IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 April 2024, accepted 25 May 2024, date of publication 28 May 2024, date of current version 8 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3406500

SURVEY

A Systematic Literature Review of Inter-Service Security Threats and Mitigation Strategies in Microservice Architectures

PHILIPP HAINDL, PATRICK KOCHBERGER, AND MARKUS SVEGGEN
Department of Computer Science and Security, St. Pölten University of Applied Sciences, 3100 St. Pölten, Austria
Corresponding author: Philipp Haindl (philipp.haindl@fhstp.ac.at)

ABSTRACT Ensuring security is of paramount importance in microservice architectures, given their distributed nature, involving numerous services and network-spanning interactions. This architectural style, which can comprise hundreds to thousands of services, inherently presents a more extensive attack surface compared to traditional monolithic applications. Moreover, the polyglot nature of microservices, which encompasses services developed and deployed using diverse programming languages and technologies, further complicates the security landscape. This paper presents a systematic literature review, analyzing 54 publications specifically in the context of security threats and mitigation strategies within the area of inter-service security in microservice architectures. We observed that the majority of studies focus on presenting methods, models, and guidelines for microservice security, with a significant portion validating these approaches. Publications in the field have increased since 2015, with conference papers being the most common type. Security threats identified are mainly related to security perimeters, attack surfaces, and inadequate monitoring and intrusion detection. There is a notable lack of comprehensive analysis on specific security threats, particularly in inter-service authentication and communication. Mitigation strategies receive more attention than security threats, with extensive discussion on infrastructure defense and secure coding practices. The identified research gap highlights the need for establishing a connection between security threats and their mitigation strategies in microservice architectures. It also underscores the necessity for a standardized taxonomy in microservice security to clarify terminology and consolidate best practices, addressing inconsistencies in the literature and guiding future empirical studies on the practical challenges of implementing security measures.

INDEX TERMS Microservices, microservice architecture, inter-service, security threats, mitigation strategies.

I. INTRODUCTION

Microservice architecture (MSA) is widely considered to be a distributed system design that highlights the deployment of small, self-governing services. These services, according to Newman [1], should be developed around business capabilities, with a strong emphasis on automation, decentralization, failure isolation, and observability, ensuring that they can be independently deployed. Richardson [2] characterizes microservices as self-contained components that prioritize specific business domain functionality. He posits that microservices should possess their own databases to facilitate independent deployment, scaling, and development procedures. This autonomy is crucial for ensuring consistent and efficient software delivery as well as promoting scalability. Lewis and Fowler [3] further elaborate on this by defining microservices as a method of creating a single application as a collection of small services, each operating in its own process and often communicating through HTTP resource APIs, advocating for minimal centralized management and the ability to use diverse programming languages and data storage technologies. According to Wolff [4], a novel

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

characteristic of microservices is their use of modules that function as separate processes. This methodology is rooted in the principles of UNIX, which may be distilled into three key elements: (a) a program ought to perform a single function with exceptional proficiency, b) programs ought to have the ability to collaborate effectively, and c) a universal interface ought to be adopted.

Other scholars engage in comparing MSA with Service-Oriented Architecture (SOA), wherein microservices are viewed by some as a distinct architectural style [3] or a refined version of SOA [5], [6], focusing on aspects like service centralization, size, and independence. The transition to microservices introduces changes in system interactions to network calls, bringing forward potential latency and security issues [3], while also enhancing scalability and the utilization of cloud computing benefits [5], [7]. Microservice architecture's compatibility with DevOps practices and containerization emphasizes its suitability for modern application development and adaptability to cloud-based environments [8]. This amalgamation of views underscores the multifaceted nature of MSA, reflecting its roots in business-driven design, technological diversity, and operational independence, while also acknowledging the transition from traditional architectural models and its implications on security and deployment practices [1], [3], [4], [5], [6], [7].

Given the distributed nature of MSA, robust security measures are essential. Also, MSA have the potential to increase a software system's vulnerability by expanding the attack surface of a software system. A further increase in attack surface is caused by the use of virtualization technology, automated pipelines, and various third-party software in the cloud for microservice applications.

In this paper we define *inter-service security* as the ensemble of strategies dedicated to securing the interactions and connections between microservices, excluding security mechanisms operating at the edge. Likewise, we define *inter-service security threats* as threats that specifically target microservices and their interconnections. In the same vein, we recognize *mitigation strategies for inter-service security threats* as approaches to alleviate dangers resulting from such threats.

A. BACKGROUND

1) INTER-SERVICE COMMUNICATION

Inter-service communication, which refers to the exchange of data between services in MSA, is a distinguishing feature that sets the microservice architectural style apart from traditional monolithic architectures. Additionally, this aspect can result in the transfer of large amounts of data within a single network or across multiple network boundaries. One of the challenges of implementing microservices is effectively managing the flow of data between services. Excessive and unnecessary network calls between microservices can lead to increased costs and slower system performance.

2) ARCHITECTURAL PATTERNS FOR MICROSERVICE SECURITY

Based on Bass et al. [9], architectural patterns are solutions to recurrent design challenges within specific contexts that detail the functions, responsibilities, and interactions of software components. The decision to use a particular architectural pattern is critical and deeply influences multiple quality attributes of a system, including security, by dictating the system's structural organization and interaction models. Selecting an architectural pattern involves balancing trade-offs among quality attributes, with security being a pivotal quality attribute in the design process. In the realm of microservice security, specific patterns have evolved.

a: API GATEWAY PATTERN

An *API Gateway* streamlines API requests between clients and backend services in an MSA. It conceals the internal structure of a microservice-based application from the client, thus presenting the client a simplified view of it. By centralizing common tasks across requests and services, such as authentication and authorization, logging, and SSL termination, it eliminates unnecessary redundancy of these tasks among microservices. Additionally, it distributes incoming requests across multiple instances of microservices, enhancing the performance and dependability of the application. Furthermore, it can combine the outcomes from multiple microservices into single responses, reducing the number of round-trip requests required and simplifying client-side logic [1], [2].

b: SERVICE MESH PATTERN

A service mesh is a layer of infrastructure that handles inter-service communication in MSA, traffic management, and observability-related tasks such as tracing requests. It also provides security enhancement capabilities like access control, communication encryption, and rate limitation. Most importantly, it facilitates the management of traffic flow between microservice by implementing a lightweight proxy, commonly known as a *Sidecar* alongside each instance of a microservice [2], [10].

c: SIDECAR PATTERN

The *Sidecar* pattern is used to enhance and extend the functionality of a single microservice without altering its core functionality. This pattern involves deploying an additional component, the *Sidecar*, alongside the main service, both running on the same host and sharing the same lifecycle and resources but operating in separate processes. The *Sidecar* handles cross-cutting concerns such as monitoring, logging, configuration, network traffic control, and encryption using, e.g., *SSL (Secure Sockets Layer)* or *TLS (Transport Layer Security)*, thus offloading these aspects from the application logic. It intercepts microservices' communication to provide a range of features [2], [10], e.g.:

- *Service Discovery:* Automatically detecting microservices within the infrastructure.
- *Load Balancing:* Efficiently distributing incoming requests across multiple instances of a microservice.
- *Encryption and Authentication:* Securing communication between microservices.
- *Monitoring and Tracing:* Providing insights into performance and the flow of requests through the architecture.
- *Fault Injection and Recovery:* Testing the resilience of the MSA by introducing controlled failures.

d: TOKEN PATTERN

This pattern revolves around the generation, distribution, and validation of tokens for authentication and authorization purposes, not limited to MSA. Upon user login, an authentication microservice verifies the user's credentials and issues a token, such as a *JSON Web Token (JWT)*, encapsulating the user's identity and their roles or permissions. This mechanism extends also to inter-service communication in MSA, where microservices authenticate and authorize requests from other microservices by validating the token and checking the encoded permissions. The self-contained nature of these tokens ensures statelessness, promoting scalability of the software system by removing the need for shared session data. Furthermore, this pattern enhances security and decoupling by allowing services to authenticate requests without directly managing user credentials, thus enabling flexible security mechanisms and facilitating easier updates to authentication processes without impacting service functionality [1], [11].

3) PERSPECTIVES ON MICROSERVICE SECURITY

The distributed nature of microservices can potentially increase the system's vulnerability [12] to security threats by expanding the general attack surface. Furthermore, microservices applications are frequently deployed in the cloud using virtualization technology, automated pipelines, and various third-party software, which further expands the attack surface. In discussing security within the context of microservices, it is essential to address several important aspects, such as the authentication and authorization [13] of both users and between services and applications.

Siriwardena and Dias [12] categorize microservice security into *edge-level security*, *inter-service communication*, *secure deployment*, and *secure development*. The objective of *edge security* is to protect against increasing threats at the edge of networked environments. This requires a thorough examination of sophisticated security measures, such as encryption protocols, identity and access management, and anomaly detection techniques, to secure microservices from vulnerabilities that are inherent in edge computing scenarios [14], [15], [16], [17]. Security in the context of *inter-service communication* in MSA [12] deals with security in communication, interaction, and operation between microservices. It covers authentication, e.g., using *Mutual Transport Layer Security (mTLS)* and authorization, secure communication between services, e.g., using encrypted channels, and other security-related topics like service discovery, communication within a *Service Mesh*, the *Sidecar Pattern* [18], network segmentation and firewalls, and intrusion detection. The use of orchestration and container technologies, such as *Kubernetes* [19] and *Docker* [20], can increase the complexity of managing microservices and extend the range of potential attacks to include third-party vulnerabilities and other related issues [13], [21], [22]. The implementation of dedicated patterns, such as the *Token Pattern* [11], the *API Gateway Pattern*, and the use of a *Service Mesh*, are particularly effective in enforcing security measures within microservices [1], [2]. Adherence to secure development principles, including automated security testing, continuous integration and deployment, and adherence to best practices, ensures that MSA are resilient to attacks, thereby improving the overall security posture of such a distributed system.

B. EXISTING LITERATURE REVIEWS

A number of literature reviews have been conducted in the field of microservice security. Yu et al. [23] presented a survey resulting from a structured literature search on security challenges in microservice-based fog applications. The work exclusively focuses on vulnerabilities present in service communication across containers and networks and related to data security and permission issues in MSA. The authors did not follow the guidelines of systematic literature reviews in the sense of the guidelines of Kitchenham and Charters [29] and restricted their search to *Scopus* and *Web of Science* exclusively. In our study, we examined a broader range of publications by directly querying the relevant scholarly libraries. This allowed us to include sources that may not have been considered in their review. Soldani et al. [24] carried out a grey literature review examining the challenges and advantages of employing microservices. The authors provided a unique perspective by drawing on industrial experiences with MSA, captured from magazines, blog posts, whitepapers, and videos. However, the sources utilized in their work inevitably have not undergone the same level of academic rigor as the sources employed in our study. Further, the paper has not addressed the classification of the challenges and advantages based on common security issues. Similarly, Pereira-Vale et al. [25] conducted a systematic literature review on security in microservice-based systems, deriving 15 security mechanisms from a selection of academic and grey literature. The authors have classified the security solutions presented in the articles based on the standard security mechanisms, scope, and relevance to specific security contexts. They noted that authentication and authorization within MSA are the most commonly discussed security mechanisms in the selected papers. While our paper and theirs share some common themes, their emphasis is on security mechanisms rather than security threats. Additionally, their classification of

TABLE 1. Existing literature reviews and comparison to this SLR.

Author(s)	Focus	# Studies	Sources		Search Date
			white	gray	
Yu *et al.* [23]	Security in microservice-based fog applications	66	●	○	2017
Soldani *et al.* [24]	"Pains and gains" of microservices	51	○	●	2017
Pereira-Vale *et al.* [25]	Security in microservice-based systems	70	●	●	12/2019
Hannousse and Yahiouche [26]	Microservice and microservice architecture security	46	●	○	12/2019
Ponce *et al.* [27]	Security smells and refactorings for microservice security	58	●	●	2021
Berardi *et al.* [28]	Microservice security in general (threat models, security measures, infrastructure, deployment)	290	●	○	2021
This study	Security threats and mitigation strategies in microservice inter-service communication	54	●	○	4/2024

mitigation strategies is less comprehensive and less specific compared to ours. Hannousse and Yahiouche [26] conducted a systematic mapping study of 46 studies on microservice security and identified unauthorized access, data exposure, and service compromise as primary concerns. The authors recommended implementing auditing and access control solutions to mitigate these risks. The study emphasized the effectiveness of these measures at the software infrastructure layer. Furthermore, an ontology was developed to assist developers in addressing specific threats and security mechanisms associated with MSA. Their work, however, does not particularly concentrate on security issues emerging from inter-service communication or the interplay between microservices, being the primary distinction to our study. Ponce et al. [27] conducted a multivocal literature review on microservice security, analyzing 58 publications and formulating ten security smells, along with a set of methods to mitigate these smells. The primary focus of this work is a trade-off analysis that aids in deciding whether to keep a security smell or to apply refactoring, taking into account the positive and negative impacts on specific code and design quality attributes. In contrast to our work, this research specifically, but also solely, concentrates on security challenges and mitigation strategies resulting from the code-based implementation of microservices. Berardi et al. [28] performed a comprehensive literature review of 290 publications, utilizing both quantitative and qualitative analysis to examine threat models, security approaches, infrastructure, and development methods. Their work provides a thorough and expansive overview of microservice security. However, it does not present a conceptualization or categorization of the identified issues, particularly in relation to inter-service communication threats and mitigation strategies in MSA.

In Table 1, we present a comprehensive summary of the dissimilarities pertaining to focus, study population, sources, and search date of the pertinent literature reviews. This table facilitates a clear understanding of the distinctions between the various existing reviews. Our research is distinct from the existing works in four aspects: a) it specifically focuses and categorizes inter-service security threats and mitigation strategies in MSA, b) by utilizing only white literature from digital scholarly libraries, c) by also assessing the level of focus that each reviewed study takes on an examined topic on an ordinal scale, and d) by also covering 17 recent studies published after 2021, which we identified as the last year a literature review in this realm has been conducted.

C. CONTRIBUTIONS AND NOVELTY

The current body of literature does yet not contain any systematic reviews that specifically and comprehensively address the topic of inter-service security in MSA. In this sense, the primary objective of this paper is to gain a more in-depth understanding of the security challenges associated with inter-service communication in MSA and to identify existing mitigation strategies. The main contributions of our work are:

- **C1.** Demographic attributes of publications w.r.t. study distribution over time, research and contribution types, publication channels and publication venues
- **C2.** Categorization and synthesis of inter-service security threats in MSA
- **C3.** Categorization and synthesis of mitigation strategies for inter-service security threats in MSA

The remainder of this paper is structured as follows: Section II outlines the methodology and research questions. In Section III we present the findings of our study in the following order: Demographic attributes of the reviewed publications are given in Section III-A, the categorization and description of inter-service security threats reported in the publications in Section III-B, followed by describing and categorizing the mitigation strategies in Section III-C. Following to that, in Section IV we condense and discuss our interpretation of the findings. In Section V, we elaborate the potential threats to the validity of this study. We conclude our work in Section VI, along with suggestions for future research directions.

II. METHOD

A systematic literature review (SLR) [30] is a widely acknowledged research method in the field of

Evidence-Based Software Engineering. It is a structured process [29] for identifying, assessing, and synthesizing all relevant evidence associated with a specific research question or topic. The process comprises three main stages: formulating a review protocol, executing the review, and documenting the review. We adhered to the guidelines provided by Kitchenham and Charters [29], which included the following key components: (i) defining research questions, (ii) devising a search strategy, (iii) establishing inclusion and exclusion criteria, (iv) selecting studies, (v) assessing the quality of the included studies, and (iv) extracting and synthesizing data. In adherence to the PRISMA 2020 guidelines [31], we have followed the framework's reporting items, ensuring transparency and reproducibility in our systematic literature review. The PRISMA 2020 guidelines aim to improve research by providing concrete measures for transparency and reproducibility. In the following, we elaborate on each step of this process as defined by Kitchenham and Charters [29].

A. RESEARCH QUESTIONS

This study aims at summarizing the current state of research pertaining to *"inter-service security threats and mitigation strategies in microservice architectures"*. To this end, we have formulated three research questions.

RQ1. What characterizes the research, contributions, publication channels, venues, and publication trends in the field of inter-service security of microservices?

RQ2. Which inter-service security threats in MSA are discussed in the studies and how can they be categorized?

RQ3. Which mitigation strategies for inter-service security threats in MSA are discussed in the studies and how can they be categorized?

The first question delves into the research and contributions facets, publication trends, and venues of the publications through a comprehensive analysis of bibliometric data. The second research question systematically identifies and categorizes the security threats discussed in the relevant literature. The third research question examines the proposed mitigation strategies and their implementations in the publications, while also grouping the identified mitigation strategies into thematic categories.

B. SEARCH STRATEGY

In order to retrieve as many relevant studies as possible, we defined a search strategy as described by Kitchenham and Charters [29] and Zhang et al. [32]. In the following, we elaborate the search method, search terms, and data sources that were used for this review.

1) SEARCH METHOD

Pilot searches were conducted between March and August 2023, with an iterative approach to refining the search string. The primary (automated) searches for this study were conducted subsequently in October 2023 and updated in April 2024.

To ensure that all relevant primary studies were included, we conducted a manual *backward snowballing* step [33] in addition to our automatic search. This was done to capture further studies that may have been overlooked during the initial search [34], [35]. There are various methods available for identifying primary studies, such as snowballing [33], quasi-gold standard (QGS) methods [32], random sampling, and margin of error [36]. These methods can be used individually or in combination to achieve the desired results. Backward snowballing involves utilizing the reference list of a publication to identify additional relevant publications. Also, this approach considers the context in which references are made, ensuring that only publications that are appropriately related to the topic are included.

2) SEARCH TERMS

We adhered to the guidelines outlined by Zhang et al. [32] and systematically refined and assessed our search terms to retrieve a suitable number of publications discussing microservices and inter-service security issues or mitigation strategies within the same context. The search string was composed of search terms concerned with *population* and *intervention* as suggested by Petticrew and Roberts [37]. The *population* encompasses microservices and inter-service communication, which are the primary areas of focus in the application domain. In this context, the *intervention* refers to security threats and mitigation strategies within the aforementioned application area. The relevant search terms were reflected through synonyms and logical operators were employed to pair them. These synonyms were selected based on the words utilized in the publications retrieved from the pilot searches, as well as the terms employed by prominent scholars in the field. Throughout the iterative refinement process, we ensured that all relevant publications that we were aware of were captured using the resulting search string. After multiple rounds of refinement, we determined the final search string as:

(microservice)
AND
("inter-service" OR "interservice" OR "service-to-service")
AND
("security" OR "threat" OR "vulnerability" OR "mitigation" OR "protection")

3) DATA SOURCES

We queried five prominent scholarly libraries, namely *IEEE Xplore*, *SpringerLink*, *ScienceDirect*, *Wiley Online Library*, and *ACM Digital Library* due to their extensive coverage of scientific literature relevant to the study's focus. Initially, we restricted the search string to keywords, titles, and abstracts in the digital libraries. We extended our search to include the fulltexts of the articles as we observed that searching within the fulltexts provided additional relevant publications. Despite the large number of retrieved publications, each publication could still be evaluated manually.

TABLE 2. Query filters used for the digital libraries.

Library	Query Filters
ScienceDirect	Article type: Review Articles and Research articles. Subject areas: Computer Science, Engineering and Decision Sciences
Wiley Online Library	Publication type: Journals
SpringerLink	Content types Conference paper and Article
ACM Digital Library	Content type: Research article
IEEE Xplore	Content type: Conference paper and Journal article

TABLE 3. Inclusion and exclusion criteria of this study.

Inclusion Criteria	
I1	Publications elaborating on security threats related to inter-service security threats in MSA
I2	Publications elaborating on mitigating strategies for security threats related to inter-service security threats in MSA
I3	Publications addressing limitations to mitigating strategies for inter-service security threats in MSA
Exclusion Criteria	
E1	Duplicates
E2	Publications in a language other than English
E3	Bachelor, Master or PhD thesis
E4	Non peer-reviewed work
E5	Insufficient microservices focus
E6	Limited security discussion
E7	Off-topic domains
E8	Insufficient quality

In addition, we employed a set of search query filters (see Table 2) for each library, which reduced the retrieved publications to those most relevant to our study.

C. INCLUSION AND EXCLUSION CRITERIA

The articles obtained from the digital libraries were assessed against a set of predetermined inclusion and exclusion criteria to systematically exclude irrelevant studies. The comprehensive list of the criteria used for selection is provided in Table 3.

The criteria for inclusion in the review were designed to limit the selection of publications to those with a specific focus on security issues or mitigation strategies in inter-service communication in MSA. Duplicate or non-peer reviewed publications as well as those in another language than English were eliminated from the outset. In order to ensure that selected papers adequately address the topics of this study and later allow us to extract relevant data, we also placed emphasis on assigning each paper to at least one inclusion criterion. This was particularly important to facilitate the later categorization of the papers into either elaborating on security threats or mitigation strategies. On the other hand, we excluded any papers that did not place sufficient emphasis on the examined topics. During the backward snowballing process, we assessed the relevance of a paper for inclusion by examining its context and reference in the main paper, and compared it to the criteria as outlined by Wohlin [33].

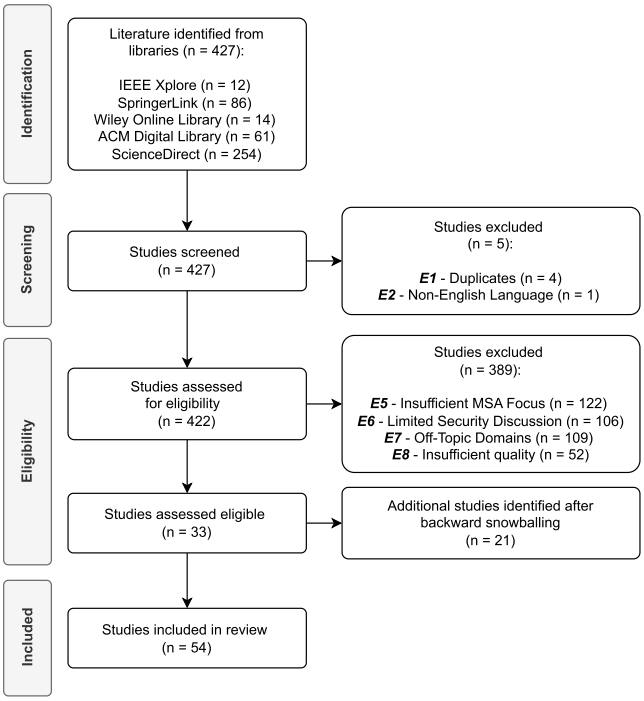


FIGURE 1. Overview of the study retrieval and selection process according to PRISMA 2020 [31].

D. STUDY SELECTION

Figure 1 shows the stages of the study retrieval and selection process according to the PRISMA 2020 guidelines [31] for reporting systematic reviews.

1) IDENTIFICATION

A total of 427 publications were retrieved from the five digital libraries using our search string and the filters, most of them from *ScienceDirect* (n = 254).

2) SCREENING

The retrieved publications were screened and duplicates (**E1**, 4 publications) and publications in languages other than English (**E2**, 1 publication) were removed in the first stage of assessment. This resulted in 422 publications remaining for the next step.

3) ELIGIBILITY

We assessed 422 publications against our inclusion and exclusion criteria based on their abstract, introduction, and/or conclusion. If a paper met any of the inclusion criteria, we thoroughly examined it. We excluded 122 publications due to insufficient focus on microservices (**E5**), further 106 publications due to limited discussion of the examined security aspects (**E6**), 109 publications due to off-topic focus

(**E7**), and 52 publication due to insufficient quality (**E8**). Therefore, we excluded 389 studies and assessed 33 studies as eligible for inclusion in this review. It is important to note that no bachelor, master, or PhD theses, nor any other non-peer reviewed works were included in these publications, so exclusion criteria **E3** and **E4** were not applicable.

The 33 studies that were identified were then utilized as the basis for the backward snowballing step, which ultimately resulted in an additional 21 studies being incorporated into this review. Just like the primary search results, all the referenced papers that were assessed during this step were also subjected to the same inclusion and exclusion criteria.

4) INCLUDED

A total of 54 studies were included in this review after retrieval, screening, and eligibility checks. Mourão et al. [38] suggest three metrics to evaluate the performance of a search method for querying digital libraries in the context of an SLR. *Precision* refers to the correctness of the results, with a score of 100% indicating that all papers retrieved were selected for the SLR. *Recall* measures the completeness of the results, with a score of 100% indicating that all papers selected for the SLR were retrieved through the search method. These two metrics are crucial in evaluating search methods for SLRs, as they balance the correctness and completeness of the results. Finally, the *F-score* provides a compromise between precision and recall, which is considered the best measure for evaluating the overall performance of a search method. These metrics can be calculated based on the below formulae.

$$Precision = \frac{|\text{Visited} \cap \text{Selected}|}{|\text{Visited}|}$$

$$Recall = \frac{|\text{Visited} \cap \text{Selected}|}{|\text{Selected}|}$$

$$F\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In Table 4 we show the performance of searching on the digital libraries and of the subsequent snowballing step. For calculating the former, the set of *visited* studies reflects the set of studies retrieved as result from this search. The metrics indicate that *IEEE Xplore* has a high precision but lower recall, meaning that it is efficient at retrieving relevant studies but may miss a significant number of them. *ScienceDirect* has a high recall, indicating that it covers a broad range of relevant studies, but its lower precision suggests that it also includes many irrelevant studies. Most of the studies used in this study were obtained from this library. Compared to the other libraries, *SpringerLink* strikes a good balance between precision and recall, as evidenced by its F-score.

When calculating the metrics for the snowballing step, the *visited* set was formed of the references of the primary studies, which we procured from a specific digital library. As depicted in the table, *IEEE Xplore* demonstrated exceptional effectiveness in revealing pertinent documents, as evidenced by its superior recall and F-score. This indicates that although it retrieved a significant quantity of irrelevant documents, it is highly adept at utilizing references to locate pertinent documents. *SpringerLink* also exhibits great potential, particularly with regard to its recall and highest F-score, suggesting a relatively effective balance in its search approach. Conversely, *ScienceDirect* and *Wiley Online Library* show significant limitations in both precision and recall, with *ScienceDirect* performing particularly poorly across all metrics.

E. STUDY QUALITY ASSESSMENT

To ensure the reliability of the studies included in our review, we established a separate exclusion criterion (**E8**) to discard studies that exhibited insuffient quality. We placed different emphasis on particular aspects of a study, depending on its type. When evaluating studies reporting evaluation or validation research, we placed emphasis on identifying biases within the studies and their internal and external validity.

For solution proposals, philosophical papers, and opinion papers, our quality assessment was less concerned with empirical validity, considering these studies' conceptual or argumentative nature. The assessment of these studies focused on the clarity of their key findings. Studies were excluded if their key findings were not explicitly stated or if their implications were not understandable. For solution proposals, sample size was taken into account when evaluating their quality. Although we acknowledge that not all solution proposals are grounded in empirical data, those that relied on empirical data were assessed for the adequacy of their sample size. Insufficient sample sizes that undermined the credibility or reliability of the findings led to their exclusion.

By differentiating the quality assessment based on the study type, we were able to include a broad range of studies, while ensuring that the quality of each study was evaluated based on criteria appropriate for its research type. In total, 52 studies were excluded due to insuffient quality.

Table 6 in the Appendix provides a comprehensive list of all the studies that finally were selected for this review. The first column of the table serves as a unique identifier for each study, with the prefix ''S'' used to refer to the study within this paper. The following column indicates the title of the study, while the subsequent columns list the authors, venue, and year of publication for each study.

F. DATA EXTRACTION AND SYNTHESIS

1) DATA EXTRACTION

Based on the properties described in Table 7 in the Appendix, we extracted all necessary information from the selected studies to answer the research questions of this systematic literature review. This information was stored in spreadsheets and made accessible to all researchers involved in the project.

2) SYNTHESIS

We have separated the data extraction form into two distinct sections. The first section involves capturing bibliometric data (**P1**–**P7**), while the second section focuses on collecting

TABLE 4. Performance of the search on the digital libraries and the snowballing process.

Library	Search			Snowballing		
	Precision (%)	Recall (%)	F-score (%)	Precision (%)	Recall (%)	F-score (%)
ScienceDirect	5.91 (15/254)	45.45 (15/33)	10.45	0 (0/459)	0 (0/21)	0
Wiley Online Library	7.14 (1/14)	3.03 (1/33)	4.26	0.34 (1/290)	4.76 (1/21)	0.64
SpringerLink	9.3 (8/86)	24.24 (8/33)	13.45	1.29 (6/465)	28.57 (6/21)	2.47
ACM Digital Library	9.84 (6/61)	18.18 (6/33)	12.77	0.95 (4/419)	19.05 (4/21)	1.82
IEEE Xplore	25 (3/12)	9.09 (3/33)	13.33	2.18 (10/459)	47.62 (10/21)	4.17

information related to security issues (**P8**–**P9**) or mitigation strategies (**P10**–**P11**) in the context of inter-service communication in MSA. As the studies examined either security issues or mitigation strategies, we were able to categorize each paper into one of these two areas based on its full text. We applied descriptive statistics to analyze the properties **P1**–**P5**.

To classify the research types (**P6**) of the selected papers, we employed six categories, namely *experience paper*, *evaluation research*, *validation research*, *philosophical paper*, *solution proposal* and *opinion paper*. These categories have been derived from the existing literature [39], [40]. The definitions of each research type are provided in Table 8 in the Appendix. In a similar vein, we extracted the contribution type (**P7**) of each paper's full text, which includes the following options: *model*, *method*, *theory*, *framework*, *guideline*, *lessons learned*, *advice* or *tool*. These options have been derived from the works of Shaw [41] and Paternoster et al. [42]. The definitions of each contribution type are provided in Table 9 in the Appendix.

The data pertaining to properties **P8** through **P11** were analyzed qualitatively, specifically applying thematic analysis to each paper's full text. Adhering to the five steps outlined by Cruzes and Dyba [43], we initially extracted and coded the primary statements in each study. This enabled us to classify each study as either describing security threats or mitigation strategies. Additionally, during this initial coding process, we established a set of pilot categories for security threats or mitigation strategies that were identified in the studies.

For each paper, we conducted multiple iterations to identify which categories were addressed within the paper. These categories were predefined, and each paper could be assigned to multiple categories, with a different level of focus. We used an ordinal scale (i.e., *low*, *medium* and *high*) to indicate the extent to which a paper focused on a particular category from the background of inter-service communication. This scale was used to differentiate between studies that briefly discussed one or more topics (e.g., systematic reviews) and studies with a more in-depth focus on one or more topics. The definition of each level of focus is provided in Table 10 in the Appendix. In cases where a paper's content did not align with any existing category, we recorded this discrepancy and scheduled the paper for another review cycle. If multiple papers addressed a topic that was not yet categorized, we created a new category specifically for that topic. This approach ensured that our categories accurately reflected the topics discussed in the paper and were tailored to the domain of inter-service microservice security. The classification of each study is given in Table 11 in the Appendix.

We continued the thematic analysis until all researchers agreed that the contents of each paper were thoroughly represented by the categories assigned to it. Table 12 and 13 in the Appendix show the extracted categories and their level of focus for each study in this review.

III. RESULTS

In the following subsections we report the results from analyzing and synthesizing the data extracted from the reviewed papers to answer the research questions. We interpret and reflect upon the results in Section IV.

A. DEMOGRAPHIC ATTRIBUTES

This subsection presents the demographic characteristics of the studies that were selected, including their distribution by year and publication type, the types of research conducted and contributions, as well as the most significant venues in the field. The findings presented in this subsection are relevant to the first research question (RQ1) as specified in Section II-A. All the selected papers and the data discussed in this section are listed in Table 11 in the Appendix.

1) STUDIES DISTRIBUTION

We extracted the year of publication of the examined studies to analyze how this research field evolved over time. As can be seen in Figure 2, all selected studies were published between 2015 and 2023, covering a duration of only eight years. This underscores the fact that microservices represent a relatively new concept. The pivotal article "*Microservices: Defining the microservices architectural style by describing their nine common characteristics*" by Lewis and Fowler [3] was published in 2014 and marked a significant moment in the field of microservices development. Consequently, the subsequent emergence of publications concentrating on microservices security, particularly the security between inter-service communications, in the following years appears to be a natural progression. The year with the most studies analyzed in this study was 2021, with 11 studies (20%) published. The year with the second-highest number of

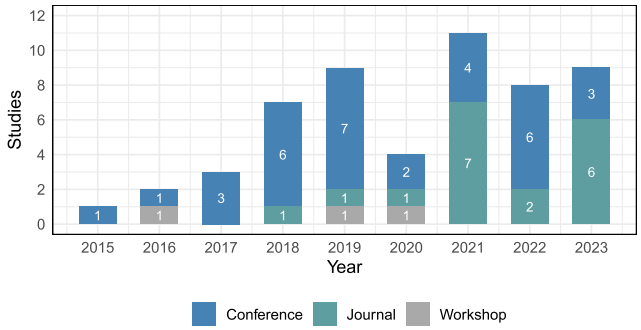


FIGURE 2. Distribution of selected studies by year and publication type.

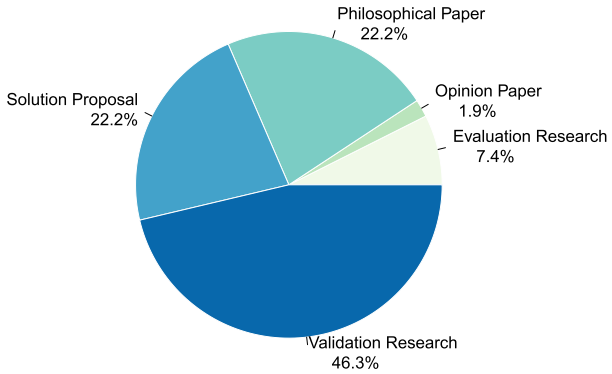


FIGURE 3. Research types of the selected studies.

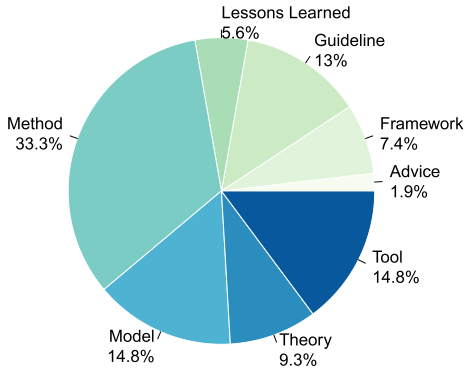


FIGURE 4. Contribution types of the selected studies.

publications was 2019. In this year, 9 studies (17%) have been published.

The majority of scholarly works in this field are typically published as conference papers, although a notable increase in journal publications was observed in 2021 and 2023. In contrast, the fewest publications were produced in any form in 2020, which may be attributed to the beginning of the Corona pandemic during that year. Given the uncertainty surrounding the possibility of conferences taking place in the following year, the increase in journal publications particularly in 2021 can be attributed to this factor. It is worth noting that only three workshop papers were published throughout the analyzed time frame, suggesting a limited relevance in this field.

2) RESEARCH TYPES

According to Figure 3, the majority of publications were categorized as *validation research*, accounting for 46.3% of the total. If we combine this with *evaluation research*, we find that more than 50% of the examined studies fall under one of these two categories, reflecting a substantial emphasis on research that either validates existing theories, models, or frameworks or evaluates their practical effectiveness. Additionally, there is a substantial volume of *solution proposals* (22.2%) and *philosophical papers* (22.2%), which tend to lack comprehensive validation or evaluation of the proposed solutions or ideas. Only one publication can be classified as *opinion paper*, representing only 1.9% of the total. The majority of the assessed publications seem to rely on existing research to formulate their content and viewpoints, or introduce new concepts and ideas that are subsequently validated and evaluated to varying extents. This suggests a strong focus on research-based content or rigorous testing of new ideas, rather than simply relying on personal opinions.

3) CONTRIBUTION TYPES

In Figure 4 we see that the majority of the examined publications primarily focus on *methods* for addressing security threats in microservice architectures, accounting for 33.3% of the total of evaluated publications. Additionally, a significant number of these works contribute *models* (14.8%) to the field, e.g., systematic literature reviews and systematic mapping studies. *Guidelines*, i.e., advices based on research results, are contributed by 13% of the publications. A small share of 7.4% of the assessed publications contribute conceptual *frameworks* for managing microservices security and categorize their content into mitigation strategies, design patterns, security tactics, and similar concepts, as exemplified in works like [S31, S11, S14]. However, a smaller segment, about 5.6%, of these publications specifically contributes *lessons learned* derived from practical experiences or empirical studies, and only one paper (1.9%) of the publications present an *advice* based on personal opinions.

To compare the distribution of research and contribution types between studies that address security threats **P8** respectively mitigation strategies **P10**, we have constructed a systematic map, as depicted in Figure 5. The systematic map highlights a prevailing interest in mitigation strategies (40 studies) compared to security threats (14 studies). Notably, among the studies focusing on mitigation strategies, there is a predominance of research related to *validation research* and *solution proposals*. This trend suggests a greater inclination towards developing and validating solutions to tackle security threats, rather than merely identifying or examining them. With regard to publications that primarily address inter-service security threats, it is evident that a substantial

TABLE 5. Distribution of the selected studies on publication venues.

Publication Venue	#	%
Journal of Systems and Software	6	11%
European Conference on Software Architecture (ECSA)	4	7%
Computers & Security	3	6%
International Conference on Availability, Reliability and Security (ARES)	2	4%
International Conference on Future Internet of Things and Cloud (FiCloud)	2	4%
IEEE/ACM International Conference on Utility and Cloud Computing (UCC)	2	4%
International Conference on Security and Privacy in Communication Networks	2	4%
Others	33	61%

number of them are categorized as *philosophical papers* and employing *evaluation research*. Furthermore, a majority of these publications contribute *lessons learned* or engage in discussions about *theory* related to the publications' topics.

4) PUBLICATION VENUES

The studies were primarily published in well-regarded journals, such as the "Journal of Systems and Software" (six studies), the "European Conference on Software Architecture" (four studies), and "Computers & Security" (three studies). These venues are widely recognized in the field of software engineering and have a strong reputation for publishing high-quality research. In Table 5, we provide a comprehensive list of the venues that have hosted at least two publications. Intriguingly, all venues with exactly two publication were conferences. Our analysis also shows that 33 of the 54 studies (61%) were hosted in single venues, i.e., each venue hosted only one publication.

B. INTER-SERVICE SECURITY THREATS IN MICROSERVICE ARCHITECTURES

In this subsection we present the results of our research into the second research question (RQ2) and categorize the inter-service security threats identified in the studies we examined. The level of focus given to each threat in each study is provided in Table 12 in the Appendix. For each category we also provide a brief summary of the key studies that are representative for that category. Figure 6 shows that many of the topics discussed in the studies were given only a low level of focus, and did not provide detailed information about specific security threats. Although many publications listed security threats and issues as risks to consider, the reasons behind these threats were often not explored in depth.

1) SECURITY PERIMETERS AND ATTACK SURFACE

As depicted in Figure 6, a substantial portion of the included publications (14 out of 54, constituting 26% of the total) concentrate on the security perimeters and the attack surface of microservices. This substantial representation reflects the popularity of this topic within the academic community. However, it is noteworthy that a substantial number of these publications exhibit a relatively low level of focus on this topic, with a particular emphasis on the attack surface of microservices.

a: INCREASED ATTACK SURFACE

The notion that adopting an MSA leads to an expansion of the attack surface, as opposed to a monolithic architecture, has been reported in numerous studies [S10, S49, S22, S24, S46, S44, S42, S35, S32]. The primary reason for this is the decentralized nature of microservices, where data are exchanged across multiple networks rather than within a single application. Furthermore, the increase in the number of open ports, accessible APIs, and access control mechanisms in MSA are identified as the primary factors contributing to the growing attack surface [S10]. According to Chondamrongkul et al. [S24], as microservices are horizontally scaled, the attack surface expands proportionally with the scaling, as new ports are opened for each deployed service.

b: PIVOTING AND TRUST ISSUES

The issue of implicit trust between services has been identified as a potential security risk in two studies [S39, S15]. According to Sun et al. [S39], this can result in scenarios where an adversary compromising one service may lead to the takeover of the entire system. Furthermore, these authors report on the limited flexibility in controlling the trust placed on different microservices. Other studies, such as [S36], also discuss the risks of pivoting and additional attacks on microservices after an initial compromise. The practice of *trusting the network* was investigated in [S15], where 13 other studies were identified to categorize this practice as negative to system security. Adherence to this practice involves trusting that all network traffic is legitimate without any verification. The study highlights that insecure inter-service communication mechanisms can be implemented by practitioners, with the justification that the network is secure and has undergone filtering and inspection by edge-level security controls. Another example of this bad practice is when microservices trust network components such as API gateways by their identity, which could allow the compromise of these network components and also pose a risk to the microservices they communicate with. The findings presented in [S8] suggest insecure configuration as a factor that explains why the compromise of one service could lead to the complete takeover of the entire system.

c: INDEFINABLE SECURITY PERIMETERS

The work of Nkomo and Coetzee [S32] highlights the challenges that arise when security perimeters are difficult to define in the context of microservices. These challenges are compounded by the implementation of containerization, which enables port mapping, dynamic addressing, and

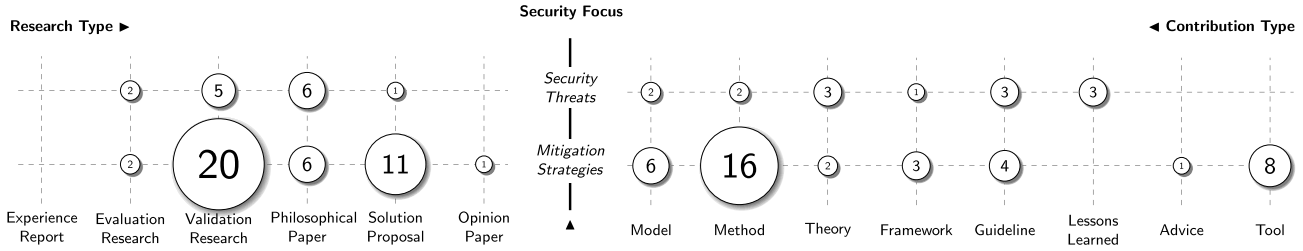


FIGURE 5. Systematic map of research and contribution types of the selected studies.

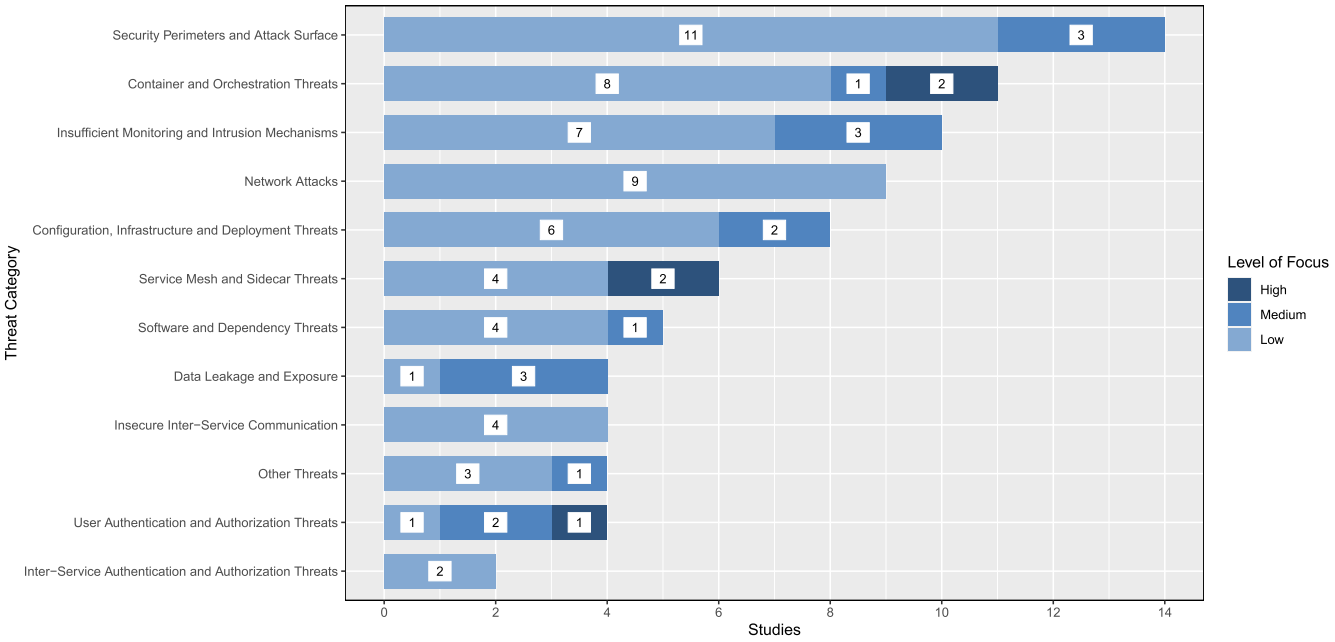


FIGURE 6. Categories of security threats and level of focus they are described in the studies.

microservice scaling. The authors argue that this added complexity makes securing microservices comparable to the challenges faced in securing monolithic systems.

d: UNNECESSARY PRIVILEGES

The issue of granting unnecessary privileges to microservices is addressed in the work of Ponce et al. [S7]. This occurs when privileges beyond what is required to carry out business functions are provided to microservices, such as when a microservice has the ability to interact with a message queue without it being essential to its business functions. In the study, it was found that 12 of the 58 selected studies reported negative security implications associated with this security smell. Additionally, the study indicates that this security smell contributes to an expansion of the attack surface for MSA.

2) CONTAINER AND ORCHESTRATION THREATS

Figure 6 shows that containerization and orchestration are crucial concepts of most MSA, as they provide the ability to scale rapidly and flexibly, which is one of the advantages of employing MSA. A significant number of the examined studies (20.3%) have reported security issues directly related to the use of containerization and orchestration tools and platforms in a microservice environment. This substantial percentage indicates that this group of security threats is relatively common among the analyzed publications.

a: VULNERABLE AND MALICIOUS CONTAINER IMAGES

Several studies [S49, S22, S46, S4] emphasize the significant number of easily exploitable vulnerabilities discovered in public container images. According to Yarygina and Bagge [S31], the utilization of vulnerable and/or malicious container images poses a significant security concern. Microservices are often deployed inside containers and the inherent mutual trust between services makes a container compromise even more dangerous, as it could allow lateral pivoting between services [S46]. Furthermore, a compromise of one container could lead to a compromise of other containers and services, and the use of vulnerable container images may be due to the use of unverified images found in public repositories [S22]. The use of images from untrusted sources that have

not been validated is considered one of the most critical image vulnerabilities [S17]. However, it is important to note that the problem with vulnerabilities found in container images is not limited to the use of unverified container images, as official *Docker* images also contain prevalent vulnerabilities [S49].

b: CONTAINERS REQUIRING LOW-LEVEL OPERATING SYSTEM FEATURES

Li et al. [S6] acknowledge that it can be challenging to containerize microservices that require low-level features from the operating system without compromising security. The research of Hannousse and Yahiouche [S8] emphasizes two main security concerns when using containers in the deployment of microservices: first, the possibility of containers being breached through unauthorized access, and second, the potential vulnerabilities that may be present within the container images themselves.

c: LACK OF AUTOMATIC UPDATES FOR CONTAINERS

According to Torkura et al. [S17], container images derived from other container images (parent images) are not automatically updated and patched in the same way as traditional software, which may leave them vulnerable, even if no vulnerabilities existed at the time of container creation. This is in line with the results of Torkura et al. [S35], which highlighted the risks associated with using homogeneous microservices. If multiple microservices are created using the same base image, a vulnerability in a base image could potentially impact all of the homogeneous microservices that were derived from it.

d: CONTAINER MISCONFIGURATION

Torkura et al. [S17] highlight the potential risks associated with misconfigurations of container images, including the possibility of vulnerabilities. Similarly, risks may also be present in container run-time and container engine configurations, as indicated by Minna and Massacci [S4], which observed a high number of container host devices using default configurations.

e: APPLICATION- AND IMAGE-LAYER VULNERABILITIES

Torkura et al. [S17] highlight the lack of research into application-layer vulnerabilities and the absence of efforts to understand the relationship between application and image-layer vulnerabilities in MSA. The authors present a study that applies vulnerability correlation to identify dependencies between different vulnerabilities, improving risk management and security hardening of microservices through a prototype implementation named *Cloud Aware Vulnerability Assessment System (CAVAS)*. This system, which outperforms traditional testing approaches in detecting vulnerabilities, underscores the importance of correcting severity metrics provided by image vulnerability scanners for effective continuous security and risk assessments in MSA.

f: CONTAINERIZATION PLATFORMS

Security vulnerabilities within containerization platforms have been examined in a study by Zeng et al. [S1], which analyzed *Docker*, *containerd* [44], and *runc* [45]) for *Common Vulnerabilities and Exposures (CVE)* that affect various containerization platforms. Among the components assessed, the *Docker* platform was determined to be the most susceptible to being compromised, with 80% of its vulnerabilities related to privilege escalation. Additionally, the study raised the concern of container escape possibilities, which could potentially allow an attacker to access files or perform actions beyond the confines of the container.

g: ORCHESTRATION TOOLS

Zeng et al. [S1] conduct a mapping of public *CVE* to the various components utilized in the*Kubernetes*orchestration platform [19]. These components include the *API Server*, *Kubectl* [46], and *Kubelet* [47]. The study found that the *API Server* component was the most vulnerable, with 30% of *CVEs* related to this component. Additionally, 80% of all identified vulnerabilities were caused by privilege escalation attacks, with a significant number resulting from tools that work together with the *Kubernetes* platform, such as plugins. The *Kubernetes* network and its different layers (Layer 2 and Layer 3) were also evaluated, and it was discovered that 90% of the vulnerabilities were related to increased privileges and 50% of these were caused by *Man-in-the-middle (MITM)* attacks resulting from misconfigurations.

h: EXCESSIVE PRIVILEGES

The potential security risk associated with granting excessive privileges to *Docker* containers is highlighted by Ibrahim et al. [S37]. Due to the fact that certain containers require extensive privileges to function effectively and are therefore granted access to the *Docker* daemon, they may have the ability to access all services within a microservice-based system, as these are all managed by this daemon.

3) INSUFFICIENT MONITORING AND INTRUSION DETECTION MECHANISMS

Figure 6 shows that 10 out of 54 publications (18.5%) address the issue of inadequate monitoring and intrusion mechanisms in MSA. These publications examine various aspects of intrusion mechanisms, including detection, response, and prevention. The research in this category focuses on the need for robust security measures to protect against potential threats and ensure the secure functioning of microservices.

a: LACK OF REACTION, DETECTION AND RECOVERY MECHANISMS

A study conducted by Márquez and Astudillo [S23] revealed that MSA primarily concentrate on preventing attacks, with insufficient focus on detecting, responding to, or recovering from them. This research also emphasizes the scarcity of discussion and development regarding recovery mechanisms

for MSA in the event of a cyber attack, which are essential to restore the system to its normal functioning state. Furthermore, Yarygina and Otterstad [S29] highlight the deficiency in research on *Intrusion Response Systems (IRS)* tailored for MSA, particularly given the increasing number of potential attacks. The study also mentions the restricted focus of research on the self-protection capabilities of microservices.

b: LACK OF TRACEABILITY AND INTRUSION DETECTION

The issue of traceability in MSA has been identified in two separate studies [S54, S22]. According to Alshuqayran et al. [S54], tracing an inter-service request through microservices is a challenging task that warrants further research. The findings of Pereira-Vale et al. [S3] indicate that only a small number of intrusion detection systems have been developed to identify anomalous behavior in MSA.

c: DIFFICULTIES WITH SECURITY MONITORING

The challenges related to security monitoring in MSA are highlighted by Nkomo and Coetzee [S32]. Specifically, the use of containers in microservice-based meetings systems can add complexity to the process of identifying network traffic for specific services. It is difficult to determine which specific service is operating within each container, further complicating the process of monitoring. Additionally, the use of cloud services that are not owned by product owners can make deployed microservices attractive targets for adversaries [S39]. This can lead to limited visibility of service failures in MSA [S22], making it difficult to track system failures. This can complicate to effectively identify and address security breaches in microservice-based systems.

d: DEEP-PACKET INSPECTION ISSUES

The complexities involved in traffic monitoring and filtering in relation to deep packet inspection are thoroughly explored in the work of Nehme et al. [S49]. The authors underscore the necessity for tailored rules in deep packet inspection within an MSA, while simultaneously acknowledging the difficulty of this endeavor.

4) NETWORK ATTACKS

Several publications discuss different network level attacks, such as *MITM* attacks and *Distributed Denial-of-Service (DDoS)* attacks (see Figure 6). Out of the 54 analyzed publications, only 9 (16,7%) discuss one or more network attacks that could pose a threat to microservice systems. Although all publications discuss these threats on a low level of focus, these threats have not been the primary focus of detailed examination by these publications.

a: DISTRIBUTED DENIAL OF SERVICE

Numerous studies have highlighted the potential vulnerability of microservices to DDoS attacks [S20, S7, S51, S31, S24, S23]. According to Zdun et al. [S20], DDoS attacks can result in service-level degradation or availability issues in MSA. The large attack surface typically associated with a microservice system, as identified by Márquez and Astudillo [S23], may contribute to increased susceptibility to DDoS attacks. Additionally, the use of utility microservices, such as sidecars, has been reported to pose an increased risk of DDoS attacks by Suneja et al. [S26], as these microservices may contain unpatched DDoS-related vulnerabilities. Furthermore, the flexibility of microservices can be exploited in DDoS attacks, where attackers could potentially scale up less secure microservices to overwhelm more secure central components.

b: MAN IN THE MIDDLE ATTACKS

The vulnerability of MSA to MITM attacks has been acknowledged in multiple studies [S20, S24, S42]. While these sources emphasize the potential threat posed by MITM attacks to microservices, they do not provide in-depth analysis or information on this topic.

c: SERVICE DISCOVERY AND HIJACKING ATTACKS

Undermining service discovery mechanisms and the possibility of enrolling a malicious node within MSA is a security concern pointed out by Yarygina and Bagge [S31]. This type of attack could lead to misdirecting communication towards the attacker. Torkura et al. [S42] provide a brief overview of the issue of session/token hijacking. However, the discussion of this threat in this paper lacks extensive elaboration.

5) CONFIGURATION, INFRASTRUCTURE AND DEPLOYMENT THREATS

Figure 6 depicts that 8 of the 54 analyzed studies (14.8%) concentrate on security threats associated with the configuration, infrastructure, and deployment of MSA. The majority of these studies take a low level of focus.

a: AGNOSTIC TECHNOLOGY AND VULNERABILITY DETECTION

Microservices can be developed using diverse programming languages, libraries, and other tools, yet they are capable of functioning and communicating collectively within a unified system. Moreover, MSA frequently employ a variety of technologies and tools, such as those for monitoring, traffic mediation, and data storage. According to Torkura et al. [S42], the technological diversity within such systems can complicate the process of vulnerability detection, as each technology employed must be separately identified and tested for vulnerabilities.

b: CLOUD INFRASTRUCTURE

The potential security threats arising from the utilization of cloud technologies in MSA are investigated by Yarygina and Bagge [S31]. The authors characterize cloud technologies as engendering "a myriad of security concerns". As an illustration, they sketch an adversary that compromises one service inside the perimeter and afterwards moves laterally through the system to gain full control. If the credibility

of the sender is not ensured in inter-service communication and microservices blindly trust whoever is calling them, then a single compromised microservice would allow an attacker to manipulate all the other services, for example by issuing arbitrary malicious requests that each microservice will fulfill. The latter is sometimes referred to as *Confused Deputy Problem*. Furthermore, the adversary could attempt to eavesdrop on the inter-service communication, insert, and modify data in transit.

c: SECRETS AND DATA AT REST

According to Ponce et al. [S7], a variety of security issues arise when data and secrets are stored in MSA. One significant concern is the *Hardcoded Secrets* security smell, which occurs when secrets are embedded directly in the application code or configuration files used for deployment. Additionally, their research emphasizes the critical problem of not encrypting data, particularly data at rest, which falls under the *Exposure of Non-Encrypted Data* security smell. Storing unencrypted data or using encryption mechanisms with known vulnerabilities can lead to this issue. Moreover, the study highlights a potentially larger risk: the use of in-house developed cryptographic code that has not undergone extensive testing and can thus create a false sense of security.

d: ATTACKS TO SOFTWARE AND HARDWARE INFRASTRUCTURE

The research conducted by Hannousse and Yahiouche [S8] explores the categorization of attacks on microservices, distinguishing between *Soft-Infrastructure* and *Hard-Infrastructure*. Attacks to the former type refer to vulnerabilities in the software components of an MSA, such as message queues, network mediation software, and monitoring tools. In contrast, Attacks to the latter type target the hardware components of a software system, focusing on vulnerabilities that may have been intentionally or unintentionally introduced during development. These hardware vulnerabilities present potential exploit opportunities for adversaries, and it has been noted that comparatively less attention is paid to securing microservices against such attacks. This suggests a potential gap in security measures being implemented, with a stronger focus typically placed on software aspects rather than hardware vulnerabilities.

e: POLYGLOT AND HOMOGENEOUS ENVIRONMENTS

According to Billawa et al. [S22], deploying microservices written in various programming languages, each with its own versions and life cycles, presents complex challenges. These challenges are further exacerbated by the difficulties of implementing effective security measures in such a diverse ecosystem. This is due to the fact that each programming language typically relies on different frameworks, which demand distinct security considerations and approaches. The need to tailor security practices to each framework adds layers of complexity to ensure robust security in these heterogeneous microservice architectures.

6) SERVICE MESH AND SIDECAR THREATS

Figure 6 illustrates that 6 out of the 54 studies (11.1%) handle topics related to service meshes and sidecar threats. By using a service mesh, much of the functionality that previously had to be implemented within the microservices codebase is shifted to the service mesh, which simplifies the implementation of commonly used functionality in MSA independent of the technologies used to develop microservices [1].

a: SERVICE MESH THREATS

Aktypi et al. [S21] identify significant design issues in contemporary service mesh technologies. One of the key challenges is the difficulty in consistently establishing a connection to a central registry. Moreover, as highlighted by El Malki and Zdun [S38], the absence of encrypted generated keys and certificates, and the lack of use of a certificate management service outside the service mesh, increase the susceptibility of the service mesh to manipulations and malicious traffic. Another issue addressed by Alboqmi et al. [S48] is the lack of policy autonomy within service meshes. This problem is compounded by the need for manual intervention to make changes within the service mesh configuration at run-time in order to protect against potential threats. Zeng et al. [S1] examined the vulnerabilities connected to *Istio* [48] and uncovered a number of substantial security issues, many of which were triggered by misconfigurations in the data and control plane or exposed vulnerabilities that were already known to the public.

b: THREATS FROM SERVICE MESH TOOLS

The study conducted by Hahn et al. [S28] investigates the security vulnerabilities present in widely used service mesh tools, which were primarily caused by misconfigurations and the absence or insufficiency of security mechanisms. The primary focus of this study was on the *Consul* service mesh [49], which served as a model, and also included evaluations of *Istio* and *Linkerd* (version 2) [50]. The study discovered that, unlike *Consul*, both *Istio* and *Linkerd* required a *Kubernetes* cluster to implement their security features, resulting in increased complexity in the deployment of these service mesh technologies.

c: SIDECAR ISSUES

According to Suneja et al. [S26], injecting utility microservices (such as sidecars) into the container of a core (regular) microservice facilitates close collaboration between services. However, this raises security concerns as the lack of isolation between the services implies that vulnerabilities affecting the utility microservice may compromise the core microservice, leading to potential risks for the entire system.

7) SOFTWARE AND DEPENDENCY THREATS

The issue of security threats arising from insecure code, poor architectural choices, and vulnerable dependencies is addressed in 5 out of the 54 publications studied (9.2%). Most of these publications give only a low level of focus to these security threats (see Figure 6).

a: DIFFICULTY IN DEVELOPING MICROSERVICES

The absence of tailored security patterns for developing MSA is highlighted in a study conducted by Pereira-Vale et al. [S3]. This research examines the scarcity of microservices security patterns in academic literature. According to Waseem et al. [S27], three out of six interviewees cited "poor design and introducing insecure code for microservice systems" as factors contributing to the complexity and difficulty of addressing security in a microservice system. Ahmadvand and Ibrahim [S50] argued that the decomposition of a software system into a microservices system is often done with the interests of a select few stakeholders in mind, resulting in a system that does not take into account the requirements of the entire system. The research presented by Waseem et al. [S9] suggests that MSA may be more susceptible to vulnerabilities than those constructed with a monolithic architecture. This increased risk is partially attributed to the greater use of third-party components in microservice-based applications.

b: HOMOGENEOUS ENVIRONMENTS

The study of Torkura et al. [S35] highlights significant risks associated with the adoption of homogeneous technology stacks in MSA. This practice, commonly used to sidestep the complexity of maintaining multiple technology stacks characteristic of polyglot architectures, can lead to the proliferation of shared vulnerabilities among services. The risk of these shared vulnerabilities is particularly acute in homogeneous microservice environments, where the uniformity of the technology stack can exacerbate security risks.

8) DATA LEAKAGE AND EXPOSURE

Out of the 54 selected publications, only 4 studies (7.4%) discuss dangers associated with the excessive exposure of data or unintended data breaches within microservice ecosystems. Given the nature of MSA, which accommodates expansive and complex systems, efforts to minimize data exposure present significant challenges, as evidenced by the selected publications.

a: LEAKAGE OF DIAGNOSTIC INFORMATION

Rezaei Nasab et al. [S2] discuss the vulnerability of diagnostic endpoints to attacks. The authors emphasize the potential risk of sensitive information leakage that may arise if these endpoints are not adequately secured.

b: EXPOSURE OF UNENCRYPTED DATA

The security smell *Exposure of Unencrypted Data*, as defined by Ponce et al. [S7], arises when microservices fail to implement adequate security controls and storage mechanisms. One example of this is storing sensitive data in plaintext without encryption. This security smell poses a threat to the confidentiality and integrity of MSA, as a breach or unauthorized access to the data could compromise the system's security.

c: DATA EXPOSURE THROUGH APIS

As described by Genfer and Zdun [S18], the exchange of information in an MSA through APIs is extensive, and data transfers typically involve sensitive and private information. Therefore, the study emphasizes that data exposure represents a significant security risk. The authors highlight the risk of excessive data exposure through APIs, which should be avoided to prevent sensitive data from being uncovered. Excessive data exposure is defined as data that are neither consumed by the receiving API nor routed to another API. The study of Zdun et al. [S41] also addresses the associated security risks of disclosing unnecessary information about system internals in API error messages, as this expands the attack surface. Furthermore, the study reports that excessive utilization of APIs by a user can negatively impact the availability of a microservice-based system.

9) INSECURE INTER-SERVICE COMMUNICATION

Insecure inter-service communication, which refers to various network communication methods utilized between services in MSA, is a prevalent issue that has been identified in 4 of the 54 analyzed studies (7.4%). This issue is discussed with a relatively low level of focus in the selected studies, as illustrated in Figure 6.

a: INSECURE INTER-SERVICE COMMUNICATION

According to Ponce et al. [S7], the security smell *Unsecured Service-to-Service Communications* represents a security concern when no safeguards are implemented to secure inter-service communication. The absence of secure communication methods can result in vulnerabilities such as MITM attacks, eavesdropping, and data manipulation, thereby compromising the confidentiality, integrity, and availability of the system. Moreover, Waseem et al. [S27] highlight that one of the complexities in securing MSA is the issue of insecure inter-service communication. This aspect significantly contributes to the challenge of addressing security concerns in the design of microservices.

b: ATTACK SURFACE IN INTER-SERVICE COMMUNICATION

Wang et al. [S12] and Pereira-Vale et al. [S3] note that the attack surface in MSA expands due to inter-service communication. This increase in vulnerability is a result of

the fact that communication between services takes place over a network rather than being limited to local interactions.

10) OTHER SECURITY THREATS

As shown in Figure 6, 4 studies out of the 54 included in this review (7.4%) address inter-service security threats that cannot be categorized under any of the previously listed categories.

a: INSIDER THREATS

Ahmadvand et al. [S14] examine the security threats that can arise when insiders have access to microservices systems. According to the study, *insiders* refer to individuals who have access to the tools that manage microservices, such as DevOps engineers and Sysadmins. The research presented various attack paths that target both data and behavior assets. For data assets, the study emphasized the security threats associated with the manipulation of logs. As for behavior assets, it highlighted the risk of manipulation of services, hardware containers, and similar modules. Additionally, it noted that such interference could facilitate in-memory attacks.

b: THREAT MODELING AND RISK ASSESSMENTS

The research of Nkomo and Coetzee [S32] underscores the inherent security challenges associated with conducting threat modeling and risk assessment for MSA. The independent nature of development teams in such architectures, coupled with continuous delivery practices, heightens the risk of releasing vulnerable code before thorough risk assessments and threat models can be developed. Moreover, the absence of threat models specifically designed for MSA, as noted by Berardi et al. [S51], adds complexity to developers' workload.

c: MISSING INTER-SERVICE SECURITY POLICIES

According to Li et al. [S46], the development of more flexible mechanisms for generating inter-service security policies for MSA is necessary. The study highlights that the current practices of manual configuration become increasingly challenging as an MSA expands in size. The authors also emphasize that their research resulted in the development of a tool aimed at addressing these challenges.

11) USER AUTHENTICATION AND AUTHORIZATION THREATS

The issue of user authentication and authorization is closely linked to inter-service security, although it is not exclusively related to it. This is exemplified by the fact that user tokens are frequently exchanged between services or used in service-specific authorization processes. As illustrated in Figure 6, a relatively small number of only 4 publications (7.4%) address security threats related to this topic, with a distribution that spans all levels of focus.

a: NON-UNIFORM ACCESS CONTROL

Ponce et al. [S7] delve into the *Insufficient Access Control* security smell, which is characterized by a lack of uniform access control enforcement across all microservices in MSA. The study points out that this deficiency may result in vulnerabilities, including the creation of potential attack vectors for the *Confused Deputy Problem*. Furthermore, the authors underscore that in MSA, access control and identity management necessitate a unique approach in contrast to a monolithic application. More specifically, automated identity verification is required when data is transferred between various services.

b: ISSUES WITH CENTRALIZED AUTHORIZATION

The security smell entitled *Centralized Authorization* as described by Ponce et al. [S7], arises in MSA when authorization duties are restricted to a solitary microservice, usually positioned at the system's periphery. This arrangement disregards the dispersed nature inherent in microservices by neglecting to implement precise access control at the level of each microservice. Instead, it opts for centralized regulation, which can also result in latency issues. This security concern, like the *Insufficient Access Control* smell, serves as an enabler for the *Confused Deputy Problem* as the services are compelled to trust the central authorization service. Nehme et al. [S49] also argue that this problem could arise when access tokens are only verified at the *API Gateway*. According to Soldani et al. [S10], the challenges related to access control in MSA can be addressed by utilizing suitable tools or technologies for decentralized access control.

c: USING MULTIPLE POINTS FOR AUTHENTICATION

Ponce et al. [S7] point out that the *Multiple User Authentication* security smell emerges in MSA when user authentication is carried out at multiple points. This practice expands the attack surface and elevates the development effort required to maintain and secure these multiple authentication points. In a similar vein, Soldani et al. [S10] state that distributed authentication and access control enlarges the overall attack surface of systems due to the numerous open ports and APIs.

d: ACCESS MECHANISM WEAKNESSES

The research conducted by Nehme et al. [S13] delves into the security concerns present in the access mechanisms utilized in MSA. The authors emphasize that *OAuth2* access scopes are confined to static and coarse-grained levels, which in turn poses challenges in devising adaptable policies. Moreover, the study underscores the intricacy involved in auditing these scopes. The study also identifies vulnerabilities in MSA associated with *OpenID Connect* [51], a framework based on *OAuth2*. It is noted that these mechanisms typically rely on a single token to access all microservices, thereby creating the issue of *Powerful Token Theft*. Furthermore, the research sheds light on the *Confused Deputy Problem*, a significant security risk in this context.

12) INTER-SERVICE AUTHENTICATION AND AUTHORIZATION THREATS

The issue of security threats related to inter-service access control mechanisms is addressed in two (3.7%) of the 54 publications analzyed for our study. As illustrated in Figure 6, this topic has yet not been extensively explored in literature.

According to Ponce et al. [S7], the concept of *Unauthenticated Traffic* was introduced as a security smell, which pertains to instances in which communication between services lacks proper authentication. Walsh and Manferdelli [S43] also point out the lack of research on authentication mechanisms between services, indicating the necessity for additional investigation and development in this particular area of study.

C. MITIGATION STRATEGIES TO INTER-SERVICE SECURITY THREATS IN MICROSERVICE ARCHITECTURES

In this section, we present the results of our investigation into the third research question (RQ3) and classify the inter-service mitigation strategies in MSA that we identified in the studies. The level of focus given to each threat in each study is detailed in Table 13 in the Appendix. For each category, we provide a concise overview of the key studies that are representative of that category.

As shown in Figure 7, a substantial number of the analyzed publications focus on mitigation strategies at a medium level. Furthermore, many of these publications contribute models and methods for mitigating security threats in MSA from an inter-service perspective. Also the figure illustrates that mitigation strategies aiming on secure coding practices and patterns as well as user authentication and authorization mitigation techniques are the most commonly discussed topics in the reviewed studies.

1) SECURE CODE, DESIGN PATTERNS AND ARCHITECTURE

This topic is discussed in 13 of the 54 publications (24%) in relation to mitigating security threats for the inter-service security domain in MSA. Most of these publications discuss the topic with a medium level of focus.

a: DEVOPS AND DEVSECOPS

Macarthy and Bass [52] define DevOps as a set of practices aimed at automating and integrating the processes between software development and IT teams, to build, test, and release software faster and more reliably. The focus is on continuous integration, continuous delivery, and automated testing to enhance the speed and quality of software development. Within the context of MSA, DevOps practices facilitate the management of independently deployable, small, modular services. In this realm, Alshuqayran et al. [S54] identified 50 effective tools for implementing MSA within DevOps, underscoring the importance of adopting DevOps practices for managing microservices effectively.

The integration of security in DevOps has led to development of DevSecOps. The central notion of DevSecOps is to prioritize security and incorporate security measures and practices into the DevOps process [53]. In the context of MSA, DevSecOps plays a critical role in continuously integrating security into the microservices' development process. Ponce et al. [S15] recommend adopting the DevSecOps practice and implementing continuous security testing as strategies to address the security issues related to the *Non-Scalable Security Controls* security smell. The authors explain that DevSecOps integrates security considerations into the development process early on, while continuous security testing ensures ongoing automated security assessments throughout the lifecycle of an MSA.

b: SECURITY BY DESIGN

Billawa et al. [S22] discuss *Security by Design* and DevSecOps in the context of securing MSA. The former emphasizes that security should be a central aspect throughout the entire microservice life cycle, working in tandem with DevSecOps to timely identify vulnerabilities in the development process. DevSecOps seeks to incorporate security principles and standards throughout the whole software lifecycle while ensuring continuous integration and fast deployment, implementing issue tracking to ensure timely identification of any defects along with continuous and automated security testing. These concepts are highlighted as best practices for securing MSA.

c: SECURITY PATTERNS AND TACTICS

Bass et al. [9] define *tactics* as deliberate design choices that significantly impact a software system's quality attributes by influencing its response to various stimuli. It is essential to note that security, as a critical quality attribute, is also profoundly influenced by the selection and implementation of suitable tactics. Tactics are implemented through employing patterns specific for a particular tactic.

A distinction must be drawn between architectural patterns presented by Bass et al. [9] (see Section I) and software design patterns, presented by Gamma et al. [54]. The former pertains to creating software architectures that meet specific quality attributes to a predetermined extent. In contrast, the latter proposes reusable software designs, such as class composition, responsibilities, and methods, to improve the maintainability of software by employing object-oriented structures and behaviors.

Billawa et al. [S22] emphasize the significance of architectural patterns in developing secure microservices. They provide examples such as the *API Gateway*, the *Command Query Responsibility Segregation (CQRS)*, and the *Circuit Breaker* pattern, contributing primarily to the availability of microservices [9]. The *API Gateway* can be employed when data from multiple microservices need to be retrieved. In particular, it encapsulates the software architecture in an API tailored for each individual internal or external actor, acting as an authentication and authorization hub.

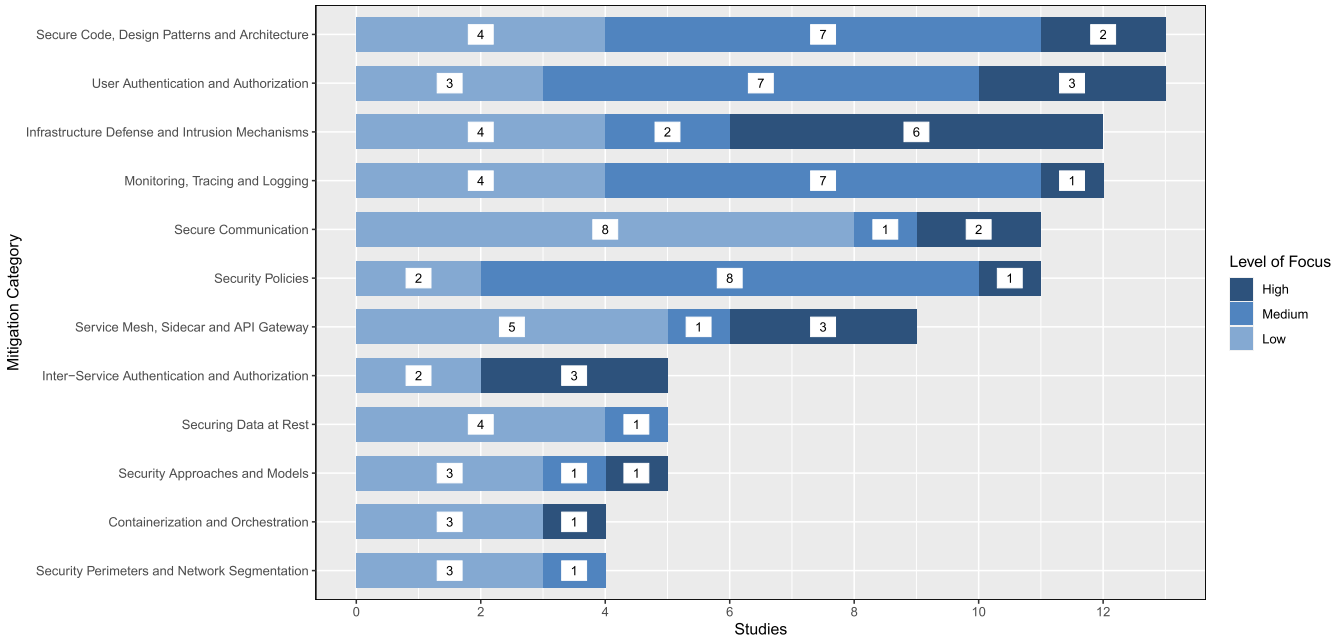


FIGURE 7. Categories of mitigation strategies and level of focus they are described in the studies.

On the other hand, the *Circuit Breaker* pattern is designed to prevent a chain reaction of security failures. By setting a threshold that determines the maximum number of failures a microservice can tolerate, it helps mitigate the impact of security breaches. The authors also consider the *CQRS* pattern to be suitable for securing MSA. It involves separating read and write operations in a data store, making it easier to ensure that only authorized entities can read and write data. Recognizing that security is significantly connected to the principles of confidentiality, integrity, and availability, these principles together constitute the widely recognized "CIA triad" [9], [55].

Additionally, Márquez and Astudillo [S23] investigate the utilization of architectural tactics and patterns in microservices development, focusing particularly on availability patterns and tactics.

d: SECURE MICROSERVICE DEVELOPMENT

Waseem et al. [S25] discuss the use of decision models in choosing appropriate design patterns and strategies for MSA, specifically in the context of microservice security. The decision model outlined in the paper requires architects to assess the relative importance of various attributes, such as security, confidentiality, integrity, and availability. Each design pattern and strategy is then connected to one or more of these attributes, either positively or negatively impacting them. Zdun et al. [S41] outline approaches for developing quality-focused microservices APIs by means of reusable decisions, which encompass design patterns, practices, or specific choices. In a similar vein, Zimmermann et al. [S45] devise a pattern language that can be applied to the creation of secure microservice APIs, among other uses. Ponce [S33] describes a methodology for developing a taxonomy of security smells and mitigation strategies that are specific to microservice security. This taxonomy provides a set of refactorings that serve as a guide for creating secure microservices. These refactorings share similarities with security tactics in their concrete definitions and in the directive manner in which they should be executed. Nkomo and Coetzee [S32] suggest a range of best practices for the secure development of microservices, including implementing and validating secure software development practices, safeguarding configurations (such as those for containers) during runtime, continuously monitoring assets, employing automated adaptation measures in response to attacks, and meticulously documenting the security requirements of the microservices architecture. Ahmadvand and Ibrahim [S50] propose a methodology for transitioning a system from a monolithic to a microservice architecture, ensuring both security preservation and optimal scalability. The authors also applied this methodology in a fictional case study to demonstrate its practical applicability.

e: VALIDATING ADHERENCE TO SECURITY TACTICS

The study conducted by Zdun et al. [S20] evaluates the implementation of security measures in MSA and assessed the extent to which these measures adhered to established guidelines. The authors identified challenges in ensuring MSA security, including the difficulties and potential for errors in manually verifying the system's compliance with various security measures. To address this issue, the authors derived a set of alternative security tactics from the reviewed literature, which they categorized into several *Architectural Design Decisions (ADD)*, each focused on a specific aspect

of microservice security, such as observability, secure communication, and authentication and identity management. Additionally, the authors developed metrics to evaluate the different security tactics and automatically determine whether a system conforms to the analyzed ADDs and security measures.

f: DETECTING VULNERABLE CODE

Waseem et al. [S25] present a method for scanning dependencies in MSA to detect vulnerabilities in the development pipeline and codebase. This strategy can be used to identify security-related issues in the source code. Schneider and Scandariato [S47] propose a technique for creating dataflow diagrams for *Java* microservices that include security annotations. By doing so, the authors can locate security-related parts of the source code. Their method utilizes the snowballing approach, which entails examining the source code for specific keywords and iteratively identifying additional keywords. The researchers achieved a precision rate of over 90% and a recall rate of 85% in validating their approach.

2) USER AUTHENTICATION AND AUTHORIZATION

Figure 7 shows that 13 of the 54 assessed publications (24%) concentrate on mitigation strategies concerning user authentication and authorization. It is noteworthy that the majority of these publications dedicate a medium level of focus to this topic. Nevertheless, there are also three studies that exhibit a higher level of interest, delving more deeply into this area with a high level of focus.

Utilizing federated identity is proposed by Li et al. [S6] as a solution to address the implementation of the *Authentication and Authorization* tactic [9] in MSA. This approach involves the use of technologies such as *OpenID Connect*, *JWT*, and *OAuth2*. Several security practices related to user authentication, authorization, and credentials have been defined by Rezaei Nasab et al. [S2]. In addition, Zdun et al. [S20] specifically address the identification and authentication of API clients. The research highlights two distinct design choices for implementing these processes for API clients. The first option eliminates any authentication measures, which is only advisable if the risks of misuse are negligible [S41]. The second approach recommends enforcing authentication, a process that can be securely implemented using protocols such as *OAuth*, *SAML*, *Kerberos* or *LDAP*.

a: TOKENS

The use of security tokens, such as *JSON Web Tokens (JWT)*, is prevalent in many of the analyzed publications. This is unsurprising, as the *Token Pattern* is a well-established security design pattern in MSA, as described by Richardson [2]. The use of security tokens has been shown to enhance security in terms of identity management in microservices. According to Billawa et al. [S22], the *JWT* standard is effective for securely transmitting claims between two parties in MSA. Waseem et al. [S25] introduce the *Access and Identity Token* pattern, which utilizes tokens and access-based protocols such as *JWT* and *OAuth1 & 2*. These tokens can carry user data that can be validated, for example, by an edge-service. The authors claim that the implementation of this pattern improves aspects such as confidentiality, integrity, accountability, authenticity, and recoverability. Rezaei Nasab et al. [S2] explored various security practices related to tokens and credentials, evaluating their effectiveness based on interviews with microservice practitioners. The study finds that the use of *JWT* for handling session revocations and expirations, as well as the practice of employing asymmetric encryption for *JWT*, was considered either absolutely useful or useful by almost all the interviewed practitioners. Additionally, 78.83% of the interviewed practitioners believed that encrypting tokens is either absolutely useful or useful when the tokens are to be exposed outside the microservices boundaries.

b: USER AUTHORIZATION

According to Nehme et al. [S49], it is essential to verify the scope of access tokens in incoming requests and to have a central authorization service, such as for audit purposes, in order to address the *Confused Deputy Problem* in microservices. Rezaei Nasab et al. [S2] found that the majority of microservice professionals they interviewed regarded the implementation of an *API Gateway* in large systems as highly beneficial for authorizing requests to microservices. Waseem et al. [S25] discuss the *Edge-Level Authorization* pattern, which enforces authorization at the *API Gateway* and enhances both the security and integrity of the system. The authors also introduce the *Service-Level Authorization* pattern, where authorization is enforced at each individual microservice through access control policies. The authors recommend this pattern over the *Edge-Level Authorization* pattern as it increases the granularity of the access control policies and improves the availability, security, and integrity of the system. Nehme et al. [S13] propose a tool that utilizes *OAuth2* and *eXtensible Access Control Markup Language (XACML)* [56], two open standards, to address the challenges associated with implementing and enforcing access control in microservices. The system consists of "*an access control server acting as an OAuth2 and XACML server, consumer microservices that contain OAuth2 client credentials and require access to resources, resource microservices that host and expose assets, and a gateway that secures each microservice*" [S13]. According to the authors, the use of the *OAuth2* authorization protocol is an effective mitigation measure for the security smell of *Insufficient Access Control* as discussed by Ponce et al. [S7]. Additionally, Billawa et al. [S22] also mention *OAuth2* as an effective security protocol.

c: DECENTRALIZED AUTHORIZATION

The research conducted by Ponce et al. [S7] highlights decentralized authorization as a potential mitigation approach for the *Centralized Authorization* security smell. By adopting this approach, which is based on a token-based authorization

protocol, authorization can be implemented at the service level. Xi et al. [S11] propose a scheme for decentralized access control in microservices, utilizing a blockchain-based distributed access control scheme and smart contracts to store policies for microservices authorization. The authors claim that their scheme not only ensures confidentiality, integrity, and non-repudiation but also exhibits better performance than classic access control schemes for microservices. This claim is supported by the authors' results of the experiment.

d: USER AUTHENTICATION

Ahmadvand et al. [S14] stress that it is essential to verify the identity of the end user before making any modifications to sensitive data in MSA to ensure data integrity. Furthermore, the authors emphasize the importance of authenticating configuration changes, such as alterations to container images. Zdun et al. [S20] suggest two architectural design decisions related to authentication: a) backend authentication, which focuses on interactions between systems like service discovery and microservices, and b) authentication on paths from clients or UIs to system services, which deals with interactions between a system and end-users. Both design decisions propose the use of *Token-based Authentication* (using identity tokens, such as JWT tokens) and *Protocol-based Authentication* (using encrypted protocols, such as *SSL*) as the most secure options for both scenarios. Ponce et al. [S15] recommend using *OpenID Connect* [51] for authentication and *OAuth2* for access control as part of the *Defense-In-Depth* and *follow The Zero Trust Principle* security practices.

e: OPENID CONNECT

Ponce et al. [S7] note that *OpenID Connect* is considered a highly effective method for mitigating the *Unauthenticated Traffic* security smell. The authors further explain that this authentication protocol is commonly employed in conjunction with a *JWT* to transfer *OpenID Connect* session information. Additionally, Billawa et al. [S22] provide recommendations for the use of *OpenID Connect*, which are supported by several gray literature sources.

3) INFRASTRUCTURE DEFENSE AND INTRUSION MECHANISMS

Security measures and mitigation strategies related to infrastructure defense and intrusion mechanisms are presented in 12 of the 54 assessed publications (22.2%). The majority of these publications have a high level of focus on the topic.

a: DATABASES AND ENVIRONMENTS

The work of Rezaei Nasab et al. [S2] discusses several security measures for databases and environments in MSA. According to the survey conducted with microservice practitioners, two security practices were considered useful by the majority of respondents. The first practice stresses the importance of implementing more stringent security policies in production environments than in development environments, while still acknowledging the importance of security measures in development environments. The second practice highlights the significance of always enforcing authentication to databases.

b: INTRUSION DETECTION AND RESPONSE

Li et al. [S6] discuss the *Intrusion Defender* security tactic, which involves identifying insecure states in MSA and selecting an appropriate response. The authors recommend four possible responses: *rollback/restart*, *isolation/shutdown*, *diversification*, and *n-variant service scaling*. These responses involve taking specific actions to address insecure states and maintain the security and integrity of the system. Yarygina and Otterstad [S29] developed a real-time system for responding to network attacks using game theory. Game theory is a branch of applied mathematics that provides tools for analyzing situations in which players make interdependent decisions. The approach operates by dynamically altering microservices, such as by removing, restarting, or relocating them. It employs the same defender actions as the approach of Li et al. [S6], along with additional actions for diversification through relocating service to another cloud provider, splitting or merging service instances on a functional code level using dedicated tools, and isolating or shutting down service instances. Chondamrongkul et al. [S40] present a method for conducting automated security analysis for MSA. According to the authors, this approach can recognize security threats and depict attack scenarios while also revealing the potential consequences of these attacks.

c: INTEGRITY PROTECTION

Ahmadvand et al. [S14] define six requirements for maintaining integrity in MSA. Their requirements for integrity protection can be listed as follows:

- Enable authentication and tracing of sensitive data changes by end users
- Protect confidentiality of system secrets in all processes
- Collect unforgeable evidence of insiders' activities in tamper-proof storage
- Detect tampering with static artifacts such as config, script files and binaries
- Raise the bar against program tampering attack (intra-service integrity protection)
- Enable services to attest to the integrity of their recipients and senders (inter-service integrity protection)

d: CIRCUIT BREAKER

This pattern involves proxying requests and rejecting them after a specific number of consecutive failures [2]. It can be used to time out calls in order to prevent system failures, as suggested by Márquez and Astudillo [S23]. Additionally, this pattern can be used to mitigate attacks such as DDoS attacks, as recommended by Yarygina and Bagge [S31]. By enabling partial failures, it allows some microservices

to fail while others remain operational, preventing the entire system from failing.

e: FIREWALLS AND PACKET INSPECTION

Chondamrongkul et al. [S24] emphasize the importance of using a firewall to restrict network traffic flowing from the public to publicly accessible microservices. The use of the *API Gateway* pattern simplifies firewalling in an MSA. According to Soldani et al. [S10], requests from the public must be routed through the gateway before they reach the microservices, making it easier to control access to the microservices.

f: DIVERSIFICATION OF MICROSERVICES

Microservice architectures offer several advantages, one of which is the ability to create a technologically diverse environment. According to Lewis and Fowler [3], this architectural pattern allows for the deployment of each microservice using various programming languages, frameworks, and technologies. Furthermore, inter-service communication in MSA can be effectively managed through the use of proxies or an *API Gateway*. Yarygina and Bagge [S31] advocate for the deployment of heterogeneous microservices to enhance diversity within MSA. This approach helps mitigate attacks such as low-level exploitation and shared code vulnerabilities, which are more prevalent in homogeneous environments. This recommendation aligns with the insights of Otterstad and Yarygina [S34], where the authors developed a security monitor service for identifying anomalies in the system and acting on this, for example, by shutting down infected microservices. Similarly, Torkura et al. [S35] present the concept of *Moving Target Defense (MTD)* to mitigate shared code vulnerabilities. Their concept enables diversification at runtime in order to promote uncertainty in the architecture to increase the effort needed to perform successful attacks against the architecture. This is achieved through their techniques for automatic code generation, where code is transformed into other programming languages at runtime.

g: GENERATION OF ATTACK GRAPHS

Ibrahim et al. [S37] introduce a solution for creating automatic attack graphs as a means to identify network threats within MSA. Their approach utilizes microservices deployed in containers, rather than traditional network nodes, in the construction of attack graphs. Designed for integration into a CI/CD pipeline, their method aims at assisting microservice practitioners in discovering potential attack paths in MSA, to mitigate the risks associated with inter-service pivoting and lateral network movement.

4) MONITORING, TRACING AND LOGGING

Out of the 54 publications, 12 studies (22.2%) focus on the topic of monitoring, tracing, and logging as a means to mitigate security threats in MSA. However, none of these publications demonstrate a particularly high level of focus on these topics.

a: SECURITY HEALTH ENDPOINT

The concept of the *Security Health Endpoint* was introduced by Torkura et al. [S42] as a component of their *Security Gateway* implementation. According to the authors, this concept enhances the monitoring capabilities of MSA, thereby promoting environmental awareness. The *Security Health Endpoint* is based on the *Health Endpoint Monitoring* design pattern, which involves providing an API endpoint for retrieving essential system information about the health of microservices. By adhering to this mitigation concept, relevant security metrics should be accessible from endpoint(s) exposed by the microservice(s), such as an security-health API endpoint. The *Security Health Endpoint* offers various types of information, including, among other things, key insights from the most recent security assessment. This includes details about specific affected components and their related vulnerabilities, as well as metrics like CVEs and *Common Weakness Enumerations (CWE)* associated with each vulnerability. The authors highlight the advantage of automatically consuming this information from other security tools, and provide an example where firewalls can retrieve updated security information from individual services to construct automated firewall rules.

b: TRACING

The *Microservice Observability (OBS)* architectural design decision, as presented in Zdun et al. [S20], encompasses various related security tactics as alternative decision points. In this context, *observance* refers to the enforcement of logging, monitoring, and the execution of request tracing. Kalubowila et al. [S16] propose an external validation solution to reduce latency in MSA. This approach utilizes a trace analyzer that detects errors by monitoring requests as they traverse through various services in MSA, identifying any errors or latency-related issues.

c: MONITORING

A critical aspect of maintaining a microservice architecture is monitoring the system to detect and respond to anomalies [S49]. Ahmadvand et al. [S14] suggest enabling tracing mechanisms for changes to sensitive data within such architectures. They also emphasize the importance of logging insider activities, such as those of DevOps personnel, to improve security. Nkomo and Coetzee [S32] advocate for monitoring the behavior of microservices during runtime to ensure system integrity. Additionally, Sun et al. [S39] introduce their prototype, *FlowTap*, which is designed for network monitoring in microservice environments. The tool monitors the security status of microservices, allowing it to enforce policies over the network traffic seen by the microservice. Further, Zeng et al. [S1] suggest monitoring the containers used for deploying microservices. Li et al. [S6] introduce the *Security Monitor* security tactic to monitor MSA for anomalies that could lead to security issues.

Otterstad and Yarygina [S34] present the *Security Monitor Service*, which is also discussed in Section III-C3.

5) SECURE COMMUNICATION

Mitigation strategies for inter-service security threats in MSA based on secure communication techniques are presented in 11 of the 54 publications studied (20.3%), although the majority discuss the topic with a low level of focus.

a: TLS

Several studies discuss the application of *Transport Layer Security (TLS)* for safeguarding communication between microservices [S3, S32, S2, S43]. Walsh and Manferdelli [S43] stress that microservices communications can be secured through multiple methods, including *TLS*. The article primarily focuses on mutual authentication, but the *TLS* mechanisms mentioned, such as *Baseline Mutual TLS* and *Centralized Attested TLS*, are integral to the secure communication process. In addition, Nkomo and Coetzee [S32] recommend *TLS* as a mitigation strategy against various security risks, including unauthorized access, insecure application programming interfaces, insecure microservice discovery, and insecure message broker. Rezaei Nasab et al. [S2] propose several security best practices to improve the security of microservices communication, based on their evaluation of gray literature and interviews with microservices practitioners. One such practice involves using *TLS* to secure service-to-database communication, ensuring that the connection is encrypted and cannot be intercepted or tampered with. Moreover, Ahmadvand et al. [S14] developed a framework that focuses on integrity protection of MSA. This framework requires services to be able to verify the integrity of other services, ensuring that a service is indeed the entity it claims to be. The use of *TLS* is a key component of this framework, as it provides encryption and authentication mechanisms that help to maintain the integrity of microservice communications.

b: MTLS

Ponce et al. [S7] suggest mitigating the security issue of non-secure inter-service communication through the use of *mTLS* for inter-service communication. The authors recommends the implementation of *mTLS* to ensure bidirectional encryption of traffic, rendering it uninterceptable and unalterable. Similarly, Ponce et al. [S15] advocate for the use of *mTLS* for the same purpose. Miller et al. [S44] also posits that employing *mTLS* for communication between pods (i.e., a group of containers on the same *Kubernetes* host) contributes to securing data in transit.

c: HTTPS AND FTPS

According to Waseem et al. [S25], an *HTTPS enforcement strategy* strategy should be implemented to promote the use of the *HTTPS* protocol over *HTTP* for inter-service communication. This is to ensure that an encrypted connection is maintained between the services, which cannot be achieved using the native *HTTP* protocol alone. Chondamrongkul et al. [S24] suggest using encrypted protocols such as *HTTPS* or *FTPS* for communication in MSA, but emphasize that this should be done exclusively on public networks to prevent data tampering and eavesdropping.

d: API SECURITY

The use of an API rate limiting strategy, as described by Waseem et al. [S25], is essential to prevent adversaries from launching brute-force attacks and making excessive API calls in MSA. Similarly, Genfer and Zdun [S18] conducted a study aimed at identifying and preventing data exposure in microservice APIs by determining the minimum amount of data required for successful exchanges. Their research objective was to establish an optimal data threshold to reduce the risk of data breaches. The authors used a source code detector, developed in a previous research project, to monitor traffic between APIs. According to the authors, data that are neither directly consumed nor routed to another API can be considered excessively exposed, serving as a criterion for identifying data overexposure. By utilizing this criterion and their source code analyzer tool, the authors discovered several instances of overexposure in two open-source microservices projects that they used for validation of their approach. However, their study did not cover all aspects of data exposure.

6) SECURITY POLICIES

Among the 54 publications that were analyzed, 10 of them (18.5%) discuss methods for enhancing security in MSA using security policies and implementing mitigation measures against inter-service security threats. Eight of these publications focus on this topic with a moderate level of attention.

a: SECURITY POLICIES

Torkura et al. [S17] developed a tool called *CAVAS*, which includes a *Security Gateway* module that functions as a *Security Enforcement Point (SEP)*. This module is designed to enforce policies for automated vulnerability assessments of container images and to provide continuous security assessments of microservices, tailored specifically to the microservices' technology stack. The *CAVAS* tool is based on the foundational concepts established by the same authors in their previous research publication [S42]. Li et al. [S6] emphasize the need for security policies for the assessment of certain vulnerabilities and areas of the network. Soldani et al. [S10] state that one of the gains of microservices is the possibility to enforce fine-grained policies, by using hierarchical groups of subsets of microservices.

b: ACCESS CONTROL POLICIES

Several authors have made significant strides in developing tools for fine-grained access control in microservices. Xi et al. [S11] present a distributed access control scheme for microservices cooperation utilizing blockchain

technology, featuring smart contracts for storing policies and a graph-based decision-making scheme for efficient access control. The scheme enhances confidentiality, integrity, and non-repudiation while outperforming traditional access control methods in efficiency, as demonstrated by security analysis and performance evaluations. Nehme et al. [S13] utilize *XACML* to create access control policies to promote fine-grained access control in microservices. Waseem et al. [S25] discusses access control policies in relation to the *Service-level Authorization* pattern and advocates this pattern as it provides more control over the enforcement of access control policies. The authors also state that the pattern covers several API policies, which can be applied in different ways. However, the authors do not note the security implications related to the different ways of applying these policies. Li et al. [S36] implemented a feature in their tool *Jarvis* that uses inter-service interactions extracted from the source code of an MSA to define such policies. Li et al. [S46] also developed *AutoArmor*, a tool that automates the generation of inter-service authorization policies, and has been proven effective with minimal overhead. Another approach was taken by Miller et al. [S44], who developed a workflow that employs the *Sidecar* pattern to enforce policies in line with the zero-trust security model. Furthermore, Sun et al. [S39] introduce *FlowTap*, a tool that enforces policies based on inter-service network traffic.

7) SERVICE MESH, SIDECARS AND API GATEWAYS

The use of mitigation strategies in relation to the *Service Mesh*, *Sidecar*, and *API Gateway* pattern is examined in 9 of the 54 publications (see Figure 7), amounting to 16.7% of the total. However, it should be noted that the majority of these publications only dedicate a low level of focus to the subject. On the other hand, three of the nine publications demonstrate a high degree of focus on the topic.

a: SERVICE MESH

A service mesh is comprised of several key components that together form the infrastructure layer. These components include the *data plane*, also referred to as the *mesh proxy*, and the *control plane* [1]. The *Themis* framework, introduced by Aktypi et al. [S21], was designed to facilitate secure peer-to-peer communication and establish a secure communication network for a service mesh without the need for a central certificate authority. The framework aims to improve security in MSA using a service mesh through a dual-layer architecture. The upper layer of *Themis* supports a peer-to-peer network, while the lower layer offers an alternative to the commonly used *mTLS* protocol for secure inter-service communication and authentication. The framework also includes features for distributed identity management [S21]. As a prototype, *Themis* was released as an open source project on *GitHub*, and its performance overhead was evaluated as minimal, averaging only a throughput overhead of 1.24%. El Malki and Zdun [S38] recommend prioritizing encryption and efficient key management for improving security in service meshes, which can be achieved through either API keys or a central certificate authority in the *control plane*. Additionally, they propose setting up authorization in either the *control plane* or *data plane*. Alboqmi et al. [S48] present a method for securing MSA using self-protection measures in the service mesh. The authors describe self-protection as a means of responding to security threats autonomously and automatically once they are detected. To enable self-protection in the service mesh, the authors overcame previous limitations of static configuration by developing a system that dynamically alters traffic routes and controls information flow based on threat assessments and defined preferences. Sedghpour and Townend [S19] highlight the increasing popularity of using *extended Berkeley Packet Filter (eBPF)* in combination with service meshes in MSA. *eBPF* is considered a lightweight, sandboxed virtual machine embedded within the Linux kernel, where calls are made through kernel hooks. The authors state that this enables security actions to be performed with low overhead.

b: SIDECAR

Suneja et al. [S26] assess various methods to securely implement container fusion. They explore the *Sidecar* pattern as a solution to ensure that container fusion does not increase the overall risk of the system. In their evaluation, they have taken into account the possibility of malicious code within the fused container. To securely attach to a microservice, they have set up a *Sidecar* while limiting access to critical system components such as disk, memory, network states, and resource statistics. The demonstration has shown that their implementation does not compromise the security of the system, and it only results in negligible performance overhead.

c: API GATEWAY AND BACKENDS FOR FRONTENDS

Several studies indicate the positive security attributes that the adoption of the *API Gateway* pattern can bring to MSA [S3, S7, S2, S25]. According to Waseem et al. [S25], the implementation of the *API Gateway* pattern can enhance the security, availability, and portability of MSA. Additionally, Waseem et al. [S25] report that the use of the *Backends for Frontend (BFF)* pattern, similar to the application of the *API Gateway*, can also improve these same security properties. Ponce et al. [S7] recommend adding an *API Gateway* to mitigate the risks associated with having publicly accessible microservices. By enforcing security at this component before the request reaches the microservices, an additional layer of security is added to the architecture, which is not present when microservices are directly exposed to the public environment.

8) INTER-SERVICE AUTHENTICATION AND AUTHORIZATION

Based on the analysis of 54 publications, only 5 of them (9.3%) discuss mitigation strategies related to this topic. While the majority of these publications demonstrate a high

level of focus on the subject matter, the overall number of publications addressing this topic remains relatively small.

a: INTER-SERVICE AUTHENTICATION

According to Walsh and Manferdelli [S43], the current *TLS* standards used for inter-service authentication and attestation (the process of verifying identity & integrity), such as *Baseline Mutual TLS* and *Federated Attested TLS-PSK*, were evaluated based on benchmarking tests. However, the authors did not assess the relative security level of these standards. In conclusion, the authors consider all the assessed protocols to be capable of providing trustworthiness to MSA, although each protocol has a different level of performance overhead. Several studies [S7, S31, S15] advocate for the use of mTLS to ensure authentication at the inter-service level. Similarly, Ponce et al. [S7] advocate the use of *mTLS* to enable authentication between services to mitigate the *Unauthenticated Traffic* security smell, which is also defined in the study. Yarygina and Bagge [S31] also present a prototype for securing and enabling inter-service authentication with *mTLS*, *Public key infrastructure (PKI)* and tokens. The study discovered that the implications of enabling these security measures provided minimal performance overhead.

b: INTER-SERVICE AUTHORIZATION

Li et al. [S36] introduce the *Jarvis* tool, developed for the automated creation of inter-service authorization policies. This tool examines the architecture of microservices prior to deployment, identifying potential communication patterns among services to form corresponding policies. Furthermore, *Jarvis* actively observes the operational behavior of MSA, gathering updated information to refine and adapt these policies accordingly. The tool was developed to mitigate the issues associated with having to manually configure inter-service authorization policies, which could be both a tedious and error-prone manual task, especially considering the flexible architecture of microservices constantly in change.

9) SECURING DATA AT REST

The issue of secure data storage is mentioned in 5 of the analyzed publications, which constitutes 9.3% of the total studies evaluated. However, the level of attention given to this topic is generally limited. Out of this group, only one study is classified as having a moderate level of emphasis on the topic, based on the established criteria. The other studies that briefly address this topic do so with a relatively low level of detail or focus.

a: ENCRYPTION OF DATA AT REST

The importance of encrypting data at rest is a central theme in the study conducted by Mateus-Coelho et al. [S5]. The study emphasizes the use of widely recognized and public cryptographic algorithms. The advantages of these algorithms, such as regular updates with security patches, comprehensive penetration testing, and ongoing scrutiny by security experts, are highlighted in the study. The authors argue that such rigorous maintenance and validation processes ensure the effectiveness and reliability of these algorithms in safeguarding data at rest. In addition, the study by Ponce et al. [S7] also recommends the use of established and widely recognized encryption technologies for encrypting data at rest. This emphasizes the significance of employing proven cryptographic methods that have undergone extensive examination and testing.

b: ENCRYPTING AND MANAGING SECRETS

The significance of encrypting confidential information when it is stored has been highlighted by Ponce et al. [S7], who recommend it as a fundamental security strategy. This approach includes avoiding the storage of secrets in repositories intended for application code and refraining from placing them in environment variables or near the application itself. This measure is critical to safeguard sensitive information and prevent unauthorized access. Billawa et al. [S22] also stress the importance of encrypting data at rest to ensure their confidentiality. Additionally, Waseem et al. [S25] propose a security strategy called *Encrypt and Protect Secrets*, which involves the use of secret vaults like *HashiVault* [57] and *Microsoft Azure Key Vault* [58] for secret storage. The work of Ahmadvand et al. [S14] emphasizes the importance of secure secret management, specifically in differentiating access between development and production environments. The authors advocate a system where developers can access secrets during the development phase, but these secrets are tightly controlled and restricted in production environments. This approach ensures that secrets are not accessed or exposed to unauthorized individuals in production, and are generated and distributed confidentially.

10) SECURITY APPROACHES AND MODELS

As depicted in Figure 7, five of the 54 publications (9.3%) center around security approaches and models that serve to alleviate inter-service security threats in the context of MSA. It is noteworthy that the majority of these publications devote only a low level of focus to this subject.

a: CONCEPT OF LEAST PRIVILEGES

Ponce et al. [S7] recommend to *follow the Least Privilege Principle* as a security refactoring strategy to mitigate the security smell of granting *Unnecessary Privileges to Microservices*, identified by the same authors. Newman [1] describes the *Minimum Privilege Principle* as a way to ensure that microservices are given the minimum privileges required to fulfill their tasks. Billawa et al. [S22] also mention the concept of least privileges as a security best practice that is recommended to adhere to when developing microservices.

b: DEFENSE-IN-DEPTH

The principle of *Defense-in-Depth* is a strategy that involves positioning defensive measures at multiple locations within

MSA [1]. To achieve this, network segmentation, limiting microservice privileges, and enforcing service-level authorization can be employed. This approach is recommended by Ponce et al. [S15] to mitigate the *No Layered Defense* security smell, which occurs when insufficient layers of security defense measures are in place. This strategy is particularly effective in implementing adequate defense measures in the outermost layer of a system, as emphasized by Ponce et al. [S15]. Implementing *Defense-in-Depth* also involves a) positioning every service behind a firewall, b) using an *API Gateway* to enforce security on incoming requests, c) employing *mTLS*, *OAuth2*, and *OpenID Connect* for encryption and access control between services, d) adhering to the *Least Privilege Principle*, e) securing sensitive data through encryption, f) verifying service input for validity, and g) enforcing observability measures such as monitoring and logging information. This pattern has been widely recognized as one of the most effective security measures in MSA. As per Billawa et al. [S22], this approach is considered to be one of the best practices for ensuring the microservice security. Moreover, Waseem et al. [S25] refer to this pattern as the *Layered Defense* design pattern, and state that it enhances the security, confidentiality, and integrity of the system. By implementing this pattern, multiple API layers and gateways are utilized, each enforcing specific authentication and authorization rules.

c: ZERO-TRUST PRINCIPLE

Ponce et al. [S15] recommend applying the *Zero-Trust Principle* to mitigate the *Trust The Network* security smell. The authors state that all their examined studies discussing this security smell agree that it can be mitigated by applying the *Zero-Trust principle*. The authors also associate a set of recommendations with the practice of implementing the *Zero-Trust Principle*, which briefly summarized includes; use *mTLS* between services, the use of *OpenID Connect* to verify identity at the edge, use *OAuth2* to enforce authorization at a service level, and enforce network segmentation. In the study by Miller et al. [S44], the authors follow the principles of zero-trust to define a secure system in which data exchanges between untrusted parties can be performed.

11) CONTAINERIZATION AND ORCHESTRATION

Figure 7 shows that 4 of the 54 analyzed publications (7.4%) discuss mitigative strategies related to containerization and orchestration, with the majority of these devoting a low level of focus to the topic. Nehme et al. [S49] suggest that container firewalls should be used to analyze all requests it receives, whether it is originating from the API gateway or other microservices.

a: MITIGATING VULNERABILITIES IN CONTAINERS

Torkura et al. [S17] explore the challenges in recognizing vulnerabilities within container images in the context of MSA and introduce their prototype called *CAVAS* (see Section III-B2), designed to detect and validate these vulnerabilities while minimizing false positives. *CAVAS* is intended for use in both development and production environments, with the aim of identifying vulnerabilities as early as possible in the software development lifecycle. The authors conducted various validation tests of the tool in a lab environment to assess the tool's ability to detect and validate vulnerabilities, as well as its efficiency in doing so. Additionally, they assessed the effectiveness of the tool's security policies and demonstrated that *CAVAS* is significantly more effective than traditional security testing techniques, showing a 31.4% improvement in the identification of vulnerabilities compared to these conventional methods. The authors also developed and integrated a specialized set of policies into the tool, with the goal of identifying vulnerabilities arising from shared code prior to their deployment in containers. Nkomo and Coetzee [S32] present several protection measures for a list of identified security threats. To mitigate the threats of an insecure runtime infrastructure, the authors list several mitigation strategies. The process of vulnerability scanning should be conducted for containers prior to their deployment to the production environment. This entails the creation and validation of secure configurations for all infrastructure components. Additionally, containers should be grouped based on their *relative sensitivity*, which refers to the potential impact of a malfunction. Billawa et al. [S22] identified the security practice of employing immutable containers as a key best practice in their research. This approach entails restricting updates to containers once they are deployed. Moreover, the authors recommend external data storage separate from the containers to ensure data preservation and easy access if container replacement is necessary.

12) SECURITY PERIMETERS AND NETWORK SEGMENTATION

Figure 7 illustrates that the use of security perimeters and network segmentation to mitigate security threats is discussed in 4 of the 54 analyzed publications (7.4%), with the majority of these studies devoting a low level of focus to the topic.

a: PRIVATE MICROSERVICES

Rezaei Nasab et al. [S2] describe two practices for managing *private microservices*, i.e., microservices that are meant to be accessed only by a specific set of end users and should be isolated from other networks. The first practice states that it should not be possible to verify the existence of a private microservice from an external microservice. The second practice suggests that service grouping should be implemented to restrict communication and visibility to the member services in the group. Both of these security practices were found to be either absolutely useful or useful by the majority of the study's respondents.

b: NETWORK SEGMENTATION

Mateus-Coelho et al. [S5] advocate for network segregation as a countermeasure against security threats in microservices. The authors propose that distributing microservices across various networks or subnets, managed through firewalls or

filtering rules, is an effective strategy to improve security. However, the authors do not provide guidance on how microservices should be effectively segregated to enhance system security. Ponce et al. [S15] further assert that network segmentation should be applied to enforce the *Zero-Trust Principle*, also known as the *Zero-Trust Security Model*. Adherence to this model implies that one assumes that the environment has already been breached. As a result, this assumption implies the need to consider a potential adversary who attempts to eavesdrop on network traffic, establish unauthorized connections, or engage in other malicious activities [1]. However, the authors do not specify how the network should be segmented or how to determine which microservices should be isolated.

c: MICROSERVICES ISOLATION

The concept of isolating microservices is thoroughly discussed in Otterstad and Yarygina [S34]. The authors explore how the isolation of microservices, in conjunction with the divergence of software, can mitigate the exploitation of system layers, referred to as *low-level exploitation* in the publication. The study examines the implementation of automatic microservice isolation when anomalies, which could pose potential security risks, have been detected by a monitoring component. The publication recommends isolating a single node or multiple nodes exhibiting similar anomalies to safeguard the microservice architecture. The authors assert that the combination of automated isolation and software diversity enhances the defense against low-level exploitation.

IV. DISCUSSION

When categorizing the studies according to their primary security focus (properties **P8** and **P10**, Table 7 in the Appendix), we observed that the majority of the analyzed publications presented mitigation strategies rather than delving into descriptions of specific inter-service security threats in MSA. Also, the elaboration of security threats was often on a relatively low level of focus, according to our classification scheme outlined in Table 10 in the Appendix.

A. SECURITY THREATS

In the context of security threats, we already noted in Section III-B1 that numerous publications have examined issues related to the security perimeters and attack surface within MSA. Nevertheless, the majority of these publications have restricted their analysis to a general overview, without delving into in-depth exploration or detailed examination of this topic.

The majority of the studies reviewed in this context concentrated on cloud technologies, such as container and orchestration threats, as delved into in Section III-B2. Microservices, often employed in conjunction with these technologies, present unique security challenges. The analyzed research delves into issues pertaining to both the technological diversity in MSA and the utilization of homogeneous technology stacks. These two concepts, despite seeming contradictory, imply that there is no definitive approach to designing secure microservices. Technological diversity in MSA can result in increased complexity and a more extensive attack surface, especially due to third-party vulnerabilities. For example, deploying 300 distinct microservices using 5 programming languages will likely yield a greater number of libraries and software packages than a monolithic application that employs a single programming language. Conversely, if these 300 microservices are constructed using the same technology stack and are horizontally scaled, the homogeneity of these services could facilitate shared-code vulnerabilities, potentially making every microservice susceptible to the same attacks.

B. MITIGATION STRATEGIES

In this realm, we found that the research was more advanced and provided us with comprehensive information on security measures and threat mitigation. Particularly, we were able to gather valuable insights on topics such as secure code, design patterns, and architecture (Section III-C1), user authentication and authorization (Section III-C2), and infrastructure defense and intrusion mechanisms (Section III-C3) from a significant number of evaluated publications.

For topics addressing mitigation strategies, we have observed more mature and detailed research, such as for the subtopics of diversification of services, as addressed in the works by Otterstad and Yarygina [S34] and Torkura et al. [S35] in Section III-C3. Additionally, we noted that many of the studies relied on conceptual frameworks to derive their presented mitigation strategies. These frameworks span a variety of approaches, including design patterns, security tactics, and mitigation decisions (refer to Section III-C1), reflecting efforts to systematically address and mitigate inter-service security threats.

C. IMPLICATIONS FOR RESEARCHERS AND PRACTITIONERS

For researchers, developing methodologies and tools that can help define precise security perimeters, identify and mitigate unnecessary privileges, and securely configure containers and orchestration platforms may be directions for future research. Our study particularly revealed a dearth of examination of the implications of proposed methods and tools, particularly with respect to their advantages and disadvantages in practical scenarios. Additionally, further advancements in monitoring and intrusion detection mechanisms are required, which are specifically tailored to the polyglot nature of microservices.

Practitioners must adopt a comprehensive approach that goes beyond traditional security measures in order to effectively protect MSA against security threats. By implementing zero-trust architectures, which minimize trust assumptions within and across services, and by employing fine-grained access control mechanisms, they can significantly enhance the security posture of microservice-based systems. Furthermore, it is crucial to securely configure and manage

container and orchestration tools, as these play a critical role in MSA and can introduce significant security risks if not properly managed. Further, dedicated training with respect to DevSecOps practices to integrate security measures throughout the development lifecycle, utilizing secure design patterns, and deploying sophisticated authentication and authorization mechanisms is highly advisable.

V. THREATS TO VALIDITY

According to the taxonomy developed by Wohlin et al. [59], there are possible threats to *external*, *internal*, *construct* and *conclusion* validity of our study.

A. EXTERNAL VALIDITY

External validity concerns the broader applicability of a set of results, particularly in situations that differ from the specific context in which they were obtained [59].

The classification schemes outlined in this SLR, specifically those presented in Table 8 to 10 in the Appendix, were developed using the retrieved papers. However, it is possible that future studies may not fit to these classification schemes, which could impact the external validity of these schemes.

B. INTERNAL VALIDITY

As per Wohlin et al. [59], internal validity pertains to the legitimacy of the methods employed to study and examine data, taking into account any biases present. Three steps of our used research method might particularly affect the study's internal validity.

In the paper selection process (see Section II-D), the decision of whether or not to include a paper can be subjective and influenced by the personal views of the authors. To address this potential selection bias, we employed an online tool for managing systematic reviews (https://www.rayyan.ai) to document the individual authors' decisions regarding inclusion or exclusion of each retrieved paper. The decisions were documented at the level of the individual inclusion criteria (**I1-I3**) or exclusion criteria (**E1-E8**). During the first round, each author used the tool to document their decisions independently. For the second round, if all authors agreed on the same inclusion or exclusion criteria for a paper, we have taken over this decision without further discussion. However, if there were discrepancies in the authors' individual decisions whether to include or exclude a paper, all authors participated in a joint discussion to reconcile their differences and arrive at a consensus decision, which was then recorded in the tool.

When assessing the quality of the studies (see Section II-E), subjective judgments of the authors may introduce biases with regards to a study's quality. This particularly pertains to identifying biases in the studies, gauging the clarity and relevance of findings and the appropriateness of sample sizes in solution proposals. To address this issue, the first and third authors collaborated in assessing the quality of each paper, while the second author independently verified a random sample of both included and excluded studies to ensure unbiased and consistent quality assessments. In cases of inconsistencies, all authors engaged in discussions to reconcile different judgments regarding a study's quality.

The classification of papers into predefined categories (experience paper, evaluation research, etc.) and the assignment of papers to specific contribution types (model, method, etc.) (see Section II-F1) may rely on subjective judgments of the authors. To mitigate this potential bias, a standardized data extraction form (see Table 7) was created and used consistently to extract and analyze data in a manner that would allow to answer the research questions. Moreover, to ensure the accuracy of the extracted data, a random subset of the data was cross-checked by the first author. Additionally, to reconcile any inconsistencies in the extracted data, the first and second author engaged in regular discussions with the third author, who carried out a substantial portion of the data extraction process.

C. CONSTRUCT VALIDITY

Construct validity pertains to extending the findings of a study to the underlying theory or concept. Various threats may arise from the design of the experiment itself or from social factors (e.g., researchers' biases) [59]. We particularly regard threats to construct validity in the search method (see Section II-B1) used for this SLR.

One of the potential threats in this context is the possibility of missing or excluding relevant papers. To mitigate this threat, we used five popular digital libraries to retrieve relevant papers, as discussed in Section II-B3. In addition, we employed three strategies to mitigate any potential threats in the search strategy. First, we developed our search string based on the *PICO* scheme, as suggested by Petticrew and Roberts [37]. Second, we improved our search string iteratively based on the results of the pilot search and tested it carefully before executing it to search for the papers retrieved for this review. Third, we employed a backward snowballing step, adhering to the guidelines of Wohlin [33], to identify as many related papers as possible by manually searching the references of selected papers.

D. CONCLUSION VALIDITY

The validity of a study's conclusion refers to the extent to which the results can be used to make accurate conclusions [59]. It is crucial to correctly interpret the data that have been extracted and to recognize potential threats, such as the researchers' biases, which could undermine the validity of the conclusions that are drawn from the data. We regard particular threats to conclusion validity when synthesizing the extracted data (see Section II-E).

The process of coding properties **P8** to **P11** (see Table 7 in the Appendix) and categorizing studies as related to security threats or mitigation strategies was prone to individual interpretation, which necessitated a predefined list of recurring themes that emerged during our initial review. This list was agreed upon by all authors prior to the synthesis and served as a guide for our thematic analysis. Each paper was read

by at least two authors prior to its analysis. Subsequently, each author independently coded a subset of papers, followed by group discussions with the other authors to reconcile any differences in coding and interpretation. This collaborative approach promoted a unified understanding and consistent application of the coding schema.

The creation of new categories for topics not initially anticipated may have been influenced by the researchers' familiarity with the domain or their subjective perception of the importance of certain themes. As such, there is a risk of overemphasis on specific areas. The third author was primarily responsible for the categorization and the creation of new categories. To ensure that the categorization was accurate and comprehensive, the following process was organized: if new categories needed to be added or existing ones refined, the third author recorded this information. The authors then engaged in iterative discussions to reconcile the categorization, particularly to revise the new categories and the papers that should be assigned to them. These discussions continued until all authors agreed on a common categorization, i.e. the refinement of existing categories or the addition of new ones.

During the coding process, we also recognized that the studies have a diverse focus on the identified categories from the background of inter-service communication. To address this disparity in focus, we utilized an ordinal scale, with levels of attention classified as low, medium, or high. However, there may be potential for interpretation bias in the use of such a subjective scale. To mitigate this bias, we have defined explicit criteria for each level of focus (see Table 10 in the Appendix). These criteria were refined and discussed among the authors to ensure consensus.

VI. CONCLUSION

Microservice architectures pursue a modular approach to developing large software systems, but they also introduce particular complexities related to inter-service communication, which requires the adoption of suitable security mechanisms to ensure resilience. The field of inter-service security presents distinct challenges for developers and security professionals who are accustomed to monolithic architectures. As a result, they may encounter difficulties in understanding the specific security threats associated with this architectural style and in identifying effective strategies for mitigating and reducing these risks. In this paper we have explored the intricate landscape of security threats and mitigation strategies for inter-service communication in microservice architectures. Through a systematic literature review encompassing 54 publications, we have analyzed the current state and trajectory of research in this field, emphasizing the prevailing focus of existing studies on contributing methods, models, and guidelines. In the following, we summarize our main conclusions:

(a) The majority of analyzed studies primarily focus on presenting methods, models, and guidelines, with a considerable number involving research with validation and evaluation. We interpret this as a commitment to rigorously examining the effectiveness of proposed security solutions, ensuring they are both innovative and practically viable for microservice architectures.
(b) The number of publications in this field has increased since 2015, with a slight decline in 2020, likely due to the impact of the Corona pandemic. Of these publications, conference papers make up the largest proportion (33 conference papers, 18 journal articles, 3 workshop papers), demonstrating the diverse range of platforms used to disseminate research in this area.
(c) A significant portion of the reported security threats are associated with the security perimeters and attack surface of microservice architectures, as well as aspects of containerization and orchestration. Another recurrent theme across is the inadequate implementation of monitoring and intrusion detection techniques in this domain, highlighting a critical area of vulnerability.
(d) There appears to be a general shortfall in the comprehensive analysis of security threats within the reviewed literature, with the overall level of focus on reported security threats being relatively low. This is especially true in the realms of inter-service authentication and authorization, and inter-service communication, where in-depth exploration is particularly scarce.
(e) We have noticed a greater emphasis on mitigation strategies compared to studies that concentrate on security threats. In this sense, a significant number of publications have been dedicated to discussing topics such as infrastructure defense, monitoring, tracing and logging, secure coding, design patterns, and architecture in detail. Each of these topics has received considerable attention and focus. The majority of studies can be classified as validation research and solution proposals, with their primary contribution being the development of methods.
(f) In the frame of mitigation strategies, we have identified advanced research on more complex methods for modifying microservice architectures at run-time to enhance their resilience against attacks. The research observed in the realm of security threats in inter-service communication can be characterized as less sophisticated.

Future studies could greatly benefit from conceptualizing the identified security threats and their corresponding mitigation strategies in relation to one another, in a manner similar to the categorization of security smells presented by Ponce et al. [27]. Establishing a clear connection between security issues and mitigation strategies to address them could offer valuable guidance, particularly for practitioners. Further investigation could involve the use of empirical methods, such as user studies, focus groups, or expert interviews, to assess the challenges associated with the

TABLE 6. Studies selected for the review.

#	Title	Authors	Venue	Year
S1	Full-stack vulnerability analysis of the cloud-native platform	Zeng et al.	Computers & Security	2023
S2	An empirical study of security practices for microservices systems	Rezaei Nasab *et al.*	*Journal of Systems and Software*	2023
S3	Security in microservice-based systems: A Multivocal literature review	Pereira-Vale *et al.*	*Computers & Security*	2021
S4	SoK: Run-time security for cloud microservices. Are we there yet?	Minna and Massacci	*Computers & Security*	2023
S5	Security in Microservices Architectures	Mateus-Coelho *et al.*	*Procedia Computer Science*	2021
S6	Understanding and addressing quality attributes of microservices architecture: A Systematic literature review	Li *et al.*	*Information and Software Technology*	2021
S7	Smells and refactorings for microservices security: A multivocal literature review	Ponce *et al.*	*Journal of Systems and Software*	2022
S8	Securing microservices and microservice architectures: A systematic mapping study	Hannousse and Yahiouche	*Computer Science Review*	2021
S9	A Systematic Mapping Study on Microservices Architecture in DevOps	Waseem *et al.*	*Journal of Systems and Software*	2020
S10	The pains and gains of microservices: A Systematic grey literature review	Soldani *et al.*	*Journal of Systems and Software*	2018
S11	Decentralized access control for secure microservices cooperation with blockchain	Xi *et al.*	*ISA Transactions*	2023
S12	Promises and challenges of microservices: an exploratory study	Wang *et al.*	*Empirical Software Engineering*	2021
S13	Fine-Grained Access Control for Microservices	Nehme *et al.*	*International Symposium on Foundations and Practice of Security*	2019
S14	Integrity Protection Against Insiders in Microservice-Based Infrastructures: From Threats to a Security Framework	Ahmadvand *et al.*	*Software Technologies: Applications and Foundations*	2018
S15	Microservices Security: Bad vs. Good Practices	Ponce *et al.*	*17th European Conference on Software Architecture. Tracks and Workshops*	2023
S16	Optimization of Microservices Security	Kalubowila *et al.*	*2021 3rd International Conference on Advancements in Computing*	2021
S17	CAVAS: Neutralizing Application and Container Security Vulnerabilities in the Cloud Native Era	Torkura *et al.*	*International Conference on Security and Privacy in Communication Systems*	2018
S18	Avoiding Excessive Data Exposure Through Microservice APIs	Genfer and Zdun	*16th European Conference on Software Architecture*	2022
S19	Service Mesh and eBPF-Powered Microservices: A Survey and Future Directions	Sedghpour and Townend	*2022 IEEE International Conference on Service-Oriented System Engineering*	2022
S20	Microservice Security Metrics for Secure Communication, Identity Management, and Observability	Zdun *et al.*	*ACM Transactions on Software Engineering and Methodology*	2023
S21	Themis: A Secure Decentralized Framework for Microservice Interaction in Serverless Computing	Aktypi *et al.*	*17th International Conference on Availability, Reliability and Security*	2022
S22	SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices	Billawa *et al.*	*17th International Conference on Availability, Reliability and Security*	2022
S23	Identifying Availability Tactics to Support Security Architectural Design of Microservice-Based Systems	Márquez and Astudillo	*13th European Conference on Software Architecture*	2019
S24	Formal security analysis for software architecture design: An expressive framework to emerging architectural styles	Chondamrongkul *et al.*	*Science of Computer Programming*	2021
S25	Decision Models for Selecting Patterns and Strategies in Microservices Systems and their Evaluation by Practitioners	Waseem *et al.*	*2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice*	2022
S26	Can Container Fusion Be Securely Achieved?	Suneja *et al.*	*5th International Workshop on Container Technologies and Container Clouds*	2019
S27	Design, monitoring, and testing of microservices systems: The practitioners' perspective	Waseem *et al.*	*Journal of Systems and Software*	2021
S28	MisMesh: Security Issues and Challenges in Service Meshes	Hahn *et al.*	*International Conference on Security and Privacy in Communication Systems*	2020
S29	A Game of Microservices: Automated Intrusion Response	Yarygina and Otterstad	*Distributed Applications and Interoperable Systems*	2018
S30	Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping	Pereira-Vale *et al.*	*2019 XLV Latin American Computing Conference*	2019

TABLE 6. *(Continued.)* **Studies selected for the review.**

S31	Overcoming Security Challenges in Microservice Architectures	Yarygina and Bagge	*2018 IEEE Symposium on Service-Oriented System Engineering*	2018
S32	Software Development Activities for Secure Microservices	Nkomo and Coetzee	*International Conference on Computational Science and Its Applications*	2019
S33	Towards Resolving Security Smells in Microservice-Based Applications	Ponce	*Advances in Service-Oriented and Cloud Computing*	2021
S34	Low-Level Exploitation Mitigation by Diverse Microservices	Otterstad and Yarygina	*European Conference on Service-Oriented and Cloud Computing*	2017
S35	A Cyber Risk Based Moving Target Defense Mechanism for Microservice Architectures	Torkura *et al.*	*2018 IEEE Intl Conf on Parallel & Distributed Processing*	2018
S36	Towards Automated Inter-Service Authorization for Microservice Applications	Li *et al.*	*ACM SIGCOMM 2019 Conference Posters and Demos*	2019
S37	Attack graph generation for microservice architecture	Ibrahim *et al.*	*34th ACM/SIGAPP Symposium on Applied Computing*	2019
S38	Guiding Architectural Decision Making on Service Mesh Based Microservice Architectures	El Malki and Zdun	*13th European Conference on Software Architecture*	2019
S39	Security-as-a-Service for Microservices-Based Cloud Applications	Sun *et al.*	*2015 IEEE 7th International Conference on Cloud Computing Technology and Science*	2015
S40	Automated Security Analysis for Microservice Architecture	Chondamrongkul *et al.*	*2020 IEEE International Conference on Software Architecture Companion*	2020
S41	Guiding Architectural Decision Making on Quality Aspects in Microservice APIs	Zdun *et al.*	*International Conference on Service-Oriented Computing*	2018
S42	Integrating Continuous Security Assessments in Microservices and Cloud Native Applications	Torkura *et al.*	*10th International Conference on Utility and Cloud Computing*	2017
S43	Mechanisms for Mutual Attested Microservice Communication	Walsh and Manferdelli	*10th International Conference on Utility and Cloud Computing*	2017
S44	Towards Secure and Leak-Free Workflows Using Microservice Isolation	Miller *et al.*	*2021 IEEE 22nd International Conference on High Performance Switching and Routing*	2021
S45	Interface Responsibility Patterns: Processing Resources and Operation Responsibilities	Zimmermann *et al.*	*European Conference on Pattern Languages of Programs 2020*	2020
S46	Automatic policy generation for inter-service access control of microservices	Li *et al.*	*30th USENIX Security Symposium, USENIX Security 2021*	2021
S47	Automatic extraction of security-rich dataflow diagrams for microservice applications written in Java	Schneider and Scandariato	*Journal of Systems and Software*	2023
S48	Toward Enabling Self-Protection in the Service Mesh of the Microservice Architecture	Alboqmi *et al.*	*2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems*	2022
S49	Securing Microservices	Nehme *et al.*	*IT Professional*	2019
S50	Requirements Reconciliation for Scalable and Secure Microservice (De)composition	Ahmadvand and Ibrahim	*2016 IEEE 24th International Requirements Engineering Conference Workshops*	2016
S51	Microservice security: a systematic literature review	Berardi *et al.*	*PeerJ Computer Science*	2022
S52	Threat Intelligence Sharing Component in the Service Mesh Architecture	Alboqmi *et al.*	*2023 10th International Conference on Future Internet of Things and Cloud*	2023
S53	A Runtime Trust Evaluation Mechanism in the Service Mesh Architecture	Alboqmi *et al.*	*2023 10th International Conference on Future Internet of Things and Cloud*	2023
S54	A Systematic Mapping Study in Microservice Architecture	Alshuqayran *et al.*	*2016 IEEE 9th International Conference on Service-Oriented Computing and Applications*	2016

practical implementation of the mitigation strategies identified in our study. Additionally, it would be beneficial to evaluate the identified security threats using predefined criteria, such as their potential risk to organizations and the effectiveness, complexity, and potential side effects of the corresponding mitigation strategies. Complementary, future research should focus on the development of methodologies and tools that define precise security perimeters, eliminate unnecessary privileges, and securely configure containers and orchestration platforms. Our study found a lack of research on the practical implications of proposed methods and tools, including their advantages and disadvantages. Additionally, improved monitoring and intrusion detection mechanisms are needed to address the unique challenges of polyglot microservices.

We also discovered a deficiency in the existing literature regarding the used terminology in the general context of microservice security. It became evident that different authors classify certain concepts differently, with some referring to them as "design patterns" while others categorize them as "security tactics". To address this issue, we advocate a taxonomy that brings together all the relevant tactics, design

TABLE 7. Properties extracted from the studies and related research questions.

#	Property	Description	RQs
P1	Title	Title of the paper.	RQ1
P2	Author(s)	Author(s) of the paper.	RQ1
P3	Year	Year of the paper.	RQ1
P4	Publication type	Type of the paper, i.e., journal article or conference paper.	RQ1
P5	Venue	Name of the publication venue.	RQ1
P6	Research type	Type of research presented in paper (see Table 8).	RQ1
P7	Contribution type	Type of contribution presented in paper (see Table 9).	RQ1
P8	Security threat(s)	Security threat(s) addressed in the paper.	RQ2
P9	Level of focus on security threat(s)	Level of focus that security threat(s) are addressed (see Table 10).	RQ2
P10	Mitigation strategy/strategies	Mitigation strategies addressed in the paper.	RQ3
P11	Level of focus on mitigation strategy/strategies	Level of focus that mitigation strategies are addressed (see Table 10).	RQ3

TABLE 8. Research types, adapted from Petersen et al. [39] and Wieringa et al. [40].

Research Types	Description
Experience Paper	A list of lessons learned from one or more projects. The study reflects the personal experience of the authors. The study often describes the use of tools/techniques in practice, with less focus on research methodology.
Evaluation Research	Techniques are applied in practical settings, along with an examination of the implications of its application in terms of advantages and disadvantages. Additionally, this involves identifying issues prevalent in industry. Examples of this type of research are industrial case studies, controlled experiments with practitioners, practitioner targeted surveys, as well as studies conducting action research or ethnography.
Validation Research	The techniques explored are novel and have yet to be implemented in practical application. Examples of this type of research are simulation of a technique as an empirical method, laboratory experiments, prototyping, mathematical analysis and proofs, and academic case study, e.g., involving students.
Philosophical Paper	Sketches a new way of looking at something, for example, by forming a taxonomy or conceptual framework.
Solution Proposal	Proposes a solution for a problem but does not completely validate it. The proposal of the solution is emphasized with a small proof of concept, such as an example or argumentation.
Opinion Paper	Personal opinion of the author on whether something is good or bad, such as a technique.

TABLE 9. Contribution types, adapted from Shaw [41] and Paternoster et al. [42].

Contribution Type	Description
Model	An abstraction of the authors observed in reality using conceptualization.
Method	Method or approach for dealing with inter-service security threats or applying mitigation strategies in microservice architectures.
Theory	Explanation for why certain results occur, based on cause-effect relationships.
Framework	Combination of multiple methods. Specifies conditions in which the use of methods is applicable, including the required input parameters and possible outcome.
Guideline	List of advices based on research results.
Lessons Learned	List of outcomes based on research results (including recommendations).
Advice	Generic recommendations based on personal opinions.
Tool	Technology or software used for addressing security threats and implementing mitigation strategies in microservice architectures.

TABLE 10. Levels of focus that studies take on security threats or mitigation strategies.

Level	Description
Low	The given content includes a limited amount of information on the topic, with only few sentences delving into the subject matter.
Medium	Substantial content, which goes beyond the surface, is related to the given topic and provides a richer understanding of it.
High	A comprehensive examination of the topic, offering insights beyond basic information, designed for a knowledgeable target audience.

patterns, methods, and other related concepts in the field. This taxonomy could provide much-needed clarity and guidance on best practices and principles to follow when developing secure microservices.

TABLE 11. Research and contribution types of the studies.

#	Research Type	Contribution Type	Focus
S1	Evaluation Research	Method	Security Threats
S2	Evaluation Research	Guideline	Mitigation Strategies
S3	Validation Research	Guideline	Security Threats
S4	Philosophical Paper	Guideline	Security Threats
S5	Opinion Paper	Advice	Mitigation Strategies
S6	Philosophical Paper	Model	Mitigation Strategies
S7	Validation Research	Framework	Security Threats
S8	Validation Research	Guideline	Security Threats
S9	Philosophical Paper	Model	Mitigation Strategies
S10	Philosophical Paper	Theory	Security Threats
S11	Validation Research	Framework	Mitigation Strategies
S12	Philosophical Paper	Lessons Learned	Security Threats
S13	Validation Research	Tool	Mitigation Strategies
S14	Solution Proposal	Framework	Mitigation Strategies
S15	Validation Research	Guideline	Mitigation Strategies
S16	Solution Proposal	Method	Mitigation Strategies
S17	Validation Research	Tool	Mitigation Strategies
S18	Validation Research	Method	Mitigation Strategies
S19	Philosophical Paper	Theory	Mitigation Strategies
S20	Evaluation Research	Method	Mitigation Strategies
S21	Validation Research	Tool	Mitigation Strategies
S22	Validation Research	Theory	Mitigation Strategies
S23	Validation Research	Model	Security Threats
S24	Validation Research	Tool	Mitigation Strategies
S25	Validation Research	Guideline	Mitigation Strategies
S26	Solution Proposal	Method	Mitigation Strategies
S27	Evaluation Research	Lessons Learned	Security Threats
S28	Validation Research	Lessons Learned	Security Threats
S29	Solution Proposal	Method	Security Threats
S30	Philosophical Paper	Model	Mitigation Strategies
S31	Philosophical Paper	Framework	Mitigation Strategies
S32	Validation Research	Guideline	Mitigation Strategies
S33	Validation Research	Tool	Mitigation Strategies
S34	Solution Proposal	Method	Mitigation Strategies
S35	Validation Research	Method	Mitigation Strategies
S36	Solution Proposal	Method	Mitigation Strategies
S37	Solution Proposal	Method	Mitigation Strategies
S38	Validation Research	Model	Mitigation Strategies
S39	Validation Research	Method	Mitigation Strategies
S40	Solution Proposal	Method	Mitigation Strategies
S41	Validation Research	Model	Mitigation Strategies
S42	Solution Proposal	Method	Mitigation Strategies
S43	Solution Proposal	Method	Mitigation Strategies
S44	Solution Proposal	Method	Mitigation Strategies
S45	Philosophical Paper	Model	Mitigation Strategies
S46	Validation Research	Tool	Mitigation Strategies
S47	Validation Research	Method	Mitigation Strategies
S48	Validation Research	Method	Mitigation Strategies
S49	Philosophical Paper	Theory	Security Threats
S50	Solution Proposal	Method	Mitigation Strategies
S51	Philosophical Paper	Theory	Security Threats
S52	Validation Research	Tool	Mitigation Strategies
S53	Validation Research	Tool	Mitigation Strategies
S54	Philosophical Paper	Model	Security Threats

TABLE 12. Categorization of the studies regarding described security threats and their level of focus on it.

Threat Category	Level of Focus		
	Low	Medium	High
Security Perimeters and Attack Surface	[S10, S8, S49, S22, S24, S46, S44, S42, S35, S39, S36]	[S7, S32, S15]	
Container and Orchestration	[S8, S31, S49, S22, S46, S35, S6, S37]	[S4]	[S17, S1]
Insufficient Monitoring and Intrusion Mechanisms	[S54, S3, S49, S22, S32, S39, S15]	[S23, S29, S30]	
Network Attacks	[S20, S7, S51, S31, S24, S42, S1, S23, S26]		
Configuration, Infrastructure and Deployment	[S31, S14, S22, S42, S17]	[S8, S7]	
Service Mesh and Sidecar	[S26, S21, S38, S48]		[S1, S28]
Software Implementation and Dependencies	[S54, S3, S27, S50]	[S35]	
Data Leakage and Exposure	[S2]	[S7, S18, S41]	
Insecure Inter-Service Communication	[S3, S7, S27, S12]		
Other	[S51, S46, S32]	[S14]	
User Authentication and Authorization	[S49]	[S10, S13]	[S7]
Inter-Service Authentication and Authorization	[S7, S43]		

TABLE 13. Categorization of the studies regarding described mitigation strategies and their level of focus on it.

Mitigation Category	Level of Focus		
	Low	Medium	High
Secure Code, Design Patterns and Architecture	[S7, S9, S45, S33]	[S20, S22, S32, S15, S23, S41, S25]	[S50, S47]
User Authentication and Authorization	[S3, S22, S6]	[S20, S7, S31, S14, S15, S41, S25]	[S11, S2, S13]
Infrastructure Defense and Intrusion Mechanisms	[S10, S24, S6, S23]	[S31, S2]	[S14, S35, S37, S29, S40, S34]
Monitoring, Tracing and Logging	[S14, S49, S32, S16]	[S20, S42, S39, S17, S6, S1, S34]	[S52]
Secure Communication	[S7, S14, S24, S44, S32, S15, S2, S25]	[S43]	[S18, S53]
Security Policies	[S10, S6]	[S11, S46, S44, S42, S39, S36, S13, S25]	[S17]
Service Mesh, Sidecar and API Gateway	[S3, S7, S2, S25, S19]	[S26]	[S21, S38, S48]
Inter-Service Authentication and Authorization	[S7, S15]		[S31, S36, S43]
Securing Data at Rest	[S14, S22, S25, S5]	[S7]	
Security Approaches and Models	[S7, S22, S25]	[S44]	[S15]
Containerization and Orchestration	[S49, S22, S32]		[S17]
Security Perimeters and Network Segmentation	[S15, S2, S5]	[S34]	

APPENDIX

SELECTED STUDIES

See Table 6.

CLASSIFICATION SCHEMES

See Tables 7–10.

CLASSIFICATION OF THE STUDIES

See Table 11.

CATEGORIZATION OF SECURITY THREATS AND MITIGATION STRATEGIES

See Tables 12 and 13.

REFERENCES

[1] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, CA, USA: O'Reilly Media, 2021.

[2] C. Richardson, *Microservices Patterns: With Examples in Java*. Shelter Island, NY, USA: Manning, 2019.

[3] J. Lewis and M. Fowler. (2014). *Microservices*. Accessed: May 25, 2024. [Online]. Available: https://Martinfowler.com/articles/microservices.html

[4] E. Wolff, *Microservices: Flexible Software Architecture*, 1st ed. London, U.K.: Pearson, 2021.

[5] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Cham, Switzerland: Springer, 2017, pp. 195–216, doi: 10.1007/978-3-319-67425-4_12.

[6] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis, "Microservices in practice, part 1: Reality check and service design," *IEEE Softw.*, vol. 34, no. 1, pp. 91–98, Jan. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7819415

[7] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proc. IEEE 9th Int. Conf. Service-Oriented Comput. Appl. (SOCA)*. Macau, China: IEEE, Nov. 2016, pp. 44–51. [Online]. Available: http://ieeexplore.ieee.org/document/7796008/

[8] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Softw.*, vol. 35, no. 3, pp. 24–35, May 2018.

[9] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice* (SEI Series in Software Engineering), 4th ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2021.

[10] L. Calcote and Z. Butcher, *ISTIO: Up and Running: Using a Service Mesh to Connect, Secure, Control, and Observe*. Sebastopol, CA, USA: O'Reilly Media, 2019.

[11] R. S. de O. Júnior, R. C. A. da Silva, M. S. Santos, D. W. Albuquerque, H. O. Almeida, and D. F. S. Santos, "An extensible and secure architecture based on microservices," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2022, pp. 1–2. [Online]. Available: https://ieeexplore.ieee.org/document/9730757

[12] P. Siriwardena and N. Dias, *Microservices Security in Action*. Shelter Island, NY, USA: Manning, 2020.

[13] U. Zdun, P.-J. Queval, G. Simhandl, R. Scandariato, S. Chakravarty, M. Jelic, and A. Jovanovic, "Microservice security metrics for secure communication, identity management, and observability," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 1, p. 16, Feb. 2023, doi: 10.1145/3532183.

[14] F. Al-Doghman, N. Moustafa, I. Khalil, N. Sohrabi, Z. Tari, and A. Y. Zomaya, "AI-enabled secure microservices in edge computing: Opportunities and challenges," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1485–1504, Mar. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9723563

[15] M. D. Hossain, T. Sultana, S. Akhter, M. I. Hossain, N. T. Thu, L. N. T. Huynh, G.-W. Lee, and E.-N. Huh, "The role of microservice approach in edge computing: Opportunities, challenges, and research directions," *ICT Exp.*, vol. 9, no. 6, pp. 1162–1182, Dec. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959523000760

[16] A. Kollu, K. K. Chennam, and D. Mahajan, "An empirical review on secure edge computing architecture," in *Proc. 2nd Int. Conf. Cogn. Intell. Comput.*, in Cognitive Science and Technology, A. Kumar, G. Ghinea, and S. Merugu, Eds. Singapore: Springer, 2023, pp. 661–668.

[17] A. H. Sodhro, A. Lakhan, S. Pirbhulal, T. M. Groenli, and H. Abie, "A lightweight security scheme for failure detection in microservices IoT-edge networks," in *Sensing Technology* (Lecture Notes in Electrical Engineering), vol. 886. Cham, Switzerland: Springer, 2022, pp. 397–409. [Online]. Available: https://link.springer.com/10.1007/978-3-030-98886-931

[18] C. Meadows, S. Hounsinou, T. Wood, and G. Bloom, "Sidecar-based path-aware security for microservices," in *Proc. 28th ACM Symp. Access Control Models Technol.* New York, NY, USA: Association for Computing Machinery, May 2023, pp. 157–162, doi: 10.1145/3589608.3594742.

[19] The Linux Foundation. (2024). *Production-Grade Container Orchestration*. Accessed: Feb. 8, 2024. [Online]. Available: https://kubernetes.io/

[20] Docker Inc. (2024). *Docker*. Accessed: May 25, 2024. [Online]. Available: https://www.docker.com/

[21] F. Dewanta, "Secure microservices deployment for fog computing services in a remote office," in *Proc. 3rd Int. Conf. Inf. Commun. Technol. (ICOIACT)*, Nov. 2020, pp. 425–430. [Online]. Available: https://ieeexplore.ieee.org/document/9332025

[22] H. Yu, X. Wang, C. Xing, and B. Xu, "A microservice resilience deployment mechanism based on diversity," *Secur. Commun. Netw.*, vol. 2022, pp. 1–13, Jun. 2022, doi: 10.1155/2022/7146716.

[23] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of microservices-enabled fog applications," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 22, Nov. 2019, Art. no. e4436, doi: 10.1002/cpe.4436.

[24] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *J. Syst. Softw.*, vol. 146, pp. 215–232, Dec. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121218302139

[25] A. Pereira-Vale, E. B. Fernandez, R. Monge, H. Astudillo, and G. Márquez, "Security in microservice-based systems: A multivocal literature review," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102200. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404821000249

[26] A. Hannousse and S. Yahiouche, "Securing microservices and microservice architectures: A systematic mapping study," *Comput. Sci. Rev.*, vol. 41, Aug. 2021, Art. no. 100415. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013721000551

[27] F. Ponce, J. Soldani, H. Astudillo, and A. Brogi, "Smells and refactorings for microservices security: A multivocal literature review," *J. Syst. Softw.*, vol. 192, Oct. 2022, Art. no. 111393. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016412122200111X

[28] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, and M. Prandini, "Microservice security: A systematic literature review," *PeerJ Comput. Sci.*, vol. 8, p. e779, Jan. 2022, doi: 10.7717/peerj-cs.779. [Online]. Available: https://peerj.com/articles/cs-779

[29] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Softw. Eng. Group, School Comput. Sci. Math., Keele Univ., Keele, U.K., Tech. Rep. EBSE-2007-01, 2007, vol. 2.

[30] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950584908001390

[31] M. J. Page et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *Systematic Rev.*, vol. 10, no. 1, p. 89, Dec. 2021, doi: 10.1186/s13643-021-01626-4.

[32] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 625–637, Jun. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584910002260

[33] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.* New York, NY, USA: Association for Computing Machinery, May 2014, pp. 1–10, doi: 10.1145/2601248.2601268.

[34] C. Wohlin, P. Runeson, P. A. da Mota Silveira Neto, E. Engström, I. do Carmo Machado, and E. S. de Almeida, "On the reliability of mapping studies in software engineering," *J. Syst. Softw.*, vol. 86, no. 10, pp. 2594–2610, Oct. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121213001234

[35] W. Afzal, S. Alone, K. Glocksien, and R. Torkar, "Software test process improvement approaches: A systematic literature review and an industrial case study," *J. Syst. Softw.*, vol. 111, pp. 1–33, Jan. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121215001910

[36] T. Kosar, S. Bohra, and M. Mernik, "A systematic mapping study driven by the margin of error," *J. Syst. Softw.*, vol. 144, pp. 439–449, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218301353

[37] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences: A Practical Guide*, 1st ed. Oxford, U.K.: Wiley, Dec. 2005.

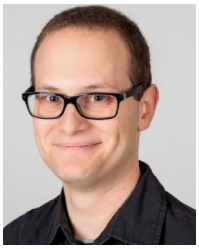
[38] E. Mourão, J. F. Pimentel, L. Murta, M. Kalinowski, E. Mendes, and C. Wohlin, "On the performance of hybrid search strategies for systematic literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 123, Jul. 2020, Art. no. 106294. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920300446

[39] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950584915000646

[40] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requirements Eng.*, vol. 11, no. 1, pp. 102–107, Mar. 2006, doi: 10.1007/s00766-005-0021-6.

[41] M. Shaw, "Writing good software engineering research papers," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 726–736. [Online]. Available: http://ieeexplore.ieee.org/document/1201262/
[42] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software development in startup companies: A systematic mapping study," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1200–1218, Oct. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914000950
[43] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2011, pp. 275–284. [Online]. Available: https://ieeexplore.ieee.org/document/6092576
[44] The Linux Foundation. (2024). *Containerd*. Accessed: May 25, 2024. [Online]. Available: https://containerd.io/
[45] Open Container Initiative. (2024). *RUNC*. Accessed: May 25, 2024. [Online]. Available: https://github.com/opencontainers/runc
[46] The Linux Foundation. (2024). *Kubectl*. Accessed: May 25, 2024. [Online]. Available: https://kubernetes.io/docs/reference/
[47] The Linux Foundation. (2024). *Kubelet*. Accessed: May 25, 2024. [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
[48] ISTIO. (2024). *ISTIO/The ISTIO Service Mesh*. Accessed: May 25, 2024. [Online]. Available: https://istio.io/latest/about/service-mesh/
[49] HashiCorp. (2024). *Consul by HashiCorp*. Accessed: May 25, 2024. [Online]. Available: https://www.consul.io/
[50] Buoyant. (2024). *Linkerd*. Accessed: May 25, 2024. [Online]. Available: https://linkerd.io/2.14/overview/
[51] OpenID Foundation. (2024). *OpenID—OpenID Foundation*. Accessed: May 25, 2024. [Online]. Available: https://openid.net/
[52] R. W. Macarthy and J. M. Bass, "An empirical taxonomy of DevOps in practice," in *Proc. 46th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2020, pp. 221–228. [Online]. Available: https://ieeexplore.ieee.org/document/9226359
[53] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," in *Software Process Improvement and Capability Determination*, A. Mas, A. Mesquida, R. V. O'Connor, T. Rout, and A. Dorling, Eds. Cham, Switzerland: Springer, 2017, pp. 17–29.
[54] E. Gamma, R. Helm, R. Johnson, J. Vlissides, and G. Booch, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Reading, MA, USA: Addison-Wesley, Oct. 1994.
[55] Nat. Inst. Standards Technol. (2006). *Minimum Security Requirements for Federal Information and Information Systems*. Federal Information Processing Standard (FIPS). [Online]. Available: https://csrc.nist.gov/pubs/fips/200/final
[56] Org. for Advancement Structured Inf. Standards. (2013). *EXtensible Access Control Markup Language (XACML) Version 3.0*. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
[57] HashiCorp. (2024). *Vault by HashiCorp*. Accessed: May 25, 2024. [Online]. Available: https://www.vaultproject.io/
[58] Microsoft. (2024). *Key Vault—Microsoft Azure*. Accessed: May 25, 2024. [Online]. Available: https://azure.microsoft.com/en-us/products/key-vault
[59] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012. [Online]. Available: http://link.springer.com/10.1007/978-3-642-29044-2

PHILIPP HAINDL received the Ph.D. degree in computer science from Johannes Kepler University Linz, in 2021. He has more than 15 years of practical experience in industrial and research-focused software projects as a Software Engineer and an Architect. He is currently a Lecturer in software engineering with the St. Pölten University of Applied Sciences. His research interests include empirical software engineering, software security, and software quality operationalization. He regularly functions as a reviewer for international software engineering conferences and journals.



PATRICK KOCHBERGER received the M.Sc. degree in IT security from the St. Pölten University of Applied Sciences. He is currently a Researcher in software security with the St. Pölten University of Applied Sciences.

MARKUS SVEGGEN received the M.Sc. degree in cyber security and resilience from the St. Pölten University of Applied Sciences, in 2024, with a thesis examining the topics of this study. During his study, he was a Security Consultant in an international consulting company in Vienna, with a focus on penetration testing and payload development.