A Multi-Layered Architecture for Microservices with Self-Protection Properties

Vinícius Matheus Romualdo Santos
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo (ICMC-USP)
São Carlos, SP, Brazil
vinicius.romualdo@usp.br

Lina María Garcés Rodríguez
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo (ICMC-USP)
São Carlos, SP, Brazil
linagarces@usp.br

Abstract

The microservices architecture has become well established in modern distributed systems, but its decentralization widens the attack surface, which extends from the external perimeter to inter-service communication and to the application *runtime*. Isolated protection mechanisms tend to offer only partial coverage in this scenario. This short paper presents MLCNP (*Multi-Layered Cloud Native Protection*), an architecture that organizes the protection of microservice-based *cloud-native* applications into three complementary boundaries, aligned with the *defense-in-depth* principle: a *gateway* with a WAF at the perimeter, a *sidecar proxy* on east-west communication, and a RASP mechanism at the application level, the last being the boundary that brings the proposal closest to the *self-protection* property. The paper describes the research method that led from a systematic mapping study to the architecture, details the architectural rationale of the boundaries, and discusses the potential of the RASP boundary to evolve toward Artificial Intelligence-oriented self-protection. The experimental evaluation of the proposal is outlined as future work.

Keywords

Microservices Architecture, Software Security, Self-protection, *Defense-in-Depth*, *Cloud-Native*, RASP

1 Introduction

Microservices have become one of the dominant architectural styles for modern software systems due to their scalability, modularity, and independent deployment capabilities [6, 15]. However, decomposing applications into distributed services significantly expands the attack surface, exposing not only the external perimeter but also inter-service communication and the runtime of each service [2, 3]. Although mechanisms such as authentication, authorization, and encryption remain fundamental, they primarily protect the system perimeter and provide limited protection against attacks that exploit compromised services or lateral movement inside the infrastructure. This challenge is becoming even more critical with AI-assisted software development, where automatically generated code can accelerate delivery while introducing security vulnerabilities that bypass traditional pre-deployment verification [17].

Security in microservices should therefore be addressed as an architectural concern rather than as a collection of isolated protection mechanisms. Security engineering recommends combining complementary tactics following the *defense-in-depth* principle, in which multiple independent layers cooperate to detect, resist, and respond to attacks, avoiding single points of failure [2, 14, 23]. Likewise, the *self-protecting* property from Autonomic Computing advocates architectures capable of continuously monitoring and reacting to security threats through coordinated protection mechanisms [22, 28].

Despite these principles, current approaches for securing microservice systems mostly focus on individual technologies, such as Web Application Firewalls (WAFs), service meshes, or Runtime Application Self-Protection (RASP), without explicitly defining how these mechanisms should be architecturally organized and coordinated [1, 7, 8, 18, 26].

Recent studies have emphasized the need for stronger security architectures for cloud-native microservices, such as the adoption of self-adaptive security and machine learning techniques, but these efforts remain focused on specific mechanisms rather than comprehensive architectural solutions [7, 11]. Various proposals examine individual protection technologies, such as service meshes for self-protection [1], the complementary use of WAF and RASP [26], and multi-layered cloud-native security models, such as the 4C architecture [25, 27]. However, these approaches either address security at a conceptual level or focus on isolated protection mechanisms, without explicitly defining how perimeter protection, secure service-to-service communication, and runtime application protection should be architecturally integrated in microservice-based systems.

To address this gap, this paper proposes **MLCNP** (*Multi-Layered Cloud-Native Protection*), a high-level architecture that organizes the protection of cloud-native microservice applications into three complementary layers: perimeter protection, secure service-to-service communication, and runtime application protection. Rather than introducing new security mechanisms, MLCNP provides an architectural rationale for combining established solutions in accordance with the principles of defense-in-depth, while creating the structural basis for self-protecting systems.

The remainder of this paper is organized as follows. Section 2 presents the research method. Section 3 introduces the proposed MLCNP high-level architecture. Section 4 discusses the proposal and its implications, while Section 5 concludes the paper, outlines its limitations, and presents future research directions.

2 Method

The MLCNP architecture was derived from a systematic mapping study of the state of the art in self-adaptive, security-oriented architectures for microservices, complemented by a directed investigation into runtime application self-protection (RASP), as detailed as follows.

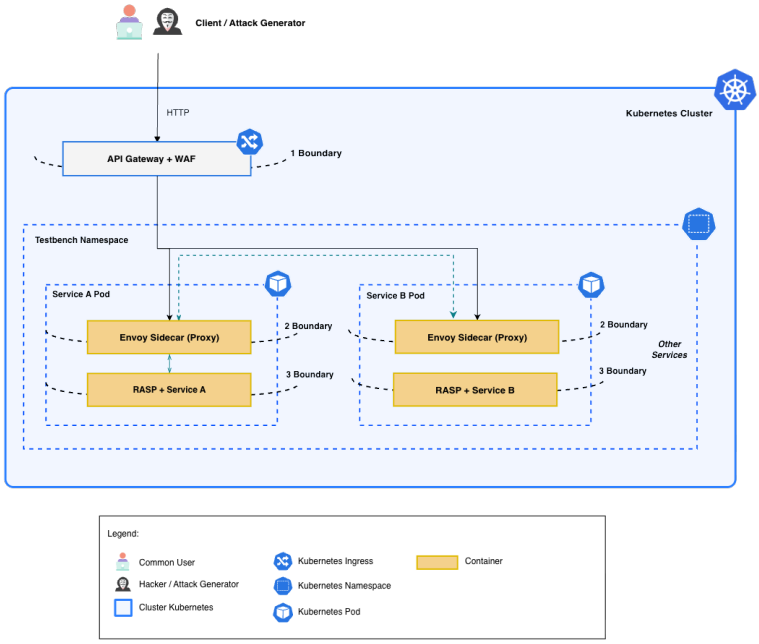

Figure 1: Configuration of the MLCNP architecture with multiple protection layers for microservices in *cloud-native* environments.

Identification of primary studies: A systematic mapping study investigated architectural approaches to self-protection in microservice systems. The search across Scopus, IEEE Xplore, and Google Scholar retrieved 197 studies, from which five primary studies [1, 5, 10, 19, 30] describing self-protection architectures were ultimately selected after a rigorous screening process. The analysis of selected architectures revealed that existing solutions predominantly focus on isolated observation and enforcement points, with reactive responses being the prevailing strategy and limited support for intelligent decision-making or empirical validation of operational impact. Furthermore, the literature lacks architectural proposals that integrate complementary protection mechanisms across multiple layers of microservice systems, highlighting the need for a coordinated defense-in-depth architecture, which motivated the proposed MLCNP architecture configuration.

Extraction of the surveyed architectures: MLCNP adopts the level/boundary abstraction that recurs across the surveyed architectures, but it adopts none of them wholesale, since each concentrates protection at a single point of the architecture. From S1 [30], the notion of automatic response at runtime is preserved, while its centralized game-theoretic engine, which selects countermeasures under a minimax criterion, is not, to avoid introducing a single point of coordination and failure. From S2 [5], the detection of illegal communications based on an explicit architectural model reinforces the value of the intermediate boundary, and S3 [19] confirms this relevance through its decentralized extension of DARE; their node-distributed control loops, however, fall outside the scope of a boundary-oriented proposal. Study S4 [1] is the closest to the proposal: its non-intrusive treatment of the sidecar/service mesh as an interception and verification point over inter-service traffic is adopted directly as the east-west communication boundary. Finally, S5 (Immunizer) [10], which structures protection as a MAPE-K framework with adaptive microagents at the application layer, reinforces the RASP execution boundary and the value of an explicit adaptation cycle. MLCNP adapts the application-layer, MAPE-K-grounded notion of self-protection from [10], but does not adopt its fully microagent-decomposed adaptation cycle, keeping the boundaries as coarser architectural units.

An additional study on self-healing [24] illustrates the use of reinforcement learning at runtime, which motivates the AI-oriented evolution of the RASP boundary to be adopted in MLCNP. A complementary, directed investigation into RASP, motivated by studies that were recovered during the search but excluded from the final corpus for not meeting the architectural criterion, showed that, building on the conceptual systematization of RASP [4], recent runtime protection work has explored the integration of RASP with predictive models [12], the analysis of RASP-generated artifacts [29], and low-overhead container-level protection [9]; however, this work remains concentrated at the application layer, without addressing the broader architectural problem of organizing heterogeneous protection mechanisms across boundaries. Even multilayer arrangements that combine WAF, *Zero Trust*, and RASP [13] remain oriented toward tool-driven validations, without formalizing an architecture specialized for *cloud-native* microservices.

This synthesis of previous architectures confirmed the gap and motivated the formalization of MLCNP as a multi-layered organization of *gateway*/WAF, *sidecar proxy*, and RASP, as presented in the next section.

3 Proposed Architecture

MLCNP is a multi-layered architecture for protecting microservice-based *cloud-native* applications. Aligned with the *defense-in-depth* principle, the architecture explicitly organizes protection mechanisms at different structural levels of the application. In microservices running on Kubernetes, traffic and business logic traverse multiple architectural layers, from the external entry point to the internal execution of the service, and the proposal formalizes these layers as successive protection boundaries. Figure 1 presents the structural view of the architecture, whose components are explained as follows.

Perimeter boundary: *Gateway* and WAF The first boundary sits at the entry perimeter, through the combination of an *API Gateway* and a *Web Application Firewall* (WAF). Its architectural role is to filter, monitor, and block HTTP/HTTPS requests before they reach the internal services, reducing the initial exposure to known attacks based on patterns observable in the traffic, such as *SQL Injection* and *Cross-Site Scripting* [20]. Its visibility, however, is limited to inbound traffic, covering neither the internal communication among services nor attacks resulting from the prior compromise of a component.

Communication boundary: *Sidecar Proxy* The second boundary is positioned on the east-west communication among services, by means of *sidecar proxies* attached to the *pods* and coordinated by a *service mesh* architecture. The proxy transparently intercepts the interactions among microservices and makes it possible to apply mutual authentication, authorization policies, traffic control, and detailed telemetry [1]. Since compromised services can exploit legitimate internal relationships for lateral movement, this layer plays a role in containment and continuous verification, acting as an architectural security connector. Even so, it operates predominantly at the protocol and metadata level, without accessing the internal semantics of the service execution.

Execution boundary: RASP at the application The third boundary sits at the innermost level of the architecture, the application *runtime*, through the integration of a RASP mechanism directly into the service. It is the boundary that most directly brings the proposal closer to the *self-protection* property, since it extends the capacity for intervention at runtime beyond traffic inspection, observing behaviors that emerge inside the application, such as improper invocations and insecure flows [4, 22]. Its limitation lies in the local nature of the observation: in isolation, it offers good visibility over the protected instance, but does not replace a distributed view of the architecture.

Integration and complementarity The central contribution of MLCNP does not lie in the isolated adoption of WAF, *sidecars*, and RASP, mechanisms already discussed in the literature, but in the formalization of their organization as complementary boundaries that differ in visibility and intervention capability: external perimeter (*gateway*/WAF), internal communication (*sidecar proxy*), and application execution (RASP). This organization reduces single points of failure and the likelihood of privilege escalation and undetected lateral movement, and it is compatible with *Zero Trust* models [21], by assuming that internal communications must also be continuously verified.

4 Discussion

Table 1 maps representative OWASP Top 10 attack classes [16] to the architectural boundaries of MLCNP capable of mitigating them. The analysis highlights the complementary roles of the three protection mechanisms: the WAF primarily inspects incoming requests, the *sidecar* monitors external and service-to-service communication, and the RASP observes the application runtime. Consequently, no individual boundary provides comprehensive protection, reinforcing the need for a defense-in-depth architecture.

Rather than considering the entire OWASP Top 10, this study selected, as an example, three representative attack classes to exercise all architectural boundaries while maintaining a manageable experimental scope. *SQL Injection* evaluates the combined capabilities of the WAF and RASP, as attacks can be detected both through malicious request patterns and anomalous runtime behavior. SSRF exercises the *sidecar* and RASP by involving unauthorized outbound communication and application-level execution. *Vulnerable and Outdated Components* relies exclusively on RASP because exploitation can only be observed during application execution. Together, these attack classes provide complete coverage of the proposed protection boundaries while avoiding unnecessary expansion of the experimental design.

This mapping provides the rationale for the MLCNP architecture, whose effectiveness depends on whether combining protection boundaries increases attack coverage without, hopefully, introducing excessive redundancy or performance degradation.

5 Conclusion And Future Work

This paper presented MLCNP, a multi-layered protection architecture for *cloud-native* microservices that organizes three complementary security boundaries, Gateway/WAF, *sidecar proxy*, and RASP, according to the *defense-in-depth* principle. By combining protection at the perimeter, communication, and runtime levels, MLCNP provides an architectural foundation for coordinated security rather than relying on isolated mechanisms. Among these boundaries, RASP emerges as the most promising point for advancing toward intelligent self-protection, since its privileged access to the application's execution context enables runtime decision-making beyond predefined security rules.

This architectural perspective is particularly relevant in the era of AI-assisted software development, where the provenance and security of generated code cannot always be guaranteed. Instead of relying solely on static assurance, MLCNP emphasizes behavior-based protection across multiple architectural layers, making runtime monitoring and response central to the overall security strategy. In this context, AI represents both a source of new security challenges and a promising enabler of adaptive runtime protection.

Future work will focus on empirically evaluating MLCNP through a controlled 2^3 factorial experiment, measuring both security effectiveness and performance trade-offs of the three protection boundaries. The study will quantify mitigation capability, false positive and false negative rates, latency, *throughput*, and runtime overhead in a Kubernetes-based environment. A subsequent research direction is the integration of AI-driven decision-making into the RASP layer, investigating its potential to evolve from rule-based protection toward truly self-protecting microservice architectures.

Table 1: Mapping among attack class, technical evidence of success, and architectural boundaries potentially in action.

Attack Class	Technical Evidence of Success	Boundaries Potentially in Action
SQL Injection	Improper data access, authentication *bypass*, or unauthorized alteration of records	WAF, RASP
SSRF	Record of an attempted or completed improper connection to an internal/external resource	*Sidecar*, RASP
Vulnerable and Outdated Components	Improper execution of vulnerable functionality or observable insecure behavior	RASP

Artifact Availability

The reproducible blueprint of MLCNP, which instantiates the three protection boundaries of Section 3 in a Kubernetes environment, is publicly available at https://github.com/vinimrs/mlcnp-blueprint.

Declaration On AI Use

AI tools were only used for language refinement and grammar correction. They were not involved in the study design, data collection, analysis, or interpretation. All scientific content and conclusions are solely the responsibility of the authors.

Acknowledgements

The authors gratefully acknowledge the PRPI-USP (Grant: 22.1.09345.01.2) and Brazilian agencies for financial support: CAPES PROEX and CNPq (Grant: 406941/2025-4, 420025/2023-5).

References

[1] Rami Alboqmi, Sharmin Jahan, and Rose F. Gamble. 2022. Toward Enabling Self-Protection in the Service Mesh of the Microservice Architecture. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE, 133–138. doi:10.1109/ACSOSC56246.2022.00047

[2] Len Bass, Paul Clements, and Rick Kazman. 2012. *Software Architecture in Practice* (3 ed.). Addison-Wesley, Boston, MA.

[3] Ramaswamy Chandramouli. 2019. *Security Strategies for Microservices-based Application Systems*. Technical Report NIST SP 800-204. National Institute of Standards and Technology, Gaithersburg, MD. doi:10.6028/NIST.SP.800-204

[4] Petar Čisar and Sanja Maravić Čisar. 2016. The framework of runtime application self-protection technology. In *2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI)*. 81–86. doi:10.1109/CINTI.2016.7846383

[5] Benoit Claudel, Noël De Palma, Renaud Lachaize, and Daniel Hagimont. 2006. Self-protection for Distributed Component-Based Applications. In *Stabilization, Safety, and Security of Distributed Systems (SSS) (Lecture Notes in Computer Science, Vol. 4280)*. Springer, 184–198.

[6] Martin Fowler and James Lewis. 2014. Microservices: a definition of this new architectural term. Blog post. Disponível em: https://martinfowler.com/articles/microservices.html. Acesso em: 24 set. 2025.

[7] Omid Gheibi, Danny Weyns, and Federico Quin. 2021. Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems* 15, 3 (2021), Article 9.

[8] Abdelhakim Hannousse and Salima Yahiouche. 2021. Securing Microservices and Microservice Architectures: A Systematic Mapping Study. *Computer Science Review* 41 (2021), 100415. doi:10.1016/j.cosrev.2021.100415

[9] Chenggang He and Chris H. Q. Ding. 2025. SecRASP: Next generation web application security protection methodology and framework. *Computers & Security* 154 (2025), 104445. doi:10.1016/j.cose.2025.104445

[10] Omar Iraqi and Hanan El Bakkali. 2020. Immunizer: A Scalable Loosely-Coupled Self-Protecting Software Framework Using Adaptive Microagents and Parallelized Microservices. In *29th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 24–29.

[11] E. Jawabreh and A. Taweel. 2026. Self-adaptation in microservice systems: A literature review. In *Service-Oriented Computing — ICSOC 2024 Workshops, Revised Selected Papers, Part I (Lecture Notes in Computer Science, Vol. 15833)*. Springer Nature Singapore, 28–40.

[12] Phuc Le-Thanh, Tuan Le-Anh, and Quan Le-Trung. 2023. Research and Development of a Smart Solution for Runtime Web Application Self-Protection. In *Proceedings of the 12th International Symposium on Information and Communication Technology*. Ho Chi Minh City, Vietnam. doi:10.1145/3628797.3628901

[13] Andrzej Mycek and Maryna Łukaczyk. 2025. Multi-Layered Security of Web Applications in Cloud Environments Using WAF, Zero Trust, AI-Driven Threat Detection, and RASP. In *Proceedings of the 39th ECMS International Conference on Modelling and Simulation*, Marco Scarpa, Salvatore Cavalieri, Salvatore Serrano, and Fabrizio De Vita (Eds.). ECMS, Europe. Communications of the ECMS, Volume 39, Issue 1.

[14] National Institute of Standards and Technology. 2004. *Engineering Principles for Information Technology Security (A Baseline for Achieving Security)*. Special Publication 800-27 Rev. A. NIST.

[15] Sam Newman. 2015. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Sebastopol.

[16] OWASP. 2021. OWASP Top 10:2021. https://owasp.org/Top10/2021/. Acesso em: 22 fev. 2026.

[17] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 754–768. doi:10.1109/SP46214.2022.9833571

[18] Aarón Pereira-Vale, Gastón Márquez, Hernán Astudillo, and Eduardo B. Fernandez. 2021. Security Mechanisms used in Microservices-based Systems: A Systematic Mapping. In *Proceedings of the 47th Latin American Computing Conference (CLEI 2021)*. IEEE, 1–10. doi:10.1109/CLEI53233.2021.9640122

[19] J. Porter and Emad Albassam. 2020. A Decentralized Approach to Architecture-Based Self-Protecting Software Systems. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 169–175.

[20] Abdul Razzaq, Ali Hur, Sidra Shahbaz, Muddassar Masood, and H. Farooq Ahmad. 2013. Critical analysis on web application firewall solutions. In *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*. IEEE, Mexico City, Mexico.

[21] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. 2020. *Zero Trust Architecture*. Technical Report NIST SP 800-207. National Institute of Standards and Technology, Gaithersburg, MD. doi:10.6028/NIST.SP.800-207

[22] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems* 4, 2 (2009), 14.

[23] Jerome H. Saltzer and Michael D. Schroeder. 1975. The protection of information in computer systems. *Proc. IEEE* 63, 9 (1975), 1278–1308. doi:10.1109/PROC.1975.9939

[24] A. Samir, Håvard Dagenborg, and Dag Johansen. 2024. QMConn: A Self-Healing Controller for Microservices Using Q-Learning and Markov Decision Processes. In *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 389–398.

[25] Bhargav Soni and Mital Padhya. 2026. Enhancing Cloud-Native Security Through 4C Architecture: A Comprehensive Analysis. In *Information Security, Privacy and Digital Forensics*. Lecture Notes in Electrical Engineering, Vol. 1456. Springer Nature Singapore, 525–546.

[26] Tomás Sureda Riera, Juan Ramón Bermejo Higuera, Javier Bermejo Higuera, Juan Antonio Sicilia Montalvo, and José Javier Martínez Herráiz. 2022. Systematic Approach for Web Protection Runtime Tools' Effectiveness Analysis. *CMES - Computer Modeling in Engineering and Sciences* 133, 3 (2022), 579–599. doi:10.32604/cmes.2022.020976

[27] Theodoros Theodoropoulos, Luis Rosa, Chafika Benzaid, Peter Gray, Eduard Marin, Antonios Makris, Luis Cordeiro, Ferran Diego, Pavel Sorokin, Marco Di Girolamo, Paolo Barone, Tarik Taleb, and Konstantinos Tserpes. 2023. Security in Cloud-Native Services: A Survey. *Journal of Cybersecurity and Privacy* 3, 4 (2023), 758–793. doi:10.3390/jcp3040034

[28] Danny Weyns, Sam Malek, and Jesper Andersson. 2012. FORMS: Unifying reference model for formal specification of distributed self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems* 7, 1 (2012), 8.

[29] Bolun Wu, Futai Zou, Mingyi Huang, Xin Sun, and Jiajia Han. 2026. Enhancing Runtime Application Self-Protection with Unsupervised Deep Learning. In *Security and Privacy in Communication Networks*, Saed Alrabaee, Kim-Kwang Raymond Choo, Ernesto Damiani, and Robert H. Deng (Eds.). Springer Nature Switzerland, Cham, 203–213.

[30] Tetiana Yarygina and Christian Otterstad. 2018. A Game of Microservices: Automated Intrusion Response. In *Distributed Applications and Interoperable Systems (DAIS) (Lecture Notes in Computer Science, Vol. 10853)*. Springer International Publishing, 169–177.