# Secure Container Orchestration in Cloud-Native Architectures

Tritons:
Marlon Brenes (1314316,mbrenesr@nyit.edu)
NYIT - Cybersecurity in Data Center Course
Spring 2025

***Abstract*—This paper explores the security challenges and solutions in container orchestration within cloud-native environments. Emphasis is placed on Kubernetes and Docker security practices, highlighting IAM policies, network segmentation, runtime security, and compliance automation. The proposed methodology leverages policy-as-code and runtime monitoring to enhance container security posture. Our results indicate measurable improvement in reducing attack surfaces across multi-tenant cloud deployments.**

## I. Introduction

In cloud-native environments, container orchestration tools like Kubernetes offer scalability and automation but introduce complex security challenges. This paper investigates how to securely orchestrate containers, ensuring availability and integrity across distributed cloud systems.

## II. Background and Related Work

Numerous studies discuss container vulnerabilities and orchestration frameworks:

- Kubernetes network policies [**?**]
- Container runtime isolation mechanisms [**?**]
- RBAC models for cluster security [**?**]

## III. Problem Statement

Despite the efficiency of orchestrators, they remain vulnerable to privilege escalation, pod escape, and misconfigured access controls. These challenges demand a more integrated approach to security.

## IV. Proposed Approach / Methodology

### A. Architecture Overview

### B. Security Measures

- RBAC enforcement and IAM mapping
- Network segmentation using Calico policies
- Runtime scanning with Falco
- Policy-as-Code using Open Policy Agent (OPA)

## V. Analysis and Discussion

The approach was tested in a simulated AWS EKS cluster. Table I shows a reduction in detected CVEs post-policy integration.

architecture_diagram.png

Fig. 1: Proposed Secure Orchestration Architecture

TABLE I: Security Scan Results Before and After Hardening

| Component | Before CVEs | After CVEs |
|---|---|---|
| Kubelet | 12 | 3 |
| Containerd | 9 | 1 |
| Ingress Controller | 7 | 2 |

### A. Code Example: OPA Policy

Listing 1: OPA Policy to Deny Privileged Pods

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "Pod"
  input.request.object.spec.containers[_].security
  msg := "Privileged pods are not allowed"
}
```

## VI. Conclusion

The proposed secure orchestration framework enhances container security by enforcing least privilege, runtime visibility, and continuous compliance checks. Future work will

address AI-driven anomaly detection and cross-cloud policy standardization.

## VII. Aditional Points

Using kubectl.Ai to complete and retrive AI yaml autogenerated:

- Kubernetes network policies [**?**]
- Container runtime isolation mechanisms [**?**]
- RBAC models for cluster security [**?**]

## References

## Appendix

- Kick-off: Define project scope
- Week 2: Setup EKS environment
- Week 4: Integrated Falco and OPA