (2020). Study of security flaws in the linux kernel by fuzzing, *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pp. 1–5.

Zhong, J. and Liu, W. (2020). Research on container security of paas, *2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 722–725.