Securing microservices: challenges and best practices

Denys Volkov[1] and Vira Liubchenko[1]

[1] Odesa Polytechnic National University, 1 Shevchenko av., Odesa, 65044, Ukraine[1]

Abstract
The research studies the security challenges and best practices in microservice architecture, emphasising the need for new security approaches due to their distributed nature. It reviews existing literature, highlighting the lack of comprehensive threat models and the predominance of theoretical over practical solutions. The authors propose a detailed threat model tailored for microservices, addressing unique challenges like inter-service communication and containerisation. The model includes specific attack vectors, sources, impact points, consequences, and mitigation strategies. The document underscores the importance of integrating academic and grey literature insights to develop effective security strategies for microservice architectures.

Keywords
microservice architecture, security challenges, threat model, attack vectors, security mechanisms

1. Introduction

Microservice architecture, one of the most popular paradigms for creating software applications, especially with the widespread adoption of cloud computing, offers numerous benefits. These include scalability, development flexibility, and deployment. In contrast to monolithic architecture, a microservices architecture allows for the development and deployment of applications by breaking business logic into separate, small services, each focused on solving specific business problems. The benefits of microservice architecture, as highlighted in the book [1], go beyond scalability, offering advantages such as development speed, heterogeneity, flexibility, failure tolerance, and improved code support. These benefits, however, come with unique security challenges that this paper aims to address.

Despite its many advantages, microservice architecture has drawbacks, especially in the context of security [2].

The list of core issues includes increased areas for attacks, communication of services, trust between services, heterogeneity, logging and monitoring, dependence on the deployment environment, and threats of cloud environments.

Due to the fundamental structural differences between monolithic and microservice architectures, the methods used to ensure protection cannot be applied to the latter. This raises the problem of finding new approaches that can be used for applications consisting of many services.

Since the emergence of the concept of microservice architecture in 2014, the number of articles on microservice security has increased significantly [3]. About 100 articles on this topic were published in 2020, compared to 20 in 2016 [4]. According to Grand View Research, the global application security market size is projected to grow 18.7% from 2024 to 2030, reaching US$7.57 billion in 2023 [5].

This underscores the growing urgency of ensuring software systems' security in scientific research and business.

This work aims to systematise security threats and strategies for combating them in the context of microservice architecture based on modern data and practices.

2. Literature review

This section reviews work completed as part of a systematic literature review (SLR) on microservices security to gain insight into the current state of research and identify areas that require further study. Both academic publications and grey literature were reviewed, including industry reports, case studies,

[1]ICST-2024: Information Control Systems & Technologies, September 23 – 25, 2024, Odesa, Ukraine

volkov.denis.v17@gmail.com (D. Volkov); lvv@op.edu.ua (V. Liubchenko)
0009-0006-0933-616X (D. Volkov); 0000-0002-4611-7832 (V. Liubchenko)

technical blogs, forums, and documentation from various organisations. This approach provides a complete understanding of the topic, as grey literature often offers up-to-date information on practical solutions and the experience of specialists, which allows one to keep abreast of new trends in the rapidly evolving field of microservice architecture.

In [4], the authors examined 290 academic papers published from 2014 to 2021 on microservice security. The most common sources of information were conference publications and journal articles, while the number of books was small, indicating the evolving nature of the field. The authors note that only 30% of the works used threat models (specifically STRIDE). In 80% of cases, these models were used for specific scenarios, indicating the lack of general threat models. Most research focuses on software engineering and programming languages rather than security, which limits the creation of universal threat models for microservices. The most common approach discussed in publications is solving specific problems, such as authentication, rather than proposing a general approach. These observations highlight the lack of security approaches that address applications across the entire stack. The authors note an increase in the citation of blockchain technologies, but at the same time, there is a lack of citation of popular technologies such as service mesh and serverless.

The work [3] focused on analysing publications on security mechanisms through the study of 26 primary studies. The security categories proposed by Fernandez et al. were used: attack detection, attack stop or mitigation, attack reaction, and attack recovery. The authors found that authorisation, authentication, and credential management were the most commonly considered security mechanisms.

In [6], the authors analysed grey literature, examining 57 sources (Blog, Presentation, Tech Article, Q&A Forum, Code Repository, and Document). They cite studies by Pereira [3], which analysed 26 academic papers from 2015-2018, and Hannoussa [7], which examined 46 academic papers from 2011-2020, which allowed the creation of an ontology of threats and mechanisms for microservices. The authors note that most knowledge in microservices security is based on a few scientific publications. At the same time, the experience accumulated by industrial practitioners remains largely unknown. By comparing the results of the grey and white literature results, it was found that the security aspects of microservice architectures mentioned in the GL sources have not been thoroughly researched and documented in scientific publications (and vice versa). Also, both works emphasise the importance of using immutable containers, mtls, JWT, security patterns, and best practices. White literature discusses the advantages of using polyglot architectures without raising the issue of related problems of different life cycles, versions, and required expertise. Grey literature reveals the disadvantages of using containerised environments (insecure images, poor authentication and authorisation mechanisms)—both white and grey literature lack information about tools for reacting and recovering from attacks. Audit logs are proposed as an option, which is a difficult task for microservices.

In [7], 46 academic publications published from 2011 to 2019 were studied. The study highlighted a preference for soft-infrastructure layer solutions over other layers, such as communication and deployment. Performance analysis and case studies emerged as the most common validation techniques for security proposals. The systematic mapping also produced a lightweight ontology for security patterns in microservice architectures. The authors advocate for further research, particularly in addressing internal attacks, proposing mitigation techniques, and focusing on various MSA layers, including communication and deployment.

In [8], the authors analysed academic and grey literature, selecting 36 and 34 grey publications. The authors point out that this work is a continuation of their previous work (in this one, they added grey literature and not only mechanisms), where they studied 26 works published from 2015 to 2019 to determine the most mentioned security mechanisms (top 3 - authorisation, authentication, credentials). They used the classification of security mechanisms proposed by Uzunov et al. [9] and the security taxonomy proposed by Fernandez et al [10]. The authors note that grey literature is characterised by the absence of works of the type "solution proposal" and "validation studies." In contrast, academic literature is not characterised by "opinions" and "personal experience." From the statistics of the number of publications from 2015 to 2019, we can conclude that with the passage of a couple of years since the beginning of the use of the term microservice, the popularity of this topic in the scientific community is growing, which is evident from the fact that the number of white literature has exceeded grey. The authors state that the grey literature does not use models (only 3% use UML). However, the complexity of security solutions requires a higher level of abstraction, which may lead to more threats in the future. The authors highlight such gaps as Lack of attention to "react to" and "recover from" attack security contexts, little work on intrusion detection and monitoring systems, lack of security patterns, secure microservice-based application development, and use of trusted hardware.

The authors of [11] analysed 58 publications (10 academic and grey) published from 2014 to 2020. The paper identifies ten security smells, each associated with specific security properties like integrity, authenticity, and confidentiality as defined in the ISO/IEC 25010 standard. It also discusses the refactoring necessary to address these smells, providing practical insights for improving the security of microservice-based applications.

The work [12] presents a systematic literature review on security threats and mitigation strategies in microservice architectures. Analysing 54 publications from 2015 to 2023, the authors identified the main security threats and methods for eliminating them in the context of inter-service interactions. Table 1 presents the general characteristics of the studied works.

Table 1
Existing literature reviews

Name	Year of public ation	Content	Number of works studied	Type of literature studied	Year of publication of the studied works
Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping [3]	2019	Mitigation strategies	26	White	2015-2018
Securing microservices and microservice architectures: A systematic mapping study [7]	2020	Threats and mitigation strategies	46	White	2011-2019
Security in Microservice-Based Systems: A Multivocal Literature Review [8]	2021	Mitigation strategies	70	White and Grey	2015-2019
Microservice security: a systematic literature review [4]	2022	Threats	290	White	2014-2021
SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices [6]	2022	Threats and mitigation strategies	57	Grey	2011-2021
Smells and Refactorings for Microservices Security: A Multivocal Literature Review [11]	2022	Threats and mitigation strategies	58	White and Grey	2014-2020
A Systematic Literature Review of Inter-Service Security Threats and Mitigation Strategies in Microservice Architectures [12]	2024	Threats and mitigation strategies	54	White	2015-2023

An analysis of the literature shows that microservice security is an area that is actively developing and has many unresolved issues. The most studied aspects are threat models and security mechanisms such as authorisation, authentication and credential management. However, there is an apparent lack of common threat models and universal security approaches. Despite significant theoretical research, the practical application of security mechanisms still requires improvement and standardisation. Grey literature complements academic study by providing up-to-date information on real-world applications and problems practitioners face.

It is essential to note the need to integrate knowledge from both sources to develop effective and comprehensive security strategies for microservice architectures.

3. Threat model design

3.1. Threat modelling problem

As we studied the SLR of the relevant research, one common problem inherent in each of the works became visible. This problem means the lack of a specialised threat model for microservice architecture. The use of a specialised threat model, which concerns not only the microservice architecture but also in general for computer science, has the following advantages:

- Individual approach depending on the context of the system and its unique security requirements.
- Improved understanding of potential threats.
- Clear clarity during communication between stakeholders.
- Flexibility as new risks and/or threats emerge.

Despite all the advantages, using a specific threat model for microservice architecture was not identified. The studied works can be divided into two categories: works that rely on generally accepted threat models, such as STRIDE, PASTA, and OWASP [4], and works that, instead of using a threat model, use threat categorisation. The difference between a threat model and categorisation is that categorisation provides a high-level view. At the same time, modelling goes into more detail, aiming to provide insight into how threats affect a system. The first group includes work [4] in which the authors considered works that use STRIDE or its components. Although the STRIDE model is a standard, it is not entirely suitable for use within a microservice architecture due to its distributed nature, many components and inter-service interaction.

It is worth noting that the same work observed no general threat model for microservices. In [6], all security problems were divided into eight categories: Trust between services, Large attack area, Testing, Container management, Low Visibility, Secret management, Polyglot Architecture, and Others. The authors of [7] developed their classification (due to the abundance of threat taxonomies) based on attack targets, identifying four main categories: User-based Attacks, Data Attacks, Infrastructure Attacks, and Software attacks.

In [11], a classification was proposed based on three security properties of microservices Confidentiality, Integrity, Authenticity and which consists of 10 categories: Insufficient Access Control, Publicly Accessible Microservices, Unnecessary Privileges to Microservices, Own Crypto Code, Non-Encrypted Data Exposure, Hardcoded Secrets, Non-Secured Service-to-Service Communications, Unauthenticated Traffic, Multiple User Authentication, Centralized Authorization.

In the last reviewed work [12], the classification of threats was presented in 12 categories: Security Perimeters and Attack Surface, Container and Orchestration Threats, Insufficient Monitoring and Intrusion Mechanisms, Network Attacks, Configuration, Infrastructure and Deployment Threats, Service Mesh and Sidecar Threats, Software and Dependency Threats, Data Leakage and Exposure, Insecure Inter–Service Communication, User Authentication and Authorization Threats, Inter–Service Authentication and Authorization Threats, Other Threats.

As can be seen, each work uses a different classification, making comparing the threats discussed in the works difficult.

In addition to the versatility of threat classifications, there is another significant problem. In [6], the authors decomposed the microservice architecture into six levels: Hardware, Virtualization, Cloud, Communication, Service/Application, and Orchestration.

The problem is that some of the classifications consider threats that are not the responsibility of the system developers, except in cases where they use their servers and hardware. Such threats include threats from the Hardware, Virtualization and Cloud levels. Accordingly, their addition to the threat taxonomy makes it less accurate.

3.2. Proposed threat model

Summarising the analysis of threat models and their classifications, we can conclude that, at the moment, there is no general threat model for microservice architecture that can be used as a basis for other research on this topic.

That is why a threat model for microservices was created to answer the research questions posed at the beginning of the section.

The process of creating a threat model for a microservice architecture was divided into two stages:

- Define a high-level view of the threat model and establish a common framework for identifying and classifying potential threats.
- Extending the model by adding threat data and protection mechanisms. Incorporate detailed threat information to improve model specificity and relevance and integrate targeted security strategies and controls to address identified threats.

Attack vectors were found to be an ideal solution to act as the basis for the proposed attack model. Attack vectors refer to areas and components of a microservice architecture that are susceptible to attack by attackers.

This makes the proposed model beneficial for microservice architecture as the model focuses on real-world threats and provides practical relevance.

It's worth noting that the order used in the list above does not correlate with the frequency with which these threats occur. However, if we talk about the most frequently discussed threats, they are authentication, authorisation, and inter-service interaction.

Compared to popular models like STRIDE and OWASP, the proposed model offers several key advantages:

- It addresses unique microservice challenges such as inter-service communication, containerisation, orchestration, and multilingual environments, making it more effective for microservices-based applications.
- The model evaluates a wide range of attack vectors, providing a thorough analysis and actionable security measures without needing extensive adaptation.
- It focuses on real-world threats, offering straightforward, actionable recommendations that are easier to implement in a microservices environment.
- The model is easily customisable and scalable, allowing it to evolve with emerging threats and specific organisational needs.
- By incorporating detailed threat information and targeted security strategies, the model maintains its relevance and effectiveness.

In the second stage, this model was expanded by defining the "source", "impact point", "expected consequences", and "mitigation strategies" characteristics for each attack vector. This approach goes beyond conventional threat models by offering a structured and detailed view of each threat. It increases clarity by identifying the origin of vulnerabilities, highlighting affected components, and providing a clear picture of the potential impact on the system.

The source determines the origin of the threat, allowing targeted prevention measures to be taken. The impact point clarifies where a threat can cause harm, helping to prioritise protection in critical areas.

Expected consequences indicate potential damage, emphasising the severity and urgency of addressing each threat. Including impact strategies directly related to each attack vector provides straightforward, actionable steps to address specific threats, ensuring that security controls are immediately relevant and practical.

It improves risk management by allowing one to accurately prioritise and develop targeted risk mitigation strategies, ensuring that security controls target the most significant threats. Additionally, this detailed model promotes better communication and understanding among stakeholders, supports compliance and auditing requirements, and ultimately provides more robust and more resilient security for microservices-based applications.

The results obtained are presented in the following list.

1. Inter-Service Communication:
    - Source. Insecure communication protocols, lack of encryption, vulnerabilities in communication protocols and encryption algorithms.
    - Impact Point. Inter-service communication channels.
    - Expected Consequences. Man-in-the-middle attacks, eavesdropping, data tampering, replay attacks, exposure of sensitive data, spoofing, data repudiation, Server-Side Request Forgery (SSRF) attacks, and session hijacking.

- Mitigation Strategies. Use strong encryption (TLS, mTLS) for service-to-service communication, validate and update protocols regularly, and use OAuth 2.0, OpenID Connect, JWT, HTTPS, FTPS, and API Security.

2. Containerization:
   - Source. Container vulnerabilities, misconfigurations, excessive privileges.
   - Impact Point. Container environments, container underlying operating systems and resources.
   - Expected Consequences. Container breaches, unauthorised access, privilege escalation, data leaks, disruption of services by exhausting the container's resources, and sandbox escape attacks.
   - Mitigation Strategies. Regularly update and patch container images, use the least privilege principle, implement runtime security tools, enforce container isolation and sandboxing, use immutable containers, container vulnerability scanning, and hardware-supported sandboxes (SGX).
3. Docker Images:
   - Source. Vulnerable or malicious images, lack of updates, using sensitive information in images, misconfiguration.
   - Impact Point. Inherited docker images and deployed containers.
   - Expected Consequences. Exposure of sensitive data, excessive privileges, introduction of vulnerabilities, execution of malicious code, compromise of entire applications, and infrastructure attacks.
   - Mitigation Strategies. Use trusted image registries, scan images for vulnerabilities, automate image updates, and avoid embedding sensitive information in images.
4. Orchestration (Deployment):
   - Source. Orchestration tools vulnerabilities, misconfigurations, compromised discovery service.
   - Impact Point. Orchestration platforms deployed application.
   - Expected Consequences. Unauthorised deployment changes, disruption of services, exploitation of orchestration tool vulnerabilities.
   - Mitigation Strategies. Secure the orchestration platform, use role-based access control (RBAC), implement configuration management tools, and regularly update orchestration tools.
5. Third-party dependencies (tools, utilities, frameworks, libraries):
   - Source. Vulnerable third-party code and outdated libraries.
   - Impact Point. Application dependencies, integration points, source code.
   - Expected Consequences. Increased attack surface, exploitation of known weaknesses, data breaches, unauthorised access, and injection attacks.
   - Mitigation Strategies. Perform regular dependency audits, use dependency management tools, update libraries frequently, avoid unnecessary dependencies, apply DevSecOps, and use SAST/DAST techniques.
6. Lack of Security Patterns & Design Principles:
   - Source. Inadequate security design and absence of standardised security practices.
   - Impact Point. Application architecture, overall system design, source code.
   - Expected Consequences. Increased attack surface, inconsistent security controls, potential for systemic vulnerabilities, data breaches, data spoofing, tampering, injection attacks, and denial of service.
   - Mitigation Strategies. Adopt security design patterns, enforce coding standards, use secure design patterns (API Gateway, Circuit Breaker, CQRS, Strangler), and secure by design.
7. Monitoring & Logging:
   - Source. Insufficient monitoring, lack of logging, misconfigurations.
   - Impact Point. Monitoring systems, log management.

- Expected Consequences. Inability to detect and respond to attacks, delayed incident response, lack of forensic data, failure to detect intrusions, breaches of sensitive information, and malicious insiders.
- Mitigation Strategies. Implement centralised logging, use monitoring tools, set up alerting mechanisms, continuous monitoring, intrusion detection systems (IDS), and log aggregation.

8. Exposing services and data publicly:
   - Source. Publicly accessible data, exposed APIs.
   - Impact Point. Public-facing microservices' APIs and data.
   - Expected Consequences. Unauthorised access, data breaches, exploitation of exposed APIs, increased attack surface, denial of services, data tampering and spoofing, Cross-Site Request Forgery (CSRF), and sniffing attacks.
   - Mitigation Strategies. Implement API gateways, use authentication and authorisation for APIs, encrypt sensitive data, restrict public access, firewall, OAuth 2.0, OpenID Connect, SSO, JWT.
9. Identity & Access Management (IAM):
   - Source. Weak identity management, improper access controls, and lack of access controls.
   - Impact Point. User authentication and authorisation systems, access control mechanisms.
   - Expected Consequences. Unauthorised access, privilege escalation, identity spoofing, repudiation, data breaches, brute force attack, malicious insider.
   - Mitigation Strategies. Enforce robust authentication mechanisms, implement multi-factor authentication (MFA), standardise access control mechanisms, use IAM tools, and enforce least privilege principles, ABAC, and RBAC.
10. Authentication & Authorization:
    - Source. Non-uniform access control, centralised authorisation issues.
    - Impact Point. Authentication and authorisation systems.
    - Expected Consequences. Access control bypass, increased risk of compromised credentials.
    - Mitigation Strategies. Use decentralised authorisation solutions, implement access control policies, OAuth 2.0, JWT, and RBAC.
11. Data Management:
    - Source. Improper data handling and lack of encryption.
    - Impact Point. Data storage, data transmission (transit channels).
    - Expected Consequences. Data leaks, unauthorised data access, data tampering, loss of data integrity.
    - Mitigation Strategies. Encrypt data at rest and in transit, enforce data handling policies, use data classification and labelling, regularly audit data access, and use secret vaults (HashiVault, Microsoft Azure Key Vault).
12. Encryption:
    - Source. Inadequate encryption practices own crypto algorithms.
    - Impact Point. Data storage, data transmission (transit channels).
    - Expected Consequences. Exposure of sensitive data, data tampering, man-in-the-middle attacks, data integrity, and sniffing attacks.
    - Mitigation Strategies. Use standard encryption algorithms, regularly update encryption protocols, implement critical management solutions, ensure end-to-end encryption, and avoid crypto code.
13. Configuration Management:
    - Source. Misconfigurations and lack of centralised management.
    - Impact Point. Configuration files and deployment scripts.
    - Expected Consequences. Security misconfigurations, unauthorised access, system disruptions, unauthorised system changes, and exposure of sensitive information.
    - Mitigation Strategies. Use configuration management tools, enforce configuration baselines, regularly review and update configurations, and automate configuration deployment.
14. Polyglot Approach:
    - Source. Use of multiple programming languages and frameworks.

- Impact Point. Application components and integration points.
- Expected Consequences. Inconsistent security practices, increased complexity, integration vulnerabilities, and increased attack surface.
- Mitigation Strategies. Standardise security practices and policies across languages, use integration testing tools, secure by design, and micro-segmentation.

15. Backup & Recovery:
    - Source. Inadequate backup procedures, lack of recovery planning, and lack of encryption.
    - Impact Point. Backup systems and disaster recovery plans.
    - Expected Consequences. Data loss, data integrity, prolonged recovery times, and inability to restore services after an incident.
    - Mitigation Strategies. Implement regular backup procedures, encrypt backups, maintain offsite backups, and use immutable backups.

In conclusion, the presented threat model provides a comprehensive framework to identify, assess, and mitigate security risks in microservice architectures. By specifying the source, impact point, expected consequences, and tailored mitigation strategies for each attack vector, this model addresses the limitations of traditional threat models, which often lack practical applicability and depth. The step-by-step approach ensures a thorough examination of potential vulnerabilities, facilitating more effective security measures.

This model enhances the understanding of security threats and empowers developers and security professionals to implement robust protection mechanisms, thereby significantly improving the overall security posture of microservice-based systems.

3.3. Application of threat model

This section will discuss the practical application of the threat model presented above. For this purpose, we will take a standard application model based on microservice architecture and use the model to identify the points of impact. It is worth noting right away that we are not tasked with defining an exhaustive list of impact points—such an endeavour would be impractical. Instead, the main goal is to demonstrate how the model can simplify identifying weak points within the application.

The considered architecture is intentionally simplified to emphasise not the architecture itself but how the threat model can be applied. The architecture from the article [13] was taken as a basis and extended slightly for a more comprehensive picture. In particular, the application's interaction with the environment in which it is deployed (cluster and node) is shown, and additional services are added for completeness. Additionally, only two services related to business logic are considered in detail. This is done to simplify the diagram and maintain focus on the threat model rather than the application architecture.

The architecture diagram, depicted in Figure 1, will serve as the basis for identifying and discussing impact points based on the proposed threat model. The impact points are marked on the diagram with circles with two numbers inside. The first number corresponds to a category in the threat model (attack vector). The second number is needed to distinguish between threats specific to one attack vector. The following will give a brief description of each of the scenarios:

1. Inter-service communication
   - 1.1. Direct communication between separate services.
   - 1.2. Indirect communication between services via message bus.
   - 1.3. Communication between orchestration services and the environment in which the application is deployed.
   - 1.4. Communication between application services and third-party services.
   - 1.5. Communication between individual service containers.
2. Containerization
   - 2.1. Individual service containers.
   - 2.2. Shared resources (CPU and memory) that are dedicated to all containers in the service.
3. Docker Images
   - 3.1. Basic docker images that are in public repositories.
   - 3.2. Custom docker images containing the business logic of the service.
   - 3.3. Service container, since the docker image is its backbone.

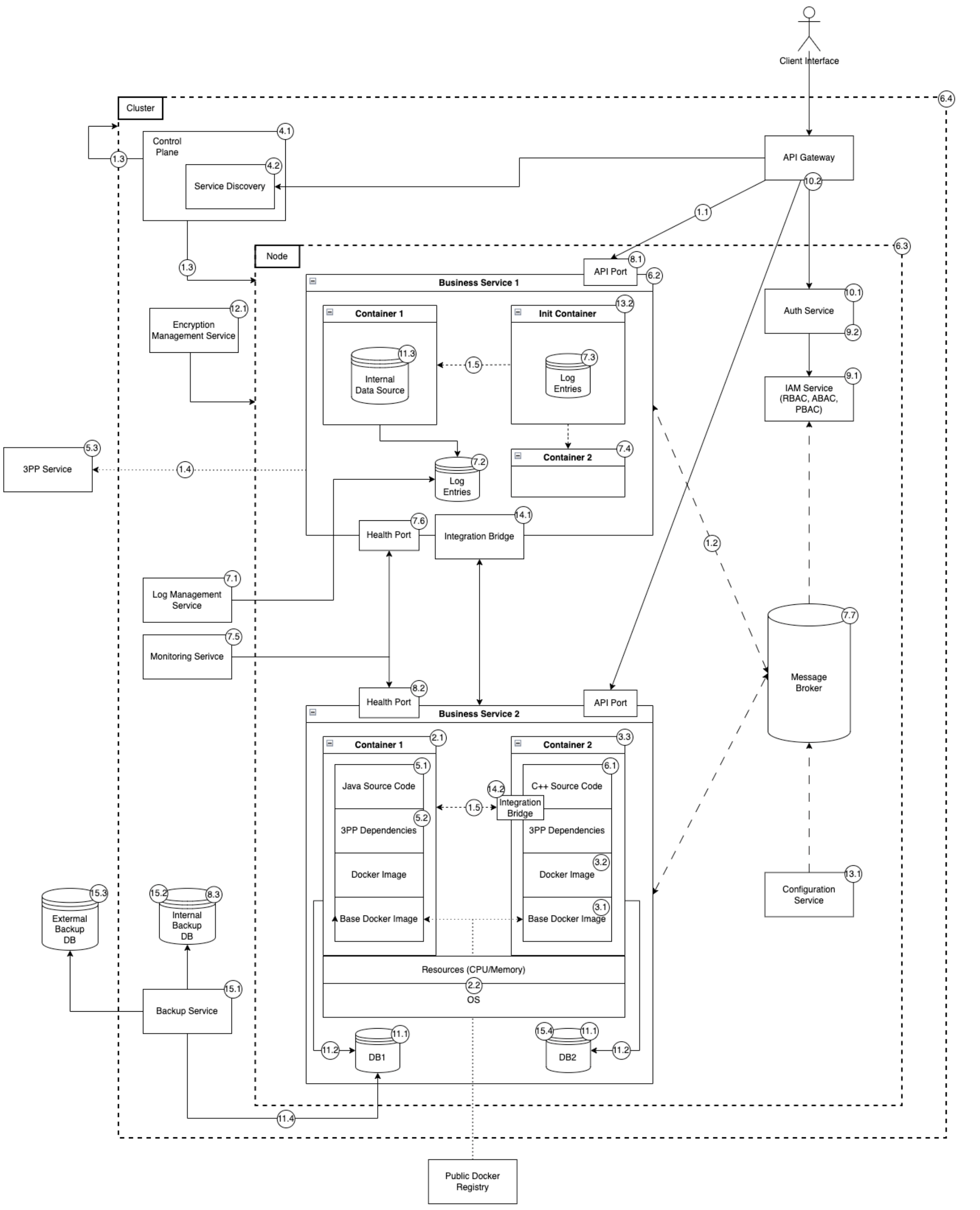

Figure 1: Microservice application with identified impact points

4. Orchestration (Deployment)
   4.1. Services responsible for administering the environment in which the application is deployed.
   4.2. Services responsible for application service discovery.
5. Third-party dependencies (tools, utilities, frameworks, libraries)
   5.1. Container source code.
   5.2. Auxiliary utilities used in the container.
   5.3. Third-party services
6. Lack of Security Patterns & Design Principles

    - 6.1. Source code.
    - 6.2. Service architecture.
    - 6.3. Application service architecture.
    - 6.4. Overall architecture.
7. Monitoring & Logging
    - 7.1. Log management service responsible for receiving logs from services and processing them
    - 7.2. Container log entries that are stored out-of-service.
    - 7.3. Container log entries that are not stored out-of-service.
    - 7.4. Container that does not create records.
    - 7.5. Monitoring service responsible for checking the status of services.
    - 7.6. Service that is monitored via a separate port.
    - 7.7. Service that is not monitored.
8. Exposing services and data publicly
    - 8.1. Port used for communication between services.
    - 8.2. Port used for internal use.
    - 8.3. Internal databases.
9. Identity & Access Management (IAM)
    - 9.1. Service responsible for checking system user rights.
    - 9.2. Authentication service.
10. Authentication & Authorization
    - 10.1. Authentication service.
    - 10.2. Communication between the application entry point (API Gateway) and authentication services.
11. Data Management
    - 11.1. Service databases.
    - 11.2. Communication between the service/container and its database.
    - 11.3. Container databases.
    - 11.4. Communication between the service database and other services.
12. Encryption
    - 12.1. The service encrypts data transferred between all application parts and stores secrets (passwords, certificates).
13. Configuration Management
    - 13.1. Service responsible for configuration of services.
    - 13.2. Container that configures its service.
14. Polyglot Approach
    - 14.1. Integration bridge between services that are not directly compatible.
    - 14.2. Integration bridge between containers that are not directly compatible.
15. Backup & Recovery
    - 15.1. Service responsible for creating copies of data stored in services.
    - 15.2. A copy of the data stored in a local database.
    - 15.3. A copy of the data stored in a remote database.
    - 15.4. Data that is not subject to the backup process.

While simplified, the considered architecture serves as a robust foundation to showcase the threat model's applicability.

It emphasises key aspects of a microservice architecture, including inter-service communication, containerisation, monitoring and logging, authentication and authorisation, data management, encryption, and more. By mapping the threat model to this architecture, we demonstrate its effectiveness in uncovering vulnerabilities and guiding the implementation of security measures.

4. Overlooked areas in security research

After introducing the threat model, we must address the identified limitations when analysing the research articles. These limitations highlight several critical gaps and challenges that must be considered to enhance the practical application of security models. Here are the expanded limitations.

Lack of Coverage of Practical Solutions. Many research articles focus heavily on theoretical knowledge and conceptual frameworks, often neglecting practical, real-world solutions. This theoretical focus can make it difficult for practitioners to apply the research findings to actual systems, as they lack concrete, actionable steps or examples. The absence of practical guidance reduces the utility of these studies for industry professionals who need to implement and test these solutions in dynamic, real-world environments. For example, we can cite the works [13]. In contrast, we can put such works as those where the authors implemented their ideas from a practical point of view [14].

Absence of Comparative Reviews of Protection Mechanisms. Authors typically fail to provide a comparative analysis of alternative solutions when discussing protection mechanisms. This omission makes evaluating different approaches' relative effectiveness, advantages, and drawbacks challenging. Without such comparative reviews, it is difficult to determine if a proposed solution is the best available option, limiting the ability to make informed decisions about security implementations. The work [15] presents the Hierarchical Role Based Access Control Model. Although it complements the RBAC mechanism, there is no comparison between them in the article, let alone a comparison with ABAC. This makes it unclear in which cases to use which of the mechanisms.

Lack of Integration of Multiple Approaches. The research often treats security mechanisms in isolation without considering the potential benefits of integrating multiple approaches to achieve a more robust security posture.

For instance, combining various detection, prevention, and response strategies could address a broader range of threats more effectively. However, the burden of integrating these approaches falls on developers, as the research does not guide how to synergise different techniques. An example is the work carried out by Ericsson employees [16], in which they presented a tool that combines several static and dynamic vulnerability scanners in place, making them easier to configure and work with. Several scanners are used due to the desire to cover as many potential vulnerabilities as possible.

Narrow Focus on Security, Ignoring Other Characteristics. Most works evaluate protection mechanisms purely from a security standpoint, neglecting other essential characteristics such as performance, efficiency, and maintainability. This narrow focus can lead to implementing security measures that, while effective, may degrade system performance, increase resource consumption, or complicate maintenance.

A more holistic approach considering these additional factors is necessary to ensure that security solutions are robust and practical for long-term use.

5. Conclusions

While microservice architectures provide substantial benefits regarding scalability, flexibility, and accelerated development cycles, they also introduce distinct security challenges that traditional monolithic architectures do not encounter. Our systematic literature review underscores the complexity and variety of these security issues, ranging from increased attack surfaces and inter-service communication vulnerabilities to the heterogeneity of services and the need for robust identity and access management.

Our analysis has identified the fragmentation in existing approaches and the lack of comprehensive, unified threat models explicitly tailored for microservices. This gap suggests that existing security frameworks and methodologies are not fully equipped to handle the nuanced demands of microservice architectures. This article presents a novel threat model designed explicitly for microservice environments. This new model addresses the identified gaps by providing a structured approach to identifying, categorising, and mitigating security threats in microservices.

The new threat model was developed through a systematic, step-by-step process:

- Defining a High-Level View: We established a broad overview of the threat landscape for microservices.
- Extending the Model: We incorporated detailed threat data to cover various potential vulnerabilities.
- Adding Protection Mechanisms: We integrated comprehensive mitigation strategies and mechanisms, ensuring the model is practical and applicable to real-world scenarios.

Moreover, our review indicates a predominant focus on theoretical aspects within academic research, with limited practical implementation and evaluation of proposed security mechanisms. This disconnect highlights the necessity for more research that bridges the gap between theory and practice, ensuring that security solutions are not only conceptually sound but also practically viable and effective in real-world deployments.

The findings emphasise the importance of adopting a holistic view of security that integrates insights from both academic research and grey literature, including industry reports, case studies, and practitioner experiences. Such integration is crucial for developing security strategies that are both comprehensive and applicable across diverse microservice environments.

Additionally, our review identifies several areas where current security mechanisms fall short. Notably, a lack of comparative analysis of different security solutions makes determining the most effective approaches challenging. Furthermore, existing research often overlooks the potential benefits of combining multiple security mechanisms to achieve enhanced protection, leaving developers to navigate these complexities independently.

To address these limitations, future research should prioritise the development of standardised, unified threat models designed explicitly for microservice architectures. These models should be flexible enough to accommodate microservices' dynamic and distributed nature while providing clear guidelines for implementation and best practices.

References

[1] I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen, Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly, Sebastopol, CA, 2016.

[2] F. Al-Doghman, N. Moustafa, I. Khalil, N. Sohrabi, Z. Tari, A. Y. Zomaya, AI-Enabled Secure Microservices in Edge Computing: Opportunities and Challenges, IEEE Transactions on Services Computing 16.2 (2023) 1485–1504. doi: 10.1109/TSC.2022.3155447.

[3] A. Pereira-Vale, G. Márquez, H. Astudillo, E. B. Fernandez, Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping, in: XLV Latin American Computing Conference (CLEI), 2019, pp. 01–10. doi: 10.1109/CLEI47609.2019.235060.

[4] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, M. Prandini, Microservice security: a systematic literature review, PeerJ Computer Science 8 (2022) e779. doi: 10.7717/peerj-cs.779.

[5] Grand View Research, Application Security Market Size & Trends, 2023. URL: https://www.grandviewresearch.com/industry-analysis/application-security-market.

[6] P. Billawa, A. B. Tukaram, N. E. D. Ferreyra, J. Steghöfer, R. Scandariato, G. Simhandl, SoK: Security of Microservice Applications: A Practitioners' Perspective on Challenges and Best Practices, in: Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22), 2022, pp. 1–10. doi: 10.1145/3538969.3538986.

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

[8] A. Pereira-Vale, E. Fernández, R. Monge, H. Astudillo, G. Márquez, Security in Microservice-Based Systems: A Multivocal Literature Review, Computers & Security 103 (2021) 102200. doi: 10.1016/j.cose.2021.102200.

[9] A. V. Uzunov, E. B. Fernandez, K. Falkner, ASE: A comprehensive pattern-driven security methodology for distributed systems, Computer Standards & Interfaces 41 (2015) 112–137. doi: 10.1016/j.csi.2015. 02.011.

[10] E. B. Fernandez, H. Astudillo, G. Pedraza-García, Revisiting architectural tactics for security, in: European Conference on Software Architecture, 2015, pp. 55–69. doi:10.1007/978-3-319-23727-5_5.

[11] F. Ponce, J. Soldani, H. Astudillo, A. Brogi, Smells and refactorings for microservices security: A multivocal literature review, Journal of Systems and Software 192 (2022) 111393. doi: 10.1016/j.jss.2022.111393.

[12] P. Haindl, P. Kochberger, M. Sveggen, A Systematic Literature Review of Inter-Service Security Threats and Mitigation Strategies in Microservice Architectures, IEEE Access 12 (2024) 90252–90286. doi: 10.1109/ACCESS.2024.3406500.

[13] R. S. de O. Júnior, R. C. A. da Silva, M. S. Santos, D. W. Albuquerque, H. O. Almeida, D. F. S. Santos, An Extensible and Secure Architecture based on Microservices, in: IEEE International Conference on Consumer Electronics (ICCE), 2022, pp. 01–02. doi: 10.1109/ICCE53296.2022.9730757.

[14] C. Meadows, T.Wood, S. Hounsinou, G. Bloom, Sidecar-based Path-aware Security for Microservices, in: Proceedings of the 28th ACM Symposium on Access Control Models and Technologies, 2023, pp. 157–162. doi: 10.1145/3589608.3594742.
[15] C. Pasomsup, Y. Limpiyakorn, HT-RBAC: A Design of Role-based Access Control Model for Microservice Security Manager, in: International Conference on Big Data Engineering and Education (BDEE), 2021, pp. 177–181. doi: 10.1109/BDEE52938.2021.00038.
[16] B. Ünver, R. Britto, Automatic Detection of Security Deficiencies and Refactoring Advises for Microservices, in: IEEE/ACM International Conference on Software and System Processes (ICSSP), 2023, pp. 25–34. doi: 10.1109/ICSSP59042.2023.00013.