Detection, Analysis and Countermeasures for Container based Misconfiguration using Docker and Kubernetes

2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS) | 978-1-6654-6883-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IC3SIS54991.2022.9885293

Vijay B Mahajan
Department of Computer Engineering and IT
College of Engineering Pune
Pune, India
Email:vijaybm20.comp@coep.ac.in

Dr.Sunil B Mane
Department of Computer Engineering and IT
College of Engineering Pune
Pune, India
Email:sunilbmane.comp@coep.ac.in

***Abstract*— In the era of a fast, productive project delivery lifecycle, the technologies like Docker containers are used in mass for deployment as well as development and production phases of the SDLC (Software Development Life Cycle) phase. Container technology changed ways of application packaging and delivery methods with the efforts of deploying them as a service in the cloud environment. Docker is a leading technology that automates the process of deployment of containers. Kubernetes is a container management tool that helps to automate multiple container deployment, descaling, and load balancing. Container technology brings a new, faster, and easier way of achieving software building phases, but it also comes with a cost of security. As containers are lightweight and are a cloud-based technology for implementing virtualization in a unique way of itself, it also brings security concerns with it, that can give adversaries a way to exploit the containerization process and cause security attacks within the containers. Such attacks may be caused due to misconfiguration of containers while deploying the containers at the early phase, which leads to a vulnerable environment ready to be exploited by adversaries or which creates vulnerabilities that can cause havoc. The purpose of the research is to analyze the container deployment misconfigurations, which may be the first step of defense for container security that is required to evade further exploitation before it occurs. The attempt is to give possible container deployment security policies that can help secure the cloud environment from security attacks with proper configurations done during the deployment phase of containers. The proposed system may be used to implement a tool that can possibly detect the misconfigurations and help secure the process.**

Keywords— Containers, cloud virtualization, Docker, Kubernetes, cloud security

I. INTRODUCTION

The era of cloud-based infrastructure for major industrial workflows to happen smoothly and provide a lateral phase implementation to increase the product delivery speed is a new trend in software industries. This is easily achieved due to all resources being on the cloud and a major resource of which is contributed by container technology. Container technology is scalable, load balancing, auto orchestrated, has less overhead, has increased portability, more consistent operations, greater efficiency, lighter weight, cost-effective, easy disposability and better application development. Process isolation and virtualization abilities with powerful Linux kernel build make containers possible solutions for enterprise-level paradigms. Some of the major abilities such as Cgroups(control groups) used as resource allocation among processes and namespaces to restrict access and visibility into resources or systems - helps to achieve multiple application components to share resources of a single instance of host OS in a similar way as hypervisor benefits from virtual machines(VM) in the process of sharing computational power, memory and other resources on a singular hardware server. Container only includes OS dependencies and processes which are necessary to execute certain code or scripts excluding executables of entire OS instance, but it can also execute OS executables fully as containers are flexible in the storage and memory resource allocation manner. So, its size is in megabytes (Gigabytes if OS is executed on the container) which helps achieve better use of hardware resources and fast bootups for OS-level executions. Containers help run multiple times several copies of an application on the same hardware, helping in cost reduction [1].

The container has become a lot popular as organizations shift towards cloud-native and hybrid multicloud environment development.

A. Docker

Docker works as a containerization platform that serves as PaaS (Platform as a service) product that uses cloud-based technology for implementing operating system virtualization for software package deliveries. Docker is simple to set up in a cloud environment and integrates with a number of great cloud service providers, including Google, Amazon Web Services, Microsoft Azure, OpenShift, and CoreOS. [1]. Docker is so popular that 'docker' and 'container' terms are used as synonyms. But before docker (released in 2013 for the public), container-related technologies were available named LinuXContainers(LXC) (released in 2008) was implemented in the Linux Kernel, with achieving full virtualization for a single Linux instance. LXC is still in use, but newer technologies using Linux kernel has taken over and are more popular (e.g., Ubuntu, a new, open-source Linux OS with container capabilities). Docker gives the ability to combine application source code with the OS dependencies and libraries which are required to run a code/script specific execution in any environment. Docker has made itself a useful toolkit for developers to build, run, deploy, update, modify and terminate containers using simpler command-based executions and gives automation through a single API. Most of the technical head of industry is already on extreme to use Docker in company standards environment.

978-1-6654-6883-1/22/$31.00 ©2022 IEEE

A.1 Docker Architecture

Following is a Docker architecture for creation of Docker images. Docker is a client-server architecture with three pillar components: (1) Docker Client, (2) Docker Host, (3) Docker Registry. The Fig.1 Depicts Docker client as the primary component for interaction with Docker Daemon. The commands like Docker build, Docker pull, Docker run for building an image file, to pull image from Docker host and to run/execute the Docker and create a container (Docker image runtime). The public repository for storing the images from where they can be extracted is called as Docker registry. The acting server (Docker Host) receives the request from Docker client. The Docker trusted registry (private or public) is used to pull the image requested by Docker registry. So public cloud vendors like Amazon Elastic Container Registry, Microsoft azure Container Registry and Google Cloud Container Registry uses the Docker Hub (Trusted Docker registry) for the image pull or extraction on cloud. The major units of Docker architecture are images, registries, containers, and Docker Host. The vulnerabilities mostly occur in the units of images and containers only. Docker when executed has a scripted file called Dockerfile which is a procedure to generated Docker image. Docker image is a ready process ready to execute and when in runtime the docker image is called as Docker Container [1].

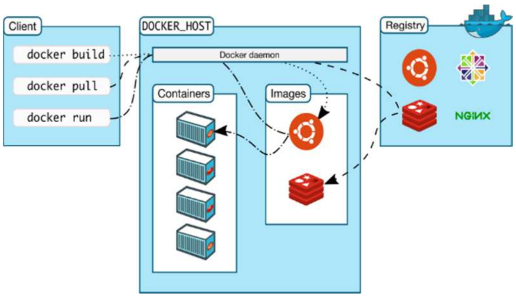

Fig. 1. Docker Architecture

B. Kubernetes

Containerized applications may be complicated to handle. In the production phase, many might require multiple containers at the same instance to run. The number might also be in thousands. So, to take advantage of this situation, docker benefits from using other tools to orchestrate or manage containers on its behalf for handling container runtime environment. One of the popular tools such as these is Kubernetes. Kubernetes is a container management tool that helps the operation of given numbers of containers together. It manages the infrastructural resources for containerized applications like the use of storage resources, network policies, computational power requirements etc. Kubernetes is a container orchestration platform that makes it easier to automate and scale container-based activities in a chain of processes. [2]. It is the fastest-growing, widely accepted containerization platform that is being used by CIOs, CTOs and developers for its functionality, ecosystem, open-source support and portability across multi-cloud services and providers.

B.1 Kubernetes Architecture

Kubernetes is a container management tool with the following infrastructure. As Fig.2 depicts Kubernetes is composed of Control plane (the master node or any worker node who executes and oversees deployment phase). Control plane deploys application across worker nodes. Control planes main units are (1) The API server (2) The Scheduler (3) etcd (4) The Controller Manager. The API server is entry point to the cluster which is implemented on RESTful interface and service provided by it acts as a proxy that iterates services which execute inside cluster. The scheduler assigns worker nodes for a job or module of application. Etcd is a key-value storage system which is distributed and is used to facilitate resources and share config data of cluster. The Controller Manager is a process which associates with several controllers for the smooth work to happen. The worker node has following components:(1) Docker, (2) Kubelet, (3) Kube Proxy. Docker is container runtime instance, or any other container runtime can be present. Kubelet provides service of communication between master node and containers on the worker node. Kube proxy helps to iterate client request across containers which needs to configure that process. Pod is the deployed unit in Kubernetes which are containers tightly coupled with shared resources [2].

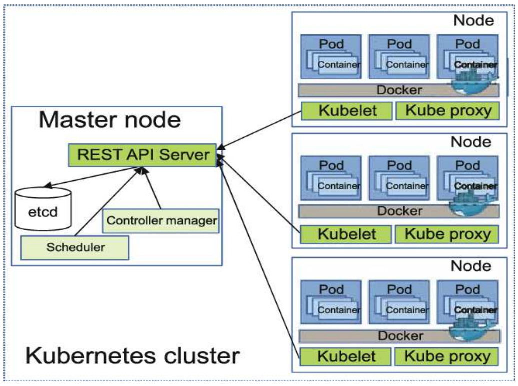

Fig. 2. Kubernetes Architecture

II. Understanding the Problem Statement

So due to this much exposure, containers security measures are that much important. Security vulnerability solutions are needed not only to public clouds but also private cloud setups where an attack might initiate in the cloud environment as Initial Access. As the cloud use is increasing in exponential form, the various security measures applied to containers is falling less to provide the actual structural secure outcomes [3]. Container usage has security vulnerabilities at SDLC phases like deployment and at runtime that must be addressed. There are some major security flaws that flow in containerization like Privilege escalation through containers, container to container attacks, container to container ransom hold, process namespace conflict, cross-site scripting attacks and application-level attacks and malware attacks [4]. Such attacks are major factors that can escalate the severity of compromise within container environments.

For the above problem, adversaries gain initial access when the containers are deployed in the environment, and the containers are misconfigured or have some loosely applied security policies, which may cause vulnerabilities to occur, which an adversary uses for exploiting container process. Misconfigured containers may help evade defenses that the administrator applies to keep security policies intact and avoid containerization exploits. This misconfiguration may occur due to network rules, root or less privileged user access, vulnerable images/OS/executables execution on the containers etc., which is the analysis part of the research study. The proposed model is in accordance with CIS Docker Benchmark [10] In NIST SP 800-190, the National Institute of Standards and Technology (NIST) publishes application container security principles. [5][13], OWASP Container Security Verification Standards [14] and CrowdStrike global threat report 2022.

III. Literature review

The literature survey is majorly dependent on the study and analysis of the containerization process and to understand possible vulnerabilities and attack vectors around the containers while the deployment of the containers in the initial access phase is taking place. Another major area of concentration is to study the possible given security solution models for the misconfiguration of containers in the research survey.

Singh S [4] the paper proposed a study of container-based services and the major role of Docker in container management services. It also explains a study of CaaS (Container As A Service) and the architecture of Docker with the benefits of the same. The objective of the paper is to understand containers and the technological implications behind them and comparison with the virtualization method and utility of containers.

Brad Kelly et al. [6] The author has proposed a model that is used to validate the security of Docker images throughout SDLC phases. The continuous Integration/Continuous Deployment (CI/CD) pipeline approach is being followed for validating the Docker image. Further, check on the vulnerabilities detection and analysis report are given as a measure of effectiveness is shown through the approach. The study on static analysis, dynamic analysis and pattern recognizing behavioural aspects are discussed. The tools like CoreOS Clair and Anchor engine are being used in the implementational model to go through static analysis, and future work of dynamic analysis have been proposed.

Pothula, D. R. [3] The paper mentions basic analysis of container security and use cases associated with the security vulnerabilities. The impact of vulnerabilities present in container security has been discussed, which will lay the structure for controls, best practices and securing methodologies.

A control map is proposed for runtime containers with a security hardening technique. Experiments to prove control map efficiency is given with comparison to container deployment defaults is being given with results confirming the thesis. The objective discussed by the author is to achieve container security control map practices that should help achieve lower vulnerabilities impacting the container's security while in the deployment and runtime phase. Analysis practices discussed for detection and mitigation of the problem too. The author successfully was prepared to illustrate that the security control map approach for containers is effective. that is proposed in the paper. Following Table 1. is a brief description of common threats present in the containers. These are most widely seen attack surfaces targeted by the adversaries while trying to penetrate the security of containers deployment.

TABLE I. COMMON CONTAINER THREATS

Sr No	Common Threats to Containers	Descriptions
1	Privilege Escalation	Gaining Root level access can make this happen
2	Container to Container Capture	If one container is compromised the other containers can be captured without root access also
3	Denial Of Attack (DOS) within Containers	Seizing of all resources can cause this and eventually can take down the system
4	Process Namespace Conflicts	Sharing of namespace with host and in multiple containers can cause root level access gain
5	Cross Site Scripting (XSS) and other Application-level attacks	Easy application-level attack on container instances can escalate the situation to total takedown of the system too.
6	Kernel System Calls exploitation	Due to kernel system calls expose, container exploitation is easy and can be used to gain access
7	Address Resolution Protocol (ARP) Attacks	ARP attacks are used to gain access to the host server using this attack for further exploiting the system.

Ahamed, W. S. S. et al. [1] The author proposed a methodology named vulnerability centric approach (VCA) for securing the docker images. The VCA helps to identify vulnerabilities in Docker images and assess them. With the help of a use case study of already existing vulnerabilities, the security audit is facilitated of Docker container images. The proposed use cases are analyzed against requirements of the OWASP Container Security Verification Standards.

The author has focused on creating use cases as a tool for security auditing for the development, deployment, and maintenance phases of the cloud environment. The objective of the author with the proposed paper is to make checks of security audit requirements of OWASP Container Security verification Standards with the help of NIST SP 800-190 Application Container Security Guide. Following in the Table. 2 is the use case study for validation of container instances and best practices that have been promoted to create safe deployment of containers. It is collection of most important use cases suggested by author and verified as system instances and through analysis of the given use case study.

TABLE II. USE CASE STUDY ESSENTIALS CHECKS

Sr No	Phase	Use Case Description
1	Docker Base Image Inspection	1. validate if image is official image 2. Validate base image is trustworthy 3. Validate the content of the image 4. Removal of unessential packages 5. Restrict the build cache usage
2	Dockerfile Configurations	1. Validate base image version 2. Running install and update checks 3. Use of command COPY instead of ADD 4. Container image status check 5. Restrict Dockerfile with private info as hardcoded values
3	Image Authentication	1. Image validated with Docket Container Trust (DCT) 2. Registry authentication
4	Image Authorization	1. Using one user one container policy 2. Restrictions on Permissions for user ID is maintained

Reeves M. [8] has given some concrete study on the security of containers by conducting vulnerabilities test on runtime containers. This was an analysis of the occurrence of container vulnerabilities and considering the impact of the exploits on the containers to see the possible vandalism. So Common vulnerabilities and Exposure (CVEs) and Proof of Concept (PoC) for exploits were utilized by the author to come up with the framework that helped to analyze PoC exploits by steps. The author created a taxonomy of classes to better understand CVEs and the cause of container escapes. The implementational model consisted of a namespace kernel feature that imprints container's super privileged user to common user on the container host. Such measures were able to achieve security hardening in containers.

Durante [9] focused on static code analysis for container security, with other aspects of the security (CIA triad). The author performed a well-defined analysis of vulnerabilities present in Docker, and all the security upgrades were applied to it. To understand how the vulnerabilities affect the containers, static code analysis was implemented. The output of the paper was the key finding that vulnerabilities can be detected by behavioural software analysis using regression testing and robustness testing. Following Table. 3 describes CVEs that most affects containers and discusses effects cause and consequences.

TABLE III. COMMON VULNERABILITY EXPOSURES,CAUSES EFFECT AND CONSEQUENCES

CVE	Cause	Effect	Consequences
2014-0047	Unprotected Resources	Exposed System	Gain Information
2014-3499	Incorrect Permission Management	Exposed System	Gain Privileges
2014-5277	Incorrect Recovery Mechanism	Restriction Violation	Gain Information
2014-5279	Unprotected Resources	Exposed System	Gain Privileges
2014-5280	Incorrect Permission Management	Restriction Violation	Execute code
2014-8178	Incorrect Configuration	Exposed System	Execute code
2014-5282	Improper Validation	Exposed System	Bypass
2014-6407	Incorrect Permission Management	Restriction Violation	Execute Code

Tunde et al. [7] analyzed static and dynamic attack detection models in Docker containers using 28 vulnerabilities. A comparison was formed based on these detection schemes amongst vulnerabilities, and a conclusive element was declared that to improve detection rate, an anomaly detection scheme works better than static analysis. An open-source tool named CoreOS Clair was used to do static analysis. The unsupervised anomaly detection scheme was facilitated to evaluate dynamic analysis.

IV. PROPOSED SYSTEM

The proposed model implements as the deployment containers are ready to be processed in a runtime environment, first all the configuration settings will be check against the model for any misconfigurations during deployment of the container. The model iterates all the security policies and guidelines that needs to be followed are getting reflected in the deployment settings or not. If the settings do not meet the requirements of secure deployment, then the process will give an alert to revaluate the deploying of the container. If the container has all settings as per security guidelines and is secure, then a green flag will be given to let the deployment proceed. This way in Initial phase of deployment only if there are any misconfigurations that may occur can be detected and corrected accordingly. The following Fig. 3 depicts the data flow diagram of the proposed system by the author. The container check model iteration amongst the security check is mentioned in it.

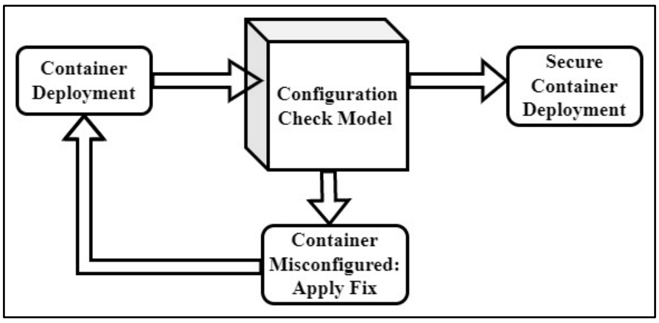
Container Deployment
Configuration Check Model
Secure Container Deployment
Container Misconfigured: Apply Fix

Fig. 3. Data Flow Diagram

Steps Incurred to get the secured Deployment of containers:

1. Deploy container with possible configuration settings as required by the user.
2. Check the configurations against the configuration check model for possible any misconfigurations.
3. If the model gives any notification for required reconfigurations, then apply it and re-deploy container.
4. If model does not need to reconfigure then the deployed container is assumed to be secured.

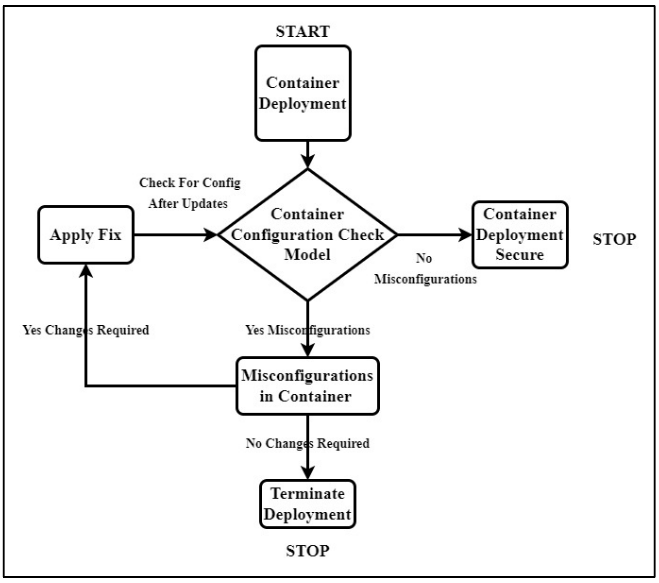

Fig. 4. Flow Chart Diagram

As depicted in the DFD diagram Fig. 3 and Flow Chart Diagram Fig. 4, the deployment of multiple of containers can be reconfigured using this configuration check model for knowing that the container deployment is secured or not. The configuration check model comprised of multiple checks that needs to be addressed as a flag of misconfigurations in containers. CIS Docker Benchmark latest addition gives the standard auditing checks for a secure docker container deployment. This documentation will be used for scripting the configuration check model to validate the deployed containers. CIS Docker Benchmark [10] is a resource that is required for overall understanding of the misconfigurations that occur. The proposed model is based on these auditing checks which incur the possible secure docker container deployment.

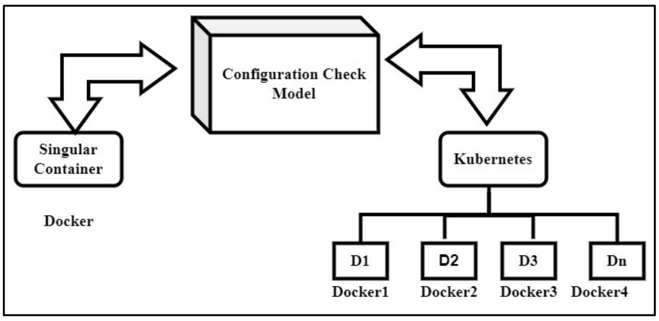

Fig. 5. Configuration Check Model on Docker and Kubernetes

As Fig. 5 depicts the Configuration Check Model can be checked on a singular Docker container or in the cluster of K8s(Kubernetes) where multiple of Docker containers are being executed. The Model can iterate to various industrial cloud virtual platforms to make it more secure while deploying containers.

Algorithmic Approach

```
While (Containers Deployed)
        Check for all possible container configurations
        For (Container Check Model)
                If (Container Config == Misconfig)
                   fix Secure Config container & redeploy
                End If
                Else If (Container Config != Misconfig)
                   The deployed container is secured and
                   no need to fix
                End Else If
        End For
End While
```

Algorithmic Approach

The above-described algorithm gives a brief idea on the model working with respect to configuration check model. The algorithm can be utilized against deployed containers and checked for any misconfigurations that occur during the deployment of the containers in Docker, Kubernetes or any other cloud virtual environment based on container technology.

V. Conclusion and Future Scope

Docker containers are widely used for deployment, development, and production phases of the SDLC (Software Development Life Cycle) phase in the era of a quick, productive project delivery lifecycle. With the efforts to distribute them as a service in the cloud environment, container technology transformed the way applications were packaged and delivered. With such wider use of the technology the security issues are also a major concern associated with it. Thus, analysis of the problem with misconfigured containers has been the main objective of this research paper. The attack vectors and environment variables affecting the problem has been analyzed and with the literature survey, to follow proper execution of the deployment of the containers is being studied from standard security guidelines using CIS docker benchmark report. Also, efficient, and smooth deployment of containers require some secure practices that are needed to be followed which is major study concern for the authors in future work that is entitled. To analyze detection and mitigation of the container's misconfiguration during the deployment of containers and forming an implementational model is the future work presented by the author.

References

[1] Ahamed, Waheeda Syed Shameem, Pavol Zavarsky, and Bobby Swar. "Security Audit of Docker Container Images in Cloud Architecture." In *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pp. 202-207. IEEE, 2021.

[2] Medel, Víctor, Carlos Tolón, Unai Arronategui, Rafael Tolosana-Calasanz, José Ángel Bañares, and Omer F. Rana. "Client-side scheduling based on application characterization on kubernetes." In International Conference on the Economics of Grids, Clouds, Systems, and Services, pp. 162-176. Springer, Cham, 2017.

[3] Pothula, Dharmanandana Reddy, Krishna M. Kumar, and Sanil Kumar. "Run Time Container Security Hardening Using A Proposed Model Of Security Control Map." In 2019 Global Conference for Advancement in Technology (GCAT), pp. 1-6. IEEE, 2019.

[4] Singh, Sachchidanand, and Nirmala Singh. "Containers & Docker: Emerging roles & future of Cloud technology." In 2016 2nd international conference on applied and theoretical computing and communication technology (iCATccT), pp. 804-807. IEEE, 2016.

[5] Souppaya, Murugiah, John Morello, and Karen Scarfone. Application container security guide. No. NIST Special Publication (SP) 800-190 (Draft). National Institute of Standards and Technology, 2017.

[6] Brady, Kelly, Seung Moon, Tuan Nguyen, and Joel Coffman. "Docker container security in cloud computing." In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0975-0980. IEEE, 2020.

[7] Tunde-Onadele, Olufogorehan, Jingzhu He, Ting Dai, and Xiaohui Gu. "A study on container vulnerability exploit detection." In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 121-127. IEEE, 2019.

[8] Reeves, Michael, Dave Jing Tian, Antonio Bianchi, and Z. Berkay Celik. "Towards Improving Container Security by Preventing Runtime Escapes." In *2021 IEEE Secure Development Conference (SecDev)*, pp. 38-46. IEEE, 2021.

[9] Duarte, Ana, and Nuno Antunes. "An empirical study of docker vulnerabilities and of static code analysis applicability." In *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, pp. 27-36. IEEE, 2018.

[10] CIS," CIS Docker Benchmark", Jan.2022 Accessed on: Jan. 2,2022 [Online] Available: https://www.cisecurity.org/benchmark/docker

[11] MITRE ATT&CK," Deploy Containers", Mar 29. 2021 Accessed on: Feb. 3,2022 [Online] Available: https://attack.mitre.org/techniques/T1610/

[12] MITRE ATT&CK," Container Administration Command", Mar 29. 2021 Accessed on: Feb. 3,2022 [Online] Available: https://attack.mitre.org/techniques/T1609/

[13] NIST, "NIST SP 800-190" Sep.2017 Accessed on: Feb. 19,2022 [Online] Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-190.pdf

[14] OWASP, "OWASP Container Security Verification Standards "July 26.2019 Accessed on: Mar. 19,2022 [Online] Available: https://github.com/OWASP/Container-Security-Verification-Standard/releases/download/v1.0/container_security_verification_standard_v1.0_en.pdf