Naif Arab University for Security Sciences
Journal of Information Security and Cybercrimes Research
مجلة بحوث أمن المعلومات والجرائم السيبرانية
https://journals.nauss.edu.sa/index.php/JISCR

JISCR

Trusted Microservices: A Security Framework for Users' Interaction with Microservices Applications

Mohamed Elkholy[1*], and Marwa A. Marzok[2]

[1] Department of Computer Engineering, Faculty of Engineering, Pharos University in Alexandria, Alexandria, Egypt.

[2] Information Technology Department Faculty of Specific Education, Matroh University, Matroh, Egypt.

Received 06 Oct. 2022; Accepted 19 Dec. 2022; Available Online 28 Dec. 2022

Abstract

Microservices architecture emerges as a promising software design approach that provides large scale software systems with flexibility, scalability and fault tolerance. Moreover, it is considered a suitable design to be implemented using software containers provided with several cloud providers. However, microservices suffer from several security challenges that hinder its progress. The concept of microservices is to break down the system functionality to a number of small coherent services. Hence, using microservices as a design approach increases the security risks by expanding the risk surface. In contrast to microservices, monolithic applications are implemented as a bulk of codes using single programming language. Such environment has several drawbacks related to flexibility and maintainability, but limits security issues. On the other hand, microservices implementation uses several programming languages and frameworks to implement small units of system functionality. Such environment opens the door to new critical security issues. The proposed work introduces the problem of securing microservices and provides a novel approach to protect microservices applications from masquerade attacks. The proposed framework also provides high protection to users from malicious services. The framework was implemented using 150 software containers to define users' HTTP requests and a set of 20 microservices were tested to proof its applicability and benefits.

I. Introduction

Security and trust are considered elementary requirements in software design. Any new design technique (such as microservices) cannot gain its popularity without solving any related security issues [1]. Microservices provide designers with an important design aim which is separation of concerns [2]. This aim is satisfied by breaking down huge systems into a set of coherent software units. Using microservices, system functionalities are divided into a set of major tasks [3, 4]. Then each task is divided into a number of sub tasks which is further divided into a set of atomic services from the perspective of business logic [5]. Each atomic service is considered the building block of a microservices system [6]. Microservices bring about a number of benefits in terms of flexibility and scalability. However, handling security in such an open

Keywords: Cybersecurity, Microservices Architecture, Monolithic Applications, Malicious Services, Masquerade Attacks.

Production and hosting by NAUSS

* Corresponding Author: Mohamed Elkholy
Email: eng_mikholy@alexu.edu.eg

doi: 10.26735/QOPM9166

environment poses several challenging issue. Microservices security should support all microservices interactions [7]. Interactions in microservices applications can be divided into three categories. First category is the messages exchange during interaction among services. While the second category is concerned with the interaction between the users and the application. The third one is the communication between the services and the database [8]. The three proposed categories of microservices interactions are clarified in Fig. 1.

Microservices have several implementation approaches. Such approaches include physical and virtual machines and even software containers [9]. A single system could be implemented using several languages and different underlying platforms [10]. Moreover, to get the full benefit of microservices, each service should have an independent access to different databases which may vary in structure and model [3, 5]. Microservices applications have different access and deployment pattern. Such heterogeneous environment is full of security gaps that could have a bad impact of using microservices [11]. A significant number of microservices applications use individual services to accept user requests acting as a gateway for the application [6]. These gateway services further route the user request to the required service. Thus, gateway approach is considered an orchestration system with a node manager that could facilitate security implementation. Other microservices applications use chirography approach without any managing nodes [10].

Heterogeneous and open environment of microservices deployment bring up several security challenges. In such environment, traditional mechanisms of user authentication are not sufficient [12]. For instance, each user should be identified and protected against masquerade attacks. Moreover, the role of each user and his authorization to access database and use queries should be defined and protected against any malicious attacks [13]. In addition, securing microservices should protect users from malicious services [14]. A security mechanism should be able to check the running services and monitor their interactions with different

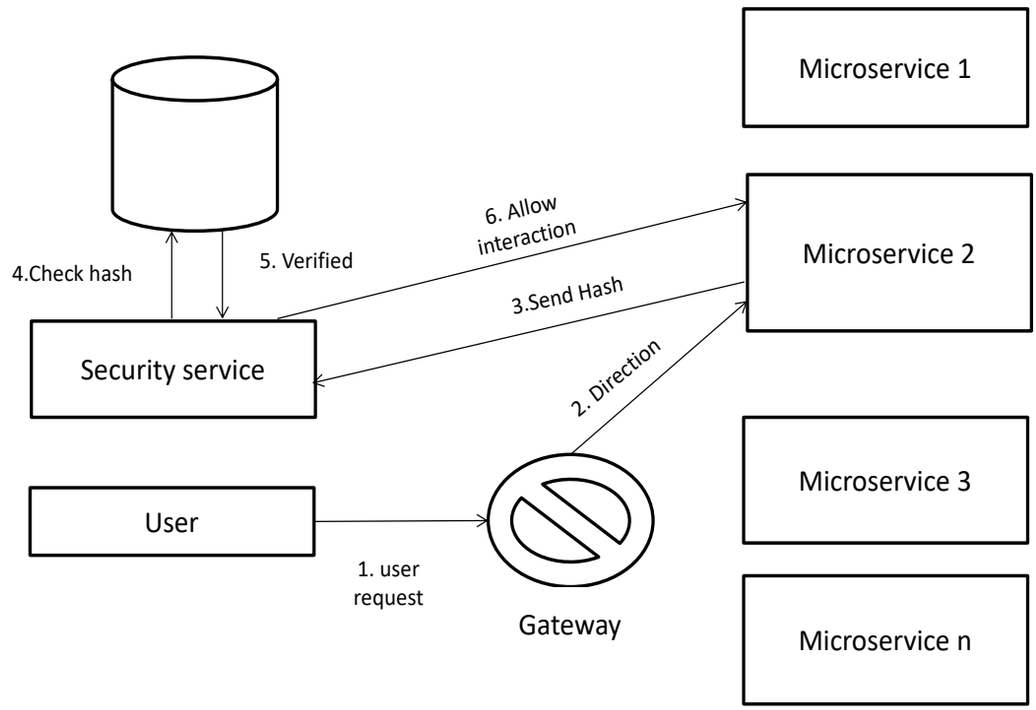

Fig. 1 Different Interactions in Microservices Applications.

users. The proposed work introduces a security framework that supports microservices with the required security needs and closes different security gaps during the interaction of users with services. The proposed research defines security gaps in microservices and analyzes the reasons of each gap. Hence, the proposed framework is designed and implemented and the experimental result proofs it enhancement and efficiency. The remaining of the paper is organized as follows; section two clarifies the microservices architecture. Section three introduces related works. Section four defines and analyzes the security risk assessment associated with microservices and problem definition. Section five illustrates the proposed framework which is implemented in section six. Section seven incorporates the conclusion and the future work.

- To analyze how using information technology highlights the risks and threats to high school students resulting from its use in the university youth category of cyber security risks.
- To determine the risks that threaten the information security of high school students.
- To determine the degree of knowledge of a sample of high school students about the dangers of using modern technology.

II. Microservices and Monolithic Architecture

A monolithic application is a bulk of huge size codes most probably written using a single language [5, 7]. The most vastly used programming languages to build are Java, C#, C/C++, Ruby and Python. The code may be divided into a group of components which are classes, methods, or functions. However, all these components are deployed together and even stopped together. Such situation provides higher security, however it lacks flexibility and fault tolerance [15]. A single failure or even a single change in a part of the system requires stopping the whole system and apply the required maintenance and then deploy the system again to be available online. Moreover, monolithic applications restrict developers from using a single set of related programming languages for the whole system functionality.

Microservices architecture rose in the last decade as a promising approach to design and implement large software systems as a set of coherent services [2, 10]. The system is broken down to a set of activities [16]. Then each activity is divided into tasks and subtasks until the whole functionality is presented as a set of independent tasks. Each task is assigned to a microservice which fulfills its functionality. Microservices inherit the basic concept of service-oriented architecture (SOA) in building a large system from integrating small reusable components [17]. Several organizations found in microservices the desired environment to design large scale systems [8]. Moreover, several implementation techniques are aligned with the concept of microservices. Software containers offered by cloud providers allow developers to focus on writing the code by offering Platform as a Service (PaaS). Such cloud service provides a developing workable environment containing the programming language compiler and all required dependencies [18]. Thus, the developer can use several languages to develop their applications without paying a large budget to install different frameworks, thereby complete the concept of microservices by enabling each task to be implemented by different languages suiting its functionality.

Microservices architecture is not considered a concrete design, as it allows several models under its umbrella [12]. For instance, microservices applications can define several roles of interaction between microservices and their databases. One model restricts database access to a single service; thus, this service acts as the database owner. When any other database wants to access this database, it communicates with its service owner [19]. Another approach allows several services to access the same database. Thus, microservices need a general framework that defines and secures each interaction between the scattered components of the application.

III. Related Work

Securing microservices is a challenge that has been largely researched in the literature and the industry. In [20], the authors provided a secure and privacy-preserving mutual authentication solution

using Elliptic Curve Cryptography (ECC). Their solution ensures mutual authentication between subscribers and brokers, and between brokers and service publishers. They tried to ensure mutual authentication, confidentiality, and message integrity, and used ECC to provide the same level of security with a key and message size lower than other public-key cryptography methods, such as RSA. Their solution was limited to using microservices for edge computing only.

In [21], Soldani et al. conducted a systematic mapping on securing microservices. Their study focused on the pains and gains of the microservices architectural style. They defined security as a design concern which provides gains as fine-grained policies, and isolation. They also defined access control, and centralized support as triggers of pain. Their work focused only on securing invocation of microservices through Application Programming Interfaces (APIs).

The authors in [22] defined an approach for monitoring and analyzing IoT service behavior using machine learning-based technique. Their approach detects any unusual service behavior by observing communication packets. They defined a mechanism to intercept the malicious traffic that may result in security or safety risks. Their approach utilized two different types of clustering algorithms, which are grid-based algorithms and k-means. The combination of microservices models and machine learning algorithms was established to enable the implementation of access control.

In [23], the authors introduced a method to classify the communication traffic between microservices using a graph-based access control model. They implemented their model with a traffic monitoring service on each communication node. The role of the monitor service is to perform Deep Packet Inspection (DPI) by intercepting each service-to-service communication. The authors enhanced traffic monitor services by a firewall mechanism to enable dropping the malicious communication packets. However, their work focused only on the inter-service communication.

In [24], Aselböck et al. defined a model that helps developers to select suitable patterns for different block chain-based applications. Their work was tested regarding its correctness and usefulness in guiding the architecture design and in understanding the rationale of various design decisions. They focused on selecting the suitable model that secures the interaction between the users and the protected system; however, they did not discuss the possibility of finding malicious services.

By analyzing different related work, we concluded that up till now there is no defined general framework that supports microservices with its security and privacy needs.

IV. Microservices Security Analyses

Microservices applications are spread across several heterogeneous services and platforms [25]. Such heterogeneity inherits difficulties while using traditional security mechanisms [26]. To provide a security mechanism, different vulnerability gaps should be analyzed.

A. Vulnerability Points and Possible Attacks

- The first security gap exists when a HTTP request is received from a user to get access to the system. Stolen credentials should be used by an intruder to get access to the system masqueraded as an authenticated user.
- Second attack may be launched by an authenticated user to access data or perform queries on a database which is not authorized to his role.
- Third attack is DoS, where an attacker sends several requests to the front end of the microservices application or to a specific service to make it go down for a period of time.
- Forth attack is done by a malicious service to steal user data during user interaction with database.
- Fifth attack takes place by a malicious service during inter-service interaction to steal user data from databases associated with the victim service.
- Sixth attack is done by an intruder how intercepts the communication links between services and listens or even changes data transferred from one service to another.

B. Research Problems

Microservices work in an open heterogeneous environment in which user authentication is essential to interact with different services. Each individual service often needs to verify that the requester is authorized to perform a certain operation. Such situation faces problem in defining the suitable mechanism to provide the user identity to each service so that to make be sure that the user has the rights to perform the required operation. One solution of such a problem is to insert a user authentication mechanism in each service. However, the security will be scattered among services and the application will suffer from different security gaps.

Moreover, in large scale applications, different users have different roles to deal with data and these roles are almost dynamic [27]. Thus, another problem will arise when dealing with modern microservices design which includes a database for each service. A user may request to access a database which is not associated with the interacting service. Hence crosscutting security aspects through services will suffer from bad performance as the user should send his credentials each time when dealing with a new service or a new database.

Another significant problem is associated with user credentials. Sending user credentials frequently to different services in the microservices application poses a threat of stealing user credentials. Afterwards, the stolen credentials might be used in a masquerade attack causing significant damage to the application.

V. Proposed Microservices Security Mechanism

The proposed framework inherits the concept of separation of concern from microservices architecture itself. Security is associated with a Microservice with acts like a guard for the other service. Thus, the security concerns are not distributed as a cross-cutting concern along all services, which may lead to degrading the performance. The proposed framework has an advantage over the previously discussed work, as it ensures the integrity of user credentials, user role, and services messages. A ticket distribution mechanism is used to identify the user and his role to access different databases. The proposed framework tracks all the security vulnerabilities that are listed in the security analyses to close all the security gaps in microservices systems.

Each user will be assigned a public and private key. The public key is distributed over all services in the application. All users registered to the application should prove their identity according to the following mechanism. On receiving the user request by the gateway services, the request is directed first to the security service. The security service is responsible of authenticating the user credentials via username and password. User credentials describe his authority to access different databases. Moreover, the security service is responsible to determine the actions the user can perform and the queries he can request. The security service issues a ticket for each user. A copy of the ticket is saved at the security service as a plain text and another copy is sent to the user. The user has to encrypt this key using his private key and to return it to the security service. User public key is used to decrypt the encrypted key and compare it with the saved plain text ticket. After matching, the user is authenticated, and the ticket becomes valid. Such process ensures the user authentication and also protects the microservices applications from masquerade attacks.

Hence, the security service responds to the gateway with a message that allows the user to be connected to the system and request the desired functionality. Thus, the user is defined and authenticated, and his role is determined. The ticket allows the user to access only the databases that are allowed to his role. Fig. 2 represents user authentication mechanism.

The gateway then directs the user request to the microservice that will fulfill the request. However, user request may be passed to several services to complete a single request. The user delivers his ticket to each service to verify his role and his kind of authorization. To close the gap of finding malicious microservices in the system, each service is required to send a hash of its code

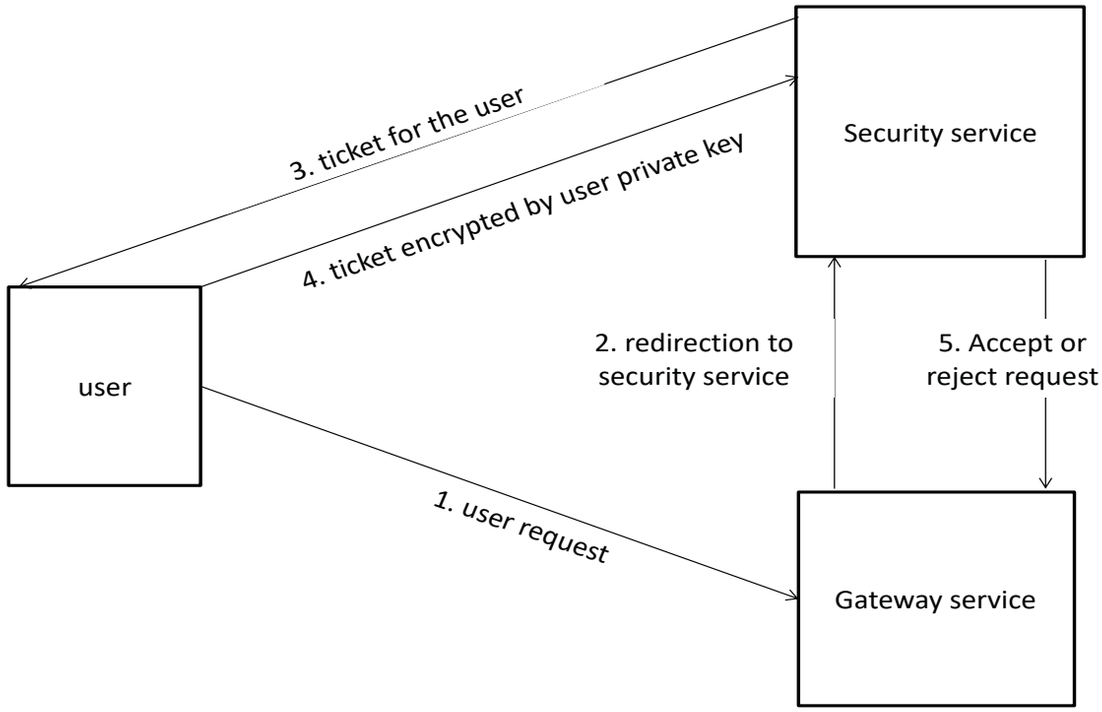

Fig. 2 User Authentication Mechanism.

to the security service before interacting with the user. Microservices are considered as stateless software components that should not get any change in code after user interaction. Hence, the security service compares the hash value sent from the service with the original hash saved in the security service database. If the two hashes are identical this indicates no change in the code and the service is allowed to interact with the user. Fig. 3 clarifies the individual service check before user interaction.

Thus, the proposed framework design covers two main issues in securing microservices systems. The first is identifying users, proving authentication and defining the role of each user. The second is securing user request from any malicious service by checking each service before user interaction.

VI. Implementation

The system was implemented using JavaScript language on Node.js. The criteria behind using Node.js is its ability to provide unblocking threads to enhance the overall performance while communicating between different components of the microservices system. The proposed system includes several checks of user ticket as well as services check. The unblocking threads features allow accepting new threads while another request is directed to check the database. The system was built on a virtual machine offered by AWS and with Linux operating system. The system used 150 software containers to define users' HTTP requests and a set of 20 Microservices. The first step is user authentication by sending username and password. Fig. 4 presents the algorithm used in authenticating and defining user role. An Apache server was used to perform the authentication using ".htaccess" and ".htpasswd" files. The username and password are stored in hash format .htaccess file references a .htpasswd file. Each line in the file consists of a username and a password separated with colon (:). Then the HTTP redirection was implemented to transfer any user request to the security service. Redirection is triggered by sending a HTTP redirect response to every user request. The individual microservices check is done by implementing MongoDB including all the

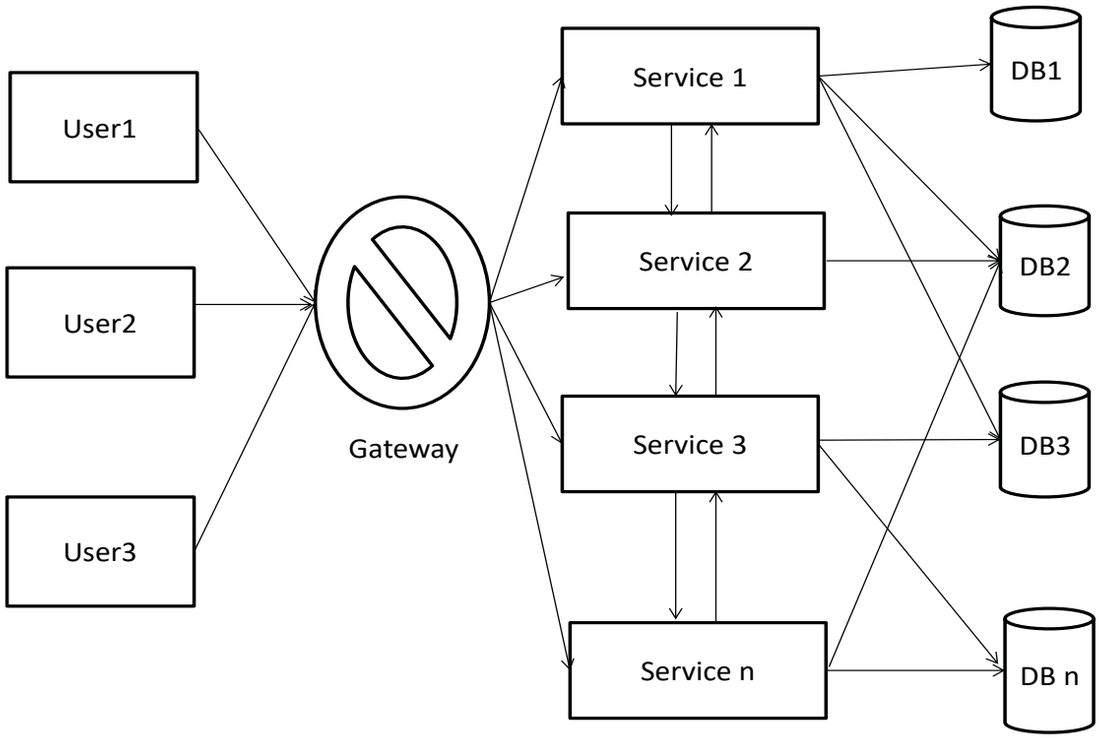

Fig. 3 Individual Service Check.

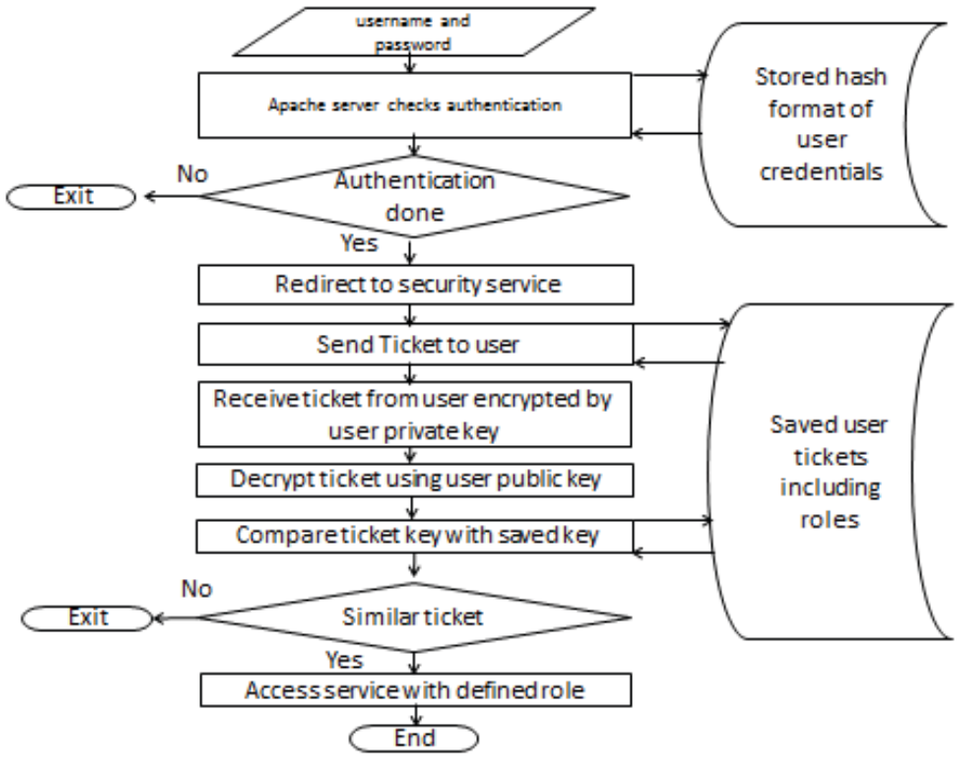

Fig. 4 Proposed Role Authentication Definition Algorithm.

hash values of different microservices. *Listing1* includes the coding concept of the proposed framework implementation.

```
AuthType Basic
AuthName "Access to the staging site"
AuthUserFile secureMicroservice/userrequest/to/.htpasswd
Require valid-user

VirtualHost *:443>
 Redirect / https://www.serviceservice.com
</VirtualHost>

const Mongoose = require("mongoose")
const localDB = `mongodb://secureMicroservice/servicehash`
const connectDB = async () => {
await Mongoose.connect(localDB, {
useNewUrlParser: true,
useUnifiedTopology: true, })
console.log("MongoDB Connected")}
module.exports = connectDB
const connectDB = require("./db");
connectDB( );
```

Listing1: Part of Framework Implementation Using JavaScript

VI. Conclusion and Future Work

The proposed work demonstrates a framework that provides microservices applications with several security needs. The framework protects microservices applications from masquerade attacks and ensures user authentication. A dedicated security service is responsible for user authentication and role identification rather than scattering security between different services. The proposed work ensures the user credentials and user role using distributed tickets. Each user is identified, and his authorization is defined according to a ticket encrypted by his private key. The security service decrypts the ticket using user public key to ensure authentication and user role. The framework also protects microservices against malicious services by checking each service before user interaction. Our future work will include securing the interaction among individual services. Several microservices applications include large size of messaging between services. Thus, it is a challenging issue to protect these messages from different attacks.

Funding

This article did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest

Authors declare that they have no conflict of interest.

References

[1] A. Chatterjee, M. W. Gerdes, P. Khatiwada and A. Prinz, "SFTSDH: Applying Spring Security Framework With TSD-Based OAuth2 to Protect Microservice Architecture APIs," *IEEE Access*, vol. 10, pp. 41914-41934, 2022, doi: 10.1109/ACCESS.2022.3165548.

[2] P. Billawa, A. B. Tukaram, N. E. D. Ferreyra, J.-P. Steghöfer, R. Scandariato, and G. Simhandl, "SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices," in *Proc. 17th Int. Conf. Availab. Reliab. Secur. (ARES '22)*, Austria, 2022, pp. 1-10, doi: 10.1145/3538969.3538986.

[3] F. Al-Doghman, N. Moustafa, I. Khalil, Z. Tari and A. Zomaya, "AI-enabled Secure Microservices in Edge Computing: Opportunities and Challenges," *IEEE Trans. Serv. Comput.*, doi: 10.1109/TSC.2022.3155447.

[4] Y. Zhang, C. Li, N. Chen and P. Zhang, "Intelligent Requests Orchestration for Microservice Management Based on Blockchain in Software Defined Networking: a Security Guarantee," in 2022 *IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Korea, 2022, pp. 254-259, doi: 10.1109/ICCWorkshops53468.2022.9814536.

[5] A. Hannousse and S. Yahiouche, "Securing microservices and microservice architectures: A systematic mapping study," *Comput. Sci. Rev.*, vol. 41, p. 100415, Aug. 2021, doi: 10.1016/j.cosrev.2021.100415.

[6] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of Microservices-enabled fog applications," *Concurr. Comput. Pract. Exp.*, vol. 31, no. 22, e4436, 2018, doi: 10.1002/cpe.4436.

[7] R. Santos, P. Soares, E. Rodrigues, P. H. M. Maia and A. Silveira, “How Blockchain and Microservices are Being Used Together: a Systematic Mapping Study,” in *2022 IEEE/ACM 5th Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, USA, 2022, pp. 39-46, doi: 10.1145/3528226.3528371.

[8] 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 *2022 IEEE Int. Conf. Consum. Electron. (ICCE)*, 2022, pp. 01-02, doi: 10.1109/ICCE53296.2022.9730757.

[9] A. Chatterjee and A. Prinz, “Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study,” *Sensors*, vol. 22, no. 5, 1703, 2022, doi: 10.3390/s22051703.

[10] M. E. Kholy and A. E. Fatatry, “Framework for Interaction Between Databases and Microservice Architecture,” *IT Prof.*, vol. 21, no. 5, pp. 57-63, 1 Sept.-Oct. 2019, doi: 10.1109/MITP.2018.2889268.

[11] D. Li, L. Deng, Z. Cai, and A. Souri, “Blockchain as a service models in the Internet of Things management: Systematic review,” *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 4, e4139, 2020, doi: 10.1002/ett.4139.

[12] X. Sun, S. Boranbaev, S. Han, H. Wang, D. Yu, “Expert system for automatic microservices identification using API similarity graph,” *Expert Syst.*, e12158, 2022, doi: 10.1111/exsy.13158.

[13] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf and D. Taibi, “Microservice Architecture Reconstruction and Visualization Techniques: A Review,” in *2022 IEEE Int. Conf. on Serv. Oriented Syst. Eng. (SOSE)*, 2022, pp. 39-48, doi: 10.1109/SOSE55356.2022.00011.

[14] F. Ponce, J. Soldani, H. Astudillo, and A. Brogi, “Should Microservice Security Smells Stay or be Refactored? Towards a Trade-off Analysis,” in *Softw. Archit. 16th Eur. Conf.*, Czech Republic, 2022, pp. 19-23, doi: 10.1007/978-3-031-16697-6_9.

[15] B. Butzin, F. Golatowski and D. Timmermann, "Microservices approach for the internet of things," in *2016 IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, 2016, pp. 1-6, doi: 10.1109/ETFA.2016.7733707.

[16] T. Yarygina and A. H. Bagge, “Overcoming Security Challenges in Microservice Architectures,” in *2022 IEEE Int. Conf. on Serv.* Oriented Syst. Eng. (SOSE), Germany, 2018, pp. 11-20, doi: 10.1109/SOSE.2018.00011.

[17] M. Elkholy, Y. Baghdadi, and M. Marzouk, “Snowball Framework for Web Service Composition in SOA Applications,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 1, 2022, doi: 10.14569/IJACSA.2022.0130143.

[18] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, “Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology,” *Internet of Things*, vol. 11, 100227, 2020, doi: 10.1016/j.iot.2020.100227.

[19] A. Pereira-Vale, G. Márquez, H. Astudillo and E. B. Fernandez, "Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping," in *2019 XLV Latin Am. Comput. Conf. (CLEI)*, 2019, pp. 01-10, doi: 10.1109/CLEI47609.2019.235060.

[20] A. Botta, W. de Donato, V. Persico, and A. Pescapé, “Integration of Cloud computing and Internet of Things: A survey,” *Future Gener. Comput. Syst.*, vol. 56, pp. 684-700, 2016, doi: 10.1016/j.future.2015.09.021.

[21] 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, 2018, doi: 10.1016/j.jss.2018.09.082.

[22] M. -O. Pahl and F. -X. Aubet, “All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection,” in *2018 14th Int. Conf. Netw. Serv. Manag. (CNSM)*, Italy, 2018, pp. 72-80.

[23] M. -O. Pahl, F. -X. Aubet, and S. Liebald, "Graph-based IoT microservice security," in *NOMS 2018 - 2018 IEEE/IFIP Netw. Oper. Manag. Symp.*, Taiwan, 2018, pp. 1-3, doi: 10.1109/NOMS.2018.8406118.

[24] S. Haselböck, R. Weinreich and G. Buchgeher, “An Expert Interview Study on Areas of Microservice Design,” in *2018 IEEE 11th Conf. Serv. -Oriented Comput. Appl. (SOCA)*, 2018, pp. 137-144, doi: 10.1109/SOCA.2018.00028.

[25] X. Zhou, et al. “Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry,” *J. Syst. Softw.*, vol. 195, 111521, 2023, doi: 10.1016/j.jss.2022.111521.

[26] A. K. Chitturi and P. Swarnalatha, “Exploration of Various Cloud Security Challenges and Threats,” in *Soft Comput. Probl. Solving*, K. N. Das, J. C. Bansal, K. Deep, A. K. Nagar, P. Pathipooranam, and R. C. Naidu, Eds., in Advances in Intelligent Systems and Computing, vol. 1057, 2019, doi: 10.1007/978-981-15-0184-5_76.

[27] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, “Security in Microservices Architectures,” *Procedia Comput. Sci.*, vol. 181, pp. 1225-1236, 2021, doi: 10.1016/j.procs.2021.01.320.