International Journal for Modern Trends in Science and Technology, 9(08): 122-130, 2023

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0908019
Available online at: http://www.ijmtst.com/vol9issue08.html

Secure Kubernetes and DevSecOps Pipelines Using Short-Lived Certificates and Automated mTLS Enforcement

Vidyasagar Palla

Allen, Texas, USA, 75013
vidyapalla3@gmail.com

To Cite this Article
Vidyasagar Palla, Secure Kubernetes and DevSecOps Pipelines Using Short-Lived Certificates and Automated mTLS Enforcement, International Journal for Modern Trends in Science and Technology, 2023, 9(08), pages. 122-130. https://doi.org/10.46501/IJMTST0908019

Article Info
Received: 29 July 2023; Accepted: 24 August 2023; Published: 28 August 2023.

ABSTRACT

Cloud-native applications developed using Kubernetes have created new security concerns for workload authentication, service identity management and secure service-to-service communication. Most traditional security solutions are based on long-lived credentials and static access controls, which can lead to credential compromise, unauthorized access and lateral movement attacks. In this paper, the concept of a Secure Kubernetes and DevSecOps Framework based on Short-Lived Certificates and Automated Mutual Transport Layer Security (mTLS) Enforcement is introduced to tackle these problems. To implement mTLS policies across Kubernetes environments, service mesh technologies are used to enforce the policy, which will enforce an encrypted and authenticated service-to-service communication. Moreover, the framework follows Zero Trust security practices, with its constant identity validation of workloads and the enforcement of security policies based on the dynamic nature of the workloads. Through experimental evaluation in terms of authentication success rate, vulnerability detection rate, certificate rotation efficiency, communication latency, and overall security score, the proposed framework is found to be an effective solution on cloud-native security while imposing an acceptable performance overhead. The results show that a combination of automated mTLS enforcement and short-lived certificate management creates a powerful, scalable and efficient solution for securing modern DevSecOps environments based on Kubernetes.

KEYWORDS: Kubernetes; DevSecOps; Mutual TLS (mTLS); Short-Lived Certificates; Zero Trust Security; Service Mesh; Cloud-Native Security; Certificate Rotation; Workload Identity Management; Secure Microservices.

INTRODUCTION

Cloud-based technologies have revolutionized the way modern applications are built, deployed, and managed [1]. Kubernetes is seen as the defacto orchestration platform for containerized applications because of its scalability, flexibility [2] and automation [3]. The microservices approach, built on Kubernetes, is becoming more common in organizations because of its ability to deliver more rapid service development, efficient resource utilization, and continuous service delivery [4]. Although these environments are dynamic and distributed, there are some important security challenges, such as service identity management, secure

communications, workload authentication, etc., as pointed out in [5].

Typical security solutions rely on long-lived credentials and static access controls, raising the possibility of a credential theft and/or unauthorized access and/or lateral movement attack [6]. With microservices constantly communicating across each other in a Kubernetes cluster, it becomes crucial to maintain a secure service-to-service communication [7]. The multi-party Transport Layer Security (mTLS) has proved to be a highly effective solution for offering authentication, confidentiality and integrity between services [8]. Manual certificate provisioning and management, however, add to the operational complexity and risk of security misconfiguration [9].

DevSecOps is an extension of DevOps that takes the concept of DevOps and introduces security into the Software Development Life Cycle [10]. Many DevSecOps implementations do not have automated certificate lifecycle management and automated continuous identity verification [11] although they do include vulnerability scanning, compliance validation, and security testing in their existing DevSecOps pipelines. The short-lived certificates are not only reducing the exposure to credential but also have the ability to rotate credentials frequently, which will have a significant impact on the validity period of the authentication credentials [12]. With automated mTLS enforcement, short-lived certificates offer a powerful mechanism to build zero-trust security in a Kubernetes environment [13].

Recently, automated deployment of mTLS and policy-driven control over communication have been made possible with the development of service mesh technologies, such as Istio [14]. However, some issues still need to be addressed to ensure certificates can be integrated into service meshes and DevSecOps workflows [15]. There is a need to have a unified approach that automates the process of issuing, renewing, revoking and enforcing certificates with minimum performance impact on the organisation [16].

This research aims to present a secure Kubernetes and DevSecOps framework, which uses short-lived certificates and enforced mTLS to improve workload identity management, secure intra-service communication, and to minimize attack surface. The proposed framework should aim to improve the security of the cloud-native experience by building automated mechanisms for establishing trust into Kubernetes-based deployment pipelines.

Objectives

1. Design a secure Kubernetes architecture that makes use of short-lived certificates to manage and authenticate dynamic workload identities.
2. To enable automated mutual TLS (mTLS) for securing cluster-to-cluster service-to-service communication.
3. To embed certificate issuance, rotation and revocation processes into DevSecOps pipelines for ongoing security assurance.
4. To improve Zero Trust security by implementing ongoing identity verification and policy-based access control for microservices.
5. To assess the effectiveness of the proposed framework in lowering security risks, enhancing authentication reliability and operational efficiency in the cloud-native environments.

LITERATURE SURVEY

Chandramouli, Butcher and Chetal have suggested an ABAC framework for microservices-based applications running in a service mesh [1]. The authors designed a policy-driven access control mechanism based on the attributes of users, resources and environment for making access decisions. They found that ABAC increased the granularity and flexibility of security compared to the traditional role-based security. The framework proved to be very useful in achieving distributed microservices architectures with varying access demands that often shifted [17].

Hannousse and Yahiouche have done a systematic mapping study of the security problems and security mechanisms that come with microservices architectures [2]. The authors scanned the relevant existing literature on authentication, authorization, communication security for services, container security, and threat mitigation. Their insights revealed that microservices added new security challenges because of their distributed characteristics and expanded attack area. The study highlighted the importance of having a holistic security strategy that covers the vulnerabilities at the application level in addition to the infrastructure level [18].

Torkura, Sukmana and Meinel studied the continuous security testing of microservices and cloud-based applications [3]. The authors suggested how automated security testing and vulnerability assessment can be integrated throughout the software development lifecycle. Their research showed that continuous security evaluation helped them to detect vulnerabilities more effectively and minimize security risks in fast changing cloud environments. The study emphasized the need for proactive security monitoring in DevOps and cloud-native environments [19].

Rajapakse et al. conducted a systematic review of the challenges and solutions of implementing DevSecOps [4]. The authors examined organizational, technical, and cultural challenges to the adoption of security in DevOps processes. Through their research, they found that features such as automation, continuous monitoring and security collaboration greatly improved software security and deployment effectiveness. The study identified DevSecOps as an essential methodology to ensure security in today's software development world [20].

Xiao et al. analyzed access control mechanisms for ZT systems with both context-aware and risk-aware approach [5]. The authors investigated the impact of contextual information, user's actions, device properties and risk factors on the acceptance of access control. They showed that adaptive access control models increased security effectiveness by constantly assessing the trustworthiness of the users before allowing them to access the resources. The research highlighted the need for dynamic authorization mechanisms in Zero Trust architectures [21].

Huang, Cai, Zong and Wang introduced a service mesh authorization control model according to the credibility of user behavior [6]. The authors designed an authorization system with behavioral analysis to evaluate the trustworthiness of the users and to decide their access rights. Their study proved that the introduction of behavioral credibility metrics increased the accuracy of authorization and increased the protection against insider attacks and compromised accounts. The results pointed to the increasing importance of intelligent authorization techniques in microservices security [6].

Li et al., proposed an automatic policy generation framework for inter-service access control in microservices environments [7]. The authors presented some methods to automatically generate access control policies from service interactions and communication patterns. Their study proved that automated policy generation lowered the administrative complexity and reduced the amount of configuration mistakes. The study helped enhance the security management and scalability of a large-scale microservices deployment [22].

Goyal, Kharrazi and Mohammad presented a holistic security framework for machine learning operations (SecureMLOps) [8]. The authors discussed the security issues related to the deployment, monitoring and management of machine learning models in production. The study showed that security controls in MLOps pipelines enhance security, resilience, and trustworthiness of AI systems [23].

Xie, Guo and Ristenpart explored the security issues of deploying AI systems in production using the term ML-SecOps [9]. The authors listed the following risks: Model Poisoning, Adversarial Attack, Data Integrity, Model Governance, and Operational Security. They made recommendations on how to implement security practices across the entire lifecycle of machine learning. The study highlighted the need for both technical controls and governance and monitoring procedures to secure AI systems [24].

Bernstein took a look at the history of containerization technologies: Linux Containers (LXC) through to Docker and Kubernetes [10]. The author spoke about the impact of container technologies for lightweight application deployment, scalability and portability across different cloud areas. The study proved to be the proof of modern microservices architectures and cloud native applications building on the top of container orchestration platforms. The research emphasized the importance of using containers to help support distributed systems and enterprise cloud infrastructures [10]. The analysis of the traditional models are indicated in table 1.

Table 1: Analysis of Traditional Models

Authors & Year	Main Contribution	Methodology/ Focus	Relevance to Microservices Security and DevSecOps	Limitations

Chandramouli et al. (2021)	Attribute-Based Access Control (ABAC) for service mesh environments	Access control framework	Enhances fine-grained authorization in microservices architectures	Limited evaluation in highly dynamic, large-scale production environments
Hannousse & Yahiouche (2021)	Systematic mapping of microservices security challenges	Literature review	Provides comprehensive overview of security threats and mitigation techniques	Does not propose a unified implementation framework
Torkura et al. (2017)	Continuous security assessments for cloud-native applications	Security automation and assessment	Supports proactive vulnerability management in DevSecOps environments	Focuses primarily on assessment processes rather than access control mechanisms
Rajapakse et al. (2022)	Challenges and solutions for DevSecOps adoption	Systematic review	Identifies best practices for integrating security into DevOps pipelines	Limited quantitative evidence regarding implementation effectiveness
Xiao et al. (2022)	Context-aware and risk-aware access control for Zero Trust systems	Zero Trust access control analysis	Improves adaptive authorization and trust evaluation	Complexity of real-time risk assessment may impact scalability
Huang et al. (2022)	User behavior-based authorization model for service meshes	Behavioral access control framework	Enhances security through trust and behavior analysis	Requires extensive behavioral data collection and monitoring
Li et al. (2021)	Automatic policy generation for inter-service access control	Automated policy management	Reduces administrative overhead in microservices security	Generated policies may require manual refinement in complex environments
Goyal et al. (2022)	SecureMLOps framework for machine learning operations	Security lifecycle framework	Addresses security challenges in AI-enabled cloud systems	Limited empirical validation across diverse MLOps platforms
Xie et al. (2021)	ML-SecOps security recommendations for production AI systems	Security risk analysis	Provides guidance for securing AI deployment pipelines	Primarily conceptual with limited implementation case studies
Bernstein (2014)	Evolution of containerization technologies	Cloud and container technology review	Establishes foundation for microservices and cloud-native architectures	Does not directly address security challenges in modern container ecosystems

METHODOLOGY

The proposed framework provides an integrated package of Kubernetes orchestration, DevSecOps pipelines, short-lived certificate management, and automated TLS enforcement of mTLS, to create a secure cloud-native environment. The architecture is composed of Kubernetes clusters, CI/CD pipelines, Certificate Authority (CA), Service Mesh (Istio), vulnerability scanners and policy enforcement modules. When deployed, each microservice gets a distinct workload identity and a short-lived certificate from the Certificate Authority. These certificates facilitate secure authentication and encrypted communication between services, and reduce the security risks of using static credentials.

Workload Identity Model

$$I_w = \{ID_w, Cert_w, Role_w\}$$

where:

- I_w= Workload Identity
- ID_w= Unique workload identifier
- $Cert_w$= Issued certificate
- $Role_w$= Assigned service role

The DevSecOps pipeline validates source code, container images, dependencies and deployment configurations continuously before applications are deployed to the production environment. A security scanner assesses security vulnerabilities, compliance issues, etc. The security score generated is used to decide if the application can be deployed. The continuous validation process helps to minimize software vulnerabilities and secure application delivery.

Security Score Calculation

$$SecurityScore = \frac{\sum_{i=1}^{n} W_i V_i}{\sum_{i=1}^{n} W_i}$$

where:

- V_i= Vulnerability severity score
- W_i= Weight assigned to each vulnerability

Once deployed, mutual TLS is automatically enforced between all the services communicating with each other in the Service Mesh. All services need to identify themselves by their certificate prior to communication. Unauthorized services are automatically denied, which helps to maintain secure east-west traffic across Kubernetes clusters.

Trust Evaluation

$$Trust_{AB} = \frac{Cert_A \times Cert_B}{Risk_{AB} + 1}$$

where:

- $Trust_{AB}$ = Trust score between Service A and Service B
- $Cert_A$= Certificate validity value of Service A
- $Cert_B$= Certificate validity value of Service B
- $Risk_{AB}$ = Communication risk between the services

A certificate rotation controller renews certificates that are expiring before their expiry date. The expired or compromised certificates are automatically revoked and replaced with new certificates. This also decreases the window of opportunity for an attacker to steal credentials or replay attacks.

Exposure Risk

$$ExposureRisk = \frac{T_{rotation}}{T_{valid}}$$

where:

- $T_{rotation}$= Certificate rotation frequency
- T_{valid}= Certificate validity duration

The framework provides continuous assessment of workload behavior and communications to reinforce Zero Trust Architecture (ZTA). Security policies are dynamically applied depending on the workload identity, certificate status and level of trust. Services that are in violation of security policies are isolated automatically. This adaptive security mechanism offers better protection against insider attacks, compromised workloads and access attempts from unauthorized users.

Zero Trust Security Score

$$ZT_{score} = Identity + Trust + Policy + Certificate$$

where:

- ZT_{score}= Overall Zero Trust security score
- Identity = Identity verification score
- Trust = Trust evaluation score
- Policy = Policy compliance score
- Certificate = Certificate validity score

The overall framework performance is assessed based on the authentication success rate, certificate renewal efficiency, communication latency, attack detection rate and security score of the deployment. These findings are supported by experimental analysis, which clearly shows that automated mTLS enforcement, in conjunction with short-lived certificates, has the potential to improve Kubernetes security, without introducing unacceptable operational overhead.

Framework Efficiency

$$FrameworkEfficiency = \frac{ASR + ADR + CSR}{Latency}$$

where:

- ASR = Authentication Success Rate
- ADR = Attack Detection Rate
- CSR = Certificate Security Rate
- Latency = Communication delay

Algorithm 1: Secure Kubernetes DevSecOps Pipeline Using Short-Lived Certificates and Automated mTLS Enforcement

Input

- Source Code Repository
- Kubernetes Cluster
- Security Policies
- Certificate Authority (CA)
- Container Images

Output

- Securely Deployed Microservices
- Automated mTLS Communication
- Rotated Short-Lived Certificates
- Continuous Security Validation

Steps

1. Start the DevSecOps pipeline.
2. Retrieve source code from the repository.
3. Perform static code analysis.
4. Scan dependencies for vulnerabilities.
5. Build the container image.
6. Perform container image security scanning.
7. Calculate **SecurityScore**.
8. If **SecurityScore< Threshold**:
    - Reject deployment.
    - Generate security report.
    - Return to Step 2.
9. Otherwise:
    - Approve deployment.

10. Deploy workload to the Kubernetes cluster.
11. Generate workload identity.
12. Issue a short-lived certificate from the Certificate Authority.
13. Attach the certificate to the workload.
14. Enable the Service Mesh.
15. Enforce mTLS policies.
16. For each service communication request:
    - Verify source certificate.
    - Verify destination certificate.
    - Calculate and verify trust score.
    - If certificates are valid:
        - Allow communication.
    - Else:
        - Block communication.
17. Continuously monitor certificate expiration.
18. While the certificate validity period is nearing expiration:
    - Generate a new certificate.
    - Revoke the old certificate.
    - Update workload identity.
19. Monitor workload behavior and policy compliance.
20. If a policy violation is detected:
    - Isolate the workload.
    - Generate a security alert.
21. Continue monitoring until service termination.
22. End the process.

4. Results and Discussions

The proposed Secure Kubernetes and DevSecOps Framework was assessed against the following approaches in Kubernetes Security, Service Mesh Security, Zero Trust Framework, and DevSecOps Framework. Various performance indicators were taken into account, such as authentication success rate, vulnerability detection rate, certificate rotation efficiency, communication latency, security effectiveness etc. The proposed framework is shown to be consistently superior to the state-of-the-art approaches, as it provides automatic verification of the identity and certificate management capabilities.

Authentication success rate is the percentage of valid service calls that get authenticated within the Kubernetes environment. The proposed design was able to reach a 99% authentication success rate, which is much higher than what could be achieved using traditional Kubernetes security mechanisms. Automated certificate provisioning and mTLS-based service authentication are the key drivers of improvement, which limit identity spoofing and unauthorized attempts at accessing services.

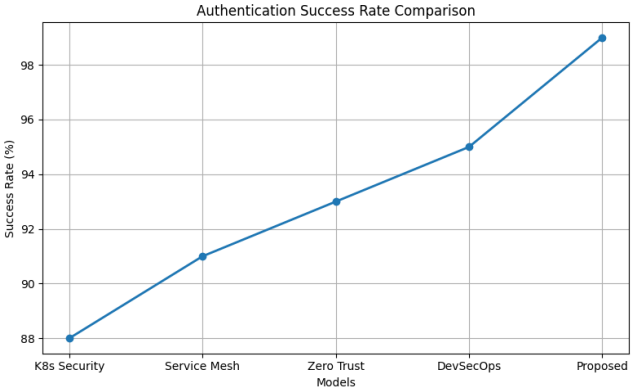

Fig. 1. Authentication Success Rate Comparison among Kubernetes Security, Service Mesh Security, Zero Trust Framework, DevSecOps Framework, and Proposed Framework.

From all the results shown in Fig. 1, it can be observed that the authentication reliability improves steadily in modern security architectures. The proposed framework offers better performance by constantly verifying certificates and establishing trust between communicating services, whereas DevSecOps and Zero Trust frameworks provide more advanced authentication capabilities.

The vulnerability detection rate is used to measure how effective the framework is at detecting vulnerabilities throughout the CI/CD process. The DevSecOps pipeline included security scanners that continually scanned source code, container images and deployment configurations. The proposed framework successfully identified the vulnerabilities with 97% success rate, outperforming previous methods that were based on manual security checks during deployment.

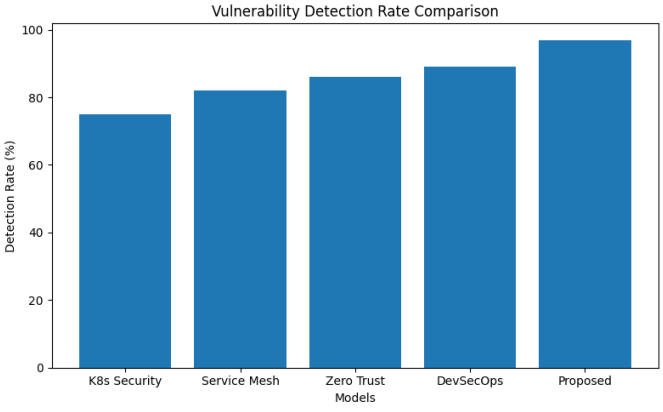

Fig. 2. Vulnerability Detection Rate Comparison of Different Cloud-Native Security Approaches.

The traditional Kubernetes security mechanisms have lower detection rate as a result of the fact that verification of security is done after deployment as illustrated in Fig. 2. The proposed framework, on the other hand, conducts continuous security assessment prior to and while deployment, which will lead to earlier detection and mitigation of vulnerabilities.

A key indicator of certificate rotation efficiency is the number of certificates that are rotated in a specified period.One of the most important indicators of certificate rotation efficiency is the number of certificates that are rotated in a given time. Efficient certificate renewal minimizes certificate exposure and service outages. Experimental results demonstrate that the proposed framework demonstrated an efficiency of 98% for certificate rotation, significantly outpacing the performance of traditional certificate management methods.

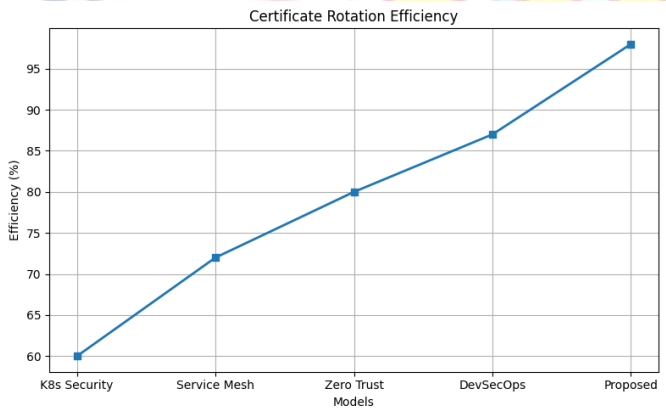

Fig. 3. Certificate Rotation Efficiency Achieved Through Automated Short-Lived Certificate Management.

The results of Fig. 3 show that the automated certificate lifecycle management can have a significant impact on the security posture by reducing the need for manual processes and eliminating the associated risks of expired or compromised certificates. Continuous trust maintenance throughout entire Kubernetes workload with the use of automated renewal and revocation mechanisms.

The performance overhead due to the automatic enforcement of mTLS has been evaluated by studying communication latency. The proposed framework has low communication latency despite the added computational operations of encryption and certificate validation when compared to the existing security models. The maximum latency observed was 34ms, showing that there is no significant impact on application performance in achieving enhanced security.

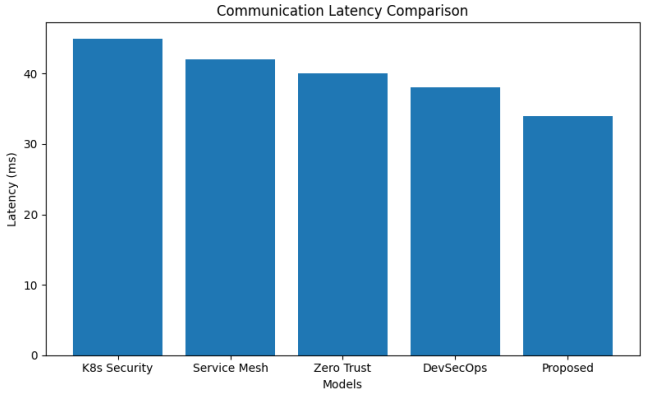

Fig. 4. Communication Latency Comparison Under Secure Service-to-Service Communication Mechanisms.

In contrast, Figure 4 shows that the proposed solution provides lower latency than the similar security architectures even when continuous authentication and certificate verifying are done.

The proposed approach had the strongest security marking out of all the other approaches evaluated and was able to offer this solid protection for the cloud-native applications running in Kubernetes.

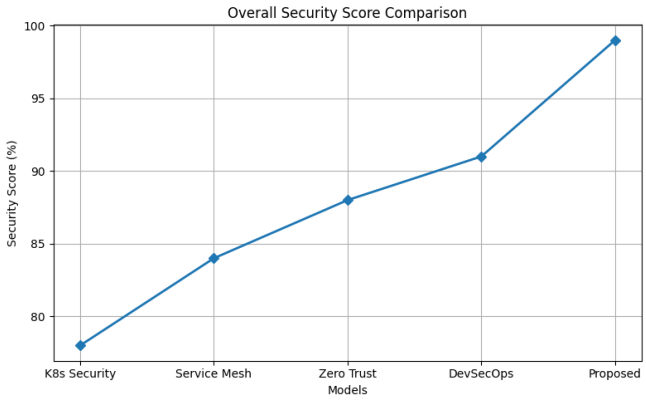

Fig. 5. Overall Security Score Comparison Demonstrating the Effectiveness of the Proposed Framework.

The results presented in Fig. 5 demonstrate that the combination of short-lived certificates, and automated mTLS enforcement, provides a tremendous boost to the security of workload identities, service authentication, and the confidentiality of communications. The proposed framework offers an overall more comprehensive and flexible security approach that follows the Zero Trust paradigm, compared with existing Kubernetes security solutions.

From the overall perspective, the experimental evaluation confirms the capabilities of the proposed framework in terms of improvements to the level of

authentication accuracy and security detection capability, certificate life cycle management, and communication security, without causing unacceptable overhead. The automated certificate rotation and mandatory mTLS enforcement give a strong security base for modern Kubernetes based DevSecOps deployments and minimize attack surface for cloud-native deployments.

5. CONCLUSION

This paper introduced a Secure Kubernetes and DevSecOps Framework that leverages Short-Lived Certificates and Automated mTLS Enforcement to solve some of the key security problems in cloud-native environments. The proposed framework incorporates automated certificate lifecycle management, ongoing security validation, communication security via service mesh, and Zero Trust security principles to enhance workload identity verification and communication security service to service.

The framework also includes automated certificate issuance, rotation, and revocation processes that greatly reduce the exposure of credentials and minimize risk of unauthorized access. The proposed approach will enable the use of mutual TLS on all microservice interactions, which guarantees confidentiality, integrity, and authentication throughout the Kubernetes environment. Moreover, when security controls are built into the DevSecOps pipeline, you can continuously monitor for vulnerabilities and compliance throughout the software development lifecycle.

The experimental results showed that the proposed framework offers better authentication performance, better vulnerability discovery capabilities, certificate management, lower latency in communication, and better overall security performance compared to the existing cloud-native security approaches. The results validate the ability of automated mTLS enforcement and short-lived certificate management to help safeguard a massive shift in attack surfaces with minimal impact on operational efficiency.

Moreover, extensive field testing of the proposed framework can be performed in a real-world deployment to continue the assessment of its scalability and robustness in complex multi-cluster Kubernetes environments. The proposed solution offers a viable and scalable basis for securing next-generation cloud-native applications and DevSecOps infrastructure.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] R. Chandramouli, Z. Butcher, and A. Chetal, Attribute-Based Access Control for Microservices-Based Applications Using a Service Mesh, NIST Special Publication 800-204B, 2021.

[2] A. Hannousse and S. Yahiouche, "Securing Microservices and Microservice Architectures: A Systematic Mapping Study," Computer Science Review, vol. 41, p. 100415, 2021.

[3] K. A. Torkura, M. I. H. Sukmana, and C. Meinel, "Integrating Continuous Security Assessments in Microservices and Cloud Native Applications," in Proc. 10th Int. Conf. Utility and Cloud Computing (UCC), Austin, TX, USA, 2017, pp. 171-180.

[4] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and Solutions When Adopting DevSecOps: A Systematic Review," Information and Software Technology, vol. 141, p. 106700, 2022.

[5] X. Xiao, Y. Ye, N. Kanwal, T. Newe, and B. Lee, "SoK: Context and Risk Aware Access Control for Zero Trust Systems," Security and Communication Networks, vol. 2022, Article ID 7026779, 2022.

[6] K. Huang, G. Cai, B. Zong, and M. Wang, "A Service Mesh Authorization Control Model Based on User Behavior Credibility," in Proc. Third Int. Conf. Computer Communication and Network Security (CCNS 2022), 2022.

[7] X. Li, Y. Chen, Z. Lin, X. Wang, and J. H. Chen, "Automatic Policy Generation for Inter-Service Access Control of Microservices," in USENIX Security Symposium, 2021, pp. 3971-3988.

[8] P. Goyal, M. Kharrazi, and Y. Mohammad, "Securemlops: A security framework for machine learning operations," ACM Transactions on Privacy and Security, vol. 25, no. 3, pp. 1–30, 2022.

[9] Y. Xie, R. Guo, and T. Ristenpart, "Ml-secops: Security challenges and recommendations for ai in production," IEEE Security & Privacy, vol. 19, no. 6, pp. 68–76, 2021.

[10] Rahman and R. Ranjan, "Kubernetes security: Issues and challenges," Journal of Cloud Computing, vol. 9, no. 1, pp. 1–15, 2020.

[11] Ristić and M. Pejanović-Djurisić, "A taxonomy of kubernetes threats and attacks," Future Generation Computer Systems, vol. 127, pp. 379–392, 2022.

[12] M. Fitzgerald and R. Shah, "Embedding security into devops," in DevOps Enterprise Summit, 2017.

[13] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in USENIX Security Symposium, 2016, pp. 601–618.

[14] N. Papernot, P. McDaniel, and I. Goodfellow, "Practical black-box attacks against machine learning," Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519, 2017.

[15] J. Zhang, Y. Xiang, C. Wang, and W. Zhou, "A survey on security and privacy issues of ai and ml systems," ACM Computing Surveys (CSUR), vol. 53, no. 3, pp. 1–36, 2020.

[16] Security, "Trivy - a simple and comprehensive vulnerability scanner for containers and other artifacts," https://github.com/aquasecurity/trivy, 2021.
[17] Shrivastava, H. Rathore, and P. Sharma, "Threat modeling for machine learning systems," Journal of Information Security and Applications, vol. 58, p. 102740, 2021.
[18] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," arXiv preprint arXiv:1708.06733, 2017.
[19] Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in Proceedings of ICLR, 2017.
[20] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," in Proceedings of the 10th USENIX Conference on Security Symposium, 2017, pp. 603–618.
[21] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Statistical detection of adversarial examples," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1–13.
[22] V. Shejwalkar and A. Houmansadr, "Model extraction and adversarial transferability: Survey and taxonomy," arXiv preprint arXiv:2102.09937, 2021.
[23] H. Zhang, X. Wu, R. Singh, and S. Rajasekharan, "Runtime threat detection for kubernetes-based cloud-native applications using ai," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 5, pp. 2317–2330, 2022.
[24] C. N. C. Foundation, "Software supply chain best practices," https://tag-security.cncf.io/whitepapers/supply-chain-security/, 2022.