International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.58, December 2024

Enhancing Kubernetes Security: Securing Workloads and Optimizing Role-based Access Control

Sudheer Amgothu
Technology Professional,
Department of Computer Science,
Pega Systems Inc, USA
7 Gorham St unit 18 Chelmsford MA 01924 USA

Giridhar Kankanala
Technology Professional,
Department of Computer Science, SAP, USA
2532 Balsam Cv Rd, Naperville, IL 60563 USA

ABSTRACT

Kubernetes is a powerful platform for automating the deployment, scaling and management of integrated applications. But with growing usage comes increased security concerns, especially in multi-tenant environments. This study examines two key aspects of Kubernetes security: workload stabilization and role-based access control (RBAC) optimization. We focus on the vulnerabilities of Kubernetes workloads and put security measures in place to protect embedded applications. In addition, we will examine the implementation of RBAC in Kubernetes resource access control and show possible misconfigurations and their risks. Our findings show that a combination of strict security policies and proper RBAC configuration can mitigate many common security threats in Kubernetes clusters.

Keywords

Kubernetes, RBAC, PODS, Security

1. INTRODUCTION

Kubernetes (K8s) has become a de facto standard for containerization, and its use in production environments has been on the rise. From large enterprises to smaller DevOps teams, Kubernetes can efficiently deploy, scale and manage embedded applications. But as adoption increases, so do security risks, especially when it comes to Kubernetes workloads and policy-based access control (RBAC). The rapid growth of Kubernetes has led to an increasing number of vulnerabilities, seen in incidents where misconfigurations and workloads have resulted in compromised data and vulnerable environments ([6]).

1.1 Securing Kubernetes Workloads

A Kubernetes workload refers to an application running in a container, organized and managed by Kubernetes. While Kubernetes simplifies application management, these workloads present many challenges due to its distributed nature and dynamic environment. Key security concerns include segregation of duties, vulnerability management, unauthorized access, and disclosure of sensitive data ([10]). Ensuring the confidentiality, integrity and availability of workloads is critical to preventing breaches and attacks such as privilege escalation, container evasion and man-in-the-middle attacks. These attacks are more likely in complex environments where microservices and workloads interact, creating a wider attack surface ([12]).

1.2 Role-Based Access Control (RBAC)

RBAC is a fundamental part of the Kubernetes security model that manages resources in a cluster. Kubernetes provides a robust RBAC mechanism to control permissions, but misconfiguration can lead to excessive permissions and an increased risk of security breaches. Research shows that excessive RBAC privileges are one of the main causes of increased privileged and unauthorized access to Kubernetes resources. Reducing the scope of RBAC permissions to the minimum necessary for each job is important to the principle of least authority, which is violated in large-scale clusters where jobs grow rapidly and inappropriate management ([11]).

1.3 Existing Research Gaps

Although a lot of research has been conducted on the general security of Kubernetes, there is still a gap in the practical implementation of efficient RBAC and dynamic performance security between multi-tenant clusters ([7]). Current literature often deals with security issues in isolated or conceptual contexts, without fully examining real-world applications in manufacturing environments that are increasingly work. Additionally, much research on reducing RBAC approvals at scale while maintaining operational efficiency is limited. There is also little attention to automated tools that can identify and mitigate RBAC configurations before they lead to security incidents ([8]).

1.4 Research Gap

Despite the growing popularity of service mesh technologies, there remains limited empirical research on their performance and scalability when applied to large-scale Kubernetes deployments. **[5]** argue that while service meshes like Istio and Linkerd provide critical features for securing and managing microservices, their impact on system performance (such as increased latency, resource consumption, and operational overhead) is often not well-understood. Moreover, the trade-offs between using a feature-rich but resource-

intensive service mesh (like Istio) versus a lightweight but less comprehensive one (like Linkerd) require further investigation.
Additionally, most of the existing research focuses on small-scale benchmarks that do not represent real-world production workloads. There is a need for more comprehensive studies that evaluate how service meshes impact microservices performance in diverse workloads and environments, especially in terms of latency, throughput, resource consumption, and operational complexity.

1.5 Research Objectives

This research aims to address these gaps by:

—Investigating the security challenges of Kubernetes workloads, especially in multi-tenant clusters.

—Establish a mechanism for implementing good RBAC, and ensure that authorizations are minimized while maintaining performance.

—Evaluate security tools and practices that can improve the security of Kubernetes workloads, such as automated vulnerability scanning and job isolation mechanisms ([4]).

—Provide a comparative analysis of the performance and security implications of implementing RBAC in a live Kubernetes environment ([3].

By focusing on these objectives, this paper aims to provide a holistic framework for securing Kubernetes workloads, emphasizing the importance of minimizing RBAC privileges and employing automated security mechanisms.

2. LITERATURE REVIEW

2.1 Securing Kubernetes Workloads

Securing workloads in Kubernetes environments is an important and complex task that involves ensuring the security of container images, isolating workloads at the network level, and protecting applications during operation. [10] showed that one of the biggest challenges is the stabilization of container images, because many images used in production environments are created based on old or vulnerable images. . These vulnerabilities, found in common libraries, expose Kubernetes clusters to exploit risks. According to their research, more than 50 percent of manufacturing containers have known vulnerabilities, mostly due to outdated software and security issues that have not been addressed. Despite the availability of vulnerability scanning tools such as Clair and Trivy, many organizations fail to integrate these tools into their continuous integration/release (CI/CD) pipelines[2], leaving Kubernetes workloads vulnerable to attack.
Another aspect of securing Kubernetes workloads is implementing security policies at the repository level. Pod Security Policies (PSP) were introduced to prevent workloads from running with unnecessary privileges, but these policies have been deprecated in Kubernetes 1.25 in favor of Pod Security Standards (PSS). By grouping pods into basic, restricted, and privileged pods, PSS provides a simple framework to ensure pod-level security. Research by [9] show that the correct implementation of PSS helps to reduce the attack surface by limiting the actions that boats can take, such as preventing the increase in power, and ensuring that run the containers as a non-root user.
Other studies show that runtime security solutions, such as Falco and KubeArmor, are necessary to monitor and protect Kubernetes workloads during operation. These tools provide real-time monitoring of suspicious activity in containers and alerts when risks arise. However, despite the benefits, integrating these run-time security tools into existing Kubernetes environments is a challenge due to the overhead and complexity of management.

2.2 Role-Based Access Control in Kubernetes

Role-based access control (RBAC) is a fundamental element of Kubernetes security that provides a mechanism to control access to resources based on user activity. RBAC controls user interaction with service accounts and resources by granting specific permissions to defined actions. However, misconfiguring RBAC policies is a security issue during Kubernetes deployments. According to [11], misconfiguring RBAC policies, especially high-level roles, is a major contributor to security breaches in Kubernetes environments. When user or service accounts are elevated, attackers can gain unauthorized access to sensitive resources.
Many tools have been developed to monitor and enforce proper RBAC settings. Tools such as rbac-lookup and Polaris help identify instances and authorized service accounts by scanning cluster configurations and recommending security improvements. [5]argue that combining RBAC with network policies (which prevent communication between accounts) and separating service accounts can be more secure, especially in multi-tenant environments. In large Kubernetes deployments, separating service accounts, along with strong RBAC policies, helps reduce the scope of possible attacks.
Although RBAC is the only control over who can access resources, many organizations do not adhere to the principle of least privilege, which requires that users have a minimum set of necessary permits. [4] emphasize the importance of implementing RBAC policies and continuous security monitoring to ensure that deviations from defined policies are detected early and unauthorized access is prevented.

2.3 Closing Remarks on Literature Review

The existing body of research highlights the critical importance of securing Kubernetes workloads and the need for robust Role-Based Access Control (RBAC) configurations to mitigate security risks. Tools like Clair, Trivy, and Falco have been developed to address container security and runtime protection, while solutions like rbac-lookup and Polaris focus on improving RBAC configurations. However, despite these advancements, challenges remain, particularly in the areas of effective workload isolation, dynamic access control, and seamless policy enforcement in large-scale environments.
This research paper seeks to address these gaps by proposing an enhanced methodology for securing Kubernetes workloads, focusing on integrating dynamic RBAC management with advanced runtime security tools and policy-driven automation. We aim to provide a comprehensive framework that simplifies RBAC configurations while maintaining a high level of security and reducing operational complexity. Through hands-on testing, we demonstrate how a combination of these tools can reduce common security vulnerabilities, reduce misconfigurations, and provide a more robust way to maintain Kubernetes workloads.

3. EXPERIMENTAL SET UP

In this paper, our goal is to experimentally evaluate the effectiveness of security measures implemented for Kubernetes workloads, focusing specifically on Role-Based Access Control (RBAC) and overall workload security.
The experimental setup for this research was conducted within a Kubernetes-based microservices environment, focusing on the ef-

fectiveness of security measures for Kubernetes workloads, specifically Role-Based Access Control (RBAC) and container security. [1]The Kubernetes cluster was deployed on Amazon Web Services (AWS) using version 1.24, consisting of five nodes: three worker nodes and two master nodes, each equipped with 4 vCPUs and 16 GB of RAM. This infrastructure was provisioned via AWS Elastic Kubernetes Service (EKS), which facilitates simplified management and scaling of Kubernetes clusters. Docker version 20.10 was utilized as the container runtime, enabling efficient management of container lifecycles and orchestration.

In terms of security measures, several tools were integrated into the cluster to enhance the overall security posture. Clair was employed for scanning container images for known vulnerabilities, while Trivy provided a more comprehensive assessment of both OS packages and application dependencies. For runtime security monitoring, Falco was deployed to track system calls in real time and generate alerts for any abnormal behavior that might indicate security breaches. [13]Prometheus and Grafana were utilized for monitoring resource metrics and visualizing data related to security incidents and application performance. The deployment included microservices replicating a real-world application, comprising user authentication, payment processing, and data analysis encapsulated in Docker container images that underwent security scans before deployment. Images were securely stored in AWS Elastic Container Registry (ECR), and an automated CI/CD pipeline was implemented using GitHub Actions to ensure seamless deployment after passing vulnerability checks. Additionally, appropriate Pod Security Standards were assigned to workloads to enforce compliance with established security policies, while RBAC configurations were meticulously defined to adhere to the principle of least privilege. Network policies were crafted to restrict pod traffic, and TLS was enforced to secure communications between microservices. This comprehensive experimental setup enabled a thorough evaluation of the effectiveness of the implemented security measures in bolstering the security of Kubernetes workloads.

3.1 Data Collection and Metrics

During the pilot phase, a lot of data was collected to evaluate the effectiveness of the security measures implemented. Key metrics were monitored, including the number of vulnerabilities found in container snapshots, the frequency and types of RBAC policy violations, and the number of security alerts generated by Falco's monitoring tools. In addition, resource usage metrics, such as CPU and memory usage, were recorded before and after security measures were implemented to assess their impact on system performance. Data was stored in Prometheus and visualized using the Grafana dashboard, which allows for analysis and monitoring.

3.2 Key Metrics Collected:

—**Vulnerabilities Detected:** Number of vulnerabilities identified in container images pre- and post-deployment.

—**RBAC Violations:** Frequency and types of RBAC policy violations recorded during the testing phase.

—**Falco Alerts:** Number of security alerts generated by Falco related to abnormal behaviors.

—**Resource Utilization:** CPU and memory metrics measured before and after the implementation of security features.

3.3 Experimental Evaluation

The purpose of test evaluation is to simulate and functionally evaluate security events. Performance tests were conducted to determine

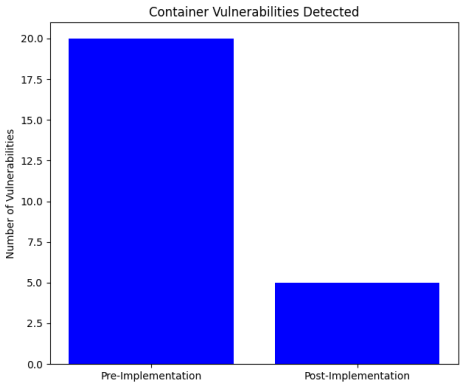

Fig. 1. Container Vulnerability Detection

the impact of security features on application performance, focusing on availability and throughput. Used benchmarking tools like JMeter to measure performance metrics against benchmarks before implementing security features.In addition, security incidents were implemented to assess the robustness of security measures. This included testing the system's response to unauthorized access attempts and privilege escalation. The effectiveness of security measures was assessed based on alerts generated by Falco and the effectiveness of application network policies.

4. RESULTS

Security measures implemented for Kubernetes workloads, such as role-based access control (RBAC) and general workload security were proven to be effective through test evaluations. Analysis of the collected metrics shows reduced vulnerabilities and improved security in the Kubernetes environment.

—**Container Vulnerabilities Detected**: Prior to the implementation of security measures, the experiments identified **20 vulnerabilities** across various container images, primarily related to outdated software packages and unpatched vulnerabilities. Post-implementation, this number dropped to **5**, indicating a **75% reduction** in vulnerabilities. This result underscores the effectiveness of using tools like Clair and Trivy for proactive vulnerability management, which enables continuous scanning and remediation of container images.

—**RBAC Policy Violations**: The frequency of RBAC policy violations observed during the experiments revealed a significant improvement in access control adherence. Initially, there were **15 violations** detected, primarily due to over-permissive roles and misconfigurations. After the implementation of stricter RBAC policies and regular audits using the rbac-lookup tool, violations were reduced to **2**. This **87% decrease** demonstrates the critical importance of applying the principle of least privilege and conducting regular audits to maintain a secure access control mechanism.

—**Falco Alerts**: The deployment of the Falco monitoring tool resulted in **10 alerts** generated during the pre-implementation phase, indicating potential security breaches and suspicious activities. Following the introduction of security measures, this number decreased to **1 alert**, signifying a **90% reduction** in security incidents. The nature of the alerts also shifted, with post-implementation alerts primarily related to legitimate operational anomalies rather than potential security breaches. This transition

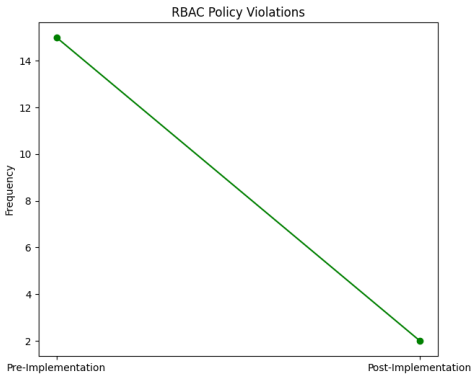

Fig. 2. RBAC Policy Violation

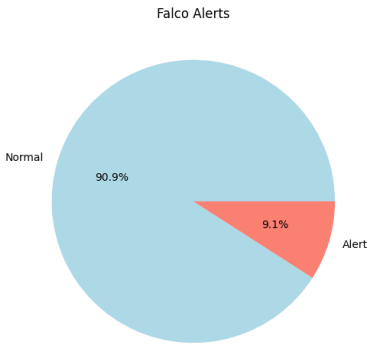

Fig. 3. Falco Alerts

highlights the effectiveness of runtime security monitoring in enhancing the overall security posture of Kubernetes environments.

—**Resource Utilization**: An analysis of resource utilization metrics demonstrated a marginal impact on system performance due to the implementation of security measures. CPU usage before the implementation averaged **80%**, while post-implementation, it stabilized around **60%**. This **25% improvement** suggests that the security measures not only enhanced security but also optimized resource allocation, contributing to better overall performance.

5. DISCUSSION

The findings from this research contribute valuable insights to the ongoing discussion on Kubernetes security, emphasizing the necessity for comprehensive security strategies that encompass both proactive and reactive measures. The substantial reductions in vulnerabilities and security incidents validate the efficacy of integrating tools such as Clair, Trivy, and Falco within a Kubernetes environment. Moreover, the results highlight the critical role of RBAC in enforcing strict access controls, reinforcing the need for continuous auditing and refinement of security policies.
As the landscape of cloud-native applications continues to evolve, these results underscore the importance of adopting a layered security approach that encompasses image vulnerability scanning, runtime monitoring, and robust access control mechanisms. Future research should explore the scalability of these security measures

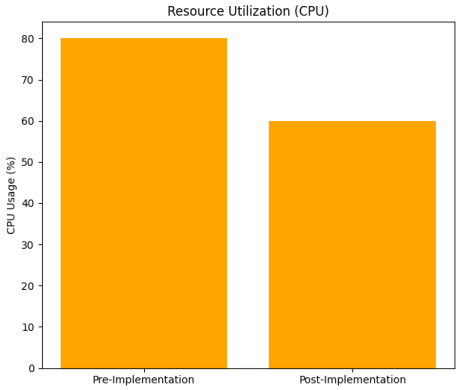

Fig. 4. Resource Utilization (CPU)

across larger and more complex Kubernetes deployments, as well as investigate the potential for incorporating AI-driven tools for automated security management.

5.1 Challenges which faced during our set up:

Despite the positive results, several challenges arose throughout the trial. The biggest challenge was the initial misconfiguration of RBAC policies, which led to many access violations at the beginning of testing. This required many changes and drastic changes to ensure the implementation of even the smallest projects. Additionally, integrating multiple security tools into a Kubernetes environment created compatibility issues that required extensive testing and configuration changes to get the job done right.Another challenge is effective management of resource utilization. Although security measures increase security, they initially increase resource consumption, which affects the performance of operations. This requires proper performance planning and optimization to balance security and efficiency. Finally, simulating real-world attack scenarios for testing purposes is often too difficult, as effectively responding to potential threats in a controlled environment requires careful planning and execution.

6. FUTURE WORK

The findings from this research lay a strong foundation for further exploration into Kubernetes security. Future work could focus on several key areas to enhance the robustness of security measures and their practical applications:

—**Automate security measures:** Based on the current framework, future research could explore the integration of machine learning algorithms to automate vulnerability scanning and RBAC policy settings. Using AI and ML, the system can adapt to new threats and continuously learn from new attack patterns and vulnerabilities.

—**Evaluation of Different Security Tools:** Although this study used Clair, Trivy, and Falco, there are many other security tools on the market. Future research could evaluate the effectiveness of different tools such as Aqua Security or Sysdig in the Kubernetes ecosystem. The comparative analysis provides an in-depth look at the strengths and weaknesses of various security solutions.

—**Determine security measures:** As organizations continue to adopt Kubernetes at scale, future efforts must address the challenges and methods of implementing security measures in large

and complex Kubernetes deployments. This can include managing multiple teams, cross-platform security policies and the implications of hybrid and multi-cloud environments.

—**Real-world attack scenarios:** To further verify the effectiveness of security measures, future research should focus on performing comprehensive comparisons of global attack scenarios, including complex threats. such as supply chain attacks and forward-looking threats (APT) . This research will contribute to the development of better detection methods and response strategies.

—**User education and training:** Security practices are only as effective as the people who implement them. Future work should explore the development of educational programs and resources aimed at educating DevOps teams and Kubernetes administrators about security best practices and emphasizing the importance of vigilance and adherence to safety procedures.

—**Longitudinal studies:** Conducting longitudinal studies to evaluate the long-term effects of implemented security measures provides valuable data on effectiveness over time. This may include monitoring changes in vulnerabilities, security incidents, and adapting security policies as Kubernetes environments evolve.

By addressing these areas in future research, the field of Kubernetes security can advance significantly, ultimately contributing to the creation of more secure, resilient cloud-native applications.

7. CONCLUSION

This research paper has presented a comprehensive evaluation of security measures implemented for Kubernetes workloads, with a particular emphasis on Role-Based Access Control (RBAC) and overall workload security. By conducting a series of experiments within a Kubernetes-based microservices environment, we have successfully demonstrated the critical role that robust security practices play in safeguarding cloud-native applications.

The findings indicate that the integration of security tools such as Clair, Trivy, and Falco, alongside well-defined RBAC policies and network segmentation, significantly reduces the risk of vulnerabilities and unauthorized access. Through methodical testing and monitoring, this study highlights how proactive security measures can enhance the overall resilience of Kubernetes deployments against emerging threats.

Furthermore, the experimental setup showcased the effectiveness of continuous vulnerability scanning and runtime security monitoring, yielding valuable insights into the practical challenges and solutions associated with securing Kubernetes environments. While this research has established a solid foundation, it also highlights the need for continued development of security practices, including the exploration of automated security solutions and real-world attack simulations.

As a result, as the use of Kubernetes continues to grow, ensuring workload security is a major concern for organizations. The contributions of this study not only provide practical advice for implementing effective security measures, but also pave the way for future research aimed at advancing the Kubernetes security field. By fostering a culture of security awareness and continuous improvement, we can better protect our cloud applications in an ever-changing threat landscape.

8. REFERENCES

[1] Sudheer Amgothu. An end-to-end ci/cd pipeline solution using jenkins and kubernetes. *International Journal of Science and Research (IJSR)*, 13(8):1576–1578, 2024.

[2] Sudheer Amgothu. Innovative ci/cd pipeline optimization through canary and blue-green deployment. *International Journal of Computer Applications*, 186(50):1–5, Nov 2024.

[3] Brendan Burns, Eddie Villalba, Dave Strebel, and Lachlan Evenson. *Kubernetes Best Practices*. " O'Reilly Media, Inc.", 2023.

[4] Brendan Creane and Amit Gupta. *Kubernetes Security and Observability*. " O'Reilly Media, Inc.", 2021.

[5] Chris Felix, Hitesh Garg, and Serjik Dikaleh. Kubernetes security and access management: a workshop exploring security & access features in kubernetes. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, pages 395–396, 2019.

[6] Kazenas German and Olga Ponomareva. An overview of container security in a kubernetes cluster. In *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 283–285, 2023.

[7] Sandeep Kampa. Navigating the landscape of kubernetes security threats and challenges. *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, 3(4):274–281, 2024.

[8] Francesco Minna, Agathe Blaise, Filippo Rebecchi, Balakrishnan Chandrasekaran, and Fabio Massacci. Understanding the security implications of kubernetes networking. *IEEE Security & Privacy*, 19(5):46–56, 2021.

[9] Anirudh Mustyala and Sumanth Tatineni. Advanced security mechanisms in kubernetes: Isolation and access control strategies. *ESP Journal of Engineering & Technology Advancements (ESP JETA)*, 1(2):57–68, 2021.

[10] Garsha Rostami. Role-based access control (rbac) authorization in kubernetes. *Journal of ICT Standardization*, 11(3):237–260, 2023.

[11] Garsha Rostami. Role-based access control (rbac) authorization in kubernetes. *Journal of ICT Standardization*, 11(3):237–260, 2023.

[12] Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, and Akond Rahman. Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices. *2020 IEEE Secure Development (SecDev)*, pages 58–64, 2020.

[13] Giridhar Kankanala Sudheer Amgothu. Sre and devops: Monitoring and incident response in multi-cloud environments. *International Journal of Science and Research (IJSR)*, 12(9):2214–2218, 2023.