applied sciences

MDPI

Article

Authentication and Authorization in Microservices Architecture: A Systematic Literature Review

Murilo Góes de Almeida * and Edna Dias Canedo *

Department of Computer Science, University of Brasília (UnB), P.O. Box 4466, Brasilia 70910-900, Brazil
* Correspondence: murilo.almeida@aluno.unb.br (M.G.d.A.); ednacanedo@unb.br or edna.canedo@gmail.com (E.D.C.); Tel.: +55-61-98114-0478 (E.D.C.)

Abstract: The microservice architectural style splits an application into small services, which are implemented independently, with their own deployment unit. This architecture can bring benefits, nevertheless, it also poses challenges, especially about security aspects. In this case, there are several microservices within a single system, it represents an increase in the exposure of the safety surface, unlike the monolithic style, there are several applications running independently and must be secured individually. In this architecture, microservices communicate with each other, sometimes in a trust relationship. In this way, unauthorized access to a specific microservice could compromise an entire system. Therefore, it brings a need to explore knowledge about issues of security in microservices, especially in aspects of authentication and authorization. In this work, a Systematic Literature Review is carried out to answer questions on this subject, involving aspects of the challenges, mechanisms and technologies that deal with authentication and authorization in microservices. It was found that there are few studies dealing with the subject, especially in practical order, however, there is a consensus that communication between microservices, mainly due to its individual and trustworthy characteristics, is a concern to be considered. To face the problems, mechanisms such as OAuth 2.0, OpenID Connect, API Gateway and JWT are used. Finally, it was found that there are few open-source technologies that implement the researched mechanisms, with some mentions of the Spring Framework.

Keywords: microservice; authentication; authorization; security; SLR

check for updates

Citation: de Almeida, M.G.; Canedo, E.D. Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. *Appl. Sci.* **2022**, *12*, 3023. https://doi.org/10.3390/app12063023

Academic Editors: Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 5 February 2022
Accepted: 8 March 2022
Published: 16 March 2022

1. Introduction

The microservice architectural style is represented by an ecosystem of small services, each running in its own process and communicating through lightweight protocols, such as HTTP (Hypertext Transfer Protocol), built around business resources and deployed independently [1]. Breaking an application into microservices can bring some benefits, such as optimizing management, scalability, availability and reliability [2,3]. However, it may bring challenges in relation to security, because, in this case, an individual attention about it must be observed in each microservice developed, different from the monolithic style where security strategies are applied in a single application [3,4]. Furthermore, there are few practical demonstrations in the literature describing solutions to improve the security of [4] service-oriented architectures.

Regardless of the implemented architecture, the authentication and authorization aspects are relevant, considering them as key elements for the security mechanisms [5]. Authentication is the process of determining whether someone or something is, in fact, who they claim to be. Authorization is the process of giving someone or something permission to do or possess something [6]. There are protocols that deal with authorization and authentication issues, such as OAuth 2.0, the standard for delegated authorization, and OpenID Connect, the authentication layer on top of OAuth 2.0 [7]. It is important to note that there is a distinction between user authentication and service authentication. In the case of authentication between microservices, there are specific mechanisms for this, such as Mutual Transport Layer Security (MTLS) [7]. Using MTLS, each microservice will legitimately identify who it talks to, while also ensures data confidentiality and integrity in this communication [8].

According to some studies, microservices are usually designed in such way that there is a relationship of trust between them [3,9]. However, it is possible to find microservice architectures that use the "zero-trust" paradigm [10]. In this last case, there is a premise that trust is never granted implicitly but must be continually evaluated [11]. Thus, a lack of observation about authentication and authorization in a single microservice can affect the entire ecosystem. It is important that studies related to security issues in microservices emphasize aspects involving authentication and authorization. Therefore, in this paper, we carried out a Systematic Literature Review (SLR) to identify in the literature the studies that address authentication and authorization in microservice environments, what are their challenges, security mechanisms used to deal with these challenges and open-source technologies that implement the mechanisms identified in the review. The focus on open-source is to provide technologies that can reduce costs, free access to source code and customization [12]. There are advantages for use open-source in the public sector, such as avoiding monopoly dominance in the market [12]. Last, but not least, even software developed by commercial firms is being released under open-source licenses as well [13]. It is important to note that the adoption of open-source, although it has the advantage of free use, it will not necessarily bring an adequate cost/benefit for the organization [14]. Therefore, it is recommended that its adoption be based on metrics such as the Total Cost of Ownership (TCO), an instrument that assesses the cost of adapting, managing and maintaining the proposed software [14].

Our main findings reveal that authentication and authorization challenges involving microservices are mostly related to the communication between them and the complexity of implementing security in each microservice, generating a complexity both in the development and in the increase of the attack surface, since individual attention must be given to each microservice. The most mentioned mechanisms in the literature that address the challenges of authentication and authorization in microservices are OAuth 2.0, JWT, API Gateway and OpenID Connect, in addition to Single Sign-on strategy. These mechanisms can be implemented together, with their respective role in the security context. The API Gateway acts as an intermediary between the external client and the microservices, providing a private network environment that allows the exchange of data between them [15]. Single Sign On (SSO) allows users to authenticate only once and use all apps associated with their user accounts, without requiring them to enter their credentials each time they access a different app [16]. Finally, we identified that the Spring Framework is widely used in the context of open-source applications.

2. Systematic Literature Review

To achieve the research goal, we performed a Systematic Literature Review (RSL), in accordance with the guidelines proposed by Kitchenham and Charles [17] and the structuring applied by Kitchenham et al. [18]. According to the authors, an RSL is "a means of identifying, evaluating and interpreting all available research relevant to a specific research question, or topic area, or phenomenon of interest" [17]. In addition, we used the online tool Parsifal [19] to support the screening and analysis of the identified studies.

2.1. Research Questions

We conducted the SLR to answer the following research questions (RQ):

1. RQ.1. What are the challenges mentioned in the literature to perform authentication and authorization in the context of microservice architecture systems?
2. RQ.2. What mechanisms are used in the literature to deal with the challenges related to authentication and authorization in a microservices architecture?

3. RQ.3. What are the main open-source technology solutions that implement the authentication and authorization mechanisms identified in the literature?

2.2. Search Process

To identify studies in the literature, we performed an automatic search in the main digital databases in the field of Computer Science. The digital databases used in the systematic literature review were: DBLP (https://dblp.uni-trier.de, accessed on 4 February 2022), IEEE Digital Library (http://ieeexplore.ieee.org, accessed on 4 February 2022) and Scopus (http://www.scopus.com, accessed on 4 February 2022). The search string used in digital databases was defined according to the keywords that must appear in the search results. The search string used was:

> (“MICROSERVICE” OR “MICROSERVICES”) AND (“SECURITY” AND “AUTHENTICATION” AND “AUTHORIZATION”) AND (“CHALLENGE*” OR “PROBLEM*” OR “ISSUE*” OR “SOLUTION*” OR “PROTOCOL*” OR “MECHANISM*” “STRATEG*” OR “IMPLEMENTATION*” OR “OPENSOURCE” OR “OPEN-SOURCE” OR “OPEN SOURCE”).

We also applied the “snowballing” process which aims to prevent relevant studies from being omitted [20]. In this process, references about the research object in each selected study are verified. Thus, we searched for papers where selected studies were cited.

2.3. Inclusion and Exclusion Criteria

The selection criteria for primary studies seek to identify papers that provide information about the research questions. Therefore, we defined the following inclusion and exclusion criteria, based on the research questions:

Inclusion Criteria

- **IC.1** Studies dealing with challenges involving authentication and authorization in microservices;
- **IC.2** Studies related to security mechanisms that deal with authentication and authorization challenges in microservices;
- **IC.3** Studies related to open-source technologies that implement security mechanisms.

Exclusion Criteria

- **EC.1** Studies that do not address the research object;
- **EC.2** Studies prior to 2010;
- **EC.3** Duplicate studies;
- **EC.4** Studies published as short paper.

2.4. Quality Assessment

To differentiate selected studies according to quality criteria we check in each selected study whether they answer the research questions. The criteria adopted were:

1. Is the research objective clearly described?
2. Do the authors describe the limitation of the study?
3. Does the study identify problems and/or challenges involving authentication and authorization in microservices architecture?
4. Does the study identify the mechanisms that mitigate the problems and/or challenges involving authentication and authorization in microservices architecture?
5. Does the study present solutions that implement security mechanisms using open-source technology?

The answer of each quality criterion question received a score, as follows:

1. Yes (1);
2. Partially (0.5);
3. No (0).

Although the primary studies were selected using specific criteria, there is an individual assessment of the quality for each study, to verify which of them are more aligned with the research questions that were defined.

2.5. Data Collection and Analysis

The following data were collected in the selected primary studies: (1) Authentication and/or authorization challenges found in microservices; (2) The mechanisms that deal with the authentication and/or authorization challenges found in microservices; (3) Open-source technologies that implement mechanisms which deal with authentication and authorization challenges in microservices.

The identified challenges, mechanisms and solutions were organized in a ranking to verify the most mentioned in the primary studies. This ranking aims to show what manuscripts have more answers about the research questions, this does not mean that the lowest rated manuscripts are worse than the first ones, it just means that the top-rated manuscripts have more information to answer our questions. Subsequently, the items most present in the studies were submitted to an individual analysis for a better understanding of their basic concepts. Finally, it was verified which specific mechanisms deal with the challenges found.

3. SLR Results

This section presents the results of performing the systematic literature review. Figure 1 shows the complete execution process of the proposed protocol to execute the SLR, with the respective quantity of studies identified in each step of the protocol. In the automatic search performed in the digital databases using the initial query, 22 papers were found. These studies were submitted to the snowballing process, resulting in 13 new selected papers. Of the 22 papers found initially, 11 were eliminated due to the exclusion criteria (5 studies that do not address the research object and 6 duplicate). Thus, 11 primary studies were selected from the digital databases and 13 studies on snowballing execution, totaling 24 primary studies. The selected primary studies are shown in Table 1. The filters applied during the SLR based on inclusion and inclusion criteria are demonstrated in Figure 2.

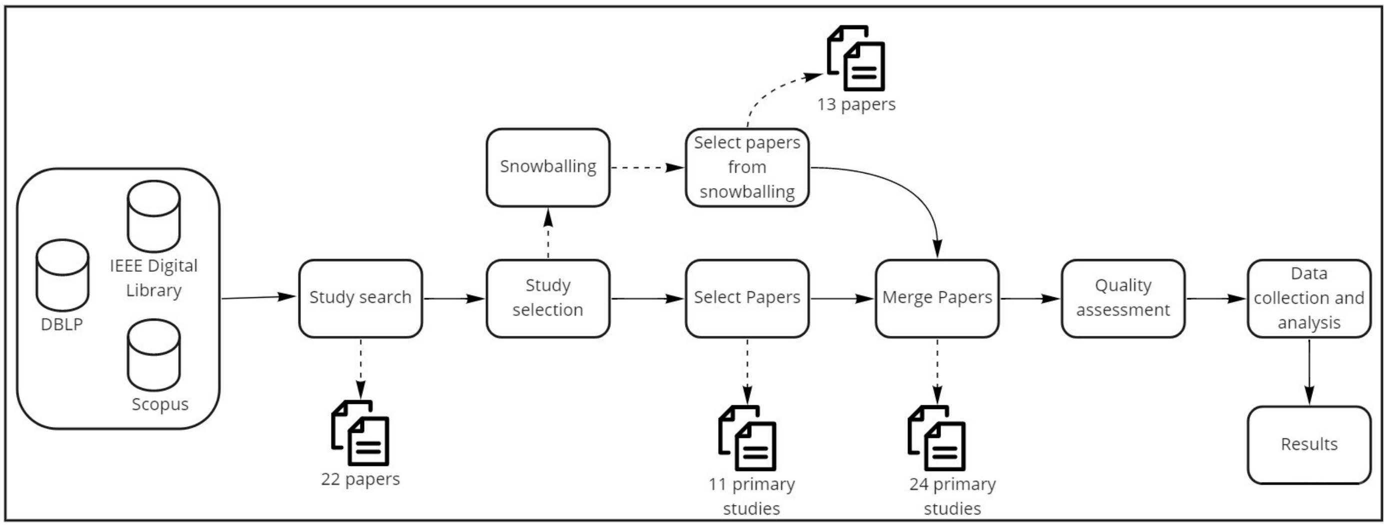

Figure 1. Protocol application process.

Table 1. Selected Studies.

ID	Year	Title	Ref
S1	2021	Security in microservice-based systems: A Multivocal literature review	[4]
S2	2021	Security in microservices architectures	[3]
S3	2020	Authentication and authorization in microservice-based systems: survey of architecture patterns	[8]
S4	2020	Information system development for restricting access to software tool built on microservice architecture	[21]
S5	2020	Research on Unified Authentication and Authorization in Microservice Architecture	[22]
S6	2020	Secure Edge Computing Management Based on Independent Microservices Providers for Gateway-Centric IoT Networks	[23]
S7	2019	Applying Spring Security Framework and OAuth 2.0 To Protect Microservice Architecture API	[24]
S8	2019	A survey on security issues in services communication of Microservices-enabled fog applications	[25]
S9	2019	Enhancing security to the MicroService (MS) architecture by implementing Authentication and Authorization (AA) service using Docker and Kubernetes	[26]
S10	2019	Implementing secure applications in smart city clouds using microservices	[16]
S11	2019	Microservice Security Agent Based On API Gateway in Edge Computing	[15]
S12	2019	Securing Microservices	[27]
S13	2019	Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping	[28]
S14	2018	Authentication and authorization orchestrator for microservice-based software architectures	[29]
S15	2018	Defense-in-depth and Role Authentication for Microservice Systems	[30]
S16	2018	Fine-Grained Access Control for Microservices	[31]
S17	2018	Overcoming Security Challenges in Microservice Architectures	[7]
S18	2018	Security considerations for microservice architectures	[32]
S19	2018	Unified account management for high performance computing as a service with microservice architecture	[33]
S20	2017	A Secure Microservice Framework for IoT	[34]
S21	2017	Access control with delegated authorization policy evaluation for data-driven microserviceworkflows	[35]
S22	2017	Authentication and Authorization of End User in Microservice Architecture	[36]
S23	2017	Integrating Continuous Security Assessments in Microservices and Cloud Native Applications	[37]
S24	2015	Security-as-a-Service for Microservices-Based Cloud Applications	[9]

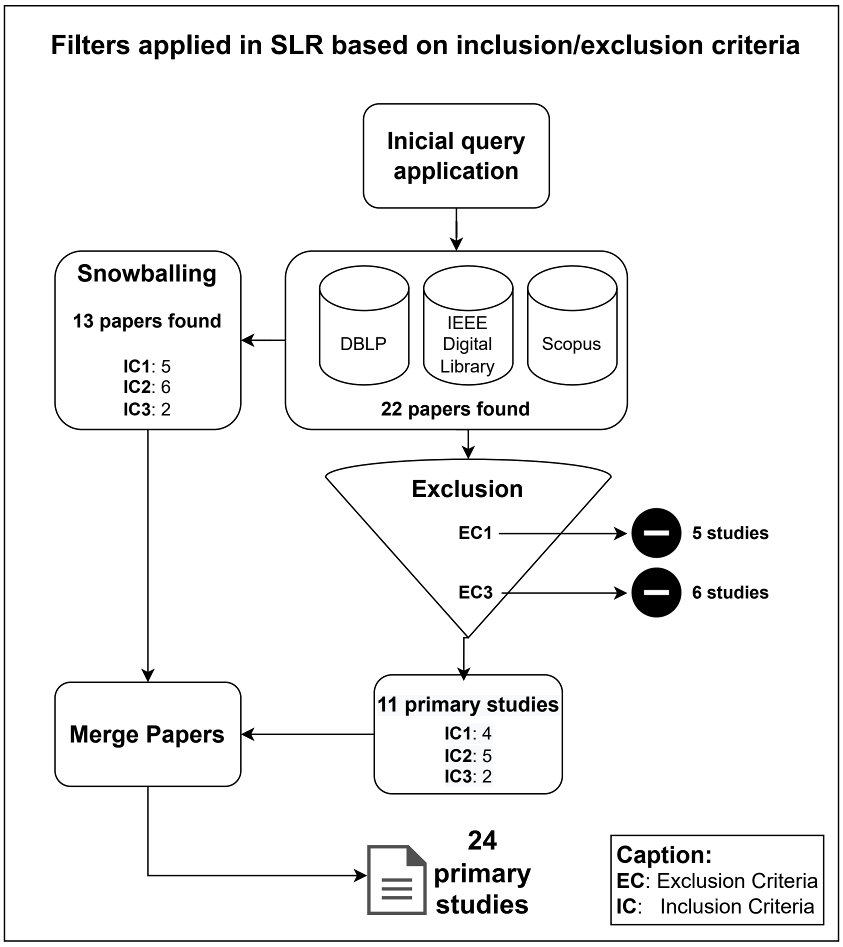

Figure 2. Filters applied in SLR process.

3.1. Quality Assessment of Reviews Carried out

According to the quality criteria, the selected studies were analyzed and scored, as shown in Table 2. All primary studies mentioned challenges involving authorization and authentication in microservices (AQ3), as well as mechanisms to mitigate such problems (AQ4), even if partially. However, there is a smaller amount of work (15) mentioning open-source technologies that implement the mechanism (AQ5). In general, the studies are clear about the objective (AQ1), but 11 of them do not describe its limitations (AQ2).

Table 2. Ranking of scores according to Quality Assessments.

ID	AQ1	AQ2	AQ3	AQ4	AQ5	Total
S1	1	1	1	1	1	5.0
S8	1	0.5	1	1	1	4.5
S16	1	1	1	1	0.5	4.5
S23	1	1	1	0.5	1	4.5
S17	1	1	1	1	0.5	4.5
S21	1	0.5	0.5	1	1	4.0
S5	1	0	1	1	1	4.0
S6	1	0.5	0.5	1	1	4.0
S13	1	1	1	1	0	4.0
S7	1	0.5	0.5	0.5	1	3.5
S3	1	0	1	1	0.5	3.5
S15	0.5	0	1	1	1	3.5
S10	1	0.5	1	1	0	3.5
S11	1	0	0.5	1	1	3.5
S20	1	1	0.5	0.5	0	3.0
S14	1	0	1	1	0	3.0
S12	1	0	1	1	0	3.0
S2	1	0	1	1	0	3.0
S19	1	0.5	0.5	0.5	0.5	3.0
S4	0.5	0	1	0.5	0.5	2.5
S24	0.5	0.5	1	0.5	0	2.5
S22	0.5	0	0.5	0.5	0.5	2.0
S18	0.5	0	0.5	0.5	0	1.5
S9	0	0	0.5	0.5	0	1.0

3.2. *Quality Factors*

We have done a verification to understand if there is any kind of relationship between the quality score and the year the study was published. Although it is possible to verify that the average score increased over the years, the standard deviation and the coefficient of variation show that the data are heterogeneous, and it is not possible to conclude that the quality has increased over the period, as shown in the Table 3. It is possible to verify in this situation that the standard deviation increases in the same proportion as the average, in addition to the coefficient of variation being in a high degree.

Table 3. Average study quality score by year.

	2015	2017	2018	2019	2020	2021
Number of Studies	1	4	6	7	4	2
Rating Average	2.5	3.38	3.33	3.29	3.50	4.00
Standard deviation	0	1.1087	1.1255	1.1127	0.7071	1.4142
Coefficient of variation	0	0.3285	0.3376	0.3386	0.2020	0.3536

We performed analyzes on data extracted from selected studies to answer the research questions.

3.3. *RQ.1. What Are the Challenges Mentioned in the Literature to Perform Authentication and Authorization in the Context of Microservice Architecture Systems?*

The challenges identified about authentication and authorization in the context of microservice architecture systems are presented in Table 4. Such challenges were presented according to the number of mentions in the selected studies, therefore, it does not mean that these are the most critical in terms of vulnerabilities or how much they occur in a microservices environment. The number of mentions of the challenges found in the studies does not necessarily reflect a level of priority in which they should be observed in a practical environment. Among the identified challenges, the five most mentioned in the literature were: "Communication between microservices" (13 mentions), "Trust between microservices compromised by unauthorized access" (12 mentions), "Individual concern for each microservice" (12 mentions), "Increased attack surface" (12 mentions), and "Microservice Access Control" (10 mentions).

Table 4. Challenges related to authentication and authorization in microservices

Pos	Challenge	ID	Number of Occurrences
1st	Communication between microservices	S1, S2, S3, S4, S7, S9, S10, S15, S16, S17, S18, S23, S24	13
2nd	Trust between microservices compromised by unauthorized access	S1, S2, S4, S6, S8, S12, S15, S16, S19, S21, S23, S24	12
3rd	Individual concern for each microservice	S1, S2, S5, S10, S12, S13, S15, S16, S21, S22, S23, S24	12
4th	Increased attack surface (compared to monolithic)	S1, S2, S3, S7, S8, S13, S14, S16, S17, S23	10
5th	Microservice access control	S5, S8, S10, S11, S14, S15, S19, S20, S21	9
6th	Authorization between services	S1, S7, S8, S15, S16, S17, S18, S21	8
7th	Lack of studies about microservices	S1, S2, S4, S8, S10, S13, S17	7
8th	Lack of security patterns in microservices	S1, S13, S15, S17, S20, S21	6
9th	Different teams working on different microservices must have the same understanding of security	S3, S15, S17, S20, S23	5
10th	Bypass on Api Gateway	S3, S4, S6, S12	4
11th	Intrusion detection/monitoring	S1, S12, S24	3
12th	Escalation of privileges	S2, S16, S24	3
13th	Lack of study demonstrating practical implementation of security in microservices	S7, S13, S23	3
14th	Coordinate authentication server with new microservices	S1, S22	2
15th	Lack of attention in attack reaction/recovery	S1, S13	2
16th	Token validation at each microservice request	S5, S6	2
17th	Public Images may be compromised	S1	1
18th	Many applications in commercial microservices without possibility to evaluate code	S4	1
19th	Use of authentication/authorization server that handles all microservices	S3	1
20th	Possibility of development in various technologies	S23	1

In general, the studies that mentioned the existing challenges made comparisons between monolithic and microservices architectures, explaining that in the monolithic model, there is only one surface to be protected, however, in the microservice environment, each autonomous service must be a point of concern regarding security, making it more complex to keep this entire ecosystem properly protected. Although each service needs particular attention, Yarygina and Bagge [7] alerted that manual security provisioning of hundreds or thousands of service instances is infeasible. Pereira-Vale et al. [4] compared monolithic with microservices using a KLOC metric (kilo Lines of Code), they say that in a monolithic application, every 100 kloc will have an average of 39 vulnerabilities. The same quantity of lines of code in a microservice application, will have an average of 180 vulnerabilities. They also alert about the decomposition of monolithic into microservices because security needs to be a global property, not the sum of local security defenses.

Nehme et al. [27] argue the importance of authentication and authorization in the context of microservice security. They mentioned that "Microservices should only be invoked after requesting authentication and, ideally, authorization if levels of privileges are available.". Pereira-Vale et al. [28] performed a systematic mapping about security mechanisms used in microservices and they discovered that the most reported security mechanisms are related to authorization, authentication and credentials. Banat et al. [29] also agreed that authentication and authorization need to be carefully observed in a microservice architecture, because this scenario presents many points of access for users and the other parts of the application. They argued that the data being especially sensitive, the crucial point of the development is the authentication and the authorization process. Cao et al. [33] proposed the implementation of a global authentication and authorization mechanism named Unified Account Management as a Service (UAMS). In this implementation, they used a RESTful API divided into several microservices. All sensitive data is transferred encrypted by the HTTPS protocol.

The studies also highlighted that microservices have the characteristic of communicating with each other, usually through the HTTP protocol, and this is a point of attention that differs from the traditional monolithic approach and should be properly analyzed and observed in terms of security. Regarding the communication issue, the authors mentioned the implementation of Transport Layer Security (TLS), used to protect the communication channels [30].

The challenges presented, in general, complement each other, or even act transversally. Mateus-Coelho et al. [3] stated that "Microservices are often designed to trust their peers and, if one of them is compromised and accessed improperly, it is possible that there is a great advantage for all others to be exploited". Dongjin et al. [25] agreed when they affirmed that "When a single service is controlled by an attacker, the service may maliciously influence other services". Nehme et al. [31] mentioned an access control problem that may be found in microservice architecture named "confused deputy attack", in their words, it is a privilege escalation attack in which a microservice that is trusted by other microservices is compromised. Sun et al. [9] brought the concern about trust between services, they affirmed that the "compromise of a single microservice may bring down the entire application". They also reported the challenge of monitoring and auditing the microservices interaction over the network and proposed a design of a security-as-service infrastructure for microservices-based cloud applications, that helps to monitor the network aiming to find possible non-expected behaviors in the communication between them.

Pereira-Vale et al. [4] stated that the communication between microservices is exposed through the network environment, which creates a potential attack surface. The authors also mentioned the problem of increasing the attack surface, as the decomposition of an application into several services increases the attack surface and the security of the application becomes more difficult to manage, because it becomes the sum of several independent defenses, rather than being managed in a global way, as was done in the monolithic approach.

Nguyen and Baker [24] warned that network communication between microservices can occur in the internet environment, which increases, in addition to exposure, the number of possible attackers. Kramer et al. [16] reinforce this concern to implement secure application in smart city clouds using microservices, because it will handle with a huge amount of data, including sensitive information about infrastructure and citizens. Xu et al. [15]

shared the same concerned, in this case, using microservice in IoT devices. Lu et al. [34] is also concerned about security in IoT devices using microservices architecture, mainly because of sensitive data that can be shared among services. They encourage the use of API gateways, that will remove all the concerns about microservices, because all interactions with the components will be performed with the API Gateway. Safaryan et al. [21] are also concerned about communication among different microservices because it is carried out through network interaction, being necessary to secure each of the service and the network. They also alert about the need of a pattern to be implemented. The lack of a correct pattern can compromise the network environment. Nguyen and Baker [24] agreed about the need to observe communication between the services. Banati et al. [29] argued that the network used in microservices communication can be secured using system administration tools such as VPN, Firewall and HTTPS.

Dongjin et al. [25] and Jander et al. [30] raised a concern that if a single service were controlled by an attacker, it could maliciously influence all other services. Concerns related to communication between microservices, increased surface exposure, access control and individual concern in each microservice, the API Gateway strategy, and use of mechanisms such as OAuth 2.0 and JWT were widely mentioned. They are also concerned about the industry, which are not fully aware about security issues involving microservices.

API Gateway helps to limit exposure between microservices as requests will be centered on it, and no longer on a microservice directly [3,8]. Jin et al. [23] mention that API gateway will secure a microservices environment because it will filter all requests. They also proposed an edge gateway to manage microservices. Nehme et al. [27] not recommended the access token validation in the gateway level, this role need to be performed in an authentication server. In contrast, Torkura et al. [37] proposed a security gateway used as security control for enforcing security policies. They also alerted about discoverability, that means, a gateway feature that allows a microservice to subscribe in it. If the discovery service can accept any subscription, vulnerable microservices could be pushed to production environments. OAuth 2.0 is a popular authorization protocol and could protect access to microservices from unauthorized access as access tokens are issued to trusted clients that could access certain services [25]. Nguyen and Baker [24] explained that OAuth 2.0 is not only used in web-based application but can be applied into backend services with no need of web browser or user interaction. ShuLin and JiePing [22] mentioned that the JWT is an open standard (RFC 7519) that defines a compact and independent way to securely transmit information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.

Barabanov and Makrushin [8] warned that implementing authorization directly in the source code of each microservice can lead to future problems, especially in different teams working on independent microservices, because of new security updates must be performed in all projects, individually. He and Yang [36] followed in the same line, they alerted that imitate the way of monolithic structure in each microservice has several deficiencies, mainly when a new service join in the system, it will be necessary to implement the security function in this case. They proposed a solution creating a specific service focused on authentication and authorization, as a result, each service is focused on its own business, ensuring better scalability and decoupling the system.

Jander et al. [30] mentioned that different teams can implement their own internal security approach in the microservice which they have responsibility for, but this may require more specialized knowledge from the teams. Torkura et al. [37] brought the concern that development by different teams can bring microservices using not only different standards in development, but also different technologies, which must implement the same security standards.

Lu et al. [34] stated that the development of microservices by different teams and even different companies is completely possible, is there is an alignment on the security implementations in each microservice. Finally, it is important to mention that the lack of security patterns in microservices, added to the few studies on the subject, both theoretical and practical, can influence the management of the development of the architecture. Torkura et al. [37] warned that there are several literatures that highlight security problems in microservice architectures, however, none of them offer practical solutions to deal with these situations.

Pereira-Vale et al. [4] alerted about the use of an authentication and authorization service to be used in an architecture of microservices. The use of this server must be robust enough to authenticate the user and carry out the token validations that are made with each client request. The lack of concern about these challenges can cause a single point of failure of failure (SPOF), that is, if this part of system fails, will affect the entire application [38]. ShuLin and JiePing [22] stated that the authentication server may affect the performance of the entire system, mainly if there are several requests to this server.

Preuveneers and Joosen [35] alerted about the flow of the data among microservices. Even if each individual microservice is protected, it is important to note if the workflow is valid. In that work, they presented a workflow-oriented framework to avoid not expected communication between microservices.

Mateus-Coelho et al. [3] enumerated some examples of mechanisms which must be observed during the developing in a microservice architecture: complex passwords, authentication, web security flaws (what are the most flaws observed), people and processes. They also listed the most critical web application security risks: injection, broken authentication and session management, cross-site scripting, broken access control, security misconfiguration, sensitive data exposure, insufficient attack protection, cross-site request forgery, components with known vulnerabilities and under protected APIs. All of them may be exploited in a microservice environment. Nguyen and Baker [24] also pointed some web security risks and carried out some experiments using CSRF attack, XSS attack and Brute Force attack in an API endpoint protected by OAuth 2.0. In this case, all of tests were prevented by the configuration proposed in the Proof of Concept presented in that work.

3.4. RQ.2. What Mechanisms Are Used in the Literature to Deal with the Challenges Related to Authentication and Authorization in a Microservices Architecture?

The mechanisms identified in the literature used to deal with authentication and authorization challenges in microservices architecture are presented in Table 5. The OAuth 2.0 protocol was the most mentioned (16 mentions), followed by JWT (14 mentions), API Gateway (14 mentions), Single Sign-ON (8 mentions) and OpenID Connect (7 mentions). Figure 3 shows the number of occurrences of the mechanisms used in the selected studies. Some mechanisms were used in only one study. It is important to emphasize that, in general terms, the identified mechanisms do not need to be implemented in a unique way, that is, they can coexist in the same environment, each one acting with a specific purpose. We identified in the selected studies some implementations in which the mechanisms act together, to mitigate possible vulnerabilities involving authentication and authorization in the microservices environment. It important to observe that some studies only point the mechanisms without explain deeply or demonstrate a practical implementation of them, such as the work of Pereira-Vale et al. [28], which is more concerned in perform a systematic mapping of security mechanisms.

Table 5. Security mechanisms used in microservices architecture

Pos	Mechanism	ID	Number of Occurrences
1°	OAuth 2.0	S1, S3, S5, S6,S7, S8, S10, S11, S12, S13, S14, S15, S16, S17, S20, S21	16
2°	JWT	S1, S3, S4, S5, S6, S9, S11, S12, S13, S14, S15, S17, S21, S22	14
3°	API Gateway	S2, S3, S4, S5, S6, S11, S12, S13, S14, S16, S17, S19, S20, S22	14
4°	Single Sign-ON SSO	S1, S2, S9, S10, S14, S19, S21, S22	8
5°	OpenID Connect	S1, S3, S8, S12, S16, S17, S21	7
6°	HTTPS	S2, S10, S14, S15, S17, S19, S20	7
7°	RBAC	S1, S3, S5, S13, S14, S17, S21	7
8°	ABAC	S1, S8, S14, S17, S20, S21	6
9°	XACML	S1, S3, S13, S15, S16, S21	6
10°	HMAC	S2, S3, S5, S14, S21	5
11º	SAML	S1, S2, S13, S14, S21	5
12º	TLS	S1, S10, S14, S15, S16	5
13º	OAuth	S1, S5, S8, S16	4
14º	Multilevel Security	S1, S3, S13	3
15º	DAC	S14, S21	2
16º	IAM	S14, S21	2
17º	RSA	S5, S11	2
18º	SASL	S1, S13	2
19º	SSL	S2, S13	2
20º	MTLS	S1, S17	2
21º	OpenID	S2, S14	2
22º	API Keys	S2	1
23º	Captcha	S19	1
24º	CAS	S8	1
25º	X509 Certificates	S1	1
26º	EAS	S3	1
27º	ECDSA	S5	1
28º	GSI	S8	1
29º	IDS	S12	1
30º	LDAP	S8	1
31º	MFA	S19	1
32º	NGAC	S3	1
33º	PBAC	S1	1

We will present a brief description of the most mentioned mechanisms in the selected studies:

- **OAuth 2.0 (Open Authorization):** The OAuth 2.0 protocol is defined by RFC 6749 [39]. According to Banati et al. [29] OAuth 2.0 is an authorization framework that allows users to access different services without having to share their credentials. In a practical scenario, the user authenticates to an authorization server and receives an authorization code or an access token, which can be used to access resources, without the need to contact the authorization server again or have to inform the username and password [25]. Access tokens are validated on each request for some service [15,35]. This procedure poses a risk for affecting the performance of a distributed architecture because if there is a large number of requests, the authentication server may be affected [22]. OAuth 2.0 is one of the protocols most used by microservice architectures for access delegation [25,31] and can be applied both in web applications and in backend services, in addition to meeting both the authentication and authorization proposal [24,28]. It is important to mention that OAuth 2.0 is widely used as an authorization protocol to protect services that use the REST (Representational State

Transfer) [23,25,27], in addition to adopting the HTTPS protocol in data communication [25].

- **JWT (JSON Web Token)**: The JWT is defined by RFC 7519 [40]. It is an open standard that provides a compact and independent way to securely transmit information between applications using a JSON (Javascript Object Notation) object. This information is verifiable and trustworthy as it is digitally signed using a secret [22]. It has a format divided into three parts: Header, Payload and Signature. The header is separated into two parts, token type and the algorithm, that may be a HMAC, SHA256 or RSA [29]. The JWT has an advantage over traditional tokens, this verification can be done directly on the resource server, without connecting to the authentication server [22,36]. Using JWT it is possible to retrieve user information directly from the token [22,26]. In addition to user information, it is common to find in a JWT their permissions and expiration time of the token [36]. The JWT has adherence to "stateless" applications, that is, those that do not keep a session on the server side, but stay data with the client side, and must be used in each client request to the resource [26]. In a microservices environment, JWTs can be transferred during the communication between them [35]. Finally, it is important to know that JWT can be integrated with the OAuth 2.0 protocol [4,23].
- **API Gateway**: In the microservices environment, API Gateway acts as an intermediary between the client and the microservices, providing a private network environment that allows the exchange of private data [3,15], that is, clients do not communicate directly with services, but only with a Gateway, which is responsible for communicates with the requested service. It can be an input that performs the filtering of client requests, making the appropriate forwarding to the microservice [23], and it can check the user's credentials, to find out if he owns the proper authorization [7,37]. We realized that API Gateway is a technique to decrease microservice exposure. Nevertheless, it is important in a future work compares the communication in different scenarios. These scenarios could be using or not using the API gateway between the client and microservices. Consequently, it will be possible to collect the strengths and weakness of both approaches and verify the possibility of hybrid scenarios.
  Lu et al. [34], stated that the API Gateway can aggregate multiple microservices in a single client interface, being an element that stands between the client and the requested service. Using the API Gateway helps to reduce the exposure of systems, then, the microservices are all protected behind the API Gateway [8]. Although several advantages for its implementation have been observed, its use may not prove advantageous when it becomes a single decision point, because, in case of failure in this element, the entire application may become inaccessible [8]. An API Gateway can use services such as JWT and OAuth 2.0 [7,8,15,23,25,34,36].
- **OpenID Connect**: OpenID Connect is an open authentication standard that ensures users have only one digital identity for multiple applications or services [29]. Dongjin et al. [25] stated that it is an authentication layer over the OAuth protocol, allowing services to read the user's basic information. Nehme et al. [27,31] observed that OpenID Connect is built on top of the OAuth protocol. Yarygina and Bage [7] reinforced that OpenID Connect provisions the user's identity. OpenID Connect can be used in conjunction with OAuth 2.0 [4,8,35]. There is a difference between OpenID and OpenID Connect (OIDC). According to the OpenID Foundation website, "OpenID Connect performs many of the same tasks as OpenID 2.0, but does so in a way that is API-friendly, and usable by native and mobile applications" [41]. They also explain that "OpenID Connect defines optional mechanisms for robust signing and encryption. Whereas integration of OAuth 1.0a and OpenID 2.0 required an extension, in OpenID Connect, OAuth 2.0 capabilities are integrated with the protocol itself" [41].
- **SSO (Single Sign-On)**: SSO allows a user to be authenticated only once when logging into a particular system, therefore, users can access all authorized resources and services on a system without needing another authentication [29,36]. According to

Banati et al. [29], the main purpose of this mechanism is the exchange of authorization credentials and not the authentication by itself. The authors also reinforced that the mechanism guarantees unified authentication in microservices and the implementation of this feature can improve the user experience [33]. In the same way as the API Gateway, the implementation of an SSO server may cause a "Single Point of Failure", that is, if there are problems in this system, every application can be compromised, as it centralizes all authentication of a system [36]. It is possible to implement a Single Sign-On system based on OAuth 2.0 [16,29,35].

- **HTTPS**: The Hyper Text Transfer Protocol Secure is defined by RFC 2818 [42]. It describes the use of HTTP over TLS (Transport Layer Security). Using the HTTPS protocol will ensure that the communication be encrypted [16]. It provides a channel between two hosts identified by certificates [30]. The use of HTTPS not just limited to encrypt data, but ensures that a given client is communication with whom he wants to [3].
- **RBAC**: Role-Based Access Control is used in authorization process [4]. It is a very know identity-based access control model [35]. The use of a role-based access control will increase the flexibility of the system because the role will define what access the client is allowed [22]. RBAC is user-centric access control model, it does not account for relationship between the requesting entity and the resource [7]. RBAC authorization roles can be incorporated into JWT tokens as an additional attribute [7].
- **ABAC**: Attribute-Based Access Control, based in the words of Preuveneers and Joosen [35] "grants access rights to subjects through the use of policies or rules that combine various types of attributes to facilitate user access to the right resources under the right conditions". They complemented that it offers more expressivity and flexibility compared to another access control models such as RBAC. The primary goal of ABAC in the words of Yu et al. [25] is "an access control model is to fulfill the requirements of highly heterogeneous environments such as multi-cloud environment". They also pointed the benefit of centralized security management and orchestration that will protect the application according to consistent policies. ABAC is recommended to be used when there is fine-grained authorization of resources, such as access to a specific API call [7].
- **XACML**: eXtensible Access Control Markup Language is defined by RFC 7061 [43]. According to this document, XACML "defines an architecture and a language for access control (authorization). The language consists of requests, responses, and policies". It is used to create access control policies and can be used with OAuth 2.0 protocol [31]. Nehme et al. [31] proposed a model using XACML along with OAuth 2.0. In this case, OAuth 2.0 acts as an authorization service and XACML with policy administration and decision points. Barabanov and Marushin [8] discourage the use of XAML because it use a complicated syntax, causing more work for developers, adding to the fact that there were not many open-source integrations.
- **HMAC**: Hash-based Message Authentication Code is defined by RFC 2104. It provides a way to check the integrity of an information transmitted in a medium [44]. In the words of Mateus-Coelho et al., HMAC consists in "hash-based messaging code to sign the request". According to the same authors, there are many examples that can be found in internet suggesting the use of HMAC over HTTP. HMAC algorithm can also be used to sign a JWT [22].
- **SAML**: The Security Assertion Markup Language (SAML) 2.0 is defined by RFC 7522 and is defined as an XML-based framework that allows identity and security information to be shared across security domains [45]. In a microservices environment, SAML is used to exchange user attributes stored at the identity provider [35]. Mateus-Coelho et al. [3] affirmed that "SAML and OpenID is perfect for Authentication and Authorization of someone's on a system but it's also great for service-to-service authentication as well". Nevertheless, they admitted that SAML is complex when it is compared to other technologies such as Api Keys.

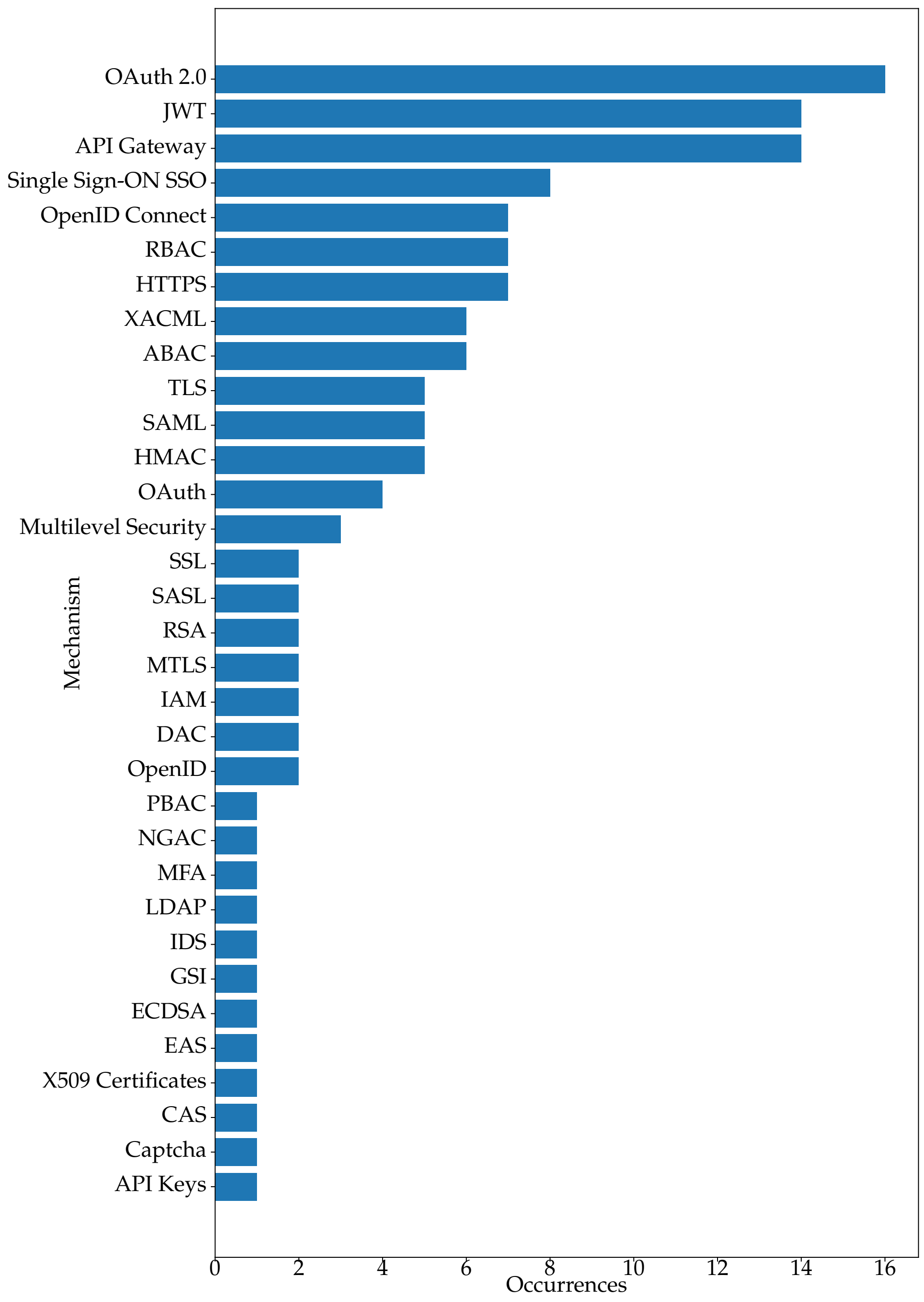

Figure 3. Number of occurrences versus security mechanisms found

3.5. RQ.3.What Are the Main Open-Source Technology Solutions That Implement the Authentication and Authorization Mechanisms Identified in the Literature?

The main open-source solutions that implement the authentication and authorization mechanisms identified in the literature are presented in Table 6. It is possible to notice that the Spring [46] ecosystem libraries are the most mentioned (Spring Security, Spring Cloud, Spring Boot, Gateway Zuul and Eureka Server), totaling 10 mentions. We realized that Spring Boot and Eureka are not focused on security, but they have specific security libraries that can be used together. For instance, implementing Spring Boot allows to implement Spring Security library, and Eureka helps to implement an API Gateway using Spring Cloud. Some of these studies demonstrated a practical implementation of Spring framework using security mechanisms [21,22,24,25].

Although the research found several references to the Spring ecosystem, which is built by Java programming language, it is important to mention that there are alternative frameworks based in other languages that implement the security mechanisms found into a microservice environment, such as GoKit (Golang) [47], Flask (Python) [48] and .NET Core (C#) [49].

The other open-source solution mentioned more than once is called Kong [50] (2 mentions), the others being mentioned only once. It is important to note that the Spring Framework uses the Java programming language and has several libraries for implementing security mechanisms in microservices, such as API Gateway, OAuth 2.0 and OpenID Connect [46]. The Kong application refers to the "Kong API Gateway", that is, among all the mechanisms raised, it supports the implementation of an API Gateway.

Table 6. Open-source technologies that implement security in microservices architecture.

Open-Source	ID
Spring Security	S1, S4, S5, S7, S8
Kong	S6, S11
Spring Boot	S6, S7
Gateway Zuul	S4, S5
Eureka Server	S5
Jadex	S15
Jarvis	S1
Lagom	S15
VertX	S15

4. Discussions

Given the challenges, mechanisms and open-source solutions presented, it was verified which of the mechanisms and solutions could be implemented to face the challenges, according to what was collected in this RSL. Table 7 presents the solutions that act on the related challenges. It was verified that part of the challenges does not have a direct link on the open-source mechanism and/or implementation. The mechanisms identified could be applied to face 09 challenges of the 20 listed in Table 4. Although it seems a low number, these challenges are the most mentioned by the authors.

The mechanisms were widely mentioned by the authors, most of them can face the challenges, with emphasis once again on the implementation of OAuth 2.0, JWT, API Gateway, OpenID Connect and Single Sign ON (SSO). Nevertheless, it does not mean that they are better or will solve any kind of security issues related to microservices architecture. Even the less mentioned mechanisms could be more appropriate, depending on the case. It is important to know what each mechanism is individually and what it does, for then, implement a good security architecture in a system.

Table 7. Linking Challenges, Mechanism and Open-Source Solutions.

Challenge	Mechanism	Open-Source
Increased attack surface (compared to monolithic)	API Gateway (S2, S3, S4, S6, S12, S13, S16, S17, S20, S22), OAuth 2.0 (S7, S12, S13), SSO (S14)	Spring (S4, S7), Kong (S6, S11)
Authorization between services	OAuth 2.0 (S1, S5, S6, S7, S8, S10, S12, S13, S14, S15, S16, S17, S21), SAML(S2), OpenID(S2, S14, S16, S17), JWT (S9, S12, S13, S14, S15, S17, S21)	Spring (S1, S5)
Bypass in Api Gateway	XACML (S3), NGAC (S3), JWT (S3, S6, S12), OpenID (S3, S12, S16), OAuth 2.0 (S3, S6, S12, S16), TLS (S3), IDS (S12)	Spring (S4), Kong (S6)
Communication between microservices	TLS (S1, S10, S14, S15), MTLS (S17), SSL (S2), HTTPS (S2, S10, S14, S15, S17, S20), SAML(S2, S14, S21), XACML (S16, S21), OpenID(S2, S3, S8, S16, S17), JWT (S4, S5, S6, S12, S13, S14,15, S16, S17, S21), OAuth 2.0 (S5, S6, S7, S8, S10, S13, S15, S17, S21), GSI (S8)	Spring (s5), Kong (S6)
Trust between microservices compromised by unauthorized access	JWT (S1, S3, S4, S5, S6, S9, S11, S12, S13, S14, S15, S17, S21, S22), OAuth 2.0 (S1, S3, S5, S6, S7, S8, S10, S11, S12, S13, S14, S15, S16, S17, S20, S21), OpenID Connect S1, S2, S3, S8, S12, S14, S16, S17, S21)	Spring (S4, S5)
Microservice Access Control	OAuth 2.0 (S1, S5, S8, S10, S11, S12, S13, S14, S15, S16, S20, S21), OpenID (S1, S2, S3, S8, S12, S14, S16), TLS(S1), MTLS(S1), SASL (S1), SSO (S1, S2), JWT (S1, S5, S11, S12, S13, S14, S15, S21), HMAC(S2, S21), ABAC (S8, S17, S20, S21), RBAC (S17, S21), CAS (S8), RSA (S11), XACML (S16), Captcha (S20), Multiple FA (S20), DAC (S21), IAM (S21)	Spring (S1, S5, S11)
Coordinate authentication server with new microservices	LDAP (S8), SSO (S1, S2, 10, S13, s14, S20, S21, S22), OAuth 2.0 (S12, S14, S16), OpenID (S12, S14, S16),	
Individual concern for each microservice	JWT (S1, S3, S4, S5, S6, S9, S11, S12, S13, S14, S15, S17, S21, S22), OAuth 2.0 (S1, S3, S5, S6, S7, S8, S10, S11, S12, S13, S14, S15, S16, S17, S20, S21), OpenID Connect (S1, S2, S3, S8, S12, S14, S16, S17, S21)	Spring (S4, S5)
Use of authentication/authorization server that handles all microservices	SSO (S1, S2, 10, S13, s14, S20, S21, S22), OAuth 2.0 (S12, S14, S16)	

Study Limitations

The study was performed with searches in DBLP, IEEE and Scopus databases. To prevent relevant works from being discarded, the snowballing process was applied. Even with this concern, it is possible that, increasing the number of databases for consultations, new studies may be found. However, as verified in some studies collected, there is currently a lack of studies on the subject [3,4,7,16,21,25,28]. It was also verified that there is a lack of studies demonstrating the practical implementation of security in microservices [24,28,37]. Hence, it is likely that over the next few years, if the research related to the subject in question increases, a new systematic review will be necessary, in order to complement the knowledge collected in this work. We cannot conclude that the mechanisms less mentioned in the studies are less used, therefore, it is important to explore all of them, that can be done in a future work. Finally, it is important to note that this study is more focused on identifying answers to the research questions, that is, it is possible that the answers to these questions may be the subject of further studies pointing out which challenges are most critical in terms of vulnerability, how much they occur in a practical environment, or even which of these challenges should be addressed with priority. The mechanisms can also be implemented and tested in order to find out in a practical environment which of the challenges are mitigated with the implemented mechanism.

5. Final Remarks

As verified during the execution of this work and demonstrated in Table 4, there is a lack of studies related to security in microservices architecture. The lack increases when the study is specific for authentication and authorization, especially in a practical approach. It is important that the subject be better explored, because, as verified in this work, within a microservice environment, it is necessary to be concerned with security aspects in each service, individually, as the adoption of this architecture can increase the attack surface and still generate attention points in the communication between them, in this way, the lack of attention in these questions can make the applications vulnerable to unauthorized accesses. Of all the points listed in Table 4, there are issues related to the implementation of technologies themselves, however, there are other aspects related to the subject, such as the organization of development teams working on different microservices within the same system, therefore, is a theme with vast field to be explored.

Several mechanisms were found that mitigate the main points of attention observed, all of them listed in Table 5, with OAuth 2.0 being the most mentioned, along with the Json Web Token (JWT) and the use of API Gateway. The correct implementation of these can reduce the possibility of any type of unauthorized access to one or more microservices, making the environment better protected. There are few studies on practical implementations, thus, a scenario for future work is foreseen, especially with proposals for specific patterns within this context.

Finally, it was found that the literature indicates few open-source solutions that implement the mechanisms found. In this case, a viable alternative expands the search into new sources, including gray literature, which is literature produced at all levels of government, academic, business and industrial, in print and electronic formats, but which is not controlled by commercial publishers, that is, where publication is not the primary activity of the producing body[51]. Such findings can be properly experimented with scientific rigor and identified as technical solutions that solve the challenges collected in this work.

Author Contributions: Writing—original draft preparation, M.G.d.A. and E.D.C.; writing—review and editing, M.G.d.A. and E.D.C.; visualization, M.G.d.A. and E.D.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lewis, J.; Fowler, M. *Microservices—A Definition of This New Archtectural Term*; EA PAD: Redwood City, CA, USA, 2014.
2. Merson, P. *Microservices beyond the Hype: What You Gain and What You Lose*; SEI Digital Library: San Diego, CA, USA, 2015.
3. Mateus-Coelho, N.; Cruz-Cunha, M.; Ferreira, L.G. Security in microservices architectures. *Procedia Comput. Sci.* **2021**, *181*, 1225–1236. doi: [CrossRef]
4. Pereira-Vale, A.; Fernandez, E.B.; Monge, R.; Astudillo, H.; Márquez, G. Security in microservice-based systems: A Multivocal literature review. *Comput. Secur.* **2021**, *103*, 102200. doi: [CrossRef]
5. Pippal, S.K.; Kumari, A.; Kushwaha, D.S. CTES based Secure approach for Authentication and Authorization of Resource and Service in Clouds. In Proceedings of the 2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011), Allahabad, India, 15–17 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 444–449.

6. Halonen, T. Authentication and authorization in mobile environment. In *Tik-110.501 Seminar on Network Security*; Citeseer: Princeton, NJ, USA, 2000.
7. Yarygina, T.; Bagge, A.H. Overcoming Security Challenges in Microservice Architectures. In Proceedings of the 12th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2018 and 9th International Workshop on Joint Cloud Computing, JCC 2018, Bamberg, Germany, 26–29 March 2018; pp. 11–20. doi: [CrossRef]
8. Barabanov, A.; Makrushin, D. Authentication and authorization in microservice-based systems: Survey of architecture patterns. *arXiv* **2020**, arXiv:2009.02114.
9. Sun, Y.; Nanda, S.; Jaeger, T. Security-as-a-service for microservices-based cloud applications. In Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015, Vancouver, BC, Canada, 30 November–3 December 2015; pp. 50–57. doi: [CrossRef]
10. Mehraj, S.; Banday, M.T. Establishing a Zero Trust Strategy in Cloud Computing Environment. In Proceedings of the 2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 22–24 January 2020; pp. 1–6. doi: [CrossRef]
11. Rose, S.; Borchert, O.; Mitchell, S.; Connelly, S. *Zero Trust Architecture*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MA, USA, 2020.
12. Rossi, B.; Russo, B.; Succi, G. Adoption of free/libre open source software in public organizations: Factors of impact. *Inf. Technol. People* **2012**, *25*, 156–187. doi: [CrossRef]
13. Hippel, E.V.; Krogh, G.V. Open source software and the "private-collective" innovation model: Issues for organization science. *Organ. Sci.* **2003**, *14*, 209–223. [CrossRef]
14. Lavazza, L. Beyond Total Cost of Ownership: Applying Balanced Scorecards to Open-Source Software. In Proceedings of the International Conference on Software Engineering Advances (ICSEA 2007), Cap Esterel, France, 25–31 August 2007; p. 74. doi: [CrossRef]
15. Xu, R.; Jin, W.; Kim, D. Microservice security agent based on API gateway in edge computing. *Sensors* **2019**, *19*, 4905. doi: [CrossRef] [PubMed]
16. Krämer, M.; Frese, S.; Kuijper, A. Implementing secure applications in smart city clouds using microservices. *Future Gener. Comput. Syst.* **2019**, *99*, 308–320. doi: [CrossRef]
17. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; EBSE: Goyang, Korea, 2007.
18. Kitchenham, B.; Brereton, O.P.; Budgen, D.; Turner, M.; Bailey, J.; Linkman, S. Systematic literature reviews in software engineering—A systematic literature review. *Inf. Softw. Technol.* **2009**, *51*, 7–15. doi: [CrossRef]
19. Freitas, V. Parsifal, 2021. Available online: https://parsif.al (accessed on 17 October 2021).
20. Wohlin, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, UK, 13–14 May 2014; EASE '14; Association for Computing Machinery: New York, NY, USA, 2014. doi: [CrossRef]
21. Safaryan, O.; Pinevich, E.; Roshchina, E.; Cherckesova, L.; Kolennikova, N. Information system development for restricting access to software tool built on microservice architecture. *E3S Web Conf.* **2020**, *224*, 01041. doi: [CrossRef]
22. Shulin, Y.; Jieping, H. Research on Unified Authentication and Authorization in Microservice Architecture. In Proceedings of the International Conference on Communication Technology Proceedings, ICCT, Nanning, China, 28–31 October 2020; pp. 1169–1173. doi: [CrossRef]
23. Jin, W.; Xu, R.; You, T.; Hong, Y.G.; Kim, D. Secure edge computing management based on independent microservices providers for gateway-centric IoT networks. *IEEE Access* **2020**, *8*, 187975–187990. doi: [CrossRef]
24. Nguyen, Q.; Baker, O. Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API. *J. Softw.* **2019**, *14*, 257–264. doi: [CrossRef]
25. Yu, D.; Jin, Y.; Zhang, Y.; Zheng, X. A survey on security issues in services communication of Microservices-enabled fog applications. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e4436. doi: [CrossRef]
26. Bhutada, S.; Jyothi, K.K. Enhancing Security to the Microservice (MS) Architecture By Implementing Authentication and Authorization Service using Docker and Kubernetes. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*, 401–407.
27. Nehme, A.; Jesus, V.; Mahbub, K.; Abdallah, A. Securing Microservices. *IT Prof.* **2019**, *21*, 42–49. MITP.2018.2876987 [CrossRef]
28. Pereira-Vale, A.; Marquez, G.; Astudillo, H.; Fernandez, E.B. Security mechanisms used in microservices-based systems: A systematic mapping. In Proceedings of the 2019 45th Latin American Computing Conference, CLEI 2019, Panama City, Panama, 30 September–4 October 2019. doi: [CrossRef]
29. Banati, A.; Kail, E.; Karoczkai, K.; Kozlovszky, M. Authentication and authorization orchestrator for microservice-based software architectures; Authentication and authorization orchestrator for microservice-based software architectures. In Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018.
30. Jander, K.; Braubach, L.; Pokahr, A. Defense-in-depth and Role Authentication for Microservice Systems. *Procedia Comput. Sci.* **2018**, *130*, 456–463. doi: [CrossRef]

31. Nehme, A.; Jesus, V.; Mahbub, K.; Abdallah, A. Fine-Grained Access Control for Microservices. *Lect. Notes Comput. Sci. Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinform.* **2019**, *11358 LNCS*, 285–300. doi: [CrossRef]
32. Richter, D.; Neumann, T.; Polze, A. Security considerations for microservice architectures. In Proceedings of the CLOSER 2018-Proceedings of the 8th International Conference on Cloud Computing and Services Science, Funchal, Portugal, 19–21 March 2018; pp. 608–615. doi: [CrossRef]
33. Cao, R.; Lu, S.; Wang, X.; Xiao, H.; Chi, X. Unified Account Management for High Performance Computing as a Service with Microservice Architecture. In Proceedings of the Unified Account Management for High Performance Computing as a Service with Microservice Architecture, Taipei, Taiwan, 16–23 March 2018.
34. Lu, D.; Huang, D.; Walenstein, A.; Medhi, D. A Secure Microservice Framework for IoT. In Proceedings of the Proceedings-11th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2017, San Francisco, CA, USA, 6–9 April 2017; pp. 9–18. doi: [CrossRef]
35. Preuveneers, D.; Joosen, W. Access Control with Delegated Authorization Policy Evaluation for Data-Driven Microservice Workflows. *Future Internet* **2017**, *9*, 58. doi: [CrossRef]
36. He, X.; Yang, X. Authentication and Authorization of End User in Microservice Architecture. *J. Phys. Conf. Ser.* **2017**, *910*, e012060. doi: [CrossRef]
37. Torkura, K.A.; Sukmana, M.I.; Meinel, C. Integrating continuous security assessments in microservices and cloud native applications. In Proceedings of the UCC 2017-Proceedings of the10th International Conference on Utility and Cloud Computing, Austin, TX, USA, 5–8 December 2017; pp. 171–180. doi: [CrossRef]
38. Dooley, K. *Designing Large Scale Lans: Help for Network Designers*; O'Reilly Media: Sebastopol, CA, USA, 2001.
39. Hardt, D. Rfc 6749: The oauth 2.0 authorization framework. *Internet Eng. Task Force IETF* **2012**, *10*, 1–75.
40. Jones, M.; Bradley, J.; Sakimura, N. *Rfc 7519: Json Web Token (jwt)*; IETF: Fremont, CA, USA, 2015.
41. Foundation, O. How Is OpenID Connect Different than OpenID 2.0? 2022. Available online: https://openid.net/connect/ (accessed on 4 February 2022).
42. Rescorla, E. *Rfc 2818: HTTP over TLS*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2000.
43. Sinnema, R.; Wilde, E. *Rfc 7061: eXtensible Access Control Markup Language (XACML) XML Media Type*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2013.
44. Krawczyk, H.; Bellare, M.; Canetti, R. *MAC: Keyed-Hashing for Message Authenticatio*; *Internet Engineering Task Force (IETF)*: Fremont, CA, USA, 1997.
45. Campbell, B.; Mortimore, C.; Jones, M. *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2015.
46. Pivotal. Spring Cloud. 2021. Available online: https://spring.io/projects/spring-cloud (accessed on 14 October 2021).
47. Kit, G. Go Kit—A Toolkit for Microservices. 2022. Available online: https://gokit.io/ (accessed on 4 February 2022).
48. Projects, P. Flask Web Development. 2022. Available online: https://flask.palletsprojects.com/ (accessed on 4 February 2022).
49. Microsoft. NET Core. 2022. Available online: https://dotnet.microsoft.com/ (accessed on 4 February 2022).
50. Kong. Kong API Gateway. 2021. Available online: https://konghq.com/kong (accessed on 14 October 2021).
51. Garousi, V.; Felderer, M.; Mäntylä, M.V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Inf. Softw. Technol.* **2019**, *106*, 101–121. [CrossRef]