ISSN:0975-9646
Lamya AL-Khuzam/ (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 15 (2), 2024, 14-18

Securing Cloud-Native Containerized Applications: Orchestration, Supply Chain, and Runtime Protection

Vishakha Sadhwani[#1]
[#]*Department of Computer Engineering, University of Maryland*
College Park, MD 20742, United States

***Abstract*—Over the past few years, cybersecurity professionals have publicly recognized that container technology has been increasingly popular and used by numerous enterprises. Cloud native environments have gained significant momentum in enabling the creation and deployment of applications across many locations, resulting in enhanced flexibility and a simplified development lifecycle. Containers present distinct cybersecurity concerns that involve several components such as images, containers, hosts, runtimes, registries, and orchestration systems. This emphasizes the imperative necessity to allocate resources towards ensuring the security of the container stack. The research, published by Aqua Security on June 21st, highlights various methods via which attackers might compromise a company's container infrastructure and the image supply chain. In addition, they projected a 600% increase in the next few years if proper measures are not taken. This article examines the security factors involved in container orchestration and the software supply chain landscape. In order to address these problems, it is crucial to implement standardized security and configuration controls. This study introduces three broad scenarios that tackle prevalent security vulnerabilities in container management, along with the corresponding solutions that are currently accessible. The use cases encompass: (I) Ensuring the security of application containers by preventing misconfigurations in the orchestrator (II) Protecting application containers from potential threats posed by insecure registries (III) Implementing a shielding cloud platform to protect against hacked containers**

Keywords*— *Cybersecurity, Containers, Orchestrations, Kubernetes, Infrastructure, Software Supply chain

I. INTRODUCTION

Cloud providers offer a purpose-built infrastructure for containerized application deployments, ensuring managed availability, scalability, resiliency, and a secure base for your software. It provides an abstract layer that facilitates deployment, scaling, expanding interconnections, and continuously monitoring this infrastructure, thereby streamlining operations and ensuring quality within the cloud environment.

However, the diverse composition of cloud infrastructure makes it susceptible to attack if even one component is compromised [1]. This is a key concern for container-based cloud deployments, where vulnerabilities can surface in various compromised resources: software packages, container images, orchestrator misconfigurations, and many more. These compromises might stem from malicious actors, bad dependencies, bypassed code reviews, or compromised systems within the deployment pipeline. As cloud infrastructure scales, the challenge of protecting it from such attacks grows exponentially [2]. Therefore, it is imperative to include additional security measures during cloud container orchestration and provisioning, alongside continuous reviews of processes and configurations. Further supplementing it with auditing systems to maintain a strong security posture for your application.

To effectively implement security measures, it's essential to understand the various elements and processes involved in building and deploying containerized applications. These elements include images, container runtimes, cluster networks, and access to data deployed in workloads across multiple clouds. As applications scale, the complexity of managing, securing, and debugging them increases due to the multitude of component workflows.

The nextsection presents the main components connected with distant settings and services, organized into categories:

1. **Protected source code**: Safeguard code across local storage, development environments, and version control management systems [1].
2. **Protected system for constructing and evaluating infrastructure**: Evaluate container orchestration platforms such as Kubernetes[3] for misconfigurations and applying appropriate authentication, authorization methods and safety controls.
3. **Vulnerability scanning:** Involves conducting scans on container images and runtimes to identify any potential weaknesses or vulnerabilities before they are deployed for production. Develop and deploy systems for verifying the authenticity and integrity of images.
4. **Container access and microservice communication**: Oversee and safeguard the access to containers and ensure secure communication between microservices.
5. **Ensuring compliance across multi cloud platforms:** Enforce compliance and standards with company policies by implementing restricted namespace access and strong authentication/authorization for applications [3].
6. **Logging and monitoring**: Implement practices to consistently record and track applications [10], thus mitigating the risk of unauthorized and harmful utilization of resources.

These are just some high-level categories that are involved in an application delivery lifecycle. While numerous studies have been undertaken on host and container level security, there is a lack of emphasis on container management and supply chain security [2]. This study explores three common use cases at the platform level, aiming to enhance users' comprehension of security issues related to container management platforms. Additionally, it provides an overview of the available strategies for securing application containers.

II. BACKGROUND

A. Challenges in Container Security

Presently, the security concerns pertaining to containers and their orchestration are of great significance. Failure to integrate security as a fundamental component of the application lifecycle not only puts corporations at risk, but also jeopardizes the businesses of their customers. Organizations encounter significant challenges in guaranteeing the security and coordination of containers [2]. Failure to adequately address security across the whole application development process exposes both organizations and their consumers to possible vulnerabilities. In addition, developers often perceive security measures as impeding innovation and speed at which products are brought to the market [6]. This emphasizes the necessity of implementing a cohesive strategy that integrates security strategies with the capability to promptly adjust and adapt. An effective solution could involve implementing a continuous security paradigm that prioritizes **"Shift Left Security"**[10]. This addresses the question of what needs to be done, but the question of **how** remains: how can these strategies be integrated into your application development process? Although the cloud promotes shared accountability and helps to tackle certain security concerns, there are still instances where both the application and the cloud platform it operates on can be vulnerable to significant attacks.

This research paper focuses on tackling a specific issue and proposes a conceptual framework that revolves around the essential elements of deploying containerized apps in the Cloud. Furthermore, it highlights the potential hazards that can occur in the absence of adequate precautions and explores the ways in which orchestration, supply chain, and cloud security tactics can be utilized to mitigate these risks. Consequently, it is necessary to analyze the challenge and divide it into the subsequent separate instances of usage:

(I) Ensuring the security of application containers by preventing misconfigurations in the orchestrator
(II) Protecting application containers from potential threats posed by insecure registries
(III) Implementing a shielding cloud platform to protect against hacked containers

B. Literature Review

This study employs a wide variety of sources, including well-regarded academic journals, industry magazines, and dissertations collected from online archives. This wide-ranging collection illustrates the need for a thorough understanding of the challenges in securing Cloud containers and practical guidance for implementing solutions. By analyzing several internet platforms, we gained significant insights into the newest market trends and unique business concerns. Furthermore, we effectively discovered recurring issues and available remedies within this domain. The selection criteria we employed focused on multiple facets of container deployment security in cloud environments, encompassing characteristics, remedies, risks, weaknesses, exploits, accessible utilities, pertinent standards, established assessment methodologies, potential applications of container technology, and alternative containerization approaches[2]. Employed Google Scholar to do searches utilizing keywords such as "application container security," "cloud platform security," and "container orchestration." Following the initial search, we excluded generic resources that were not directly applicable to container security. In the end, we utilized a retrospective citation search strategy to expand the scope of our literature analysis by examining the references cited in the selected works.

III. THREAT MODELS

A. Attack Scenarios and Proposed Solutions

Although there are many instances that illustrate the security of containerized applications, attempting to compile a comprehensive list would be impractical and burdensome for readers. Hence, we suggest a novel classification of utilization scenarios, with a primary emphasis on registries and orchestration[3][4]. The objective of this research is to provide a comprehensive understanding of the potential hazards, weaknesses, and supply chain considerations linked to application containers and orchestration systems like Kubernetes.

I. Use Case 1 : Protecting application containers from orchestrator misconfigurations

Kubernetes has emerged as the prevailing method for managing and coordinating containers, especially in cloud-based settings[6]. Due to its robust power and capacity to handle large workloads, it is a crucial tool for contemporary cloud-based applications in diverse sectors. Unresolved Kubernetes security misconfigurations can present substantial hazards to your cloud infrastructure and applications[7]. It is essential to analyze and comprehend the typical weaknesses in Kubernetes manifests (configuration files) in order to ensure the security of containerized workloads in the cloud.

According to an empirical study that examined 2,039 Kubernetes manifests from 92 open-source repositories[6], common misconfigurations include the lack of resource restrictions, the absence of securityContext settings, and other similar issues[7]. These vulnerabilities can result in container breaches, illegal entry to cloud resources, data extraction, and a compromised cloud environment. To prevent these attacks, it is crucial to do security-focused code reviews and utilize static analysis tools[8] on Kubernetes manifests.

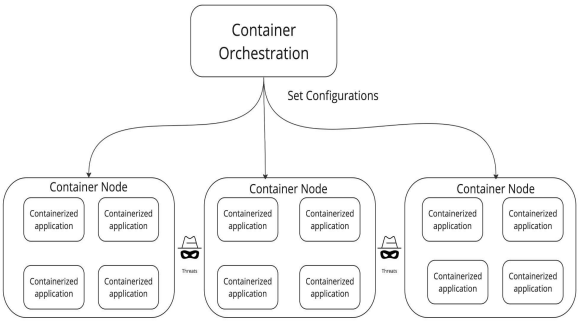

Figure 1. High Level Overview of container stack with the Orchestration Platform

Table 1 Scenarios of Attack for Use Case (I) - Protecting application containers from orchestrator misconfigurations

Category	Possible Attack	Scenario	Solution
Missing Resource Limits	Resource Exhaustion	Without defined resource limits (CPU, memory) containers can consume an excessive amount of resources within a node. This can lead to a denial-of-service situation, affecting the performance of other containers or crashing the node entirely. Adversaries could deliberately introduce such workloads to disrupt cluster operations.	1. Enforce resource requests and limits 2. Leverage Kubernetes tools lik Limit Ranges and Resource Quot to help enforce these policies at t namespace level 3. Leverage Open Policy Agent(OPA) to define & enforce fine grained policies across your clusters during container creatior
Missing Security Context	Privilege Escalation	A container could run as root or have escalated privileges within the host. This grants access to attackers who compromise the container, full control over the underlying node.	1. Run containers with non-root users and drop unnecessary linux capabilities 2.Prevent malicious modificatior container filesystem with setting "readOnlyRootFilesystem" to "tr 3. Use tools like Kyverna or OPA Gatekeeper for policy enforceme during container creation.
hostIPC Enablement	Host Compromise	Potential manipulation of host-level processes and resources from within the container, severely undermining isolation.	1. Avoid setting it true unless strictly necessary 2. Define policies with OPA that prevent the deployment of pods with hostIPC Enablement: true unless explicitly exempted.
hostNetwork Enablement	Lateral Movement and Network Compromise	An attacker within the container could move laterally across the host network, access other systems, or exfiltrate data.	1. Enforce isolation by utilizing kubernetes network policies to define allowed network communication for pods 2. Define policies with OPA that prevent the deployment of pods with hostNetwork: true unless explicitly exempted.
hostPID Enablement	Information Disclosure and Evasion	An attacker can view sensitive processes running on the host and potentially mask malicious activities or hide malicious processes.	1. Avoid setting it true unless strictly necessary 2. Define policies with OPA that prevent the deployment of pods with hostPID Enablement: true unless explicitly exempted
Docker Socket Mounting	Host Takeover	An attacker can obtain complete control over the Docker daemon on the host, and could use this to spawn new containers, modify existing ones, or even delete them entirely, potentially taking over the host.	Use the Container Runtime Interface (CRI) instead of directly mounting the Docker socket.
Hard-Coded Secrets	Credential Theft and Unauthorized Access	If an attacker gains access to the container or its configuration, they can steal sensitive information(passwords, API keys, or access tokens) and potentially gain unauthorized access to sensitive resources.	1. Leverage cloud-provider speci secrets management tools along with kubernetes secrets to store a manage sensitive data securely. 2. Integrate secrets management into your CI/CD pipeline to preve accidental hard-coding of secrets 3. Use a centralized Secret management and encryption tool like Vault(Hashicorp)
Insecure HTTP	Man-in-the-Middle Attack (MitM)	An attacker who intercepts the HTTP traffic can potentially read sensitive information or even modify it in transit (MitM attack).	1. Encrypt data in transit, protecti it from eavesdropping or tamperi 2. Enforce network segmentation using kubernetes network policies 3. Use service meshes(Istio, Linkerd) to implement observability, traffic control between microservices with mTL for encryption.

II. Use Case 2 : Safeguarding applications containers against unsecured registries

Unsecured registries present significant risks to containerized applications as they have the potential to store malicious images that contain backdoors, malware, or obsolete images that are filled with vulnerabilities[1]. In order to reduce these risks, it is advisable to give priority to the utilization of reliable and protected registries, such as cloud-native solutions (such as Google Artifact Registry, Amazon ECR, Azure artifacts or any internal repository) or well-managed private registries. Deploy a system for ongoing vulnerability scanning to identify potential vulnerabilities in pictures both prior to and subsequent to their storage[10]. Utilize image signature and verification techniques to guarantee the integrity and source of the image. Implement stringent access controls by utilizing IAM technologies and network limitations to restrict unauthorized interactions with the registry. The image below illustrates the locations where attacks can occur, followed by a table that categorizes and presents distinct attack scenarios.

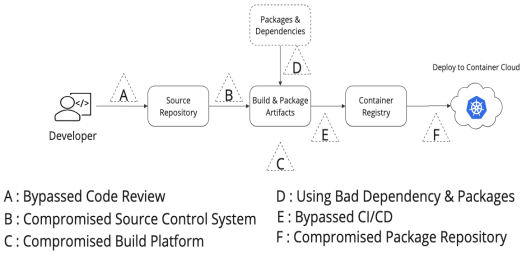

Figure 2. Overview of Security protection requirements for the supply chain components within the cloud environment.

Table 2 Scenarios of Attack for Use Case (II) - **Safeguarding applications containers against unsecured registries**

Category	Possible Attack	Scenario	Solution
Bypassed Code Review	Malicious updates, or theft of sensitive data and credentials.	Without thorough reviews, a bad actor can inject malicious data into your application source code	1. Mandatory Code Reviews with strict policies, and require senior level approvals before merges. 2. Implement static code analysis tools that flag potential security issues and suspicious code patterns 3. Use cloud specific build servers to start the build process only after successful code reviews. Monitor & maintain logs to audit activities. 4. If using cloud source repositories, implement fine-grained access controls to ensure only authorized personnel can approve changes and merge into protected branches.
Compromised Source Control System	Code alteration, backdoor insertion, or theft of sensitive data and credentials.	An attacker can gain access to a developer's source control credentials and modifies code to include a vulnerability they can later exploit.	1. Enforce MFA for all access to source control repositories, and adhere to the principle of least privilege, granting users only the minimum necessary access. 2. Implement robust auditing and monitoring to detect suspicious activity within repositories. 3.Monitor all activities related to the repository through cloud logging & monitoring platform to catch any suspicious behavior.
Compromised Build Platform	Dependencies and build process alterations	Potential manipulation of container image builds due to outdated software that an attacker can exploit	1. Regularly patch and update software on build platforms. Isolate them from other networks to reduce attack surface. 2. Digitally sign container images using cloud solutions to confirm their origin and integrity before deployment. 3. Monitor all activities related to the build platform through cloud logging & monitoring platform to catch any suspicious behavior.
Using Bad Dependency & Packages	Data Theft, Ransomware Attacks, Lateral movement	Through malicious dependency, an attacker can exfiltrate sensitive data and they can also deliver ransomware payloads.	Continuously scan dependencies for known vulnerabilities using tools like Snyk, Trivy, container analysis etc. Establish a curated set of libraries and dependencies and maintain a Software bill of materials(SBOM) to keep track of all the components in your container images, making it easier to identify and manage risks.
Bypassed CI/CD	Untested/Vulnerable images in production	Without automated security checks, the application team can skip the security holes and make it more vulnerable for attacks	Mandate that all deployments go through the defined CI/CD pipeline with automated security checks. Use strict access controls to prevent unauthorized access. Enforce authorization tools to prevent unauthorized or vulnerable images being deployed. Maintain logs for all activities
Compromised Packaged Repository	Repository Takeover, Social engineering	The malicious repository can become a centralized distribution point for infected container images	1.Implement strict access controls and MFA for repositories and always verify container image integrity using digital signatures before pulling and using them. 2. Consider using private repositories with additional security measures for sensitive images

III. Use Case 3 : Shielding platform from compromised containers

A container that has been compromised in cloud environments presents a substantial risk, as it can be utilized to gain unauthorized access to additional cloud resources, sensitive data, or propagate horizontally inside your network[9]. To protect your platform, prioritize proactive containment by using runtime security technologies that can identify abnormal container behavior. Employ micro-segmentation to restrict network traffic, so restricting the extent of an attacker's access, even in the event of a container breach. Adopt a zero-trust strategy in your cloud environment, where access is limited to the minimum necessary based on the concept of least privilege. Whenever feasible, adopt immutable architecture to impede an attacker's capacity to establish a

lasting presence within a compromised container. The following table elucidates the different attack situations and proposes methods that can be taken to fortify your platform.

Table 3 Scenarios of Attack for Use Case (III) - **Shielding platform from compromised containers**

Category	Possible Attack	Scenario	Solution
Container Breakouts	Exploiting Vulnerabilities and initial compromise	An attacker gains access to a container through an unpatched vulnerability within the application or by injecting malicious code via a bad dependency.	1. Employ runtime security tools that monitor container behavior in real-ti detecting anomalies, and enforcing security policies> Enforce network segmentation to limit communication between containers & between contai & hosts. 2. Consider solutions like:Falco, Sysdig Secure, Cloud based agents that detects suspicious container activity using system call analysis. 3. Run containers with minimal privileges & ensure proper filesystem permissions, and use container optimized images to reduce the attack surfa
Cloud Resource Abuse	Privilege Escalation	The attacker leverages techniques to elevate their privileges within the container or attempts to break out into the host.	1. Implement Least Privilege/ Hardening by running containers with the lea possible privileges. Harden the host OS of cloud instances running containe and consider running containers in rootless mode whenever possible. 2. Runtime Security: Use runtime protection tools to detect and block anomalous container behavior (file access patterns, network connections, et 3. Enforce Network segmentation and implement Monitoring and Anomaly Detection mechanism, consider cloud based or 3rd party solutions
Data Exfiltration	Exploiting Cloud Environment	The attacker targets cloud services, APIs, or configurations with weaknesses, aiming to access more privileged resources or sensitive data.	1. Encrypt sensitive data at rest and in transit within the cloud environment 2. Firewalls and Network Segmentation to restrict outbound traffic and segr networks to prevent unauthorized data movement. 3. Implement intrusion Detection/Prevention mechanism to monitor networ traffic and container behavior for signs of lateral movement attempts. Also, incorporate DLP tools that detect, classify, and potentially block the transfe sensitive data. 4. Cloud Security Posture Management (CSPM): Continuously monitor clo configurations and assets, alerting and potentially fixing misconfigurations could be exploited.
Lateral Movements	Data Theft, Ransomware Attacks	The attacker compromises additional workloads, explores cloud assets, and potentially exfiltrated data.	1. Zero-Trust: Implement a Zero-Trust approach in your cloud environment constantly authenticate and authorize access to resources. 2. Kubernetes Network Policies or cloud-native firewalls help isolate work and reduce the potential for lateral spread. 3. Implement intrusion Detection/Prevention mechanism to monitor networ traffic and container behavior for signs of lateral movement attempts.

IV. Conclusion

Cloud-native containerized apps have significant advantages in terms of scalability, portability, and improved security for software development [5]. This makes them very suitable for massive language models in the era of Generative AI. Kubernetes, specifically, offers a resilient orchestration platform for hosting containers securely. However, a significant obstacle to the broad use of containers is the management of the various security problems they entail. Currently, there's a lack of comprehensive guides addressing orchestration platform oversights, vulnerabilities, attack scenarios, and cloud-agnostic solutions. This work aims to bridge this gap by analyzing the primary threats arising from images, registries, orchestration misconfigurations, and runtime risks.

Therefore, I have presented three use cases encompassing orchestrated containers and the software supply chain. These use cases demonstrate how existing cloud solutions can be strategically implemented to bolster containerized application security. This research contributes to a more secure and standardized approach to containerized application deployment in cloud environments.

This work presented three critical use cases for container security: (I) protecting against orchestrator misconfigurations, (II) safeguarding against insecure registries, and (III) shielding the platform from compromised containers.

To address these use cases, a range of solutions can be leveraged. These include policy enforcement tools (OPA, Kyverno), security scanning (open-source solutions), and cloud-focused solutions like trusted registries, vulnerability scanning, image signing, runtime security, micro-segmentation, immutable infrastructure, and zero-trust principles. While these solutions offer significant advantages, open challenges remain. Further research is needed to focus on enhanced vulnerability management and digital investigation capabilities within containerized environments. By addressing these challenges, we can further strengthen the security posture of container technologies within the cloud, fostering their wider adoption and facilitating innovation.

In this research paper, we have explored the critical domain of container security within cloud-native environments. Our exploration highlights the urgency of implementing robust security measures to protect the integrity of applications, the underlying orchestration layer, and the overall platform against a landscape of potential threats and vulnerabilities.

The analysis identified three core use cases:

(I) **Protecting application containers from orchestrator misconfigurations:** Orchestrator misconfigurations can leave containers vulnerable. This research emphasizes the importance of enforcing strong configuration policies, including the use of policy enforcement tools.

(II) **Safeguarding application containers against unsecured registries:** Unsecured registries can be sources of compromised or malicious container images. Rigorous image scanning, registry authentication, and robust access controls are essential. This includes vulnerability detection tools, security scanning (including open-source solutions), and cloud-focused CI/CD mechanisms like trusted registries, vulnerability scanning, and image signing.

(III) **Shielding the platform from compromised containers:** A compromised container can endanger the entire platform. Strategies such as network isolation, minimizing container privileges, and continuous runtime monitoring are crucial to limit the potential impact.

The findings of this research highlight the multifaceted nature of container security challenges. To achieve comprehensive protection, a defense-in-depth approach is vital [10], encompassing elements such as:

1. **Robust image governance:** Strict image provenance tracking, vulnerability scanning, and secure image registries.
2. **Stringent configuration management:** Enforcing configuration best practices and employing tools for policy validation and drift detection.
3. **Runtime security mechanisms:** Runtime monitoring, behavior analysis, and network micro-segmentation to prevent and contain malicious activity.

V. Future Research Directions

Further research avenues could investigate emerging areas like:

1. Confidential Computing for Enhanced Container Isolation

Several studies indicate that data exfiltration is a major concern within container security. It would be valuable to analyze tools that provide hardware-level encryption to shield the contents of containers while in use. However, encryption and decryption processes can introduce performance overhead [11]. Further research is needed to explore optimal solutions that balance security gains against potential performance impacts.

2. Integration of Machine Learning Techniques for More Sophisticated Anomaly Detection

Incorporating machine learning (ML) algorithms can enable proactive threat identification by detecting deviations from normal behavior patterns within containerized environments. This approach requires the development of trained models using container-specific attack data, as well as measures to reduce the attack surface [12]. Additionally, research should address how ML models can continuously adapt to evolving attack patterns and the dynamic nature of cloud environments.

By addressing these use cases and recommendations, researchers and practitioners can significantly advance the state of container security, fostering a more secure and resilient cloud-native landscape.

REFERENCES

[1] Bhowmik, S., Saira Bhanu, S. M., & Rajendran, B. (2020, February). Container Based On-Premises Cloud Security Framework. *2020 International Conference on Inventive Computation Technologies (ICICT)*. https://doi.org/10.1109/icict48043.2020.9112561

[2] Sultan, S., Ahmad, I., & Dimitriou, T. (2019). Container Security: Issues, Challenges, and the Road Ahead. *IEEE Access*, *7*, 52976–52996. https://doi.org/10.1109/access.2019.2911732

[3] Paladi, N., Michalas, A., & Dang, H. V. (2018). Towards Secure Cloud Orchestration for Multi-Cloud Deployments. *Proceedings of the 5th Workshop on CrossCloud Infrastructures & Platforms - CrossCloud'18*. https://doi.org/10.1145/3195870.3195874

[4] Sadhwani, V. (2022). *Cloud Container Security Next Move*. Digital Commons at Harrisburg University. https://digitalcommons.harrisburgu.edu/csms_dandt/3/

[5] Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2019, July 1). Cloud Container Technologies: A State-of-the-Art Review. *IEEE Transactions on Cloud Computing*, *7*(3), 677–692. https://doi.org/10.1109/tcc.2017.2702586

[6] Rahman, A., Shamim, S. I., Bose, D. B., & Pandita, R. (2023, May 26). Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study. *ACM Transactions on Software Engineering and Methodology*, *32*(4), 1–36. https://doi.org/10.1145/3579639

[7] Shamim, S. I. (2021, August 18). Mitigating security attacks in kubernetes manifests for security best practices violation. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. https://doi.org/10.1145/3468264.3473495

[8] Bose, D. B., Rahman, A., & Shamim, S. I. (2021, June). 'Under-reported' Security Defects in Kubernetes Manifests. *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. https://doi.org/10.1109/encycris52570.2021.00009

[9] Islam Shamim, M. S., Ahamed Bhuiyan, F., & Rahman, A. (2020, September). XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices. *2020 IEEE Secure Development (SecDev)*. https://doi.org/10.1109/secdev45635.2020.00025

[10] Sojan, A., Rajan, R., & Kuvaja, P. (2021, November). Monitoring solution for cloud-native DevSecOps. *2021 IEEE 6th International Conference on Smart Cloud (SmartCloud)*. https://doi.org/10.1109/smartcloud52277.2021.00029

[11] Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2019, July 1). Trusted Container Extensions for Container-based Confidential Computing, Cryptography and Security, https://doi.org/10.48550/arXiv.2205.05747

[12] Lin, Y., Tunde-Onadele, O., & Gu, X. (2020, December 7). CDL: Classified Distributed Learning for Detecting Security Attacks in Containerized Applications. *Annual Computer Security Applications Conference*. https://doi.org/10.1145/3427228.3427236