engineering proceedings

MDPI

Proceeding Paper

Container Security in Cloud Environments: A Comprehensive Analysis and Future Directions for DevSecOps †

Santosh Ugale [1,*] and Amol Potgantwar [2]

[1] Department of Computer Science and Engineering, MET Institute of Engineering, Affiliated to Savitribai Phule Pune University (SPPU), Nashik 422003, Maharashtra, India
[2] Department of Computer Engineering, Sandip Institute of Technology and Research Center Affiliated to Savitribai Phule Pune University (SPPU), Nashik 422213, Maharashtra, India; amol.potgantwar@sitrc.org
* Correspondence: ugalesantosh5@gmail.com; Tel.: +091-7620301496
† Presented at the International Conference on Recent Advances in Science and Engineering, Dubai, United Arab Emirates, 4–5 October 2023.

Abstract: In recent years, the security of containers has become a crucial aspect of modern software applications' security and integrity. Containers are extensively used due to their lightweight and portable nature, allowing swift and agile deployment across different environments. However, the increasing popularity of containers has led to unique security risks, including vulnerabilities in container images, misconfigured containers, and insecure runtime environments. Containers are often built using public repository images and base image vulnerability is inherited by containers. Container images may contain outdated components or services, including system libraries and dependencies and known vulnerabilities from these components can be exploited. Images downloaded from untrusted sources may include malicious code that compromises other containers running in the same network or the host system. Base images may include unnecessary software or services that increase the attack surface and potential vulnerabilities. Several security measures have been implemented to address these risks, such as container image scanning, container orchestration security, and runtime security monitoring. Implementing a solid security policy and updating containers with the latest patches can significantly improve container security. Given the increasing adoption of containers, organizations must prioritize container security to protect their applications and data. This work presents automated, robust security techniques for continuous integration and continuous development pipelines, and the added overhead is empirically analyzed. Then, we nail down specific research and technological problems the DevSecOps community encounters and appropriate initial fixes. Our results will make it possible to make judgments that are enforced when using DevSecOps techniques in enterprise security and cloud-native applications.

Keywords: container security; DevSecOps; DevOps; automation; containerization

Citation: Ugale, S.; Potgantwar, A. Container Security in Cloud Environments: A Comprehensive Analysis and Future Directions for DevSecOps. *Eng. Proc.* **2023**, *59*, 57. https://doi.org/10.3390/engproc2023059057

Academic Editors: Nithesh Naik, Rajiv Selvam, Pavan Hiremath, CS Suhas Kowshik and Ritesh Ramakrishna Bhat

Published: 18 December 2023

1. Introduction

In recent years, containerization has become a widespread technique in software development due to its scalability, efficiency, and portability benefits. However, this technology also has unique security threats that are addressed to ensure modern software applications' protection and integrity. Container security involves securing the various components of container architecture, such as container images, container orchestration platforms, and container runtime environments. These components are vulnerable to several security risks, including malware, unauthorized access, and data breaches. Therefore, it is critical to enforce appropriate security strategies and best practices to safeguard containerized applications against potential threats—the cloud-native applications building block for microservices applications using regular Linux containers. The security of container runtime has become a crucial problem to be solved [1]. To improve the security of the container,

it is necessary to create more robust kernel isolation mechanisms, perform systematic and exhaustive security evaluations on current container methodologies, and build all-encompassing configuration verification tools [2]. The speed and agility of DevOps prevent the use of traditional security methods. Software developers cannot integrate security into CI/CD pipelines for various reasons, notably a lack of understanding [3]. When dealing with security challenges, DevOps raises multiple problems. As software items are delivered to production via DevOps, automation is essential for testing them. Although regarded as an excellent solution, manual security testing and inspections are irrelevant at the delivery pace [4]. Computing environments known as Linux virtual machines and containers mix different IT components while isolating them from the rest of the system. A container's and a virtual machine's difference is that a container is built in small compared to a virtual machine. Due to the lightweight and shared operating system, the container is easily movable across different environments. The lightweight nature of containers allows them to move across different cloud deployment models suitable for multi-cloud applications. Automation is required for repetitive DevOps operations where people are prone to making mistakes or misconfigurations. The cloud is a critical DevOps participant, involving integration, continuous delivery, and security [5].

Open-source software projects driven by businesses must enhance their security practices because protection coverage is poor. For instance, the authors claim that, despite the maintainer's perception that security is crucial, just 6.83% of the 8243 analyzed projects used security automation technologies in their continuous integration pipelines. This implies that security concerns are considered throughout the creation of open-source software. Furthermore, the responsibility for security is often shifted onto the user or integrator by the maintainer, resulting in inadequate security measures being implemented [6].

Adherence to the best security practices to enhance cluster security is advisable. The method of applying authentication and authorization rules prevents malicious attacks. Vulnerability scanning is used to scan K8S components and continuously delivers for vulnerability; logging is the practice of enabling and monitoring K8S cluster logs; namespace separation is the practice of separating namespaces so that the resources of one namespace are segregated from another; access to internal databases can encrypted and restricted using the etcd protocol; continuous updates are the practice of patching container orchestration layer and host operating systems; limited CPU and memory quota limits the CPU and memory to specific pods to avoid over-utilization; SSL/TLS support enables SSL and TLS protocols to ensure secure, encrypted communication; to separate sensitive workloads, sensitive applications are run on individual servers or hosts to reduce the security surface [7]. Conducting detailed research is essential to evaluate the security and features of various cloud service providers. Each provider has strengths and weaknesses regarding security measures, offered services, associated costs, and SOA. Thoroughly analyzing these factors is necessary to make an informed decision for users to select the appropriate Cloud Service Provider (CSP) [8].

Various sources can target cloud-native services [9–11]. There are different methods to safeguard cloud servers from attacks. The initial approach permanently blocks the malicious IP address if the attack is from a static IP address. On the other hand, if the attack originates from a dynamic IP address, the most effective solution is to employ a web application firewall or CDN to block malicious traffic. The future of cloud computing services is heavily reliant on containers. These are closely linked to microservices, with containers providing a uniform approach for deploying microservices [12]. Cloud-native applications are typically built from Linux containers. The characteristic of cloud-native applications makes them impossible to secure with complex overlapping tools spanning development and production. Cloud-native applications are usually built from containers and serverless platforms as services. However, it constantly communicates with virtual machine workload, and data centers complicate security protection strategies [13,14]. VeriDevOps aims to enhance the feedback loop, expediting the validation of security requirements and additional quality attributes [15]. Further, [16] aims to analyze threats and improve

supply chain security using machine learning. The ProSPEC model predicts future critical events and prevents security violations for large container environments, such as ProSPEC model integrated with Kubernetes orchestration. Because of its limitations, identification of essential events, security policy, and event typing must be performed manually. Dynamic learning predictive models are recommended as future research work [17]. The HSAG model is proposed to extract specific risks, attacks, and threats. This proposed method can improve defense efficiency. However, it inevitably introduces time overhead [18].

The container offers many advantages over monolithic deployment but also impersonates particular security challenges that can be difficult to manage. Containers create a more extensive security surface because of the massive number of containers based on different underlying images, which can have various vulnerabilities and security challenges [19]. Another issue is the shared kernel architecture used by containers. A DevOps architecture comprising multiple components, including container runtime, container images, image registry, and the host machine, can pose various security risks Figure 1.

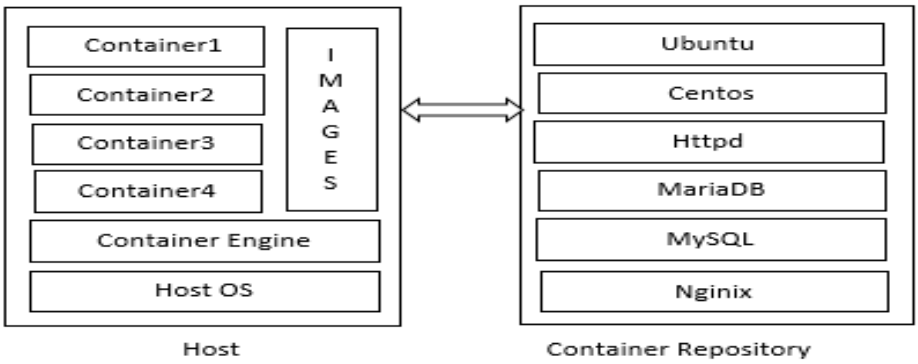

Figure 1. Container deployment architecture.

1.1. The Importance of Container Orchestration and Security Measures

Security measures such as access control, network segmentation, and encryption play essential roles in deployment. Access control helps with proper authentication and authorization for accessing container orchestration. They ensure that only authorized processes or individuals can interact with the system. Network segmentation involves dividing the containerized environment into isolated or virtual network segments. Network policies and firewalls enforce network segmentation between different layers of applications and services. Encryption guarantees the privacy and security of data while it is being transmitted and stored within the containerized environment. Transport Layer Security (TLS) protocol can be implemented for communication between various components like containers, services, and external systems.

1.2. Vulnerabilities and Risks Associated with the Container

1.2.1. Vulnerabilities in Container Images

Container images rely on software packages and libraries. Outdated versions of dependencies introduce security risks. Mitigation includes keeping all software updated to the latest version. Ensuring configuration allows unauthorized access or provides excessive privileges. Images must be built with best practices and hardening guidelines to mitigate this risk.

1.2.2. Misconfigured Container Access

Weak authentication and misconfigured container access can escalate privilege. Containers running with excessive privileges, like root users, increase the impact of security breaches. Insecure network configurations can expose sensitive services and unauthorized access. Proper network segmentation, firewalls, and secure protocol are required to mitigate these risks.

1.2.3. Insecure Runtime Environments

All containers use the shared kernel of the host system, making the entire environment vulnerable to kernel-level exploitation. Keeping the kernel and system updated to the latest version mitigates this risk. Resource limits and isolation are implemented to avoid CPU, memory, and network exhaustion attacks.

2. Materials and Methods/Methodology

2.1. Factors Affecting Container Security

There are various factors affecting container security. A few of them are container vulnerability, host vulnerability, improper deployment, vulnerability of applications or services running on the container, untrusted container images, images from untrusted sources, application attacks on containers, and container attacks on other containers within the same environment. Also, other factors include container-based attacks on the host operating system and host-based attacks resulting from external attacks on the container environment.

2.2. Agile Software Development

Agile development methodologies such as DevOps and Scrum are tailored to address the demand for accelerated growth in software development. However, when it comes to fulfilling security compliance and testing, which can be challenging to execute effectively using automated tests, the high-speed delivery of code is heavily dependent on automation. Lack of attention to security across the development phase and the integration of third-party software systems makes it crucial to incorporate security measures from the starting stage of the development process and perform regular security assessments throughout the CI/CD cycle.

2.3. Security in DevOps

Security testing in the DevOps process poses critical challenges, notably in achieving the necessary level of automation for continuous integration (CI) and delivery (CD). Integrating security into the DevOps process is difficult for ensuring that software and platforms are secure and compliant. However, security testing in DevOps poses unique challenges due to the swift and automated nature of DevOps. Therefore, it is necessary to implement standard security practices in the primary stage of the software development lifecycle (SDLC) to minimize security risks, vulnerabilities, and data breaches.

2.4. Container Security Practices

Industry follows various best security practices when it comes to container security. A few of the best container security practices are listed in the following sections.

2.4.1. Use of Minimal Installation

Installing the operating system in its minimal version is recommended to minimize the attack surface. Minimum packages and libraries reduce the attack surface.

2.4.2. Reduce the Lifespan of Containers

Reducing the lifespan of containers is a recommended security practice for deploying container-based workloads security. This involves limiting the time containers are used and regularly rotating them out with the new ones.

2.4.3. Expose Required Ports

Keeping the required ports open for the application or system to function correctly is essential. However, keeping unnecessary ports open can increase the system's attack surface and potential security breaches.

2.4.4. Use Minimal Base Images and Remove Unwanted Components

Using minimal base images and eliminating unwanted binary and libraries are recommended for container-based workloads. Minimal base images contain only the bare minimum software components, binary, and libraries required to run the application. This can help minimize the container's attack surface and reduce potential security vulnerabilities.

2.4.5. Use Trusted Images

To improve the security of container-based workloads, it is advised to use trusted container images and download from trusted sources. Therefore, the trusted images were verified as secure and free from known vulnerabilities. Using trusted images can help reduce the risks associated with deploying containers, as they are less likely to contain malicious code or security vulnerabilities that attackers can exploit.

2.4.6. Incorporate Image Scanning into the CI/CD Pipeline

Integrating image scanning into the continuous integration and continuous deployment pipeline is a recommended best practice for enhancing the security of container-based workloads. This involves automatically scanning container images for vulnerabilities and other security issues in the development and deployment process.

2.4.7. Secure Software Applications Inside a Container

Securing software applications inside a container is essential for securely deploying container-based workloads. Therefore, various security measures were implemented to safeguard containerized applications from security risks such as data breaches and cyber attacks.

2.4.8. Secure Container Runtime

Securing the container runtime is critical for deploying container-based workloads securely. The container runtime and orchestration are responsible for controlling the lifecycle of containers and providing a secure environment for them to run in. To secure the container runtime, it is essential to implement various security measures, such as using secure container runtimes, using platforms such as Docker, and configuring the container runtime to use appropriate security settings.

2.5. Effectiveness and Limitations of Container Images Scanning Techniques

A vulnerability scanner specifically designed for vulnerability scanning from container images. It usually leverages multiple vulnerability databases, including national (NVD) databases. Effectiveness includes vulnerability detection, where container image scanning techniques excel at detecting known vulnerabilities in container images; timely updates, scanning techniques which typically receive regular updates to their vulnerability databases, ensuring they stay updated with the latest vulnerabilities; integration with CI/CD pipelines, where container image scanning is integrated into continuous integration and deployment; and compliance and policy enforcement, whereby some scanning tools offer features to enforce compliance and security policies. The limitations of these techniques include false positives and negatives, where container image scanning techniques may generate false positives, incorrectly flagging components as vulnerable; performance impact, whereby scanning container images can introduce additional time and resources overhead; and limited coverage, which means that even though scanning techniques cover a wide range of vulnerabilities, they may have limitations in specific areas.

2.6. Research and Technological Challenges Faced by the DevSecOps Community

The DevSecOps community faces various research and technological challenges in ensuring container security.

2.6.1. Vulnerability Detection and Remediation

The rapid pace of container deployment makes it challenging to identify and remediate vulnerabilities effectively, especially considering the many container images and dependencies involved. The proposed methodology provides automated vulnerability scanning and continuously detects vulnerabilities in container images, along with continuous monitoring and automated patching.

2.6.2. Runtime Threat Detection

Detecting and responding to threats in real time within the containerized environment is complex due to the dynamic nature of containerized applications and distributed deployments. The proposed methodology leverages container-specific security technologies like runtime security containers and security frameworks designed for cloud-native applications for runtime threat detection and prevention.

2.6.3. Compliance and Regulatory Challenges

Complying with industry regulations and security standards while embracing the agility of containerized environments can be challenging due to the evolving nature of regulatory requirements. The proposed methodology gives a brief security posture to enforce compliance controls and regularly audit containerized environments for adherence.

2.6.4. Cross-Platform and Multi-Cloud Security

Ensuring consistent security controls and practices across multiple container orchestration platforms and cloud providers can be complex due to architecture, APIs, and security tooling variations. The proposed methodology provides a unified view and management of security across heterogeneous cloud environments.

2.7. Methodology

This section provides a summary of the proposed vulnerability scanning container image methodology. The vulnerability scanning process involves analyzing the container images for known vulnerabilities, such as outdated software components, configuration issues, and other security weaknesses. The results of the vulnerability scans provide valuable information about the security posture of the container images and the potential risks they pose to the systems in which they are deployed Figure 2.

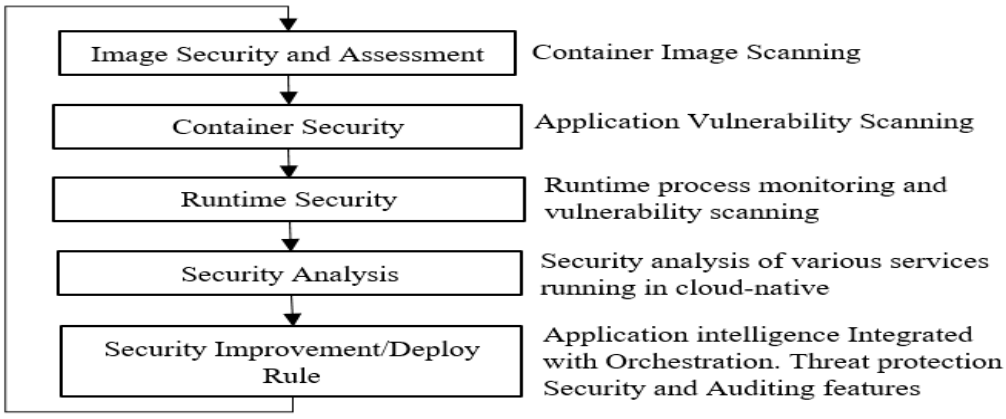

Figure 2. Proposed container security framework flowchart.

2.7.1. Container Security Framework

This section comprehensively describes the proposed framework and approach to improve container security. The previous section outlined how security vulnerabilities, misconfiguration, and loopholes can exist at different stages throughout the lifecycle of a container. During container deployment, misconfigurations or inadequate access controls can introduce security risks. For instance, default passwords may remain unchanged, or

permissions may be misconfigured, leading to unauthorized access or another security breach. Therefore, we proposed a complete container security framework to identify vulnerabilities and threats in the container lifecycle Figure 3.

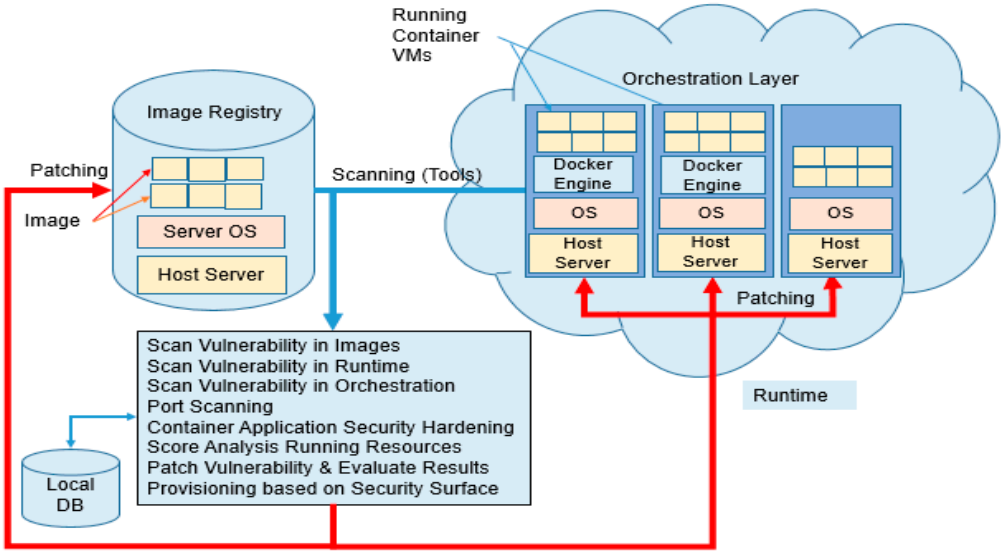

Figure 3. Proposed container security framework.

The proposed container security framework offered to conduct vulnerability scans at various levels, including container images and orchestration layer. It also scans applications running on the containers for misconfiguration, default settings, and secretes. Upon analyzing the results, auto-remediation techniques were applied to mitigate the identified vulnerabilities. Fixing identified issues makes the container environment secure [20].

2.7.2. Host OS Scanning

Host operating system security is securing the host operating system from different types of cyber-attack and malware. As cloud-native application development technology grows rapidly, the host server operating system that hosts a container environment is the critical layer for cyber security. An attack and malware that compromises the host environment could give intruders access to the complete stack. That is why host operating systems need to scan for vulnerabilities and harden them based on CIS Benchmarks or any other remediation technique. In addition, the operating system is protected against improper access control, file integrity, and OS commands. In the proposed framework, the host operating system scans for vulnerability. Scan reports enable us to monitor networks, operating systems, applications, and the container's runtime for security vulnerability.

2.7.3. Runtime Security Container

The third step was to design and implement a Runtime Security Container (RSC) to minimize security attacks. RSC is a security guard between the external world and active running containers. RSC analyzes incoming traffic coming to the live containers and blocks malicious traffic. In addition, RSC optimizes and is trained to protect against various attacks based on behavioral changes. Finally, RSC was integrated with the deployment pipeline and the container production environment to speed up deployment.

2.7.4. The Roles of Runtime Security Monitoring Tools and Techniques in Detecting and Mitigating Threats

Runtime security tools are crucial in identifying and mitigating threats and malicious activities in containerized environments.

- Threat detection: The runtime security system continuously monitors the running containers' behavior in case of suspicious behavior that may indicate a security issue.

Various runtime parameters like resource utilization and network traffic are used to identify potential threats.

- Container image integrity: Runtime security tool verifies the integrity of container images during runtime by comparing the expected state defined by the image.
- Compliance: The runtime security tool helps to enforce the security policy and compliance requirements. The tool can detect default configuration saved passwords and misconfiguration compared with standard procedure.
- Incident response: If a security issue arises, the proposed system provides valuable data for incident reporting.

3. Results

This section summarizes the results obtained from vulnerability scanning of different container images. The vulnerability scanning process involves analyzing the container images for known vulnerabilities. The results of the vulnerability scans provide valuable information about the security posture of the container images and the potential risks they pose to the systems in which they are deployed. A free vulnerability scanner performs vulnerability scans on five different container images. Images were pulled from the Docker hub. Images detail Mariadb with 13,942 stars, Nginx with 18,237 stars, httpd with 4369 stars, MySQL with 5314 stars, and Debian with 4606 stars, all scanned using the free image scanners Trivy and Grype. Testing details (OS CentOS 7.9, 4 CPU, 2 GB RAM, 10 GB Hard Disk). Table 1 and Figure 4. show scan reports using Grype and Table 2 andf Figure 5 Show the Trivy scan.

Table 1. Test results using the GRYPE image scanner.

Vulnerability	MariaDB	MySQL	Httpd	Nginx	Debian
Critical	0	0	2	3	1
High	1	5	16	18	11
Medium	3	9	10	11	4
Low	8	0	6	7	6
Negligible	8	0	67	79	52
Packages	156	142	118	143	96

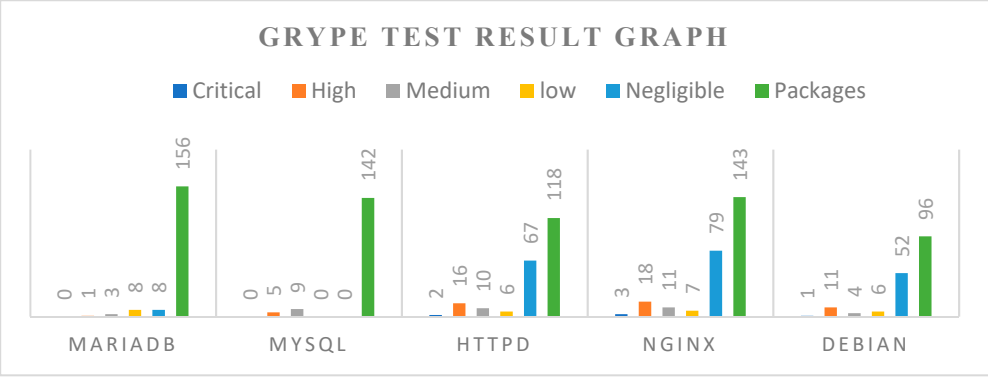

Figure 4. Test result graph using GRYPE.

Table 2. Test result using the Trivy image scanner.

Vulnerability	MariaDB	MySQL	Httpd	Nginx	Debian
Critical	0	0	2	3	1
High	1	3	13	18	11
Medium	3	8	10	15	5
low	16	0	74	92	56
Packages	156	142	118	143	96

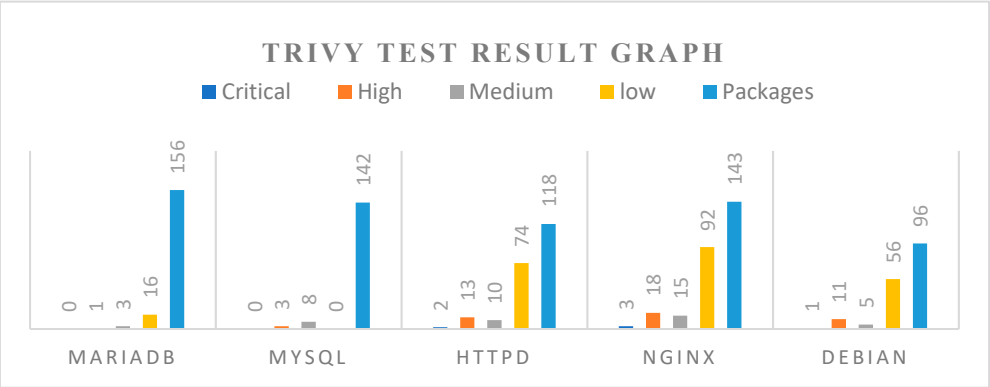

Figure 5. Test result graph using Trivy.

4. Discussion

The results using GRYPE include two critical vulnerabilities that were detected in the httpd image with CVE-2019-8457 libdb5.3 and CVE-2023-23914 libcurl4.2; three critical vulnerabilities that were detected in the Nginx image with CVE-2023-23914 curl, CVE-2023-23914 libcurl4 and CVE-2019-8457 libdb5.3. 3; and one critical vulnerability that was detected in the Debian image CVE-2019-8457 libdb5.3.

The results using Trivy included two critical vulnerabilities that were detected in the httpd image with CVE-2019-8457 libdb5.3 and CVE-2023-23914 libcurl4.2; three critical vulnerabilities were detected in the Nginx image with CVE-2023-23914 curl and CVE-2023-23914 libcurl4 and CVE-2019-8457 libdb5.3. 3; and one critical vulnerability that was detected in the Debian image CVE-2019-8457 libdb5.3.

Implementing an automated dynamic security approach within a CI/CD pipeline can have the following implications and potential trade-offs:

Overhead and development speed: Automated dynamic security approaches introduce additional overhead, such as runtime monitoring and vulnerability scanning. Scanning and monitoring processes require time and other resources for additional RSC containers. False positive results: An automated security approach may generate false positives. Addressing false positives can require additional time and effort and inkling manual verification and validation. The proposed system is fine-tuned regularly to reduce manual verifications. Integration challenges: Integrating an automated security approach in the CI/CD pipeline is challenging—the proposed methodology using automation and scripting helps to streamline the integration process.

5. Conclusions

This paper focused on the security concerns associated with running containers in the cloud. This study examines security threats and risks when deploying containers in cloud environments, including vulnerabilities and attacks on the container orchestration platform, runtime, and host operating system. The paper also explores different security mechanisms and strategies to mitigate security risks, such as container images and host operating system scanning for vulnerability and malware. By studying these security concerns and analyzing the effectiveness of different security approaches, the paper provides insights and recommendations for organizations seeking to secure their container-based workloads in the cloud environment. By following the proposed methodology, it is possible to systematically assess the security posture of container deployment. In addition, the proposed system identifies any gaps in the security controls and configuration of the containers, container runtime, host systems, or the container orchestration platform. Through this process, organizations can proactively address potential security risks and reduce attacks against their container environments. DevSecOps teams can benefit from this work, as it provides a reference architecture, or framework, to make the container environment more secure.

Author Contributions: Conceptualization, S.U. and A.P.; methodology, S.U. and A.P.; formal analysis, S.U. and A.P.; writing—original draft preparation, S.U. and A.P.; writing—review and editing, S.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the article.

Acknowledgments: The authors thank Viraj Nevase, at ESDS Software Solution, for his valuable input and support during this research work. We also thank MET Institute of Engineering Research Center Nashik and ESDS Software Solution for providing resources and support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Subrahmanya, S.V.; Shetty, D.K.; Patil, V.; Hameed, B.M.; Paul, R.; Smriti, K.; Naik, N.; Somani, B.K. The role of Data Science in healthcare advancements: Applications, benefits, and future prospects. *Ir. J. Med. Sci. (1971-)* **2021**, *191*, 1473–1483. [CrossRef]
2. Kumar, S. Reviewing Software Testing Models and Optimization Techniques: An Analysis of Efficiency and Advancement Needs. *J. Comput. Mech. Manag.* **2023**, *2*, 43–55. [CrossRef]
3. Kumar, S.; Gupta, U.; Singh, A.K.; Singh, A.K. Artificial Intelligence: Revolutionizing Cyber Security in the Digital Era. *J. Comput. Mech. Manag.* **2023**, *2*, 31–42. [CrossRef]
4. Arnold, B.; Qu, Y. Detecting software security vulnerability during an agile development by testing the changes to the security posture of software systems. In Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence CSCI, Las Vegas, NV, USA, 16–18 December 2020; pp. 1743–1748.
5. Gokarna, M.; Singh, R. DevOps: A historical review and future works. In Proceedings of the 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 19–20 February 2021; pp. 366–371.
6. Angermeir, F.; Voggenreiter, M.; Moyon, F.; Mendez, D. Enterprise-driven open source software: A case study on security automation. In Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Madrid, Spain, 25–28 May 2021; pp. 278–287.
7. Islam Shamim, M.S.; Ahamed Bhuiyan, F.; Rahman, A. Xi commandments of Kubernetes Security: A systematization of knowledge related to Kubernetes Security Practices. In Proceedings of the 2020 IEEE Secure Development (SecDev), Atlanta, GA, USA, 28–30 September 2020; pp. 58–64.
8. Guptha, A.; Murali, H.; Subbulakshmi, T. A comparative analysis of security services in major cloud service providers. In Proceedings of the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 6–8 May 2021; pp. 129–136.
9. Diaz, J.; Perez, J.E.; Lopez-Pena, M.A.; Mena, G.A.; Yague, A. Self-service cybersecurity monitoring as enabler for devsecops. *IEEE Access* **2019**, *7*, 100283–100295. [CrossRef]
10. Nadeem, M.; Arshad, A.; Riaz, S.; Band, S.S.; Mosavi, A. Intercept the cloud network from brute force and ddos attacks via intrusion detection and prevention system. *IEEE Access* **2021**, *9*, 152300–152309. [CrossRef]
11. Avritzer, A. Challenges and approaches for the assessment of Micro-Service Architecture Deployment Alternatives in devops: A tutorial presented at ICSA 2020. In Proceedings of the 2020 IEEE International Conference on Software Architecture Companion (ICSA-C), Salvador, Brazil, 16–20 March 2020; pp. 1–2.
12. Sultan, S.; Ahmad, I.; Dimitriou, T. Container security: Issues, challenges, and the road ahead. *IEEE Access* **2019**, *7*, 52976–52996. [CrossRef]
13. MacDonald, N.; Winckless, C. *Innovation Insight for Cloud-Native Application Protection Platforms*; Gartner: Stamford, CT, USA, 2021.
14. Tripwire. *Stay Ahead of Ransomware with Tripwire Enterprise: Best Practices for Ransomware Prevention and Detection*; Tripwire: Eden Prairie, MN, USA, 2021.
15. Sadovykh, A.; Widforss, G.; Truscan, D.; Enoiu, E.P.; Mallouli, W.; Iglesias, R.; Bagnto, A.; Hendel, O. VeriDevOps: Automated Protection and Prevention to meet security requirements in DevOps. In Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 1330–1333.
16. Yeboah-Ofori, A.; Islam, S.; Lee, S.W.; Shamszaman, Z.U.; Muhammad, K.; Altaf, M.; Al-Rakhami, M.S. Cyber threat predictive analytics for improving Cyber Supply Chain Security. *IEEE Access* **2021**, *9*, 94318–94337. [CrossRef]
17. Kermabon-Bobinnec, H.; Gholipourchoubeh, M.; Bagheri, S.; Majumdar, S.; Jarraya, Y.; Pourzandi, M.; Wang, L. Prospec: Proactive security policy enforcement for containers. In Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, Washington DC, USA, 25–27 April 2022; pp. 155–166.
18. Li, Y.; Hu, H.; Liu, W.; Yang, X. An optimal active defensive security framework for the container-based cloud with deep reinforcement learning. *Electronics* **2023**, *12*, 1598. [CrossRef]

19. OWASP. Available online: https://owasp.org/Top10/ (accessed on 28 March 2023).
20. Sun, J.; Wu, C.; Ye, J. Blockchain-based Automated Container Cloud Security Enhancement System. In Proceedings of the 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 6–8 November 2020; pp. 1–6.