A Layer-Aware Container Image Threat Scanner for Automated Vulnerability Detection and Forensic Analysis

M S Sunitha Patel[1], Assistant Professor and B D Sona[2], PG Student

[1,2] Department of Computer Science and Engineering, SJCE, JSS Science and Technology University, Mysuru, Karnataka, India
sunithapatel@jssstuniv.in, sonashetty1623@gmail.com

Abstract. Container technology has quickly gained popularity in the realm of cloud computing and DevOps, and there have been many security problems associated with its integration, particularly when it comes to security vulnerabilities that exist in container images such as those created by Docker. There are many publicly accessible container images that contain obsolete software, which can make their associated systems prone to cyberattacks. Often, manual testing for vulnerabilities is ineffective and not applicable on a wide scale. This paper seeks to present a solution in the form of Container Image Threat Scanner with Layer-Aware Forensics. The suggested system combines automatic image capture, vulnerability scan, layer identification, and reporting process within the DevSecOps process. With the use of Trivy vulnerability scanning tool, the system detects vulnerabilities such as CVE that arise due to operating system or dependency application packages. Vulnerabilities are then classified depending on their levels of risk into Critical, High, Medium, and Low categories for effective evaluation. In addition, the system performs layer-based forensics to determine the exact layers contributing to the vulnerabilities. The implementation shows how automation could improve security monitoring, reduce manual effort, and support container security procedures in modern cloud-native architectures. Experimental data proves that the introduced system is capable of discovering vulnerabilities in the container images and generating complete security reports ready for CI/CD integration. This system will improve proactive threat discovery and contribute to creating reliable containerized applications.

Keywords: container security, Docker, vulnerability scanning, Trivy, CVE, layer-aware forensics, DevSecOps, CI/CD pipeline

1 Introduction

The rapid advancements in cloud computing and microservices architectures have significantly hastened the uptake of containerization tools. Containers provide a more efficient method compared to VMs as they enable applications to utilize the underlying operating system without requiring VM creation. From the list of available containers, Docker emerges as one of the most prominent platforms for developing, deploying, and packaging applications [1], [2]. However, while containers are advantageous, they present several security issues that cannot be overlooked. The first issue involves the vulnerability of the container images themselves. This arises due to such aspects as the use of out-of-date libraries, configuration errors, or unnecessary code included in the image. The said vulnerabilities could be taken advantage of by an attacker who seeks to gain access [3]. Recent studies reveal that a considerable number of the container images that are available in the public domain contain identified security weaknesses, such as High and Critical CVEs. For instance, Shu et al [4] found that many container images available in public registries have various vulnerabilities, which proves the importance of using secure containers to mitigate the dangers involved. This stresses the requirement for automatic security systems that continuously monitor and analyze container environments. In order to overcome these problems, several scanning techniques like Trivy have emerged. These software packages perform an in-depth analysis of container images by checking their installed software against reliable vulnerability databases like the National Vulnerability Database (NVD) [5]. However, manually performing vulnerability scans is a tedious and error-prone process.

The suggested research presents a system that automates the process of scanning containers in order to detect any kind of vulnerabilities. The objective is to:

- automatize the process of scanning images inside containers;
- identify and categorize vulnerabilities based on their level of threat;
- create security reports;
- strengthen security integration in DevSecOps practices.

To ensure that the current cloud computing framework is addressed by this research, realistic methodologies and approaches are used. The format of this paper can be summarized as follows: Literature Review can be found

in Section 2, System Architecture in Section 3, Implementation Approach in Section 4, Experimental Results in Section 5, and Conclusion in Section 6.

2 Related Works

Security in containers has become one of the most important areas of study because of the quick adoption of containers in cloud computing environments. Many research studies have been carried out on the weaknesses and threats in such systems.

Sultan et al. [6] have provided an extensive review of the numerous risks that exist when it comes to securing the container, highlighting some of the critical areas that need to be covered in terms of threats. They include image risk, container execution risk, orchestration risk, and host risk, among others.

An analysis of the Docker security aspect was performed by Combe et al. [7], who pointed out that despite the benefits in terms of efficiency offered by containers, they operate on the same kernel as the host machine, making them susceptible to attacks leading to a violation of the isolation principle.

According to Kaur et al. [8], there have been empirical investigations of container images used in scientific purposes, and many container images were observed with hundreds of vulnerabilities due to outdated and unnecessary package versions. It was found that vulnerabilities decreased after upgrading and cleaning the package versions.

Wist et al., in their study [9], performed a comprehensive analysis of Docker Hub images and found that these images were highly susceptible to vulnerabilities. In other words, there could be some security concerns in highly regarded images on Docker Hub.

Container Escape Vulnerability is discussed by Aktolga [10], where the ability of an attacker to exploit containers to bypass their isolation and gain access to the underlying operating system is considered a major threat.

Recently, Chennareddy and Ali [11] have done research on the methods used to find vulnerabilities in Docker systems, comparing static and dynamic approaches. They have concluded that using several approaches together can increase the efficiency of such analysis.

Additionally, the literature review concerning container security threats has classified container security threats, and several gaps have been discovered in the current literature. Sroor et al. [12] reviewed over 100 papers, highlighting the importance of having automatic security integration.

In general, based on the available literature, container images are vulnerable most of the time, and the current approach to dealing with such issues is not sufficient. In fact, although several tools and methodologies have been introduced in recent times, there still seems to be a need for an automated solution that is effective and practical. This is what makes the proposed solution interesting.

3 System Architecture

The proposed Container Image Threat Scanner with Layer-Aware Forensics system will help ensure automatic vulnerability scanning and forensics capabilities for Docker container images. The architectural design is based on the concept of integrating automatic vulnerability scanning and forensics into the process of container image security.

The overall architecture consists of six key components:

3.1 Container Image Repository

A container image is held in a repository, which may be public, such as Docker Hub, or private. The images include application binaries, libraries, runtime environment, and OS packages required for running them. Since potential threats could lie in any layer, the repository is the main starting point for conducting security assessments [13].

3.2 Image Acquisition Module

This module plays a very important role in acquiring the images of containers either locally or remotely. The automation of this process is achieved by utilizing the Docker API to retrieve these images. Once the image is downloaded, its metadata is extracted; this metadata includes information like the image ID, operating system type, tags, etc.

3.3 Vulnerability Scanning Engine

The vulnerability scanning engine performs a static analysis of the container image for securing it. The Trivy scanning tool is used to evaluate the vulnerabilities in the container image considering the packages and libraries installed in the operating system and the programming languages [14]. Identified vulnerabilities are correlated to the database entries like NVD and CVE.

The vulnerabilities are classified according to their severity level as follows:

- Critical
- High
- Medium
- Low
- Unknown

3.4 Forensic Investigations Based on Layered Approach

Instead of using the traditional scanning methods, this method uses the forensic approach based on layering. In fact, every Docker image is made up of layers, with each layer consisting of files, packages, and configurations. The forensic tools can discover the following issues:

- Vulnerabilities in layers
- Files installed maliciously
- Issues with package configurations
- Additional software installation
- Duplicate inclusion of dependencies

Each time there is an issue related to security, its source layer should be discovered [15].

3.5 Report Generation Module

Once scanning and forensics have been performed, the system will automatically generate comprehensive security reports. Such reports will comprise the following information:

- Vulnerability summary
- Vulnerability CVE number
- Security risk levels
- Vulnerable package list
- Vulnerable image layers
- Mitigation procedures recommended

Such reports can be outputted in JSON, HTML, and PDF format.

3.6 Integration of DevSecOps Module

The following module can be introduced to integrate the system into the CI/CD pipeline to allow for the automatic continuous monitoring of the security level of the container. The automation process will perform scanning both in the building phase and during the deployment phase to guarantee identification of any vulnerabilities before deployment.

The processes flow within the proposed module are as follows:

1. Upload/Selection of container image by user
2. Image acquisition module searches for metadata within the container image
3. Vulnerability scanning for package and dependency
4. Forensic engine scanning for layers with vulnerabilities
5. Classification based on the vulnerability's severity
6. Automatic generation of security report
7. Notifications sent to CI/CD

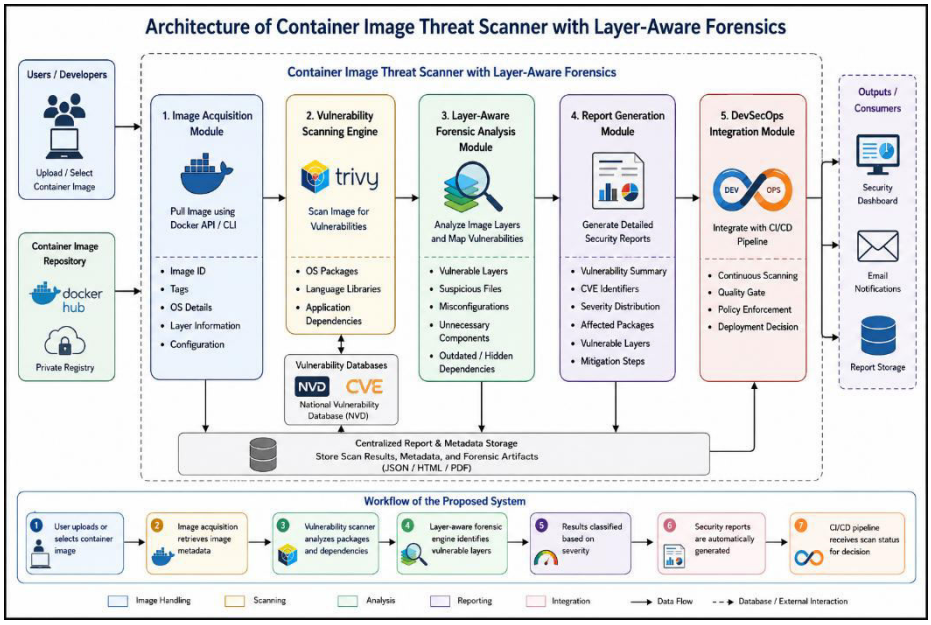

Fig. 1. Container image threat scanner architecture diagram

4 Methodology

The deployment of the suggested Container Image Threat Scanner with Layer-Aware Forensics is oriented toward the automated evaluation and analysis of vulnerabilities within Docker container images through the use of open-source security applications and scripting approaches. The approach aims at ensuring effective detection of vulnerabilities and visibility of threats associated with layers in the image.

4.1 Implementation Environment

The following software/hardware components/tools that will allow implementation to take place are:

- Operating System: Linux or Windows with WSL (Windows Subsystem for Linux)
- Containerization: Docker
- Programming Language: Python
- Vulnerability Scanner: Trivy
- Processing JSON files to create an HTML report

Python was selected due to its potential for automation, subprocess management, JSON file parsing, and interaction with different software tools. On the other hand, Docker helps to manage containers efficiently, while Trivy assists in identifying vulnerabilities within the container images [17, 18].

4.2 Image Scanning Procedure

The scanning procedure begins when the user defines a Docker container image that will be analyzed for vulnerabilities. The suggested system automatically fetches the image from the user's local host or remote container registry, and then starts scanning the vulnerabilities present in the image by using Trivy. The scan process is initiated programmatically by using the subprocess library in Python through the following commands:

trivy image nginx:latest

In the scanning process, Trivy checks for:

- OS packages
- App libraries
- Language dependencies
- Configurations

Trivy obtains all the required data related to the vulnerabilities like:

- CVE ID
- Name of the package
- Version being used

- Fixed version
- Severity of the vulnerability

The collected data about the vulnerabilities is compared against public databases including NVD and CVE [19]. The scanned data is stored in the form of a JSON file.

4.3 Layer Extraction Analysis

A docker container is made up of a series of immutable layers created during the build phase. With security concerns related to vulnerabilities in particular layers coming into light, the current research uses the concept of layer extraction forensics to establish and analyze sources of vulnerability in relation to the Docker layers. The forensic engine makes use of the following command for layer analysis:

docker history

After getting the layers from the Docker image, the forensic engine proceeds to check for any vulnerabilities in those layers and maps each of them to the particular build step. Forensic analysis helps us map several things as follows:

- Vulnerable layers in terms of software
- Duplicate software dependencies
- Malformed file and service layer
- Legacy components layer
- Suspicious layer addition to the layer

The above method helps trace any vulnerability in an image down to the Docker file build step that causes it.

4.4 Automated Reporting

After carrying out vulnerability scans and forensics on the systems, automated reports are generated for both developers and security personnel, detailing:

- Number of discovered vulnerabilities
- Level of severity of the vulnerabilities
- CVEs
- Names of vulnerable packages
- Layers within the Docker images that are impacted
- Mitigation strategies

These reports are available in different formats, such as JSON, HTML, and PDF. Automation of reporting leads to reduced time spent manually analyzing reports and faster mitigation of vulnerabilities [7].

4.5 Continuous Integration Flow

To facilitate DevSecOps, the suggested system will be able to scan vulnerabilities in Continuous Integration/ Continuous Deployment (CI/CD) pipelines.

Compatible with:

• Jenkins
• GitHub Actions
• GitLab

Upon building a new container image, the CI/CD pipeline will automatically initiate the scanning process. With high/ critical severities found, it can automatically block deployment if necessary to avoid pushing insecure container images to production.

This allows for more robust DevSecOps practices by integrating security checks into the software development life cycle [16].

4.6 Threat Detection Algorithm

Threat detection algorithm will follow the following steps:

Step 1: Input the docker image name.
Step 2: Obtain the docker image from the repository locally available or Docker Hub.
Step 3: Perform vulnerability analysis using Trivy.
Step 4: Analyze the vulnerability information from JSON output file produced by the tool.
Step 5: Extract Docker image layer metadata.
Step 6: Correlate identified vulnerabilities with corresponding layers.
Step 7: Categorize identified vulnerabilities according to severity levels.
Step 8: Develop forensic security report.
Step 9: Output scanning result to users.

Automating such tasks minimizes the dependency on manual work and increases the efficiency in monitoring of containers' security. Integrating vulnerability analysis with forensic analysis that considers layers helps achieve better understanding of potential threats related to container images.

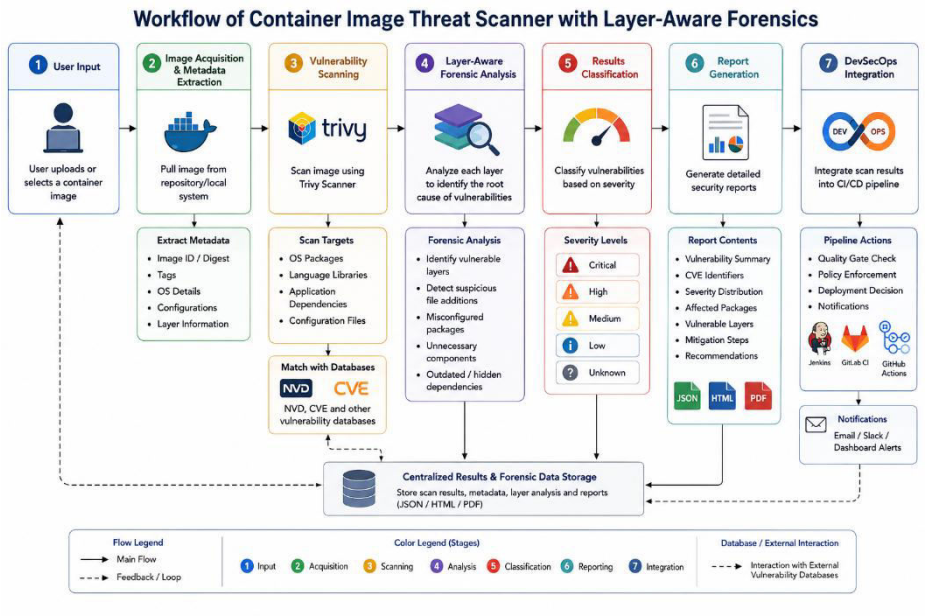

Fig. 2. Container image threat scanning workflow

5 Experimental Results

Testing of the proposed approach was carried out through the use of a number of publicly available images for Docker containers to test the ability of the approach not only to detect vulnerabilities but also its forensic value.

5.1 Experimental Setup

All experiments were carried out using the following hardware:
• Intel Core i5 CPU
• 8GB RAM
• Docker Engine
• Python environment
• Trivy vulnerability detector
Several different Docker images such as Ubuntu, Nginx, Node.js, and MySQL were analyzed.

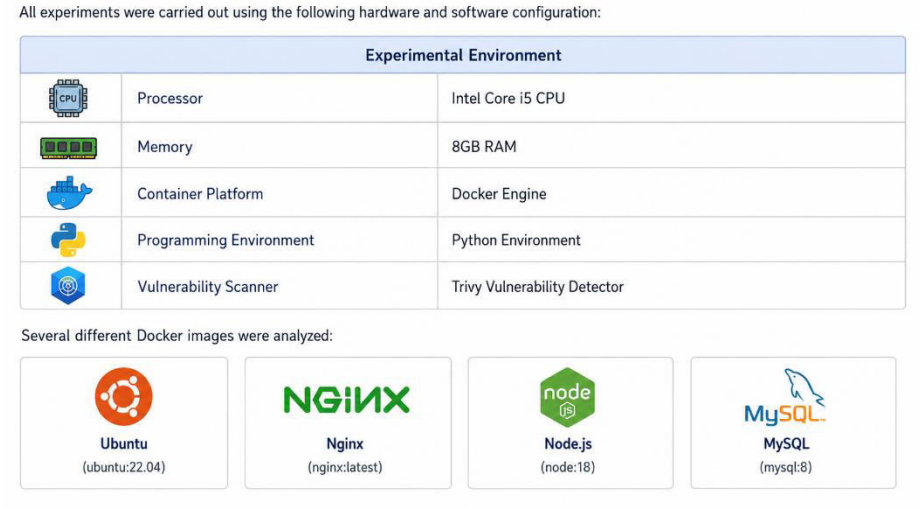

Fig. 3. Experimental setup and environment configuration

5.2 Vulnerabilities Detected

The system successfully detected several vulnerabilities among the various container images used. From the results, a number of images were found to have outdated packages that could be associated with critical or high-level CVEs. Examples include:

- Outdated system libraries in Ubuntu images
- Vulnerable npm packages in Node.js images
- Configuration issues in MySQL images

The vulnerabilities were successfully categorized according to their severity.

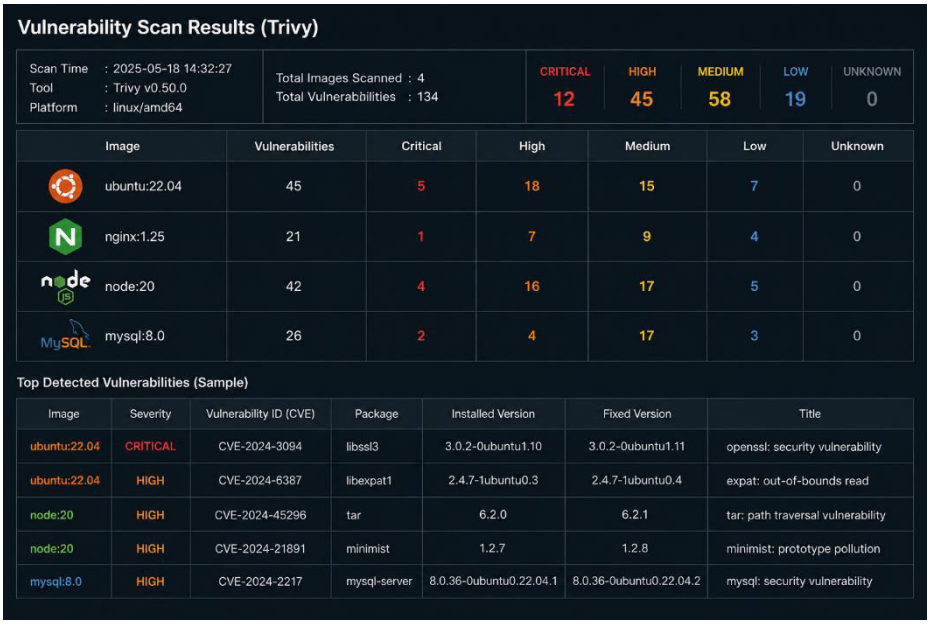

Fig. 4. Vulnerability scan report dashboard

5.3 Layer-based Forensic Results

The forensic analysis component was able to detect the layers that caused the vulnerabilities. The analysis showed that:

- The majority of the vulnerabilities were created by base layers of the images.
- More software installations increased the vulnerability space.
- Superfluous packages also led to vulnerability creation.

The layer mapping process proved more effective compared to conventional vulnerability scanners.

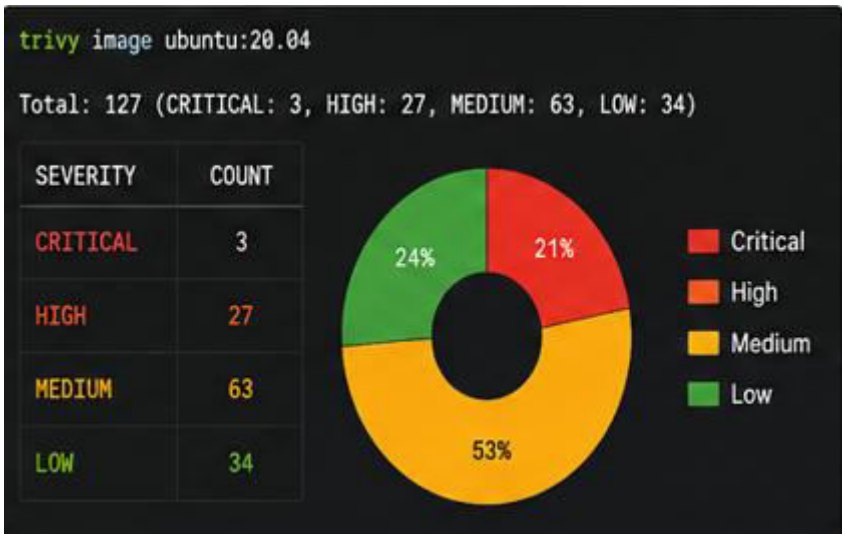

Fig. 5. Vulnerability analysis of Ubuntu container

5.4 Performance Assessment

Assessment of performance depended on the following criteria:

- Size of the image
- Installation of the package
- Internet connection for updating the vulnerability database

Small images entailed faster scan times, while large enterprise images entailed relatively longer scan times. However, automated testing reduced the need for manually conducting performance evaluations.

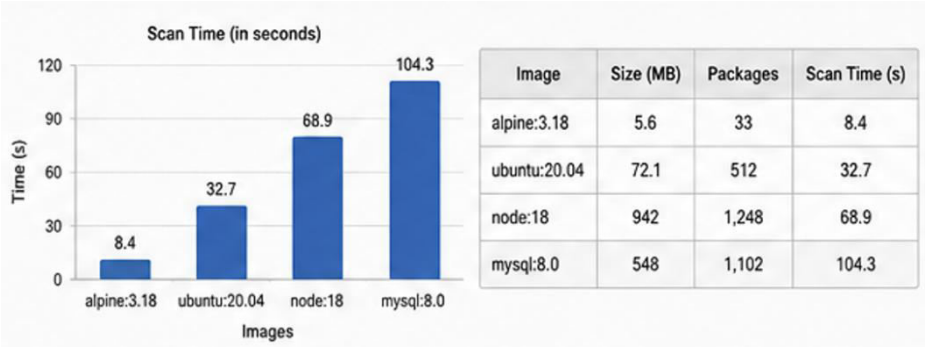

Image	Size (MB)	Packages	Scan Time (s)
alpine:3.18	5.6	33	8.4
ubuntu:20.04	72.1	512	32.7
node:18	942	1,248	68.9
mysql:8.0	548	1,102	104.3

Fig. 6. Performance assessment report summary

5.5 Discussion

Results obtained from experimental analysis demonstrate that the designed system is an efficient approach to incorporating vulnerability assessment with forensics.

Layer-aware analysis is demonstrated as a reliable method for investigating causalities and assessing the overall security stance of container platforms.

The methodology can be used with:

- Native cloud applications
- DevSecOps frameworks
- CI/CD systems
- Enterprise containers

6 Conclusion

The use of containers has become an integral part of modern cloud-native architectures. However, the vulnerabilities that exist within container images pose serious security threats to the extent that they can lead to system breaches. In this paper, we propose Container Image Threat Scanner with Layer-Aware Forensics that incorporates the concepts of automatic vulnerability scanning, forensic analysis of container layers, and automation in DevSecOps. Our proposed solution employs Trivy to detect vulnerabilities in containers and extends traditional vulnerability scanning by tracing vulnerabilities to their respective layers. The experimental verification proved that this system is capable of detecting vulnerabilities, analyzing them in terms of severity, and offering valuable forensics information. The automation of reports generation and CI/CD pipelines make the system useful for security monitoring practices.

Areas for future research could include:

- Runtime monitoring
- Anomaly detection using AI techniques
- Kubernetes security integration
- Automatic repair for identified vulnerabilities
- Behavior analysis of containers

The system presented above improves container security through its practical, scalable, and automated design.

References

1. Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." Linux j 239, no. 2 (2014): 2.
2. Pahl, Claus. "Containerization and the paas cloud." IEEE Cloud Computing 2, no. 3 (2015): 24-31.
3. Zhan, Dongyang, Zhaofeng Yu, Xiangzhan Yu, Hongli Zhang, and Lin Ye. "Shrinking the kernel attack surface through static and dynamic syscall limitation." IEEE Transactions on Services Computing 16, no. 2 (2022): 1431-1443.
4. Shu, Rui, Xiaohui Gu, and William Enck. "A study of security vulnerabilities on docker hub." In Proceedings of the seventh ACM on conference on data and application security and privacy, pp. 269-280. 2017.
5. National Institute of Standards and Technology (NIST), “National Vulnerability Database (NVD).” [Online]. Available: https://nvd.nist.gov
6. Sultan, Sari, Imtiaz Ahmad, and Tassos Dimitriou. "Container security: Issues, challenges, and the road ahead." IEEE access 7 (2019): 52976-52996.
7. Combe, Theo, Antony Martin, and Roberto Di Pietro. "To docker or not to docker: A security perspective." IEEE Cloud Computing 3, no. 5 (2016): 54-62.
8. Kaur, Bhupinder, Mathieu Dugré, Aiman Hanna, and Tristan Glatard. "An analysis of security vulnerabilities in container images for scientific data analysis." GigaScience 10, no. 6 (2021): giab025.
9. Wist, Katrine, Malene Helsem, and Danilo Gligoroski. "Vulnerability analysis of 2500 docker hub images." In Advances in Security, Networks, and Internet of Things: Proceedings from SAM'20, ICWN'20, ICOMP'20, and ESCS'20, pp. 307-327. Cham: Springer International Publishing, 2021.
10. Aktolga, İlter Taha. "A study on analysis and detection of container escape vulnerabilities in docker." Master's thesis, Middle East Technical University (Turkey), 2024.
11. Ajith, Vishnu, Tom Cyriac, Chetan Chavda, Anum Tanveer Kiyani, Vijay Chennareddy, and Kamran Ali. "Analyzing docker vulnerabilities through static and dynamic methods and enhancing iot security with aws iot core, cloudwatch, and guardduty." IoT 5, no. 3 (2024): 592-607.
12. Sroor, Maha, et al. "A Systematic Mapping Study on Risks and Vulnerabilities in Software Containers." arXiv preprint arXiv:2512.11940 (2025).
13. Mouat, Adrian. Using docker: developing and deploying software with containers. " O'Reilly Media, Inc.", 2015.
14. Threatt, Andrew Kyle. "Some analysis of common vulnerabilities and exposures (CVE) data from the national vulnerability database (NVD)." (2024).

15. S. Tak, “Forensic Analysis of Docker Containers and Images,” Digital Investigation Journal, vol. 32, 2020.
16. Humble, Jez, and David Farley. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.
17. Docker Official Documentation
https://docs.docker.com
18. Trivy Official Documentation
https://trivy.dev
19. National Vulnerability Database (NVD)
https://nvd.nist.gov