# Zero Trust Framework for Enhancing Security in Container Ecosystems

Santosh Kumar Upadhyaya[1], Dhanvi Medha Beechu[1], and B Thangaraju[1]

International Institute of Information Technology, Bangalore, India, Department of Computer Science and Engineering.

**Abstract.** As organizations accelerate their digital transformation, container technologies have emerged as a cornerstone for application deployment, emphasizing the need for robust security measures. Traditional perimeter-based security models are insufficient to address the dynamic and distributed nature of container environments. This paper introduces a Zero Trust Container Framework (ZTCF), rooted in the "never trust, always verify" principle, to enhance container ecosystem security. The framework integrates identity verification, continuous monitoring, and automated threat response to address specific vulnerabilities and requirements throughout the container lifecycle. Experimental analysis comparing legacy systems with the proposed framework demonstrates significant improvements in security posture, compliance, and resilience. This research offers actionable insights for designing secure, scalable, and resilient containerized systems, guiding organizations in strengthening their security strategies as they adapt to the evolving digital landscape.

**Keywords:** Zero Trust · Container Security · Network Security · Container Technologies · Digital Transformation · Cloud Computing.

## 1 Introduction

The rapid adoption of container technologies has transformed the way organizations develop, deploy, and manage applications. Containers offer scalability, portability, and efficiency, making them a critical component in modern software architectures. However, this widespread adoption also introduces unique security challenges, particularly in dynamic, distributed, and cloud-native environments. Traditional perimeter-based security models, which rely on implicit trust within the network, are no longer adequate for securing containerized ecosystems. This gap underscores the need for a paradigm shift toward more robust security approaches. In this context, the Zero Trust (ZT) security model, based on the principle of "never trust, always verify," provides a promising framework. Unlike legacy models, ZT assumes that no system component—whether inside or outside the network—can be inherently trusted. By enforcing strict authentication, continuous monitoring, and policy-driven access control, ZT minimizes the attack surface and mitigates potential risks. However, implementing Zero Trust principles in containerized environments presents challenges, such as managing complex networks[1], maintaining real-time compliance [2], and ensuring interoperability across diverse platforms. This research focuses on the development and application of a Zero Trust Container Framework (ZTCF) tailored specifically for container ecosystems. The proposed framework addresses security vulnerabilities across the entire container lifecycle, from development to deployment, incorporating cutting-edge practices like identity verification, posture management, and automated policy enforcement. By leveraging this framework, organizations can secure their containerized applications against modern threats while maintaining operational efficiency. The proposed framework is particularly valuable in industries where data security and system reliability are paramount. For example:

- Financial Services: With the growing reliance on microservices and cloud-based applications, financial institutions can use ZTCF to secure sensitive data transactions and ensure compliance with regulations like General Data Protection Regulation(GDPR) and Payment Card Industry Data Security Standard (PCI DSS).
- Healthcare: ZTCF can safeguard patient data in healthcare systems, addressing vulnerabilities in containerized environments that host electronic health records (EHR) and telemedicine platforms.
- E-commerce: Online retailers can protect user data and payment systems by implementing Zero Trust security measures in their container-based architectures.

Consider the case of a leading financial services company adopting containerized microservices for its payment gateway. Traditional perimeter security left the system vulnerable to lateral movement attacks. By integrating Zero Trust principles, the organization implemented continuous monitoring, multi-factor authentication (MFA), and real-time policy enforcement. This transformation significantly reduced the risk of breaches, improved compliance, and enhanced user trust in its digital services. This paper systematically explores the challenges and opportunities of adopting Zero Trust principles for container security. It evaluates existing practices, presents the architecture and modules of the proposed ZTCF, and validates its effectiveness through experimental analysis. The findings provide actionable insights for practitioners and researchers aiming to strengthen security in containerized environments, ensuring scalability and resilience in an evolving digital landscape.

### 1.1 Our Contribution

While Zero Trust Architecture (ZTA), network security, audit, and compliance are often discussed independently as mentioned in [3], [4], their integration into a unified framework is essential. The primary contribution of this paper is the analysis of existing security issues within the container lifecycle and the proposal of a comprehensive security framework based on the principles of Zero Trust (ZT) called Zero Trust Container Framework (ZTCF). This ZTCF is tailored for the entire container lifecycle and applies to both development and deployment processes. The security analysis is conducted by comparing the experimental results from the legacy system with those from the ZTCF.

### 1.2 Organization of the Paper

The remainder of this paper is organized as follows: Section 2 reviews the existing literature on Zero Trust Architecture (ZTA) and its application to container security. Section 3 provides an overview of the Zero Trust principles and their relevance to modern IT environments. Section 4 provides an overview of the container security. Section 5 discusses the unique security challenges in containerized ecosystems and highlights the need for a specialized framework. Section 6 presents the proposed Zero Trust Container Framework (ZTCF), detailing its architecture and key modules. Section 7 describes the experimental setup, security analysis methodology, and comparative results, illustrating the efficacy of the framework. Finally, Section 8 concludes the paper by summarizing the findings and outlining potential directions for future research, with a focus on enhancing posture management and compliance auditing in containerized environments.

## 2 Literature Review

The literature on implementing Zero Trust Architecture (ZTA) in container security is comprehensive, addressing theoretical foundations and practical strategies. Ray et al. review the background, technologies, and challenges of ZTA [5]. Kindervag J. introduced the Zero Trust model, advocating a departure from 'trust but verify' [6]. Surantha et al. explore using ZT to secure container deployments within network boundaries [7]. Leahy et al. offer a framework for ZT in container security, emphasizing trust evaluation [8]. NIST views ZTA as a strategic framework focusing on trust-building [9]. Moric et al. discuss a security framework for container orchestration [10]. This research builds on these studies to explore the challenges and opportunities of ZTA in container security, informing further analysis and discussion.

## 3 Understanding Zero Trust (ZT)

Zero Trust (ZT) represents a shift from traditional perimeter-based security to a model that assumes no implicit trust, inside or outside the network. ZT ensures thorough authentication, authorization, and encryption for every access request, reducing lateral movement risks through continuous identity verification. It utilizes technologies like multifactor authentication (MFA), micro-segmentation, and real-time analytics to enforce strict access controls, focusing on least privilege and continuous verification. This framework secures modern IT environments, including cloud services and IoT devices, emphasizing trust over specific technical solutions. ZT benefits from advances in AI, ML [11], [12], IAM, encryption, and analytics, aiding real-time access management. In container security, ZT enhances security but also presents unique challenges, which this research explores.

## 4 Overview of Container Security

Container security [13] is vital for safeguarding modern applications, addressing challenges like vulnerabilities in container images [14] and runtime threats [15]. A multi-layered approach is needed, including vulnerability scanning, trusted registries, robust access controls, and continuous monitoring, as analyzed by Eldjoy et al. [16]. While tools like Kubernetes and Docker provide basic security features, integrating additional solutions strengthens the protection of containerized applications.

## 5 Implementing ZT in Container Security

Adopting Zero Trust (ZT) for container security involves a thorough strategy to protect containerized environments. Containers offer scalability and uniformity but present unique security issues that Zero Trust Architecture (ZTA) can address. Runtime security in a ZT framework involves continuous monitoring and real-time threat detection. Tools like Kubernetes and Docker offer built-in security features, but integrating solutions like intrusion detection systems (IDS) and endpoint detection and response (EDR) enhances protection. Implementing ZT requires a paradigm shift, emphasizing continuous verification and enforcing the principle of least privilege. Regular audits and compliance checks ensure security policies are enforced and updated for emerging threats. By adopting ZT principles, organizations can secure containerized environments, ensuring that access requests are authenticated, authorized, and encrypted, thus enhancing IT infrastructure resilience.

## 6 Proposed ZT Container framework (ZTCF)

The fundamental principles of ZT are as follows.

- **Foster Trust:** Establish and maintain a secure environment where trust is continuously evaluated.
- **Apply Trust-Based Access Policies:** Implement access controls based on the trustworthiness of users and devices.
- **Persistent Trust Verification:** Continuously verify and validate trust for all access requests.
- **Dynamics Trust Management:** Adapt trust levels in real-time based on changing conditions and behaviours.

Building on the 4 principles discussed above, the proposed ZTCF is presented in the following Fig. 1. ZTCF consists of several key modules, each essential for maintaining robust container security. The following sections outline the key modules.

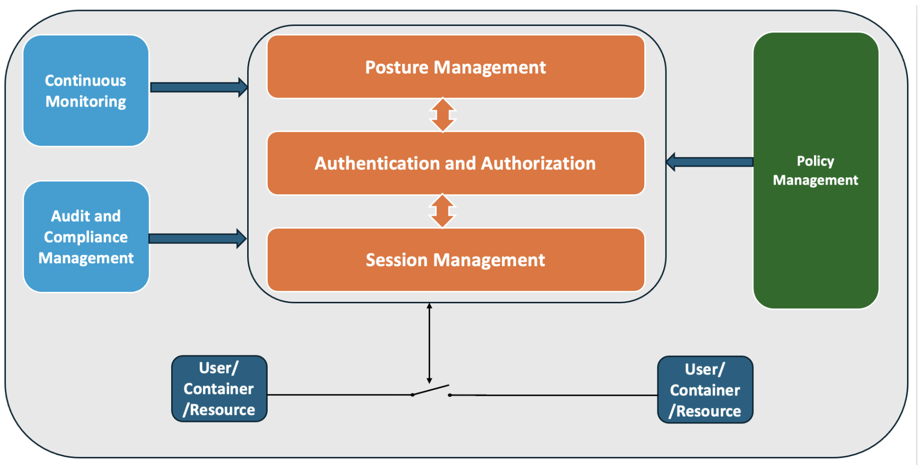

Continuous Monitoring
Audit and Compliance Management
Posture Management
Authentication and Authorization
Session Management
Policy Management
User/ Container /Resource
User/ Container /Resource

**Fig. 1.** Architecture of the Zero Trust Container Framework.

## 6.1 Session Management

When an entity requests resource access, the session management layer processes the request. It forwards the request to the authentication and authorization layer and acts based on the response, serving as a policy enforcement point to either allow or deny access. The session management layer is responsible for creating sessions for all requests, handling resource access requests, and accepting only encrypted requests. It verifies the authenticity and integrity of access requests, creates and forwards authentication requests to the authentication and authorization layer, handles responses from this layer, and enforces policies. Additionally, it manages dynamic trust.

## 6.2 Authentication and Authorization

The Authentication and Authorization layer is vital for system trust, working closely with posture management to ensure robust authentication and authorization. It verifies user identities via authentication servers and uses posture management for secondary context-based checks, enhanced by MFA. It determines access rights through the authorization server, retrieving and conveying policies. Strong encryption for data in transit and at rest is essential, using secure protocols for data exchanges. These servers can be internal or external, offering flexible deployment and integration.

## 6.3 Posture Management

Posture involves verifying that endpoint devices meet required security standards before network access. This process evaluates devices like computers and smartphones for compliance with security policies, preventing potential risks.

The posture management layer [17] enhances authentication and authorization and supports continuous monitoring. It collects contextual data from users and devices to inform decisions, considering factors like time, location, and other contexts for accurate security assessments. For instance:

– Time: Monitors access patterns to detect anomalies.
– Geographical Location: Confirms access from expected locations, flagging high-risk areas.
– Contextual Information: Uses device type, network, and user behavior for informed access decisions.

### 6.4 Audit and Compliance

This layer handles detailed logging of access and system events. To ensure container compliance before deployment, integrate automated compliance scans into the Continuous Integration and Continuous Deployment (CI/CD) pipeline [4]. Defining security policies as code ensures consistency and simplifies compliance management. Using AI and ML enhances audit data analysis, providing insights for continuous monitoring. These technologies help organizations detect and respond to security issues more effectively.

### 6.5 Continuous Monitoring System

Continuous monitoring uses real-time surveillance and analysis to detect threats in container environments, reducing breaches. It maintains a dynamic security posture with regular trust monitoring and communication with the session manager. Key components include automated compliance scans, AI-driven anomaly detection, and alerts, all of which enhance visibility, compliance, and security.

### 6.6 Policy Management System

The policy management system handles static and dynamic policies, sharing them with the session management layer. Insights from continuous monitoring and posture management inform dynamic policy creation. Additionally, role-based access policies are developed and managed within the system to govern access requests. These modules collectively form a cohesive security model that mitigates risks throughout the container lifecycle.

## 7 Security Analysis

The container lifecycle includes both development and deployment phases. According to [8], the development phase involves activities such as coding, writing Docker files, creating images, running containers, and testing. If requirements are met, changes are pushed to the code repository; if not, the process returns to further development. Once code is committed, it moves to the deployment phase. Our proposal addresses both the container development and deployment processes.

### 7.1 Security analysis methodology

The security analysis process involves initially performing tests without the proposed ZTCF, identifying security issues, and then repeating the same tests with the proposed ZTCF. This is followed by a security analysis and a comparison between the security of the legacy framework and ZTCF. The experiment is carried out for container development and deployment scenarios using both the legacy framework and ZTCF, and the results are then compared against the following criteria.

**Table 1. ZTCF Security Analysis Criteria**

| Criteria | Explaination |
|---|---|
| Strength of Authentication Mechanisms | Evaluates authentication methods' robustness levels |
| Granularity of Authorization | Analyzes system's fine control over permissions/access |
| Posture Management | Examines each system's asset inventory maintenance |
| Logging Capabilities | Assessing logging mechanisms' comprehensiveness |
| Compliance Adherence | Checks systems' adherence to regulatory requirements |
| Real-Time Monitoring | Assesses real-time monitoring of activities and threats |
| Dynamic Policy Management | Evaluates ease of updating policies for current threats |
| Context-Awareness | Analyzes each system's contextual decision-making ability |
| Dynamic Trust Adjustment | Examines dynamic trust adjustment via continuous verification |
| Incident Response | Examines dynamic trust adjustment via continuous verification |
| Disaster Recovery | Evaluates disaster recovery plans and their efficacy |
| Ease of Use | Assesses each system's user-friendliness for end-users |
| Administrative Overhead | Compares administrative effort needed for system upkeep |

### 7.2 Applying ZTCF for Container Development

Implementing the Zero Trust Container Framework (ZTCF) in container development enhances security by verifying all entities continuously, without default trust. This involves rigorous authentication and authorization, network segmentation, and strict access controls. Comprehensive monitoring and logging offer real-time visibility into container activities, enabling prompt threat detection and response.

In the coding phase, workplace security is enforced through session management and multi-factor authentication, with policy-driven access controls and continuous monitoring. During image creation, Docker files guide the secure construction of immutable images. For running containers, only authenticated users with sufficient privileges are allowed, with ongoing endpoint and host security checks. Testing involves integrating unit and smoke tests into continuous monitoring to ensure workload security, logging events as part of audit and compliance management. When pushing code, ZTCF ensures secure authentication and authorization, with continuous monitoring to meet all conditions.

**Table 2. ZTCF Security Analysis for Container Development**

| Criteria | Legacy | ZTCF |
|---|---|---|
| Strength of Authentication Mechanisms | Simple Auth | MFA, Biometric |
| Granularity of Authorization | Basic | RBAC, ABAC |
| Posture Management | Periodic | Real-time |
| Logging Capabilities | Less detailed, De-centralised | Detailed, Centralised |
| Compliance Adherence | Manual | Inbuilt |
| Real-Time Monitoring | Periodic, Rule based | Real-time, AI/ML Support |
| Dynamic Policy Management | Static policy | Dynamic Policy support |
| Context-Awareness | None | Supported |
| Dynamic Trust Adjustment | Static | Supported |
| Incident Response | Traditional | Advanced |
| Disaster Recovery | Less automated | Fully automated |
| Ease of Use | Simple | Complex |
| Administrative Overhead | Minimum | Visible |

**Summary of Security Analysis:** Each phase of container development is integral to workplace and workforce security. Security analysis is detailed in Table 2. Throughout coding, Docker file creation, image creation, container execution, and testing, workforce security is maintained via session management, authentication, and authorization, while workplace security is ensured through posture management. Continuous monitoring, auditing, and compliance are crucial across all phases. Session management addresses changes in trust, taking necessary actions. During development, integrating static analysis tools, code reviews, and regression tests into continuous monitoring boosts security. If issues arise, session management prevents code from being committed to the repository.

## 7.3 Applying ZTCF for Container Deployment

In the deployment phase, one or more containers operate coherently to meet specific requirements. There are primarily 4 scenarios:

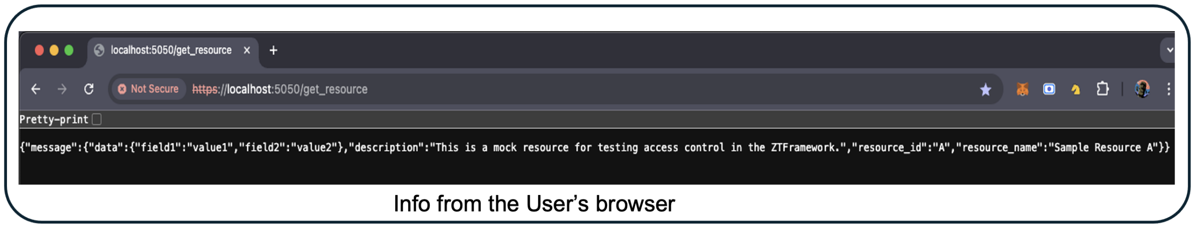
- User-to-Container Communication
- Independent Container
- Container-to-Container Communication
- Container-to-Resource Communication

The security analysis is conducted for each of the aforementioned scenarios.

**User-to-Container Communication (use case: Grant User Access to a resource through remote API)** We can get access to the resource inside the container by running `http://<ip:port>/get_resource` on the browser. Access to the resource is granted, and the user can view the details in the browser.

There are following issues with this.

- Request over the non-secure channel is accepted


localhost:5050/get_resource
Not Secure
https://localhost:5050/get_resource
Pretty-print
Info from the User's browser

```
{"message":{"data":{"field1":"value1","field2":"value2"},"description":"This is a mock resource for testing access control in the ZTFramework.","resource_id":"A","resource_name":"Sample Resource A"}}
```

**Fig. 2.** resource access through remote API

– Missing authentication
– Missing authorization
– Missing explicit policy for the resource access
– The device's posture has not been verified

The proposed ZTCF ensures that the session management layer accepts only encrypted requests. Authentication and authorization management verify that resource requests are authenticated and authorized. Posture management confirms that endpoint devices run the latest software with all patches applied. The policy management layer creates a new policy to authorize specific resource requests, enhanced by specifying allowed IP addresses and device types for access.

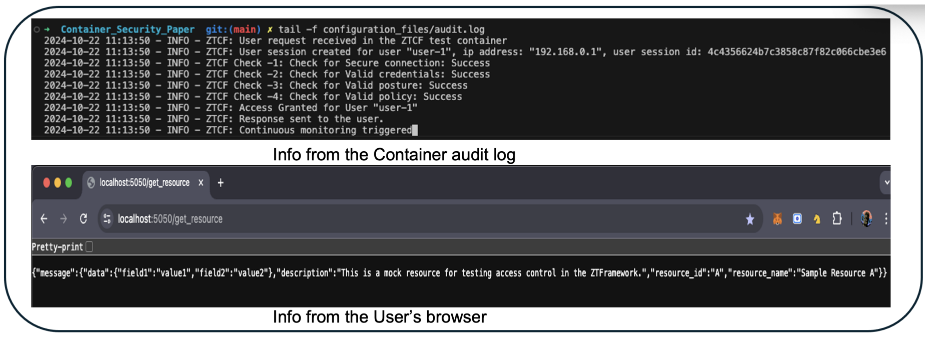
The test was repeated using ZTCF, and the observations are detailed below.

– Request on insecure connection (HTTP) is denied. It is only accepted on secure connection (HTTPS).
– User authentication is done, after initial authentication done, MFA is triggered.
– The posture of the endpoint is checked against the posture policy. If the posture meets the required policy criteria, then the authentication moves to the next layer. Otherwise, authentication fails.
– For testing the policy, the resource policy is changed to deny and confirmed that the resource access is denied.
– Updated the policy to allow access from mobile and retested with an Android phone that is not up to date. Observed that the audit message was generated for the recommendation to update the device software version, and access is allowed. Fig.3

**Summary of Security Analysis:**

The summary of security analysis for grant user access to remote api use case is provided in Table. 3.

– Authentication Mechanisms: Legacy systems rely on insecure credentials, while ZTCF uses multi-factor, biometric, and context-based methods.
– Authorization Granularity: Legacy systems have coarse controls; ZTCF provides fine-grained, role-based, and attribute-based access.
– Posture Management: Legacy systems use periodic scans; ZTCF continuously assesses posture in real-time with compliance integration.



```
○ → Container_Security_Paper git:(main) ✗ tail -f configuration_files/audit.log
  2024-10-22 11:13:50 - INFO - ZTCF: User request received in the ZTCF test container
  2024-10-22 11:13:50 - INFO - ZTCF: User session created for user "user-1", ip address: "192.168.0.1", user session id: 4c4356624b7c3858c87f82c066cbe3e6
  2024-10-22 11:13:50 - INFO - ZTCF Check -1: Check for Secure connection: Success
  2024-10-22 11:13:50 - INFO - ZTCF Check -2: Check for Valid credentials: Success
  2024-10-22 11:13:50 - INFO - ZTCF Check -3: Check for Valid posture: Success
  2024-10-22 11:13:50 - INFO - ZTCF Check -4: Check for Valid policy: Success
  2024-10-22 11:13:50 - INFO - ZTCF: Access Granted for User "user-1"
  2024-10-22 11:13:50 - INFO - ZTCF: Response sent to the user.
  2024-10-22 11:13:50 - INFO - ZTCF: Continuous monitoring triggered
```

Info from the Container audit log

```
{"message":{"data":{"field1":"value1","field2":"value2"},"description":"This is a mock resource for testing access control in the ZTFramework.","resource_id":"A","resource_name":"Sample Resource A"}}
```

Info from the User's browser

**Fig. 3.** Authentication Success by ZTCF.

- Logging: Legacy systems have decentralized logging; ZTCF offers centralized, comprehensive logs for real-time analysis.
- Compliance: Legacy systems require manual checks; ZTCF automates compliance checks and reporting.
- Monitoring: Legacy systems monitor infrequently; ZTCF enables real-time monitoring with advanced analytics.
- Dynamic Policy Management: Legacy systems have static policies; ZTCF supports dynamic, automated updates based on threats.
- Context-Awareness: Legacy systems lack context-awareness; ZTCF integrates user behavior and device posture.
- Dynamic Trust Adjustment: Legacy systems have static trust; ZTCF continuously adjusts trust based on assessments.
- Incident Response: Legacy systems rely on manual responses; ZTCF automates detection and remediation.
- Disaster Recovery: Legacy systems use outdated technologies; ZTCF automates recovery with regular testing.
- Ease of Use: Legacy systems feature outdated interfaces; ZTCF prioritizes intuitive user experiences.
- Administrative Overhead: Legacy systems increase manual tasks; ZTCF reduces overhead through automation.

**Independent Container (Use case: Review an Existing Container)** Continuously scanning images during deployment and maintaining an audit log, the ZTCF provides a comprehensive security trail, enabling prompt detection and remediation of potential threats. This continuous monitoring approach offers significantly enhanced security for container environments compared to traditional offline image scanning methods.

**Container-to-Container Communication** The experimental result for container-to-container communication is similar to the user-to-container experimental result.

**Table 3. ZTCF Security Analysis for Grant User Access to Remote API**

| Criteria | Legacy | ZTCF |
|---|---|---|
| Strength of Authentication Mechanisms | No Auth | Auth with MFA |
| Granularity of Authorization | No policy | RBAC |
| Posture Management | No Support | Supported |
| Logging Capabilities | Basic | Advanced with AI/ML capability |
| Compliance Adherence | Periodic | Inbuilt, Compliance as a service |
| Real-Time Monitoring | Periodic | Real-time |
| Dynamic Policy Management | NA | NA |
| Context-Awareness | NA | NA |
| Dynamic Trust Adjustment | NA | NA |
| Incident Response | Slower, Manual | Faster, Automatic |
| Disaster Recovery | Manual | Auto failover |
| Ease of Use | Easy | Complex |
| Administrative Overhead | Minimal | Significant |

**Container-to-Resource Communication** The experimental result for container-to-resource communication is similar to the user-to-container experimental result.

The complete result for Container-to-Container Communication and container-to-resource communication will be uploaded to GitHub and the link will be shared once the work presented in this paper is accepted.

## 8 Conclusion

The increasing adoption of container technologies necessitates robust security measures to protect containerized environments from evolving threats. This research addresses this critical need by proposing the Zero Trust Container Framework (ZTCF), a comprehensive solution designed to enhance security across the container lifecycle. Rooted in the Zero Trust principles of "never trust, always verify," the ZTCF integrates advanced features such as continuous monitoring, dynamic policy enforcement, posture management, and multi-factor authentication to mitigate vulnerabilities inherent in containerized ecosystems.

The experimental results highlight the effectiveness of the ZTCF in comparison to legacy systems. Key improvements include stronger authentication mechanisms, granular access controls, enhanced compliance adherence, and real-time monitoring capabilities. These enhancements significantly reduce the attack surface, ensure dynamic trust adjustments, and provide automated incident response mechanisms, leading to a marked improvement in the security posture of container environments. Furthermore, the proposed framework demonstrates its practical applicability in addressing critical industry needs, such as securing sensitive data in financial services, ensuring regulatory compliance in healthcare, and protecting user information in e-commerce platforms. By addressing both development and deployment phases of the container lifecycle, the ZTCF offers a holistic approach to container security.

The ZTCF not only meets the security demands of dynamic, distributed environments but also sets a foundation for future enhancements. As container technologies evolve, incorporating advancements in artificial intelligence, machine learning, and compliance automation will further strengthen the framework. Future work will focus on refining posture management, integrating advanced analytics, and extending ZTCF's applicability to emerging use cases, ensuring resilience in the face of ever-changing security landscapes.

## References

1. Sushant Chamoli and Varsha Mittal. Docker networking: A security review. *2023 7th ICOEI.*
2. Xin Lin, Lingguang Lei, Yuewu Wang, Jiwu Jing, Kun Sun, and Quan Zhou. A measurement study on linux container security: Attacks and countermeasures.
3. Farhan A Qazi. Study of zero trust architecture for applications and network security. 2022.
4. Santosh Kumar Upadhyaya and B. Thangaraju. A novel method for trusted audit and compliance for network devices by using blockchain. In *2022 IEEE CONECCT)*, 2022.
5. Partha Ray. Web3: A comprehensive review on background, technologies, applications, zero-trust architectures, challenges and future directions. 2023.
6. Kindervag. No more chewy centers: The zero trust model of information security, 2010. Forrester Research Inc, Cambridge.
7. Nico Surantha and Felix Ivan. Secure kubernetes networking design based on zero trust model: A case study of financial service enterprise in indonesia. 2019.
8. Darragh Leahy and Christina Thorpe. Zero trust container architecture (ztca). 2022.
9. Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture, 2020.
10. Zlatan Moric, Vedran Dakic, and Matej Kulic. Implementing a security framework for container orchestration. 2024.
11. Saubhagya Munasinghe, Nuwan Piyarathna, Erandana Wijerathne, Upul Jayasinghe, and Suneth Namal. Machine learning based zero trust architecture for secure networking. 2023.
12. Jayama Pinnamaneni, Nagasundari S, and Prasad B. Honnavalli. Identifying vulnerabilities in docker image code using ml techniques. *2022 2nd ASIANCON.*
13. Olufogorehan Tunde-Onadele, Jingzhu He, Ting Dai, and Xiaohui Gu. A study on container vulnerability exploit detection. *IEEE IC2E 2019.*
14. Soonhong Kwon and Jong-Hyouk Lee. Divds: Docker image vulnerability diagnostic system. *IEEE Access*, 2020.
15. Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. Container security: Issues, challenges, and the road ahead. *IEEE Access*, 2019.
16. Amina Eldjou and Moahmed Elhadi Amoura. Enhancing container runtime security: A case study in threat detection. 2023.
17. Perera. Docker container security orchestration and posture management tool. In *2022 13th ICCCNT.*