Journal of Global Research in Electronics and Communication (JGREC)
Volume 2 | Issue 7 | July 2026 | Pages 15-21
DOI: https://doi.org/xxxxxxxxxxxxx
e-ISSN: 2321-3175
REVIEW ARTICLE OPEN ACCESS Available online: https://jgrec.info/index.php/jgrec/index

AI-Powered Security for Microservices and Container Platforms: A Review Analysis

Mr. Mohit Sahu
Assistant Professor
Department of Computer Sciences and Applications
Mandsaur University, Mandsaur
mohit.sahu@meu.edu.in
Corresponding Author* Mohit Sahu (mohit.sahu@meu.edu.in)

Received: 10 July 2026 | **Revised:** 15 July 2026 | **Accepted:** 25 July 2026

***Abstract*—The rapid adoption of microservices architecture and containerized platforms has transformed the development, deployment, and management of modern cloud-native applications. However, their distributed, dynamic, and ephemeral nature introduces significant security challenges, including unauthorized access, API attacks, anomalous behaviour, resource exploitation, and distributed denial-of-service attacks. Traditional rule-based and static security mechanisms often struggle to detect evolving and previously unknown threats in such environments. This review analyzes the application of Artificial Intelligence (AI) for strengthening security across microservices and container platforms. It examines microservices architecture, containerization technologies, orchestration platforms, service mesh technologies, and their associated security challenges. The study further reviews AI-based security approaches, including machine learning, deep learning, anomaly detection, intrusion detection, adaptive API protection, and intelligent resource management. A taxonomy of anomaly detection based on metrics, logs, and traces is presented to highlight different observability perspectives. Recent studies are comparatively analyzed according to their methodologies, findings, limitations, and future directions. The review identifies AI's potential to enable adaptive, scalable, and proactive security while highlighting challenges related to computational cost, data availability, model complexity, false positives, and real-time deployment. Future research should focus on explainable, lightweight, and autonomous AI-driven security frameworks.**

Keywords*—*Artificial Intelligence, Microservices Security, Container Security, Anomaly Detection, AI-Powered Security.

I. INTRODUCTION

Software development involves developing software against a set of requirements. Software testing is needed to verify and validate that the software has been built to meet these specifications [1]. Microservices architecture is a software architectural style that consists of developing an application as a set of services, where each service is executed independently and communicates with the others through lightweight mechanisms [2]. Some of its advantages compared to a monolithic architecture include supporting technological heterogeneity, resilience, scalability, and ease of deployment, among others. It also stands out that microservices help organizations manage development teams better, as they minimize the number of people working on the same source code [3]. This architectural style has been an emerging trend originated in the industry [4], whose popularity has led many organizations to adopt it. Netflix is one of the first success stories known worldwide [5].

The foundational element of microservices architecture is Docker and other containerization technologies. With Docker, developers can create packages for applications that include necessary dependencies in portable containers. The standardized execution environment of these containers provides consistent results throughout different deployment stages, from development to production. All application requirements can be packaged inside containers which addresses the common issue of applications working differently from one machine to another when traditional deployment methods are used. Amazon Elastic Container Registry (ECR) functions as a fully managed Docker container registry service, making the image management process easier. The integration of Amazon Elastic Container Registry with other AWS services enables smooth and scalable deployment of containerized applications. Organizations benefit from the Docker and ECR combination to implement DevOps methodologies, support continuous integration and deployment (CI/CD), and improve development team collaboration [6][7].

Microservices architecture is a design pattern that breaks complex applications into smaller, self-sufficient, and deployable services that communicate via well-defined public APIs, providing benefits such as increased scalability, technology heterogeneity, and easier development. However, this decentralized organization also brings with it major difficulties in maintaining system reliability and in achieving effective fault tolerance [8]. In the microservices world, many components interact with each other via API calls, so only the endpoints are visible to the outside world [9]. Although this makes ESB more integration and extensible, this openness also makes ESB more vulnerable to attacks such as unauthorized access, injection attacks, data tampering, and DDoS attacks[10]. Some conventional API security strategies

J. Glob. Res. Elec. Comm., 2 (7):15-21
DOI: https://doi.org/xxxxxxxxxxxx

that do not effectively address the security challenges posed by microservice constraints include static access control lists and rule-based security, which are ineffective in modern dynamic microservices. In particular, legacy security systems fail to address rapidly evolving threats and lack sufficient capabilities to secure large, distributed environments. In response to these issues, there is a strong demand for intelligent and asymptotic security models for API. To that end, this paper presents the Dynamic Adaptive API Security Framework, which incorporates AI and Blockchain technologies. Incorporating these two state-of-the-art technologies, the proposed framework is envisioned to overcome the existing API protection approaches' drawbacks and provide effective protection schemes for microservices in the future [11].

This paper is divided into five sections: Section II introduces Microservices and Container Foundations. Section III explores AI-Powered Security for Microservices and Container Platforms. Section IV reviews recent literature, and Section V concludes the paper and outlines future research directions.

II. Microservices and Container Foundations

In a microservices-based system, each service is responsible for a specific function and communicates with other services through well-defined APIs [12]. By contrast, containerization enables lightweight virtualization by customizing containers as application correspondences from separate images that use fewer resources and time [13].

A. *Microservices Architecture*

The most significant aspect between various services in microservices architecture is loose coupling [14]. The individual services may have different technology requirements and may share or have their own database and servers, as shown in Fig. 1.

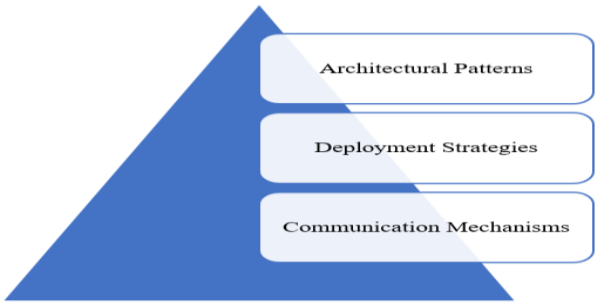

Fig. 1. Microservices Architecture

- **Architectural Patterns:** To adopt the microservices architecture, the choice of an appropriate pattern is an essential aspect [15]. Some of the patterns for implementing microservices are as follows: Backend for frontend: Key-value store, Document store, Log aggregator, Service registry, Database as a service.
- **Deployment Strategies:** Apart from building and developing microservices, the management and maintenance are also key aspects. The requirements of each microservice differ due to variability in its resources and technology stack.
- **Communication Mechanisms:** As microservices consist of a collection of small services, they require a proper and sophisticated channel for inter-service or inter-process communication.

B. *Security-based Microservices Challenges*

Microservices architecture introduces new concepts related to the communication layers and technologies. The microservices development process lacks the adoption of security-by-design [16], such that security in microservices frequently comes as an afterthought.

- **Distributed Communication:** It places an overhead on security tasks, especially for security assessments which are traditionally configured for static network resources, hosts and applications [17].
- **Ephemeral Nature of Resources:** It means that an ephemeral microservice can be destroyed at any time and then resurrected to its last known state immediately.
- **Trust Among Inter-Communicating Microservices:** All inter-communicating service instances are assumed to operate within a common security trust domain. However, this could introduce security issues.

C. *Containerization Technologies*

A method of operating system virtualization in which applications are run in isolated user spaces, called containers, while using the same shared operating system kernel in Fig.2 [18].

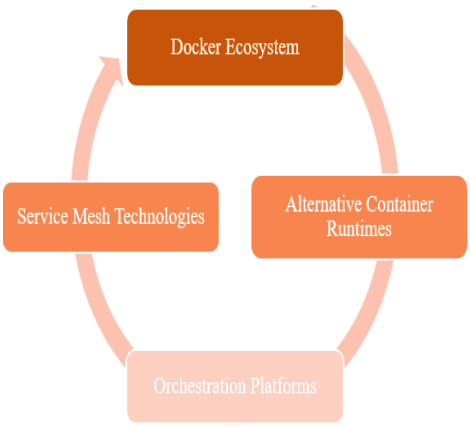

Fig. 2. Containerization Technologies

1) *Docker Ecosystem*

Docker remains the most widely adopted containerization platform, offering:

- **Docker Engine:** The core container runtime.
- **Docker CLI:** Command-line interface for container management [19].
- **Docker Compose:** Tool for defining multi-container applications.
- **Docker Hub:** Public registry for container images.
- **Docker Swarm:** Native clustering and orchestration.

2) *Alternative Container Runtimes*

Docker pioneered the modern containerization movement, several alternative runtimes have emerged:

- **Container:** A lightweight, portable container runtime used by Docker and other platforms.
- **CRI-O:** A Kubernetes-focused container runtime interface implementation.
- **RKT (Rocket):** A security-focused container runtime originally developed by CoreOS.

DOI: https://doi.org/xxxxxxxxxxxx

- **Padman:** A daemonless container engine compatible with Docker's CLI.
- **LXD:** A container hypervisor building on LXC technology.

3) Orchestration Platforms

As containerized deployments scaled beyond individual hosts, orchestration platforms emerged to manage distributed container workloads:

- **Kubernetes:** Kubernetes is a widely adopted container orchestration platform that provides advanced scheduling, service discovery, load balancing, and self-healing capabilities.
- **Docker Swarm:** Docker's native clustering solution, offering simplicity for smaller deployments.
- **Amazon ECS/EKS:** Amazon's container management services.
- **Azure AKS:** Microsoft's managed Kubernetes service.
- **Google GKE:** Google's managed Kubernetes offering.
- **OpenShift:** Red Hat's enterprise Kubernetes platform with additional developer and operational features [20].

4) Service Mesh Technologies

As containerized microservices architectures grew more complex, service mesh technologies emerged to manage service-to-service communication:

- **Istio:** Provides traffic management, security, and observability for microservices.
- **Linkerd:** A lightweight service mesh focused on simplicity and performance.
- **Consul Connect:** HashiCorp's service mesh solution integrated with Consul.
- **AWS App Mesh:** Amazon's managed service mesh implementation.

D. Hypervisor vs Container-Based Virtualization

Hypervisor-based virtualization and containerization are two most commonly used techniques for server abstraction and efficient resource utilization in data centre environments[21].

- **Hypervisor-Based Virtualization:** It is carried out at the hardware level by virtually distributing physical resources (using a software layer, called hypervisor or VMM, placed between OS and hardware) among multiple co-located VMs running parallel [22].
- **Containerization:** Its containerizations virtualize OS level resources encapsulating standard OS processes along with their dependencies to create containers that are collectively managed by underlying host OS kernel and can be seen as sandboxes running applications in isolated ways.

E. Tools for Containers

A wide range of tools is available for containers, spanning from container runtimes to orchestrators. This section describes the most popular offerings in different categories:

- **Docker:** Docker is perhaps the most well-known container platform as it offers an easy-to-understand set of tools to package, deploy, and run applications across different platforms and devices [23]. It is an open-source platform and an establishing member of the OCI, which aims to establish a set of open standards for containers.
- **Kubernetes:** The deployment, management, scaling, networking, and self-healing of containerized applications may all be automated with Kubernetes, a platform for container orchestration. With capabilities like resource provisioning, service discovery, load balancing, automatic scaling, and fault recovery, it facilitates the effective administration of several containers across distributed computing systems [24].
- **Anaconda:** It is a package manager responsible for installing and managing Python packages and libraries. It was created as an alternative to pip, the default package manager used in Python. Pip installed the requested package and all the required dependencies without checking for any conflicts with previously installed packages.

III. AI-Powered Security for Microservices and Container Platforms

The reason why AI-supported models are accepted to secure microservices and containers can be justified with the limitations of standard defences [25]. Unstable load attacks and polymorphic attacks cannot be addressed with policy-based rules. Moreover, institutional or business entities face increased pressure to protect sensitive information in the cloud.

A. Taxonomy of Anomaly Detection in Microservices

To frame the landscape of AI-based microservice reliability, categorize anomaly detection along three dimensions: the observability signals being analyzed, the modelling techniques employed, and the deployment layer where detection is performed [26].

1) Observability Signals

Microservice anomaly detection relies on diverse observability signals that provide complementary information about system performance, operational events, request execution, and abnormal service behaviour.

- **Metrics:** Metrics are quantitative time-series measurements reflecting system performance and resource usage [27]. Metrics are typically collected at regular intervals from infrastructure components and from applications.
- **Logs:** Logs are unstructured or semi-structured text records emitted by applications or system components. They contain fine-grained information about events, failures, and program states, but their free-text nature makes automated analysis challenging.
- **Traces:** Traces capture the end-to-end execution path of individual requests across multiple microservices. A trace typically forms a directed acyclic graph where each node represents a span and edges represent causal or temporal relationships.

2) Modelling Techniques

Various modelling techniques analyze observability data to identify deviations from normal behaviour, ranging from conventional statistical approaches to ML, DL, and generative models.

DOI: https://doi.org/xxxxxxxxxxxx

- **Statistical-Based Methods:** Detect anomalies using statistical patterns, distributions, and deviations from normal behaviour.
- **Machine Learning Methods:** Use algorithms such as Random Forest, SVM, KNN, and clustering to identify abnormal patterns.
- **Deep Learning Methods:** Apply CNN, RNN, LSTM, Autoencoders, and Transformer models for complex anomaly detection.
- **Generative Models:** Use GANs and related generative approaches to learn normal behaviour and identify deviations.

3) Deployment Layer

Anomaly detection can operate across multiple deployment layers, enabling monitoring and analysis at application, container, infrastructure, cloud, and edge environments[28].

- **Application Layer:** Detect anomalies from microservice applications and service-level behaviours.
- **Container Layer:** Monitor containers, pods, and their resource and execution behaviour.
- **Infrastructure Layer:** Analyse servers, networks, and underlying computing resources.
- **Cloud/Edge Layer:** Perform detection closer to data sources at the edge or centrally within cloud environments[29].

B. Deep Container Framework Design

The Deep Container framework implements a layered architecture designed for real-time anomaly detection in cloud-native container environments [30].

- **Real-time Data Collection and Preprocessing:** The data collection subsystem implements distributed telemetry capture mechanisms optimized for container environments [31]. Advanced preprocessing pipelines perform feature extraction and normalization operations in real-time.
- **Deep Learning Model Architecture:** The neural network architecture implements specialized layers designed for container telemetry analysis. Model optimization techniques include dynamic batch processing and automated parameter tuning mechanisms [32].
- **Anomaly Detection Algorithm:** The Deep Container anomaly detection algorithm implements a hybrid approach combining DL inference with statistical analysis. The detection mechanism utilizes multi-dimensional feature analysis to identify behavioural deviations in containerized environments.

C. Benefits of AI in Microservices

The adoption of AI microservices in enterprise applications offers several benefits, including improved scalability, flexibility, and efficiency[33]. These benefits are particularly important in the context of AI, where the dynamic and resource-intensive nature of workloads can pose significant challenges for traditional architectures.

- **Scalability:** One of the primary advantages of AI microservices is their ability to scale independently [34]. In an architecture, scaling an application typically involves scaling the entire system, which can be inefficient and costly.
- **Flexibility:** AI microservices also offer greater flexibility in terms of development and deployment. Since each microservice is developed and deployed independently, enterprises can adopt a more agile approach to software development, enabling them to respond quickly to changing business needs and technological advancements.
- **Efficiency:** The modular nature of AI microservices also contributes to improved efficiency in terms of resource utilization and performance. By decomposing applications into smaller, self-contained units, enterprises can optimize the use of computational resources, such as CPU, memory, and storage.

D. AI-Powered Intrusion Detection in Microservices and Containers

There are many Ai-Powered Intrusion Detection like ML, DL and feature extraction.

- **Machine Learning:** The analysis of the network's activities within the microservice-based architecture is made possible using ML algorithms, which are critical in AI-based intrusion detection systems [35]. It is in contrast with traditional rule-oriented systems which require specific signature of the virus in ML approach, one can analyze previous data and detect unknown threats.
- **Deep Learning:** The advanced techniques, widely known as deep learning or DL, again improve the accuracy level in intrusion detection by feeding vast data on the network traffic and analyzing the same for creating further complex pattern identification. These models apply chaptalized and multi-layered neural networks to achieve feature sharing and features categorization, enabling them to easily identify complex cyber threats [36].
- **Feature Extraction:** Feature extraction is a rather important element in the framework of IDS creation because it converts raw data into a set of features for further analysis by AI models. In the case of the AI-powered detection system, this is ratioed with a qualitative and relative measure of extracted features.

Table I compares major ML and DL algorithms for cybersecurity threat detection, highlighting their advantages, limitations, and applications. Supervised and unsupervised methods address known and novel attacks, while CNNs and RNNs analyze traffic and temporal patterns. Hybrid models provide comprehensive detection but require greater computational resources.

TABLE I. COMPARISON OF MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

Algorithm Type	Advantages	Limitations	Best Use Cases
Supervised Learning	High accuracy with labelled data; interpretable models	Requires large, labelled datasets; limited adaptability to new threats	Detecting known attack patterns and established threats
Unsupervised Learning	Can detect novel and unknown attacks; no need for labelled data	Higher false positive rates; complex tuning required	Identifying anomalous behaviours and zero-day attacks

DOI: https://doi.org/xxxxxxxxxxxxx

Convolutional Neural Networks (CNNs)	Effective for pattern recognition in network traffic	Requires significant computational resources	Analyzing structured network traffic for intrusion detection
Recurrent Neural Networks (RNNs)	Suitable for sequential data analysis; detects temporal attack patterns	Susceptible to vanishing gradient problem; high training time	Monitoring time-series logs and detecting persistent threats
Hybrid AI Models	Combines strengths of multiple algorithms for improved detection	Increased model complexity; requires high processing power	Comprehensive threat detection across different attack vectors

IV. Literature Review

This section highlights related studies on AI-powered security for microservices and container platforms, emphasizing intelligent API gateways, anomaly and threat detection, zero-trust security, container orchestration, secure communication, resource management, and automated security mechanisms to improve the protection, scalability, and reliability of modern distributed applications.

Karan Kumar Ratra et. al. (2026) propose an AI-enhanced API Gateway architecture with intelligent security and performance features using advanced ML models. Key innovations include zero-trust security, anomaly detection, adaptive rate limiting, and predictive load balancing. LSTM, Isolation Forests, Gaussian Mixture Models, and Deep Q-Networks detect malicious traffic, adjust request thresholds, and route traffic using historical and real-time metrics, providing a scalable solution for securing and optimizing complex AI microservices environments [37].

Santosh Kumar Kotakonda` (2026) compares the two available options from AWS, Elastic Container Service (ECS) and Elastic Kubernetes Service (EKS) running on serverless AWS Fargate compute engine. The comparison mainly focused on operational complexity, performance metrics, security architecture and total cost of ownership (TCO), when deployed to a stateful Spring Boot microservices environment. ECS on Fargate has operational simplicity and is very quick in task startup, whereas EKS maintains finer granular controls over resource scheduling and provides advanced security models such as service mesh architecture. EKS infrastructure cost looks cheap but organizations end up paying more for operational labour costs due to EKS complexity[38].

Pedro Melo et. al. (2025) introduce a benchmarking framework designed to systematically evaluate software ageing in container platforms. The approach employs well-defined metrics to characterize aging behaviors under representative workloads and stress conditions. Using this framework, conduct a comparative study of two widely used container platforms, Docker and Padman, across multiple deployment environments [39].

Meenaxi M Raikar et. al. (2024) explore the integration of Docker and microservices architecture for developing scalable web applications. Docker's containerization technology packages applications and dependencies into portable containers, enabling rapid startup, reduced resource consumption, and flexible deployment. The framework includes client applications, Docker containers, microservices, and a database layer, with Angular for the front end, Node.js for the back end, and deployment on Google Cloud Platform (GCP). SSL/TLS, middleware-based authentication and session management, and MongoDB with Mongoose support secure communication and robust database connectivity [40].

Pethuru Ra, Skylab Vanga & Akshita Chaudhary (2023) focus on integrating security layers into microservices applications by discussing security challenges and relevant solutions. Microservices provide optimized software building blocks through distributed architectures involving backend services, middleware, and user interfaces. Authentication validates users through correct credentials, while API gateways improve the performance and scalability of the MSA [41].

Lamees M. Al Qassem et. al. (2022) formulate the efficient resource management of cloud microservice resources as a constrained optimization problem that is amenable to a solution in real time. A novel optimisation-based reactive resource manager is implemented on an open-source platform and compared with the academic state of the art and existing cloud-provider solutions [42].

Table II provides a summary of key studies on AI-powered security, microservices, and container platforms, highlighting their study focus, methodologies, key findings, challenges and limitations, and proposed future research directions across intelligent API gateways, container orchestration, container platforms, microservices security, and AI-driven resource management approaches.

TABLE II. Comparative Analysis of AI-Powered Security Approaches for Microservices and Container Platforms

Reference	**Study Focus**	**Approach / Methodology**	**Key Findings**	**Challenges / Limitations**	**Future Directions**
Karan Kumar Ratra et al. (2026)	AI-enhanced API Gateway for secure and efficient microservices communication	Integrated LSTM, Isolation Forest, Gaussian Mixture Models (GMM), and Deep Q-Networks (DQN) for anomaly detection, adaptive rate limiting, predictive load balancing, and zero-trust security	Improved detection of malicious traffic, dynamic request control, and intelligent traffic routing in AI-based microservices environments	Complexity of integrating multiple AI models; requires continuous monitoring and sufficient real-time data	Develop more adaptive AI gateways with improved real-time learning, scalability, and automated security responses
Santosh Kumar Kotakonda (2026)	Comparison of AWS ECS and EKS on Fargate for stateful Spring Boot microservices	Comparative evaluation based on operational complexity, performance, security architecture, and total cost of ownership	ECS provides simpler operations and faster task startup, while EKS offers greater scheduling control and advanced security capabilities	EKS introduces higher operational complexity and labour costs	Develop simplified Kubernetes management and cost-efficient orchestration solutions for enterprise microservices
Pedro Melo et al. (2025)	Software ageing in container platforms	Proposed a benchmarking framework using ageing-related metrics and evaluated	Provided systematic measurement and comparison of software	Aging behavior can vary with workloads and deployment	Extend benchmarks to more container platforms, workloads,

		Docker and Podman under workloads and stress conditions	aging behavior across container platforms	environments; benchmarking may not cover all real-world scenarios	and long-term production environments
Meenaxi M. Raikar et al. (2024)	Integration of Docker and microservices for scalable web applications	Developed a Docker-based microservices web application using Angular, Node.js, MongoDB, and GCP, with SSL/TLS and authentication mechanisms	Demonstrated portability, rapid deployment, reduced resource consumption, and scalability of Docker-based microservices	Security and deployment configurations may require additional management as systems scale	Improve automated security, orchestration, monitoring, and deployment of Docker-based microservices
Pethuru Raj, Skylab Vanga & Akshita Chaudhary (2023)	Security challenges and solutions in microservices architecture	Examined authentication, API gateways, distributed services, and security practices for microservices applications	Highlighted the importance of authentication and API gateways for improving security, performance, and scalability	Distributed microservices introduce complex authentication, communication, and security-management requirements	Develop stronger integrated security frameworks, automated authentication, and intelligent API gateway solutions
Lamees M. Al Qassem et al. (2022)	Efficient cloud resource management for microservices	Formulated resource management as a constrained optimization problem and implemented a reactive resource manager for comparative evaluation	Demonstrated an optimization-based approach for improving real-time cloud microservice resource management	Reactive approaches may have limited adaptability to rapidly changing workloads	Explore predictive and AI-driven resource management for proactive scaling and dynamic workload optimization

V. Conclusion and Future Work

Artificial Intelligence in strengthening security for microservices and container platforms. Microservices and containerization provide scalability, portability, flexibility, and efficient resource utilization, but their distributed, dynamic, and ephemeral nature introduces complex security challenges. Traditional rule-based security mechanisms are often insufficient for detecting evolving, unknown, and sophisticated attacks. AI-based approaches, including ML, DL, anomaly detection, and intelligent intrusion detection, provide promising capabilities for identifying abnormal behaviours and emerging threats. The reviewed studies also demonstrate the potential of AI-enhanced API gateways, zero-trust security, adaptive rate limiting, predictive traffic management, and intelligent resource allocation to improve security and operational efficiency. However, challenges remain concerning computational requirements, data quality, model complexity, false positives, scalability, and real-time deployment. Overall, AI offers a promising foundation for developing adaptive, proactive, and scalable security mechanisms capable of protecting modern cloud-native microservices and container environments against increasingly sophisticated cyber threats.

Future research should focus on developing lightweight and explainable AI models capable of real-time threat detection across large-scale microservices and container environments. Greater attention is needed toward zero-day attack detection, adaptive API protection, automated security responses, and privacy-preserving learning. Integrating AI with zero-trust architectures, service meshes, and container orchestration can further improve security, resilience, scalability, and autonomous protection.

- **Funding:** This research received no external funding.
- **Institutional Review Board Statement:** Not applicable.
- **Informed Consent Statement:** Not applicable.
- **Data Availability Statement:** The data supporting the findings of this study are available from the corresponding author upon reasonable request.
- **Conflicts of Interest:** The author declares no conflict of interest.

How to cite this article:
M. Sahu, "AI-Powered Security for Microservices and Container Platforms: A Review Analysis," *Journal of Global Research in Electronics and Communication*, vol. 2, no. 7, pp. 15-21, Jul. 2026, Doi: XXXX

References

[1] N. Anwar and S. Kar, "Review Paper on Various Software Testing Techniques & Strategies," *Glob. J. Comput. Sci. Technol.*, pp. 43–49, May 2019, doi: 10.34257/GJCSTCVOL19IS2PG43.

[2] J. Vučković, "You Are Not Netflix," in *Microservices*, Cham: Springer International Publishing, 2020, pp. 333–346. doi: 10.1007/978-3-030-31646-4_13.

[3] O. Al-Debagy and P. Martinek, "A Comparative Review of Microservices and Monolithic Architectures," in *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, IEEE, Nov. 2018, pp. 000149–000154. doi: 10.1109/CINTI.2018.8928192.

[4] P. Di Francesco, I. Malavolta, and P. Lago, "Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption," in *2017 IEEE International Conference on Software Architecture (ICSA)*, IEEE, Apr. 2017, pp. 21–30. doi: 10.1109/ICSA.2017.24.

[5] S. Pinto-Agüero and R. Noel, "Microservices Evolution Factors: A Multivocal Literature Review," *IEEE Access*, vol. 13, pp. 88707–88730, 2025, doi: 10.1109/ACCESS.2025.3570658.

[6] V. R. Gudelli, "Containerization Technologies: ECR and Docker for Microservices Architecture," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 11, no. 3, pp. 1–10, Jun. 2023, doi: 10.37082/IJIRMPS.v11.i3.232165.

[7] S. B. S. Ali, S. Tarakampet, and S. Tatavarthi, "Reusable, Secure, and Compliance-First CI/CD Pipeline Architectures for Regulated Enterprise Environments," in *2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISET)*, 2026, pp. 1–6. doi: 10.1109/ICAISET66439.2026.11541732.

[8] S. Yalamati, "Ai-Enhanced Fault Tolerance In Microservices : Predictive Failure Models For Resilient Software Systems," *Int. J. Appl. Eng. Technol.*, vol. 7, no. 2, pp. 26–35, 2025, doi: 10.13140/RG.2.2.33410.34242.

[9] K. R. Kiran, "API Integration Barriers in Case-Based Process Management: A Review of Interoperability and Real-Time Response Constraints Outline," *Zenodo*, vol. 25, no. 03, 2025, doi: 10.5281/zenodo.16782616.

[10] B. Singh, M. A. Augie, H. Singh, and T. Banerjee, "Strengthening Modern IAM Authentication with Quantum Cryptography and Anti-Phishing Techniques," *Sarcouncil J. Eng. Comput. Sci.*, vol. 04, no. 10, pp. 17–31, October, 2025, doi: 10.5281/zenodo.17260292.

[11] D. Kaul, "Dynamic Adaptive API Security Framework Using AI - Powered Blockchain Consensus for Microservices," vol. 08, no. 04, pp. 402–420, 2020, doi: 10.18535/ijsrm/v08i4.ec03.

[12] J. Willard and J. Hutson, "The Evolution and Future of Microservices Architecture with AI-Driven Enhancements," *Int. J. Recent Eng. Sci.*, vol. 12, no. 1, pp. 16–22, Feb. 2025, doi: 10.14445/23497157/IJRES-V12I1P103.

[13] S. B. Verma, B. Pandey, and B. Kumar Gupta, "Containerization and its Architectures: A Study," *ADCAIJ Adv. Distrib. Comput. Artif. Intell. J.*, vol. 11, no. 4, pp. 395–409, Jun. 2023, doi: 10.14201/adcaij.28351.

[14] N. Torvekar and P. S. Game, "Microservices and Its Applications An Overview," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 4, pp. 803–809, Apr. 2019, doi: 10.26438/ijcse/v7i4.803809.

[15] K. Brown and B. Woolf, *Implementation Patterns for Microservices Architectures*. 2016.

[16] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, and M. Prandini, "Microservice security: a systematic literature review," *PeerJ Comput. Sci.*, vol. 7, p. e779, Jan. 2022, doi: 10.7717/peerj-cs.779.

[17] A. S. Abdelfattah and T. Cerny, "Microservices Security Challenges and Approaches," Sep. 2022. doi: 10.62036/ISD.2022.27.

[18] J. Felter, Wes; Ferreira, Alexandre; Rajamony, Ram; Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," 2015, pp. 171–172. doi: 10.1109/ISPASS.2015.7095802.

[19] D. Manor, D. Rod, H. Fang, and J. Watkins, "Containerization in Cloud Computing: A Review," 2025.

[20] S. K. Malaraju, "Process of OS Security Hardening - Red Hat Enterprise Linux," *Int. J. Lead. Res. Publ.*, vol. 1, no. 2, pp. 1–11, October, Oct. 2020, doi: 10.70528/IJLRP.v1.i2.1474.

[21] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and Virtual Machines at Scale : A Comparative Study," no. 1, pp. 1–13, 2016, doi: 10.1145/2988336.298833.

[22] J. Watada, S. Member, A. Roy, H. Pham, and B. Xu, "Emerging Trends , Techniques and Open Issues of Containerization : A Review," *IEEE Access*, vol. 7, pp. 152443–152472, 2019, doi: 10.1109/ACCESS.2019.2945930.

[23] L. Urblik, E. Kajati, P. Papcun, and I. Zolotová, "Containerization in Edge Intelligence: A Review," *Electronics*, vol. 13, no. 7, p. 1335, Apr. 2024, doi: 10.3390/electronics13071335.

[24] E. Casalicchio, "Container Orchestration: A Survey," in *Systems Modeling: Methodologies and Tools*, A. Puliafito and K. S. Trivedi, Eds., Cham: Springer International Publishing, 2019, pp. 221–235. doi: 10.1007/978-3-319-92378-9_14.

[25] M. Mishra, "Securing Cloud-Native Microservices Using AI-Driven Threat Detection Models," *Int. J. Res. Rev. Appl. Sci. Humanit. Technol.*, pp. 337–341, 2025, doi: 10.71143/ka63xh42.

[26] S. Sreevathsa, "Strategies for Reducing Alert Fatigue and Improving Signal Quality in Enterprise Monitoring Architectures," *Int. J. Adv. Eng. Manag. Sci.*, vol. 12, no. 3, pp. 444–451, 2026, doi: 10.22161/ijaems.123.42.

[27] M. Mohammad, "Survey on AI-Based Reliability and Anomaly Detection in Microservices," *Int. J. Comput. Appl.*, vol. 187, pp. 56–63, 2026, doi: 10.5120/ijca2026926263.

[28] K. K. Mohammed, "The Future is Cloud : Modernizing Big Data for the Cloud Era," *Int. J. Sci. Res. Eng. Trends*, vol. 11, no. 5, pp. 1–5, 2025, doi: 10.5281/zenodo.17339856.

[29] K. Jangiti, "Design and Validation of a Machine Identity Governance Framework for AI Agents in Multi-Cloud Environments," in *SoutheastCon 2026*, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/SoutheastCon63549.2026.11476363.

[30] K. Xiong, Z. Wu, and X. Jia, "DeepContainer : A Deep Learning-based Framework for Real-time Anomaly Detection in Cloud-Native Container Environments," vol. 5, no. January, pp. 1–17, 2025, doi: 10.69987/JACS.2025.50101.

[31] S. Wang, J. Chen, L. Yan, and Z. Shui, "Automated Test Case Generation for Chip Verification Using Deep Reinforcement Learning," *J. Knowl. Learn. Sci. Technol. ISSN 2959-6386*, vol. 4, no. 1, pp. 1–12, Jan. 2025, doi: 10.60087/jklst.v4.n1.001.

[32] M. Li, M. Shu, and T. Lu, "Anomaly Pattern Detection in High-Frequency Trading Using Graph Neural Networks," *J. Ind. Eng. Appl. Sci.*, vol. 2, no. 6, pp. 77–85, Dec. 2024, doi: 10.70393/6a69656173.323430.

[33] S. Tatavarthi, S. Tarakampet, and R. R. Koilakonda, "Leveraging Generative AI for Adaptive Compliance Workflows in Enterprise Platforms," *Int. J. Res. Appl. Sci. \& Eng. Technol.*, vol. 14, no. 1, 2026, doi: 10.22214/ijraset.2026.77213.

[34] J. Mwangi and T. Bablu, "AI Microservices in Enterprise Applications: A Comprehensive Review of Use Cases and Implementation Frameworks," *Q. J. Emerg. Technol. Innov.*, vol. 07, 2025.

[35] O. N. I. S. Boluwatife, "AI-Powered Intrusion Detection for Microservice-Based Architectures," *IRE Journals*, vol. 8, no. 6, pp. 1064–1071, 2024.

[36] T. Chen and C. Guestrin, "XGBoost," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[37] K. K. Ratra, D. Kumar Seth, D. Verma, and H. Burman, "AI-Enhanced API Gateway: A Unified Framework for Security and Performance for Next Generation Microservices," in *2025 IEEE 16th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Berkeley, CA, USA: IEEE, 2025, pp. 577–584. doi: 10.1109/IEMCON67450.2025.11381185.

[38] S. K. Kotakonda, "Benchmarking Container Orchestration Platforms: A Comparative Analysis of Kubernetes and AWS ECS for Stateful Microservices," in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, IEEE, Feb. 2026, pp. 1–5. doi: 10.1109/ICAIC67076.2026.11395772.

[39] P. Melo, J. Ferreira, J. Araujo, and M. Vieira, "Benchmarking Software Aging Effects in Container Platforms," *IEEE Trans. Reliab.*, vol. 74, no. 4, pp. 5015–5029, Dec. 2025, doi: 10.1109/TR.2025.3612809.

[40] M. M. Raikar *et al.*, "Leveraging Docker Containers for Deployment of Web Applications in Microservices Architecture," in *2024 First International Conference for Women in Computing (InCoWoCo)*, IEEE, Nov. 2024, pp. 1–6. doi: 10.1109/InCoWoCo64194.2024.10863683.

[41] P. Raj, S. Vanga, and A. Chaudhary, "Microservices Security," in *Cloud-Native Computing*, Wiley, 2022, pp. 289–298. doi: 10.1002/9781119814795.ch14.

[42] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. A. M. Elfadel, "Optimal Resource Allocation for Containerized Cloud Microservices," in *2022 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, IEEE, Nov. 2022, pp. 271–274. doi: 10.1109/ICECTA57148.2022.9990377.