**EVALUATE THE IMPACT OF MICROFRONTENDS ON CLOUD-BASED APPLICATION ARCHITECTURE**

**MAI CHUNG TUAN**

**FINAL THESIS**

**MAY 2024**

# DEDICATION

I want to dedicate this thesis work to my wife, Quynh Nguyen. She has been a consistent source of support and motivation during the difficulties of graduate school and life. I am genuinely grateful to have her in my life. I also want to dedicate this work to my parents, who have always unconditionally loved me and whose positive examples have guided me to strive hard for the things I aim to accomplish.

# ACKNOWLEDGEMENTS

I would like to thank Upgrad and LJMU for giving me the chance to do this thesis work for them, and especially, big thanks to my supervisor at LJMU, Dr. Ehsan Rana, for her constructive feedback and unwavering guidance throughout the thesis process.

I must also extend my sincere appreciation to all the interviewees and participants who generously shared their time and invaluable insights during this research. Their contributions form the foundation of this thesis, and I am truly grateful for their perspectives.

Lastly, thank you to all who have played a role, large or small, in making this endeavor possible.

# ABSTRACT

This study investigates the impact of applying micro-frontend development to cloud-based applications. It will explore the impact of micro-frontend to the scalability, cost-effectiveness, and securing of cloud-based applications. The primary objectives are structured around comprehensive case studies, performance evaluations, and the formulation of best practices. Through these objectives, the research seeks to provide actionable guidelines and insights for effectively integrating micro-frontends within cloud-based applications. Real-world implementations of micro-frontends will be examined to gain practical insights, performance assessments will be conducted in simulated cloud environments, and best practices will be developed for seamless integration. The scope of the study is broad, encompassing a thorough examination of how micro frontend development influences the scalability, cost-effectiveness, and security of cloud-based application architectures. Employing a multifaceted approach, including case studies and performance evaluations, the study aims to address key aspects of this integration. By considering a variety of real-world scenarios and diverse cloud platforms, the research aims to provide a holistic understanding of the challenges and opportunities associated with the integration of micro-frontends into cloud-based applications. The methodologies, case studies, and analyses detailed in subsequent sections contribute to the overarching goal of offering valuable insights to the dynamic landscape of web application development, enabling organizations to make informed architectural decisions and achieve optimal outcomes.

***Keywords:******Security, systematic mapping, micro-frontends, cloud-based application***

# LIST OF ABBREVIATIONS

AWS Amazon web service

MA Monolith architecture

MFE Micro frontend

MFA Micro-frontend architecture

MSA Microservice architecture

GCP Google cloud platform

SaaS Software as a service

IaaS Infrastructure as a service

PaaS Platform as a service

CI Continuous integration

CD Continuous deployment

JCR Journal Citation Reports

PoLP

# LIST OF FIGURES

Figure. Monolith architecture: ...

Figure. Micro service architecture: ...

Figure. Micro frontend architecture: ...

Figure. E2E micro-frontend architecture: ...

Figure. Overview systematic mapping: ...

# LIST OF TABLES

Table. Research questions and motivation: ...

Table. Data extraction: ...

Table. Number of studies returned by each repository: ...

Table. Inclusion criteria: ...

Table. Exclusion criteria: ...

Table. Quality assessment criteria: ...

Table. MFA security threats per category: ...

Table. MFA security mechanism per category: ...

Table. Distribution of proposed solutions by application layers: ...

Table. MFA security targets multiple platforms: ...

Table. Verification and validation methods: ...

**TABLE OF CONTENT**

[DEDICATION i](#_Toc170603545)

[ACKNOWLEDGEMENTS ii](#_Toc170603546)

[ABSTRACT iii](#_Toc170603547)

[LIST OF ABBREVIATIONS iv](#_Toc170603548)

[LIST OF FIGURES v](#_Toc170603549)

[LIST OF TABLES vi](#_Toc170603550)

[CHAPTER 1 INTRODUCTION 1](#_Toc170603551)

[1.1 Background 1](#_Toc170603552)

[1.2 Problem Statement 2](#_Toc170603553)

[1.3 Significance of The Study 4](#_Toc170603554)

[1.4 Scope of The Study 5](#_Toc170603555)

[1.5 Aim of The Study 5](#_Toc170603556)

[CHAPTER 2 LITERATURE REVIEW 6](#_Toc170603557)

[2.1 Background of Microservice 6](#_Toc170603558)

[2.2 Micro-frontends Architecture 8](#_Toc170603559)

[2.2.1 Benefits of micro-frontends 10](#_Toc170603560)

[2.2.2 Challenges in micro-frontends 11](#_Toc170603561)

[2.3 Cloud-based Architecture with Micro-frontends 12](#_Toc170603562)

[2.3.1 Deployment models 13](#_Toc170603563)

[2.3.2 Scalability in cloud-based application 14](#_Toc170603564)

[2.3.3 Security in cloud-based application 15](#_Toc170603565)

[2.4 Systematic Mapping 16](#_Toc170603566)

[2.5 Summary 17](#_Toc170603567)

[CHAPTER 3 RESEARCH METHODS 19](#_Toc170603568)

[3.1 Research Objectives for Security in MFA 19](#_Toc170603569)

[3.2 Research Questions 19](#_Toc170603570)

[3.3 Study Searching Process 20](#_Toc170603571)

[3.4 Study Selection Process 21](#_Toc170603572)

[3.5 Inclusion and Exclusion Criteria 22](#_Toc170603573)

[3.6 Quality Assessment 23](#_Toc170603574)

[3.7 Data Extraction Process 25](#_Toc170603575)

[3.8 Data Synthesis 26](#_Toc170603576)

[CHAPTER 4 RESEARCH RESULTS 28](#_Toc170603577)

[4.1 Overview of Selected Studies 28](#_Toc170603578)

[4.2 MFA Security Threats (SRQ1) 32](#_Toc170603579)

[4.3 MFA Security Mechanisms (SRQ2) 35](#_Toc170603580)

[*4.3.1* Architectural layered approach (SRQ3) 38](#_Toc170603581)

[*4.3.2* Verification and validation techniques (SRQ4) 41](#_Toc170603582)

[CHAPTER 5 VALIDATION 43](#_Toc170603583)

[5.1 Insider Threats and External Attacks in MFA 43](#_Toc170603584)

[5.2 Emphasis on Security Techniques in MFA 43](#_Toc170603585)

[5.3 Focus on Securing Individual Micro-frontends 44](#_Toc170603586)

[5.4 Lack of Appropriate Solutions to Emerging 45](#_Toc170603587)

[5.5 Absence of Appropriate Comparison Techniques 46](#_Toc170603588)

[CHAPTER 6 CONCLUSION AND FUTURE WORK 47](#_Toc170603589)

[6.1 Conclusion 47](#_Toc170603590)

[6.2 Limitations 48](#_Toc170603591)

[6.3 Future Work 49](#_Toc170603592)

[REFERENCES 51](#_Toc170603593)

[APPENDIX A 54](#_Toc170603594)

[APPENDIX B 57](#_Toc170603595)

# INTRODUCTION

## Background

Frontend development has undergone significant transformations over the past few decades. Initially, web applications were monolithic, meaning that both the frontend and backend were tightly coupled and deployed as a single unit. This monolithic architecture often resulted in several challenges, including difficulty in scaling, slow deployment times, and a lack of flexibility in adopting new technologies (Martin Fowler, 2015).

In the early days of web development, applications were typically built using a monolithic approach. All the code, including HTML, CSS, and JavaScript, was bundled together and served from a single server. This setup was simple and easy to manage for small-scale applications. However, as applications grew in complexity and size, the limitations of monolithic architecture became apparent. Any change in one part of the application required redeployment of the entire codebase, leading to longer development cycles and increased risk of introducing bugs (Cam Jackson, 2019).

To address the challenges posed by monolithic architectures, the industry began adopting microservices for backend development. This approach inspired a similar shift in frontend development, leading to the emergence of micro-frontends (MFEs). Micro-frontends apply the principles of microservices to the frontend, breaking down the user interface into smaller, more manageable pieces that can be developed, deployed, and maintained independently (Leitner et al., 2020).

MFEs enable teams to work on different parts of the application simultaneously without interfering with each other's work. This modular approach enhances scalability and flexibility, allowing developers to use different frameworks and technologies within the same application. It also facilitates easier testing and maintenance, as each micro-frontend can be tested in isolation (Gioia et al., 2019).

Cloud computing has revolutionized the way applications are developed and deployed. By leveraging cloud services, organizations can avoid the upfront costs and complexity of owning and maintaining physical servers. Instead, they can rent computing resources on-demand from cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) (Armbrust et al., 2010).

Cloud computing offers various deployment models, including Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Each model provides different levels of control and management over the infrastructure:

* SaaS delivers fully managed applications over the internet, removing the need for users to handle the underlying infrastructure.
* PaaS provides a platform for developing, testing, and deploying applications, allowing developers to focus on coding without worrying about infrastructure management.
* IaaS offers virtualized computing resources over the internet, giving users maximum control over their infrastructure and the ability to configure it according to their needs (Mell & Grance, 2011).

The combination of micro-frontends and cloud-based architecture provides a powerful synergy that enhances the development and deployment process. Cloud platforms offer the scalability and flexibility required to support micro-frontend architectures (MFAs). By hosting micro-frontends in the cloud, organizations can take advantage of automated scaling, continuous integration and deployment (CI/CD) pipelines, and robust monitoring and logging tools (Hashizume et al., 2013).

Micro-frontends benefit from cloud services by enabling independent deployment and scaling of individual frontend components. This means that different teams can deploy updates to their respective micro-frontends without affecting the entire application. Cloud-based deployment models, particularly PaaS and IaaS, are well-suited to support this level of modularity and independence (Leitner et al., 2020).

Moreover, cloud services provide essential tools for managing the complexity of micro-frontend architectures. For example, containerization technologies like Docker and orchestration tools like Kubernetes facilitate the deployment and management of micro-frontends in a distributed environment. These tools ensure that micro-frontends can be easily scaled and maintained, providing a robust foundation for modern web applications (Burns & Oppenheimer, 2016).

## Problem Statement

Micro-frontend architecture offers several benefits, such as improved modularity, scalability, reusability, flexibility, and alignment with agile development methodologies. However, implementing this architecture also presents a range of challenges that can impact both small and large organizations.

* Redundant dependencies: Each MFE operates independently, leading to the duplication of libraries and dependencies across different parts of the application. This redundancy can result in larger application sizes, increased load times, and negatively impact performance metrics and SEO rankings. To mitigate this, teams can create shared MFEs that consolidate common dependencies, though this approach introduces complexity in maintaining the independence of each micro-frontends (OpenReplay Blog).
* Consistency in design and UX: Ensuring a uniform look and feel across all micro-frontends is difficult, as different teams may have varying design standards. This can lead to inconsistent user interfaces and overlapping CSS rules. Organizations need to establish comprehensive design systems and enforce strict communication and coordination among teams to maintain visual consistency.
* Performance issues: Running multiple micro-frontend applications simultaneously can strain resources such as CPU, RAM, and network bandwidth, leading to poor performance. This issue may not be apparent during independent testing but becomes significant when all components are integrated. Effective team communication and optimized resource sharing are crucial to address this challenge ([OpenReplay Blog](https://blog.openreplay.com/common-problems-with-micro-frontends-and-how-to-avoid-them/)).
* Increased complexity: The shift from a monolithic architecture to micro-frontends increases the complexity of the overall system. Managing numerous independent modules requires sophisticated orchestration and a robust DevOps infrastructure. Organizations need to invest in advanced developer tools, continuous integration/continuous deployment (CI/CD) pipelines, and comprehensive testing frameworks to handle this complexity ([McKinsey & Company](https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-forward/permanent-revolution-how-micro-frontends-can-help-to-overcome-the-struggle-of-continuous-frontend-modernization)) ([FloQast](https://floqast.com/engineering-blog/post/the-journey-to-micro-frontends-challenges-and-solutions/)).

Large organizations adopting MFE often encounter unique difficulties. For example, IKEA implemented micro-frontends to enhance its e-commerce platform. The transition resulted in a 50% reduction in development time and a 75% reduction in page load times. However, IKEA faced challenges related to data consistency and performance optimization, which they addressed through robust caching strategies and ensuring clear data ownership. With Upwork and OpenTable, both companies leverage micro-frontends to handle high user traffic and ensure scalability. They faced challenges related to increased complexity and the need for additional tools and infrastructure to manage the distributed architecture. Implementing a robust state management strategy and ensuring effective inter-module communication were key to overcoming these obstacles ([ThinkSys](https://thinksys.com/development/micro-frontend-architecture/)). Moreover, companies like FloQast have reported issues with modularity and dependency management, where components from different micro-frontends need to interact and share information without compromising the system’s integrity and performance (FloQast).

## Significance of The Study

The importance of this study is underscored by the intersection of two revolutionary architectural concepts: micro-frontends and cloud-based application architectures. As businesses embrace these paradigms at an escalating rate, it becomes crucial to grasp how they interact synergistically or present challenges. The critical need to explore the implications of integrating micro-frontends into cloud-based application architectures, particularly with regard to security measures, arises from a variety of essential factors:

* Rapid technological advancements: With the constant evolution of technology, particularly in cloud computing and web development, understanding the implications of incorporating micro-frontends is crucial to stay current and competitive in the industry.
* Increasing demand for scalability: As the demand for scalable and responsive web applications continues to grow, exploring the scalability benefits of micro-frontends in cloud architectures becomes imperative for meeting user expectations and business requirements.
* Cost-efficiency considerations: Cost optimization is a key concern for organizations operating in the cloud. Investigating the cost-effectiveness of micro-frontends could lead to more efficient resource utilization and budget management strategies.
* Heightened focus on security: In an era of heightened cybersecurity threats, enhancing the security measures of cloud-based applications is paramount. Understanding how micro-frontends impact security indexes is vital for ensuring robust protection against potential vulnerabilities.
* Industry competitiveness: Delving into this topic is essential for organizations looking to differentiate themselves in the competitive landscape by leveraging the advantages of micro-frontends within cloud infrastructures.
* User experience enhancement: Improved user experience is a top priority for web applications. Exploring how micro-frontends influence user experience within cloud environments can lead to enhanced customer satisfaction and retention.
* Alignment with industry trends: Researching this topic aligns with current industry trends and best practices, allowing organizations to adapt proactively to new methodologies and technologies shaping the future of cloud-based web development.

By addressing these urgent considerations, this research can provide timely insights and recommendations to industry professionals, researchers, and decision-makers navigating the complexities of cloud-based application development.

## Scope of The Study

This study will encompass a comprehensive examination of the impact of micro frontend development within cloud-based application architectures. It will concentrate on two primary subject groups:

* Leading technology companies: Including large companies such as Spotify, IKEA, etc. and others that have implemented the micro-frontend architecture in their systems. These companies provide practical cases and abundant data on effectiveness and security issues.
* Frontend experts and technical architects: This group consists of software developers and technical architects who are directly involved in designing, developing, and maintaining micro-frontend applications. They offer deep insights into technical challenges and applied security measures.

Data for this research will be collected over 10 years, from 2014 to 2024. The data collection methods include:

* Document analysis: Gathering and analyzing reports, studies, and technical documents from leading technology companies.
* Surveys and interviews: Conducting surveys and interviews with frontend experts and technical architects to gather information about their experiences and security measures.
* Analysis of real-world data: Using data from actual micro-frontend systems to assess security metrics and the effectiveness of security measures.

## Aim of The Study

This study focuses on analyzing security metrics when implementing the micro-frontend architecture in cloud-based systems. Specifically, the research aims to:

* Evaluate common security issues encountered during micro-frontend deployment.
* Identify effective security measures to minimize risks.
* Compare the security effectiveness between traditional methods and micro-frontends in a cloud environment.

# LITERATURE REVIEW

In this chapter, we provide the background information of micro-frontends and establish the context for our research work. By reviewing related work, researchers can understand the existing knowledge in their field and situate their own work within the broader scientific community.

## Background of Microservice

In the past, software applications often followed a monolithic architecture (MA) approach. This involved consolidating all application processes into a single self-contained unit. Within this monolith, the user interface, business logic, and data layer were tightly integrated (Alpers et al., 2015). Although this architecture simplified deployment and operation, it also had inherent limitations.

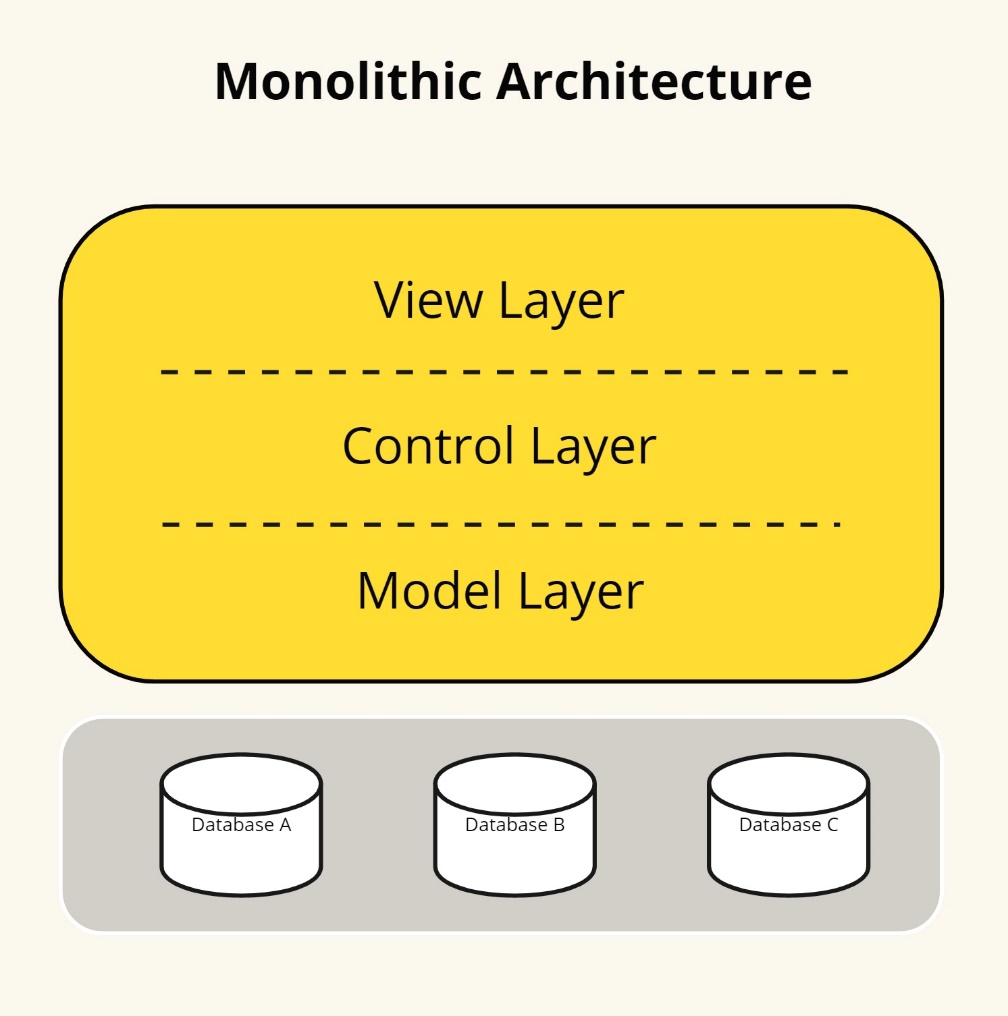
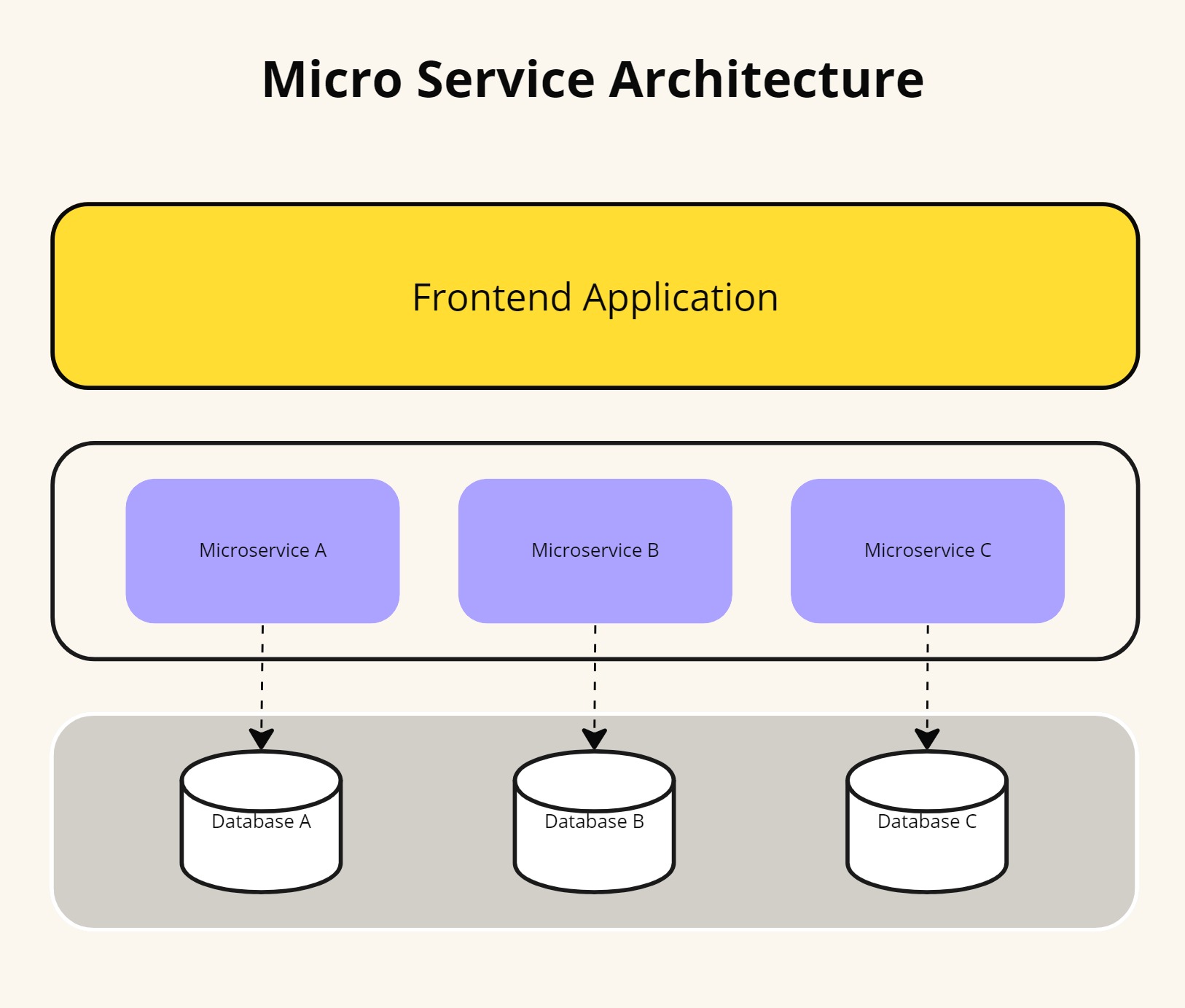


Figure 1. Monolith architecture

Monolithic architectures present significant challenges, such as inflexibility, instability, and inefficiency (Thatikonda, 2023). As monoliths expand, they become more difficult to comprehend and modify. Scalability becomes an issue as it requires changes to the entire application rather than specific components. To tackle these challenges, the concept of microservices architecture (MSA) emerged. MSA involves breaking down complex applications into smaller, loosely connected services that can be developed, deployed, and scaled independently (Agarwal et al., 2022). This transition aims to enhance maintainability, scalability, and productivity while enabling rapid and secure adaptation to increasing workloads.

*Figure 2. Micro service architecture*

## Micro-frontends Architecture

Microservices are commonly used to separate different components of an application, primarily in the backend. They structure a distributed application as a collection of services, dividing the backend into distinct functional units. Each application function becomes an independent service, which helps avoid bottlenecks in the database.

Micro frontends, however, extend the microservice concept to the frontend domain. In today’s web applications, the frontend is growing larger, while the backend is becoming less significant. Inspired by the microservices model, MFE offers a solution for modern complex web applications by allowing them to be divided into smaller, independent modules. This architectural approach involves splitting a web application into individual modules or functions, each implemented autonomously, providing frontend teams with flexibility and speed similar to what microservices offer to backend teams.

The term micro frontends or micro-frontends first appeared in 2016, in the [Thought Works Technology Radar](https://www.thoughtworks.com/es-es/about-us/news/2016/technology-radar-nov-2016) guide (Cam Jackson, 2019). A micro frontend can represent an entire page or specific fragments of a page that other teams can incorporate into their own development. Unlike reusable components, micro frontends can be implemented independently as individual projects. The approach for implementing micro frontends involves developing each part separately and dynamically extracting and using other components at runtime. A micro frontend comprises several independent and modular components that are displayed as needed. In other words, only the necessary components are loaded for a particular page. These components directly interact with data and do not rely on a centralized server to route requests or process information. Additionally, a micro frontend may include utility components that interact with the application environment, such as user-related features or other components.

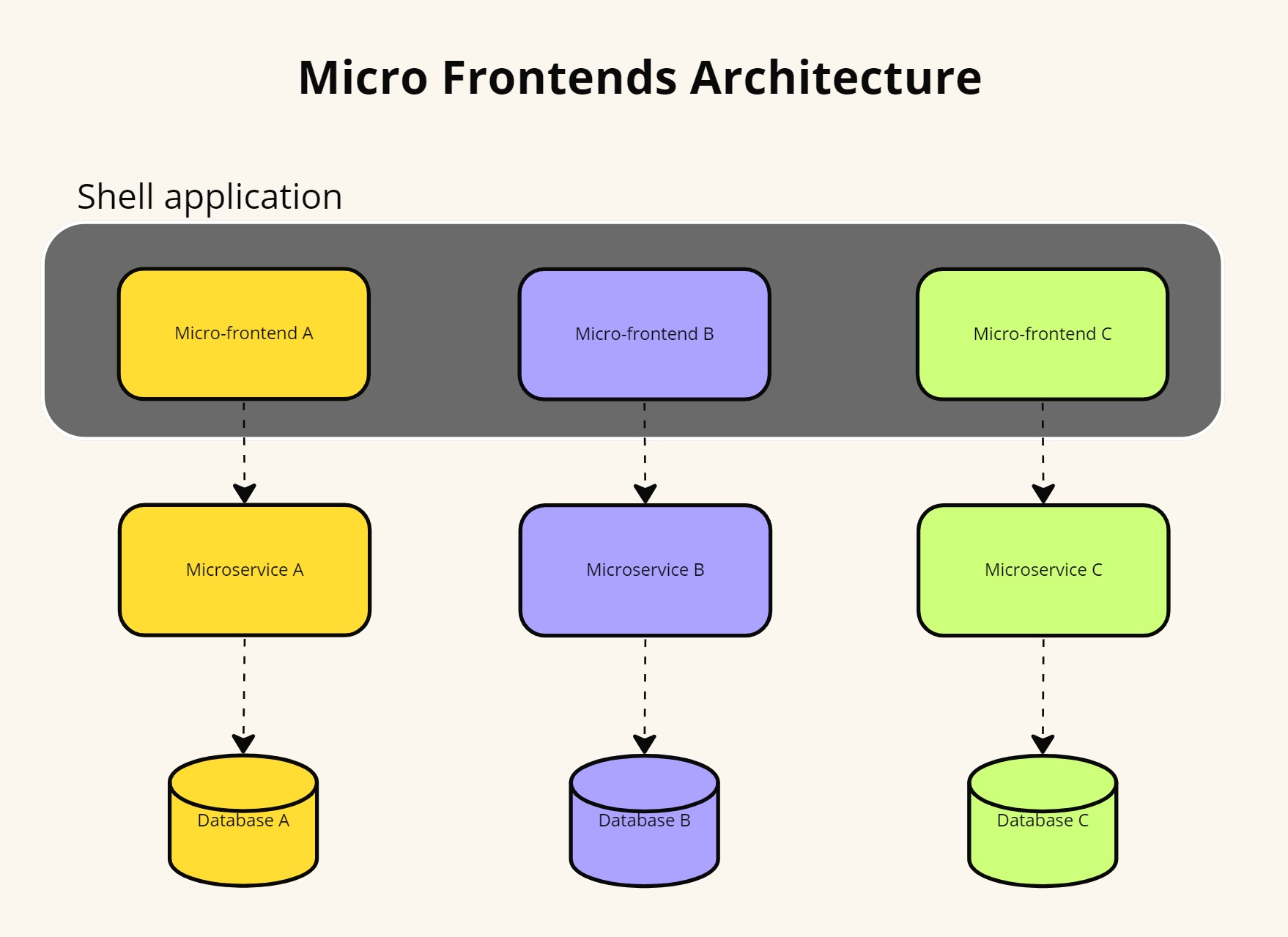


Figure 3. Micro-frontends architecture

This involves organizing the application vertically, which is made easier with micro-frontends. Micro-frontends split the application into small, independent functions, each developed simultaneously from the backend to the frontend by a dedicated team. This approach has become popular because of the challenges associated with monolithic architectures. As the frontend grows rapidly, it becomes more and more difficult to maintain a monolithic structure. Micro frontends provide the same scalability, security, and cost-effectiveness as backend microservices. The resulting application is less complex and more user-friendly. Furthermore, each micro frontend can be developed using different frameworks.

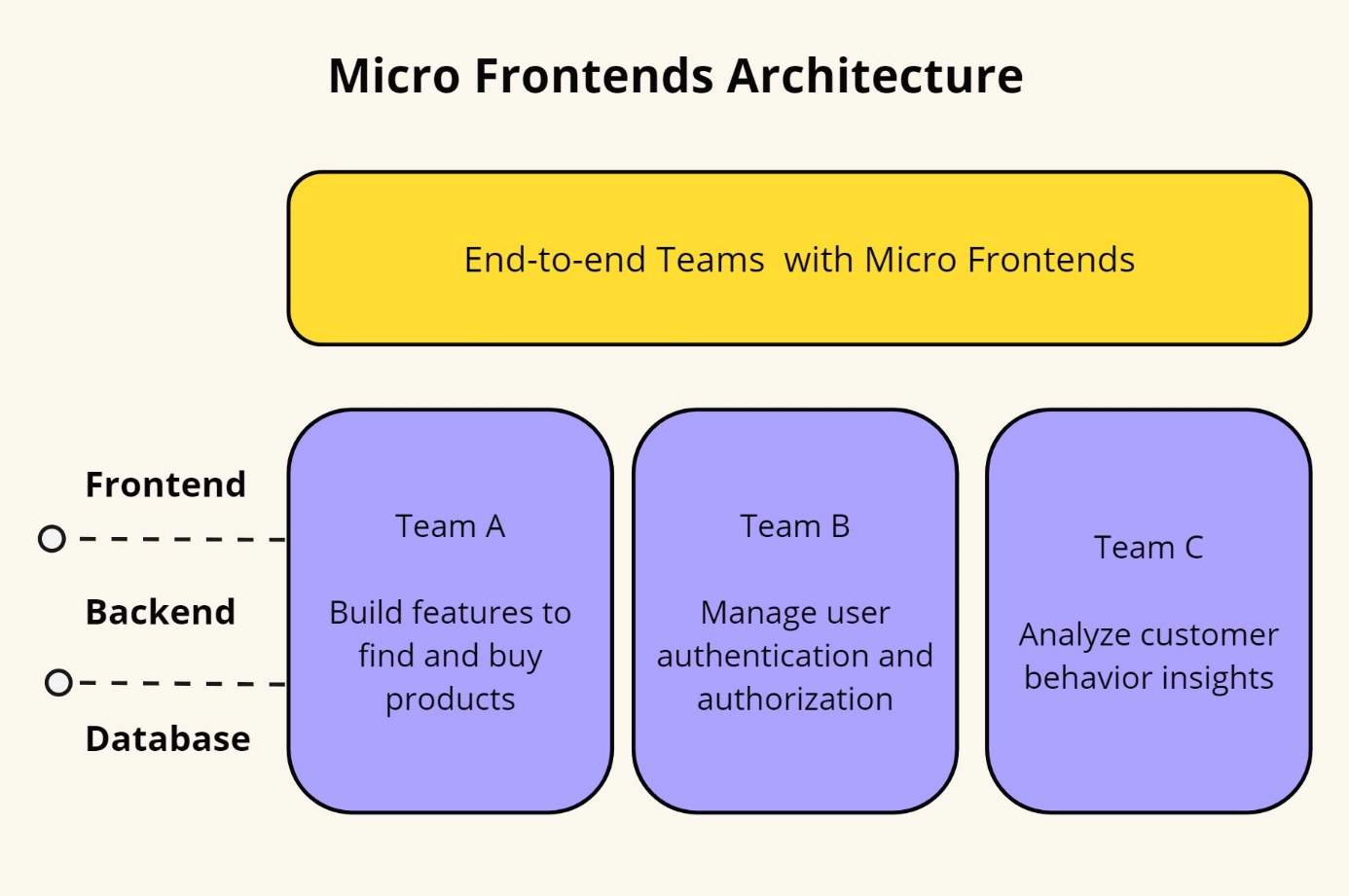


Figure 4. E2E of micro-frontends

This effort enhances development speed and streamlines the digital product's complexity by reducing inter-team dependency.

### Benefits of micro-frontends

Micro-frontend architecture offers many advantages that greatly enhance web application development and maintenance. Among these benefits are improved maintainability, scalability, and the facilitation of autonomous team operations.

Micro-frontends enhance the maintainability of large web applications by decomposing them into smaller, more manageable pieces. This modular approach allows development teams to focus on individual components without affecting the entire system. As noted by Guimarães et al. (2020), this separation of concerns simplifies debugging and reduces the complexity associated with maintaining monolithic applications. Each micro-frontend can be updated, tested, and deployed independently, streamlining the development process and facilitating easier management of codebases (Pereira et al., 2021).

Scalability is another significant advantage of MFA. By enabling different parts of an application to be scaled independently, micro-frontends can better handle varying loads across different components. For instance, if a specific micro-frontend experiences high traffic, only that component needs to be scaled, rather than the entire application. This efficient resource utilization leads to improved performance and cost savings (Santos & Costa, 2020). The ability to deploy and scale individual components independently aligns with modern DevOps practices, further enhancing the agility and responsiveness of development teams (Cam Jackson, 2019).

Micro-frontend architecture promotes the autonomy of development teams, allowing them to work on separate components concurrently without dependencies on other teams. This autonomy fosters a faster development cycle and encourages innovation as teams can experiment with new technologies and approaches within their specific domains. According to Mark Richards (2016), this decoupling of teams and responsibilities leads to better collaboration and productivity, as each team can operate within its own context, using the most appropriate tools and methodologies for their part of the application.

Moreover, micro-frontends facilitate technological diversity within a single application. Teams can choose different frameworks, libraries, and tools that best suit their specific component needs. This flexibility empowers teams to leverage the latest technological advancements while enabling a gradual transition from legacy systems to modern stacks without requiring a complete overhaul. The literature highlights that this capability to integrate diverse technologies mitigates the risk of technology lock-in and promotes a more adaptable development environment (’Berns & ’Kratzke, 2018).

### Challenges in micro-frontends

Despite the benefits of micro-frontend architecture, several challenges can complicate its implementation and maintenance. These challenges include increased complexity, testing difficulties, performance issues, and the need for consistent user experiences across different micro-frontends.

One of the primary challenges of micro-frontend architecture is the increased complexity that comes with decomposing a monolithic front-end into smaller, independent components. While beneficial for maintainability and scalability, this decomposition introduces significant complexity in terms of coordination and integration. Each micro-frontend can be developed using different frameworks and technologies, leading to a heterogeneous technology stack that can be difficult to manage (Mannisto et al., 2023). This complexity necessitates sophisticated orchestration mechanisms to ensure seamless integration and communication between micro-frontends (’Berns & ’Kratzke, 2018).

Testing in a micro-frontend architecture also presents significant challenges. Ensuring that individual micro-frontends function correctly in isolation and when integrated with other components can be complex and time-consuming. According to Pereira et al. (2021), the isolated nature of micro-frontends necessitates comprehensive end-to-end testing to verify that the entire application works as intended. Additionally, testing must account for the interactions between different micro-frontends and the overall system's behavior, which can be challenging given the potential variations in technology and implementation (Sam Newman, 2021).

Performance optimization is another critical challenge. The independent deployment of micro-frontends can lead to issues such as increased loading times and inefficient resource utilization if not managed properly. Each micro-frontend may introduce its own set of resources (e.g., JavaScript files, stylesheets), leading to redundancy and potential performance bottlenecks (Santos & Costa, 2020). Effective strategies are required to manage these resources and ensure that the overall application performs efficiently.

Maintaining a consistent user experience across multiple micro-frontends is crucial but challenging. Different teams may develop each micro-frontend using varying design patterns and user interface frameworks, making it essential to implement robust design systems and shared component libraries to ensure a unified look and feel (Richards, 2015). Inconsistencies in user interfaces can detract from the user experience, necessitating meticulous coordination and standardized design guidelines (Guimaraes et al., 2020).

Finally, micro-frontends can introduce complex dependency management issues. Ensuring that all micro-frontends operate harmoniously without version conflicts or compatibility issues requires careful planning and continuous monitoring (Berns & Kratzke, 2018). Dependency management becomes particularly challenging when different micro-frontends rely on shared libraries or global state management systems, necessitating strict version control and dependency resolution strategies (Cam Jackson, 2019).

## Cloud-based Architecture with Micro-frontends

The architectural landscape of cloud-based applications has witnessed a transformative shift, marked by the integration of micro-frontends as a key architectural paradigm. Traditionally, cloud-based applications have relied on monolithic structures, where the frontend, backend, and database components are tightly coupled into a single unit. However, the emergence of micro-frontends has challenged conventional approaches to frontend development, promoting a more modular approach (Perlin et al., 2023). Micro-frontends enable the decomposition of the user interface into smaller, independent components, each serving a specific feature or function (Cam Jackson, 2019). The adoption of cloud-based applications has brought about a significant shift in architectural design. This shift has allowed for increased scalability, improved performance, and enhanced flexibility, making cloud-based applications an asset for businesses looking to stay ahead of the curve. The modular nature of micro-frontends facilitates independent development, testing, and deployment of frontend components, enhancing agility and scalability.

### Deployment models

The choice of cloud deployment model can significantly impact the efficiency and performance of micro-frontend architectures. The primary cloud deployment models—Software as a Service, Platform as a Service, and Infrastructure as a Service — each offer distinct advantages and potential challenges for micro-frontend implementations.

SaaS provides complete software solutions that are hosted and managed by service providers. It offers a high level of abstraction, freeing developers from concerns about underlying infrastructure and platform management. This model is highly suitable for micro-frontend architectures where applications need to be delivered quickly and updated frequently. SaaS solutions can facilitate seamless integration of various micro-frontends, providing a unified user experience across different services (Marston et al., 2011). However, the primary limitation of SaaS in the context of micro-frontends is the potential lack of customization and control over the underlying architecture. Since SaaS applications are managed by third-party providers, developers may face restrictions in implementing custom micro-frontend solutions that require specific configurations or optimizations (Benlian et al., 2012).

PaaS offers a middle ground by providing a platform that includes operating systems, middleware, and runtime environments while leaving the application code development to the users. This model is particularly beneficial for micro-frontend architectures as it supports the development, deployment, and scaling of individual micro-frontends. PaaS solutions often come with built-in tools for CI/CD, version control, and other development aids that streamline the micro-frontend lifecycle (Armbrust et al., 2010). PaaS enables developers to focus on building micro-frontend components without worrying about infrastructure management. It also allows for greater customization compared to SaaS, as developers can tailor the platform services to better fit the specific needs of their micro-frontends. However, dependency on a specific PaaS provider can sometimes lead to vendor lock-in, making it challenging to migrate to other platforms if needed (Hashizume et al., 2013).

IaaS provides the most control by offering virtualized computing resources over the internet. It includes services such as virtual machines, storage, and networking. This model is ideal for micro-frontend implementations that require high customization and control over the environment. IaaS allows developers to configure their infrastructure according to the specific requirements of their micro-frontends, enabling optimized performance and security (Mell & Grance, 2011). Using IaaS, development teams can deploy and manage each micro-frontend independently, ensuring that each component can be scaled and updated as needed. The flexibility of IaaS makes it a robust choice for complex micro-frontend architectures that need to integrate diverse technologies and services. However, this model also requires significant expertise and resources to manage and maintain the infrastructure, which can be a challenge for smaller teams (Botta et al., 2016).

### Scalability in cloud-based application

Cloud scalability is the ability of a cloud computing system to adjust to varying computing demands by dynamically changing its resources, such as computing power, storage, or network capacity, on demand (Kerimovs, 2023). This flexibility allows the system to optimize its resources based on the workload, ensuring that required performance levels are consistently met. Scalability often involves increasing or decreasing the number of servers, storage units, or other computing resources as needed.

This scalability is essential because it allows organizations to quickly adapt to changes in their computing needs while making efficient use of resources. The primary goal is to enable cloud services to scale cost-effectively and handle increased loads by adding physical or virtual resources (Rashid Dar, 2016). This feature is a major advantage of cloud computing, as it allows businesses to expand their operations quickly and easily without significant upfront investments in hardware and other infrastructure. It ensures that resources are available when needed and not over-provisioned during periods of low demand, thus optimizing both cost and performance.

In cloud computing, vertical scaling refers to the process of enhancing the capabilities of an existing server or instance by increasing its resources such as RAM, CPU cores, or storage capacities like hard disks or solid-state drives (Debski et al., 2018). This allows applications to run more efficiently and handle increased load without the need for a new server or instance. It is a preferred method due to its simplicity and the fact that it does not require changes to the existing infrastructure. On the other hand, horizontal scaling involves expanding a system by adding more nodes or servers to the infrastructure, which is commonly used to increase the processing capacity of the cluster (Rashid Dar, 2016). This enables applications and services to handle a larger number of concurrent requests or process larger volumes of data efficiently. Typically, horizontal scaling is achieved by adding additional virtual machines (VMs), containers, or other resources to an existing cluster (Li et al., 2020), and is often used to improve performance or manage increased traffic. When implemented correctly, both vertical and horizontal scaling can effectively enhance a system’s performance.

### Security in cloud-based application

Despite the advantages of adopting MSA for developing complex systems, MSA introduces several challenges, notably in the realm of security (Yarygina & Bagge, 2018). Security, a longstanding issue in networking systems, becomes more complex with microservices due to the increased number of entry points and the communication traffic burden caused by decomposing systems into smaller, independent, and distributed units. Additionally, trust between individual microservices, often originating from different and unknown providers, cannot be easily established.

Due to the significant security breaches reported by companies adopting microservice architectures, such as Netflix and Amazon, addressing these vulnerabilities has become imperative. Numerous studies have highlighted the critical need to investigate MSA security (Baškarada et al., 2020). However, security threats are diverse and continually evolving. Consequently, security solutions are also proliferating, ranging from measures to secure individual microservices to comprehensive architectural and infrastructural protections.

In another study, Yarygina & Bagge aim to clarify microservice security by developing a taxonomy of microservice security, evaluating the security implications of microservice architecture, and reviewing contemporary solutions such as Docker Swarm and Netflix's security strategies. We present two key insights. First, microservice security is a multifaceted issue requiring a layered security approach, which is not yet available as a standard solution. Second, if these security challenges are resolved, microservice architectures can enhance security; their features of loose coupling, isolation, diversity, and rapid failure response contribute to greater system robustness. To address the lack of security guidelines, this paper outlines the design and implementation of a straightforward security framework for microservices that practitioners can use. Proof-of-concept evaluation results indicate that the performance overhead of the security mechanisms is approximately 11%.

## Systematic Mapping

A systematic mapping is a form of evidence-based software engineering (EBSE) (Kitchenham et al., 2015) that aims to provide an overview of a research area. It achieves this by creating a classification scheme and organizing evidence related to a specific research field.

In 2015, Peterson et al. proposed an overarching process for conducting systematic mappings, which consists of three main steps: planning, conducting, and reporting. During the planning phase, researchers justify the need and scope of the mapping, formulate research questions, and develop a protocol that outlines key decisions for conducting the mapping. This protocol includes defining search terms, specifying the search strategy, identifying literature sources, selecting relevant papers, and determining how data will be extracted and synthesized. In the conducting phase, the validated protocol from the planning step is put into action. Researchers use identified sources to retrieve papers, assess their relevance, extract useful data, and classify it. Finally, during the reporting phase, data from primary papers are visualized, results are interpreted, research questions are addressed, and the mapping process is validated and documented. *Figure 5* illustrates the overall systematic mapping process. Note that the quality assessment step is optional according to Kitchenham and Peterson.

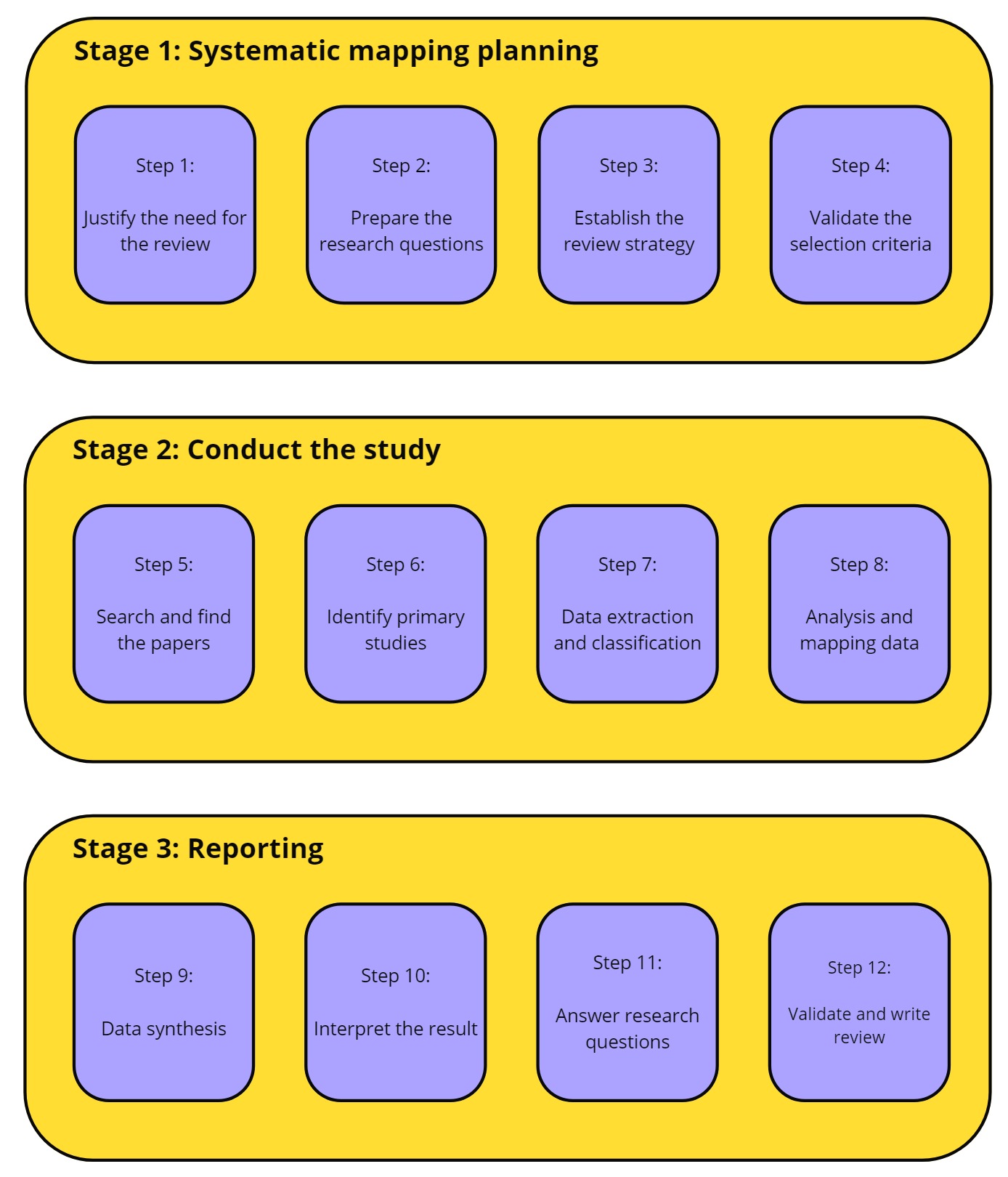


Figure 5. Overview systematic mapping

## Summary

In conclusion, the benefits of micro-frontend architecture, including enhanced maintainability, scalability, autonomous team operations, technology diversity, and improved user experience, make it a compelling choice for modern web application development. These advantages are well-documented in the literature, providing a solid foundation for adopting this architectural approach.

Choosing the appropriate cloud deployment model for micro-frontend implementations depends on the specific needs and resources of the development team. SaaS offers simplicity and ease of integration but may lack flexibility. PaaS provides a balanced approach with greater control and development support. IaaS offers the highest level of customization and control but requires extensive management. Understanding the strengths and limitations of each model can help in making informed decisions that align with the goals and constraints of the micro-frontend project.

# RESEARCH METHODS

In this section, we outline the protocol adopted for conducting this mapping study. Following the guidelines of Petersen et al. (2008) and Felderer and Craver (2017), a systematic mapping study involves several primary steps: defining research questions, searching for relevant papers, screening the identified papers, proposing or using an existing classification scheme, extracting data, and mapping the studies. The subsequent sections describe each step in detail.

## Research Objectives for Security in MFA

When implementing micro-frontends in critical systems like those used in the industrial sector, there are concerns about the potential risks of deploying micro-frontends on distributed and easily accessible platforms. The aim of this study is to identify potential threats to individual micro-frontends and cloud-based architecture that could compromise system security. Additionally, we aim to compile and categorize existing threats and the strategies used to mitigate and prevent them in easily manageable, reliable, and adaptable formats, such as ontologies. In this approach, researchers and practitioners can understand the potential risks to their current or future systems while also sharing experiences on how to mitigate and prevent security threats. Overall, the research aims to achieve the following objectives:

O1. Recognize and classify threats aimed at individual micro-frontends and cloud-based architectures.

O2. Identify, classify, and address efforts to identify, reduce, and prevent recognized security threats.

O3. Identify the validation techniques and tools used to confirm security measures.

## Research Questions

We conducted a comprehensive study of 10 open-source systems employing micro-frontends. For each system, we thoroughly analyzed the source code, manually annotating each security feature. These systems were developed and published by practitioners with a background in frontend development.

To expand our dataset and explore the design space, we created 20 additional models representing system variants. These variants were adapted from a published example, guided by discussions in relevant literature. Aside from the specific variations described in Tables 1 and 2, all other system aspects remained consistent with the base models. In total, we analyzed 30 models, summarized in Tables 1 and 2. We consider these evaluation systems to be practical examples that either reflect or closely resemble real-world microservice architectures.

Table 1. Research questions and motivation

|  |  |  |  |
| --- | --- | --- | --- |
| No | Research Question | Description | Objectives |
| SRQ1 | What are the primary security threats associated with micro-frontends, and how can they be categorized? | This research question distinguishes the list of mostly treated vulnerabilities from those needing further investigations. | O1 |
| SRQ2 | What are the different methods used to secure micro-frontends and microservice architecture, and how can we group them? | This research question identifies what security solutions are applied. | O2 |
| SRQ3 | What is the applicable level for securing micro-frontends with the proposed techniques and approaches? | This research question shows where security is implemented, emphasizing the overlooked levels in cloud-based architecture. | O2 |
| SRQ4 | What evidence supports the evaluation and validation of proposed methods for securing micro-frontends and cloud-based architectures? | This research question examines the effectiveness of current security methods and focuses on the empirical strategies employed to validate proposed solutions. | O3 |

## Study Searching Process

The search string utilized in this study is intentionally designed to be generic and simple. It is constructed based on search terms related to the population and intervention, as recommended by Petticrew and Roberts (2006). In this context, the population refers to the application area, which includes micro-frontends, micro-frontend architectures, and cloud-based architectures, while the intervention pertains to security, vulnerabilities, and attacks. Consequently, the final search string adopted is:

(" microfrontends" OR "micro-frontends")

AND

(“cloud-based” OR "architecture" OR "design" OR "system" OR "structure")

AND

("security" OR "vulnerability" OR "attack")

To retrieve relevant studies, we followed Kuhrmann et al.'s (2017) guidelines and utilized the online academic libraries:

• IEEExplore (<https://ieeexplore.ieee.org/>)

• ACM Digital Library (<https://dl.acm.org>)

• SpringerLink (<https://link.springer.com>)

• ScienceDirect (<https://www.sciencedirect.com/>)

• Wiley Online Library (<https://onlinelibrary.wiley.com>)

In order to ensure that we didn't miss any relevant studies, we used both backward and forward snowballing in addition to our automatic search, as recommended by Wohlin (2014, 2016). In backward snowballing, we checked the references of the approved papers for relevance, and in forward snowballing, we evaluated the relevance of papers that cited the approved papers. This snowballing process was applied recursively to each newly approved paper. We used Google Scholar exclusively for the forward snowballing.

## Study Selection Process

The set of papers retrieved through automatic search underwent two screening stages. Firstly, titles and abstracts were assessed for relevance, followed by a second stage where the full texts were examined to determine if they met our inclusion criteria. Each paper was independently screened by both authors, with decisions exchanged and conflicts resolved through discussion.

Additionally, papers identified through snowballing underwent separate screening by the two authors before inclusion or exclusion decisions were made.

In our study, we systematically assessed the support or violation of the collected security tactics. The following steps were followed:

1. Recommendation as Ordinal Ratings: Other authors applied our recommendations as an ordinal rating scheme to each model variant summarized in Tables 1 and 2, creating a ground truth for our study.

2. Expert Review: Three industrial security experts from our author team and two experts from another company reviewed the rating scheme and the ratings in the ground truth, contributing their insights and expertise to the robustness of our findings.

3. Statistical Analysis:

- Spearman Rank Correlation: Initially, we examined the correlation between the independent variables and the dependent variable using Spearman rank correlation, a method commonly used for analyzing the relationship between continuous and discrete ordinal variables.

- Ordinal Regression: To assess how well the hypothesized metrics predicted the ground truth data, we performed an ordinal regression analysis. This involved modeling the dependence of an ordinal response on a set of independent predictors. The lrm function from the rms package was utilized for this analysis.

Through expert judgment and rigorous statistical methods, our study provides valuable insights into the proposed metrics' effectiveness.

## Inclusion and Exclusion Criteria

We narrowed down the number of papers retrieved from online academic libraries by applying strict inclusion and exclusion criteria. For this study, we only included peer-reviewed papers from journals and conferences. The automatic search covered all publications from 2016 onwards, as there was no consensus on the term "micro-frontends" before 2016. We included only English-language papers that address security aspects or security solutions for micro-frontends or cloud-based architectures. The complete list of inclusion and exclusion criteria is presented in Tables 2 and 3, respectively.

Table 2. Inclusion criteria

|  |  |
| --- | --- |
| ID | Criteria |
| I1 | Include papers published since 2016, including early publications |
| I2 | Papers written in English |
| I3 | Papers include peer reviews |
| I4 | Papers that focus on studies conducted on security aspects of micro-frontends or cloud-based architectures |
| I5 | Papers that propose frameworks, techniques, methods, or tools to secure micro-frontends or cloud-based architectures |
| I6 | Papers presenting qualitative or quantitative evaluations of security techniques used for micro-frontends or cloud-based architectures |

Table 3. Exclusion criteria

|  |  |
| --- | --- |
| ID | Criteria |
| E1 | Publications that discuss security in distributed platforms and technologies, such as cloud, without explicitly referencing micro-frontends or microservices |
| E2 | Publications that are not focusing on security concerns of micro-frontends and cloud-based architecture |
| E3 | Publication is not a duplicate |
| E4 | Studies reviewing, surveying, or conducting secondary analysis on security measures within micro-frontend or cloud-based architectures |
| E5 | Publications such as book chapters, keynote abstracts, and presentation because these works typically receive minimal peer scrutiny and often discuss broad concepts previously introduced in academic journals or at conference events. |

## Quality Assessment and Publication Bias

This section presents the detailed quality assessment criteria of the research data, and introduction of publication bias assessment including calculations of effect size and standard error, followed by a funnel plot analysis to visualize the distribution of effect sizes and identify potential biases.

As advised by Petersen et al. (2015), conducting a quality assessment of identified papers is essential for mapping studies to ensure that sufficient information is available for data extraction. This step is crucial for evaluating the overall robustness and details provided in the selected papers. Therefore, we undertook a quality assessment process to meticulously evaluate these aspects.

Table 4 shows the quality assessment utilized a questionnaire consisting of four carefully chosen items, inspired by previous studies from Hannousse & Yahiouche (2021) and tailored to fit our specific research topic. Each item in the questionnaire served as a criterion for evaluating the quality and reliability of the studies under consideration.

Table 4. Quality assessment criteria

|  |  |  |
| --- | --- | --- |
| ID | Criteria | Score |
| QA1 | Does the study thoroughly discuss any security concerns in micro-frontend architectures? | {0, 1} |
| QA2 | Is there a clear solution presented in the study for any security threats in micro-frontend architectures? | {0, 0.5, 1} |
| QA3 | Has the study been referenced in other articles? | {0, 1} |
| QA4 | Is the study published in a reputable journal or conference proceedings? | {1, 0.5, 0} |

For QA1, a score of 1 was awarded if a criterion could be answered with "Yes," indicating that the paper met the criterion fully, while a score of 0 was given if the answer was "No," indicating that the paper did not meet the criterion. This binary scoring system ensured clarity and straightforward evaluation.

For QA2, a more nuanced scoring system was applied: a score of 1 was given to studies presenting a detailed and validated solution to a security threat, demonstrating a thorough approach to addressing the issue. A score of 0.5 was assigned to studies providing only an overview of the solution or framework, reflecting partial but incomplete information. Studies lacking clear solutions received a score of 0, indicating insufficient detail or validation.

QA3 evaluated the academic impact of the studies based on citation counts. A score of 1 was assigned to studies with three or more citations, reflecting recognition and validation by the academic community. In contrast, studies with fewer than three citations received a score of 0. Google Scholar was used to determine citation counts to avoid penalizing recent publications that might not yet have accumulated significant citations.

For QA4, the source and prestige of the paper were assessed using the CORE conference rankings and the Journal Citation Reports (JCR). Papers were ranked as follows: CORE A or B conferences were given a score of +1, CORE C conferences received a score of +0.5, and unranked conferences were given a score of 0. For journal sources, JCR Q1-2 journals received a score of +1, JCR Q3-4 journals got a score of +0.5, and unranked journals received a score of 0. This criterion ensured that the source's credibility was factored into the overall assessment.

All four criteria were treated equally, ensuring a balanced evaluation of each study. The total score for each study was calculated by summing the four values, providing a comprehensive quality metric. Only studies with a total quality score of 2 or higher were included in this study, ensuring that only the most robust and detailed papers were considered.

The assessment criteria and their associated scores are summarized in Table 4, which provides a clear overview of the evaluation process and the standards applied in this study. This rigorous approach ensures that our data extraction is based on high-quality, reliable sources, enhancing the credibility and validity of our findings.

## Data Extraction Process

Following the guidelines of Petersen et al. (2015), a comprehensive data extraction form was designed to ensure systematic and detailed collection of information from each selected paper. This form is illustrated in Table 5, providing a structured template for consistent data gathering.

Each paper included in the study is meticulously described in terms of its metadata, which encompasses essential details such as the year of publication, the source (journal or conference), and the type of publication (e.g., empirical study, theoretical paper, case study). This metadata provides a foundational context for understanding the background and relevance of each paper within the broader research landscape.

In addition to metadata, the data extraction form captures a set of specific information critical to our analysis. This includes a comprehensive list of the security threats or attacks addressed by the study. Identifying these threats is vital for mapping out the security challenges that microservice architectures face and understanding the scope of each paper's focus.

Furthermore, the form documents the proposed solutions for the identified security threats. This involves detailing the nature of each solution, whether it is a technical mechanism, a theoretical framework, a practical implementation, or a combination thereof. The specificity of these solutions is crucial for assessing their applicability and potential effectiveness in real-world scenarios.

The application level of the proposed solutions is another key piece of information extracted. This involves determining whether the solutions are applicable at the level of individual microservices, the microservice architecture as a whole, or other levels such as network or infrastructure. Understanding the application level helps in categorizing the solutions and identifying gaps or overlaps in the existing research.

Additionally, the form records the validation methods used to assess the proposed solutions. This includes whether the solutions have been validated through empirical testing, simulation, case studies, expert reviews, or theoretical analysis. The robustness of these validation methods is critical for evaluating the reliability and generalizability of the findings presented in each paper.

Finally, the application platforms on which the proposed solutions have been tested or are intended to be applied are also documented. This includes information about the specific technologies, tools, and environments used in the studies, providing insight into the practical applicability and limitations of the solutions.

By following this rigorous data extraction process, we ensure that all relevant information is systematically collected and organized, facilitating a thorough analysis of the security landscape in microservice architectures. This methodical approach allows for a comprehensive understanding of the current state of research, identifying both strengths and areas in need of further investigation.

Table 5. Data extraction

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Data Item | Description | Research Question |
| D1 | Study ID | First author name + year |  |
| D2 | Year | Year of the publication |  |
| D3 | Source | Source of the publication |  |
| D4 | Type | Conference or journal paper |  |
| D5 | Category | Analysis, solution proposal or case study |  |
| D6 | Threats | Addressed security threats | SRQ1 |
| D7 | Source of Threats | Internal or external | SRQ1 |
| D8 | Solution type | General protection measures, framework, technique, tool or methodology proposal | SQR2 |
| D9 | Applicability level | Architectural level where the security mechanism is applied | SRQ3 |
| D10 | Validation method | Verification and validation techniques used to check the feasibility of the proposed solution | SRQ4 |

## Data Synthesis

We observed a significant lack of consensus on detailed taxonomies for security threats and mechanisms, which hindered our ability to categorize all selected studies appropriately and distinctly to answer research questions SRQ1 and SRQ2. Additionally, the diversity of applications, targeted platforms, and verification and validation techniques used in the selected studies necessitated a proper categorization to effectively address research questions SRQ3, and SRQ4.

To map all the selected studies accurately to appropriate categories for each research question, we leveraged our experiences and the existing research of Aguiar Monteiro et al. (2020) in identifying categories and their relationships. We also employed grounded theory as a complementary approach to generate missing categories from the extracted data items. Specifically, we utilized open coding and selective coding to identify categories and their relationships with existing categories from D6, D9, and D10. In this study, grounded theory was applied iteratively, where categories and subcategories were refined in each iteration until a stable state was achieved.

## Grey Literature

To complement our academic search, we conducted a grey literature review to gather insights on industrial contributions and practices for security defense mechanisms in micro-frontends. Grey literature, which includes non-commercially published data, can significantly enhance a systematic review by mitigating publication bias, improving comprehensiveness, and providing a balanced overview of existing information. Additionally, grey literature often features practical, up-to-date information that is easily accessible and beneficial for practitioners and decision-makers across various fields.

For our grey literature review, we established the following criteria for selecting articles, blogs, and journals:

* Search platforms: We used Google Scholar and StackOverflow to identify relevant studies.
* Professional experience: Only studies published by professionals with expertise in this field were selected.
* Industrial case studies: Each study had to mention at least one industrial case study involving a measurable number of microservices.
* Practitioner insights: The studies needed to provide a balanced perspective on the advantages and disadvantages of the issues and topics discussed.

After identifying the relevant articles, we categorized them and removed duplicates to ensure a clear and comprehensive collection of grey literature sources. This methodology enabled us to gather a diverse range of practical and timely insights on security defense mechanisms in micro-frontend architectures.

# RESEARCH RESULTS

In this section we describe and detail the results of the mapping study answering all research questions outlined in Chapter 3.2.

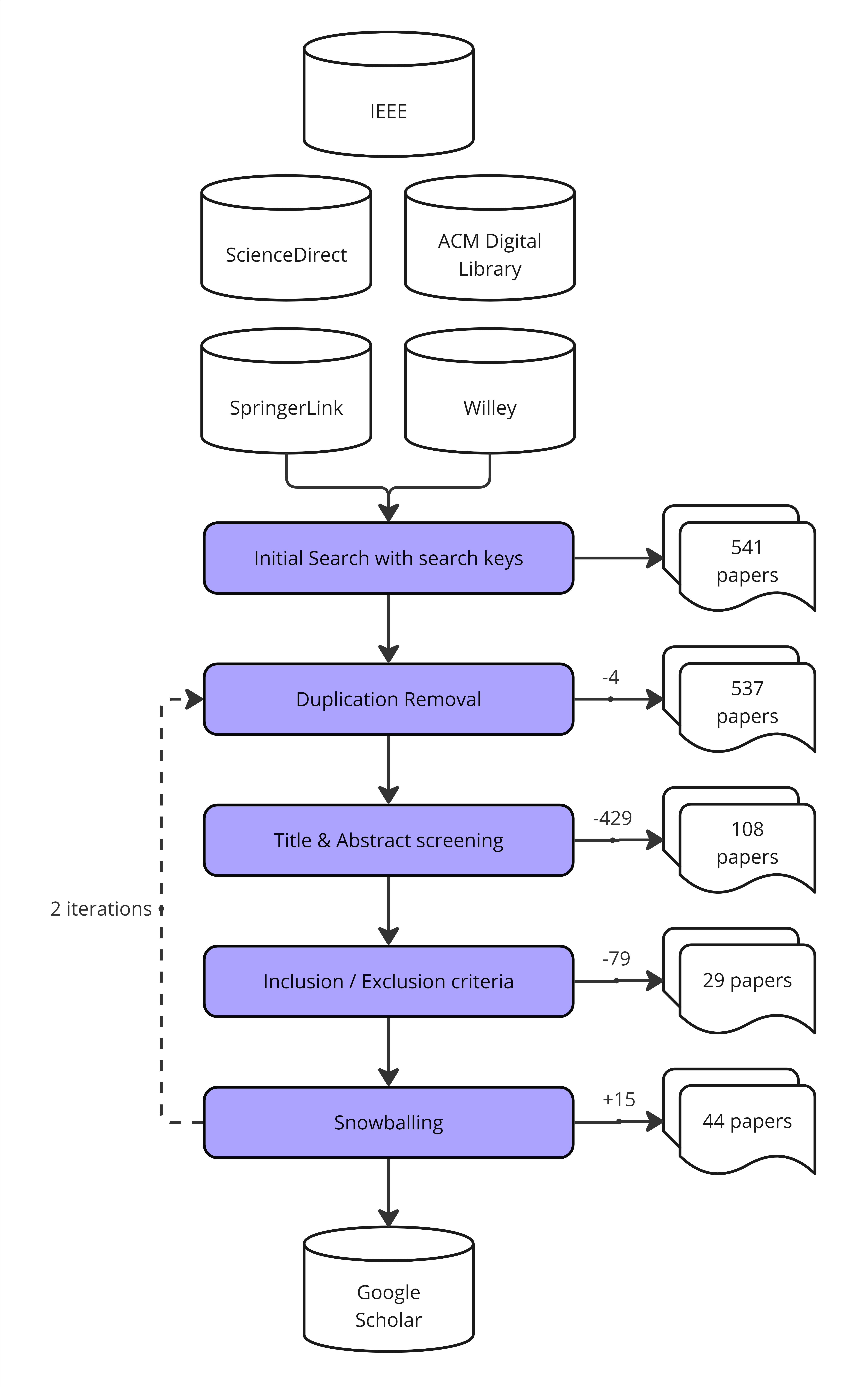
## Overview of Selected Studies

The search process, conducted in April 2024, identified 541 distinct papers on micro-frontends architecture published since 2005. The search query was applied across several academic libraries, with the results summarized in Table 6, indicating the number of papers returned by each library. Initially, 541 papers were retrieved from various search engines. After removing duplicates, the number was reduced to 537.

Table 6. Number of studies extracted per repository

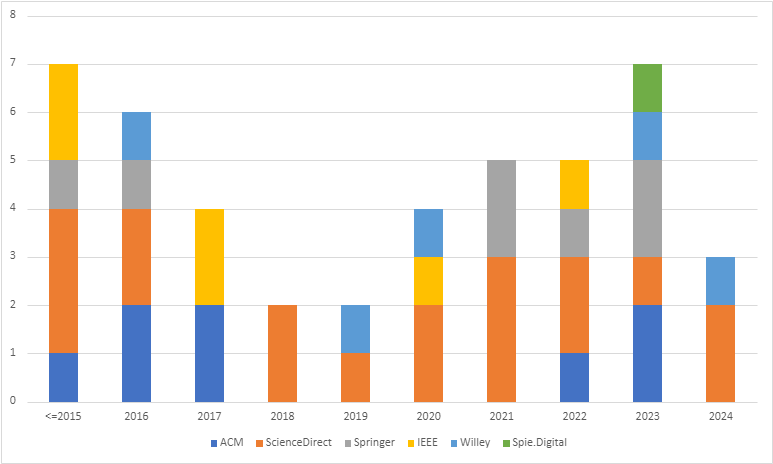
|  |  |
| --- | --- |
| Repository | Search results |
| IEEE | 30 |
| ACM Digital Library | 312 |
| SpringerLink | 20 |
| ScienceDirect | 113 |
| Wiley Online Library | 66 |
| Total | 541 |

We then screened the titles and abstracts of the remaining papers, excluding 429 papers due to irrelevance. Applying the inclusion and exclusion criteria further narrowed the list to 29 approved papers. To ensure comprehensive coverage, we employed recursive backward and forward snowballing, which added 15 more papers to our study. This snowballing process involved two cycles to achieve a steady state: the first cycle included 10 new papers, and the second cycle added 5 more papers. Figure 3 illustrates the overall selection process in detail.



*Figure* 6*. Overview selection studies*

Figure 4 presents a distribution of research papers published by various sources from 2015 to 2024. Over the years, ScienceDirect has been the most prolific publisher, consistently producing papers each year with notable peaks in 2020 and 2022, each with three publications. This highlights ScienceDirect's sustained contribution to micro-frontend security research. In contrast, Springer and IEEE have had sporadic yet significant contributions, particularly in recent years. Springer, for example, shows a resurgence in activity with two papers in both 2023 and 2024, indicating a growing interest or perhaps increased funding for research in this area. IEEE's contribution is noteworthy in the early years but diminishes, except for a single paper in 2022, suggesting a potential shift in focus or resources over time. ACM and Willey show more intermittent contributions. ACM's publications are scattered with an early focus (2016-2017) and a recent increase in 2023 and 2024. Willey has a similar sporadic pattern, with contributions in 2016, 2020, 2021, and a notable increase in 2023 and 2024. SPIE Digital, appearing only in 2024, suggests either a new interest in micro-frontend security or the inclusion of a new research topic within their scope. Overall, the table reflects a dynamic and evolving research landscape with varying levels of engagement from different publishers over the years.



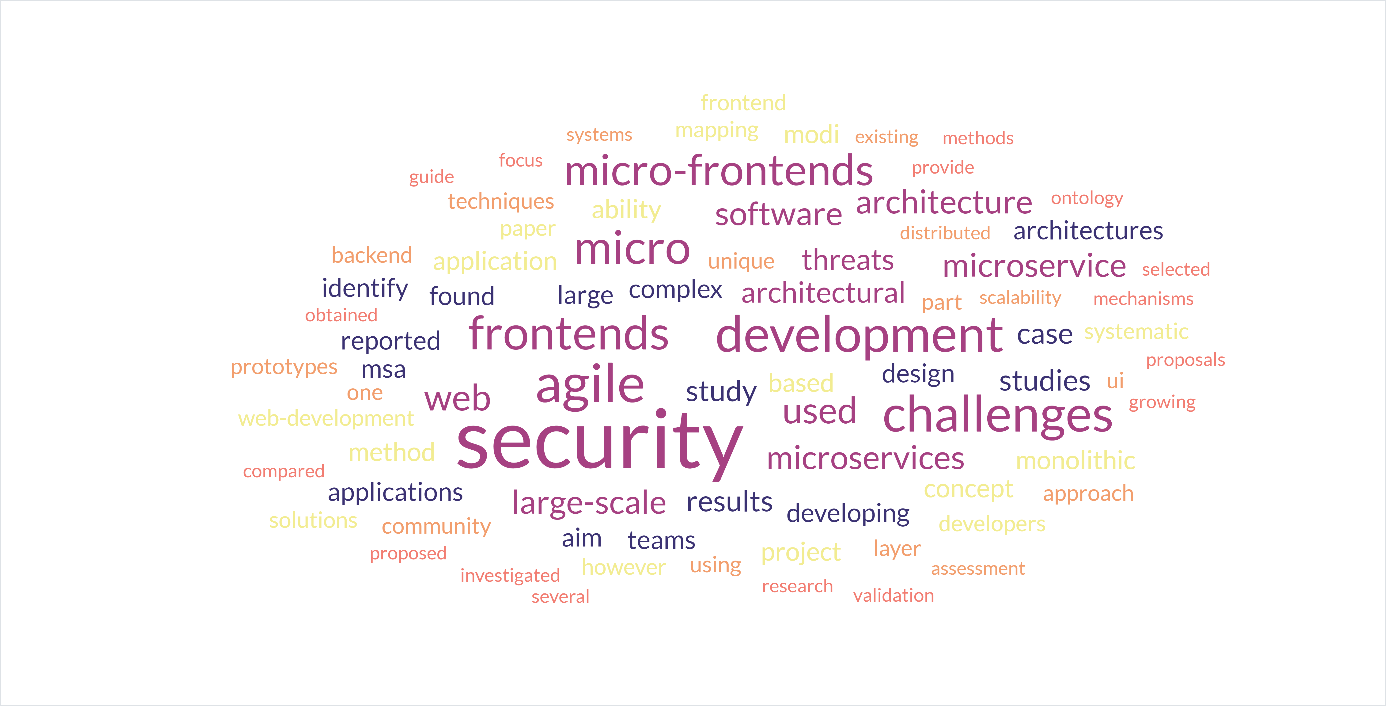
*Figure* 6*. Distribution of selected studies*

Table 7 presents the complete list of selected studies, including their year of publication, type of publication, method of discovery, and quality assessment scores. This detailed presentation helps understand the breadth and depth of research conducted in micro-frontends and their security considerations.

Table 7. Extracted primary research papers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Title | Year | Publisher | Score |
| P1 | Executing Microservice Applications on Serverless, Correctly | 2023 | ACM | 3 |
| P2 | Modularity, Code Specialization, and Zero-Cost Abstractions for Program Verification | 2023 | ACM | 2 |
| P3 | Practical and Secure Outsourcing of Discrete Log Group Exponentiation to a Single Malicious Server | 2017 | ACM | 3 |
| P4 | Reconciling Security and Functional Requirements in Multi-tenant Clouds | 2017 | ACM | 3 |
| P5 | Community-Based Secure Information and Resource Sharing in Azure Cloud IaaS | 2016 | ACM | 3 |
| P6 | Towards micro architecture for security adaptation | 2016 | ACM | 3 |
| P7 | Application of microservices patterns to big data systems | 2023 | Springer | 2 |
| P8 | A survey of optical wireless technologies: practical considerations, impairments, security issues and future research directions | 2022 | Springer | 1 |
| P9 | An empirical study of the systemic and technical migration towards microservices | 2023 | Springer | 1 |
| P10 | Applying Model-Driven Engineering to Stimulate the Adoption of DevOps Processes in Small and Medium-Sized Development Organizations | 2021 | Springer | 1 |
| P11 | Industry practices and challenges for the evolvability assurance of microservices | 2021 | Springer | 2 |
| P12 | An architecture to manage security operations for digital service chains | 2021 | ScienceDirect | 2 |
| P13 | Adaptive evidence collection in the cloud using attack scenarios | 2016 | ScienceDirect | 3 |
| P14 | Design, monitoring, and testing of microservices systems: The practitioners’ perspective | 2021 | ScienceDirect | 2 |
| P15 | Secure software development and testing: A model-based methodology | 2024 | ScienceDirect | 2 |
| P16 | An empirical study of security practices for microservices systems | 2023 | ScienceDirect | 3 |
| P17 | A flexible Compilation-as-a-Service and Remote-Programming-as-a-Service platform for IoT devices | 2022 | ScienceDirect | 3 |
| P18 | Iris: Secure reliable live-streaming with opportunistic mobile edge cloud offloading | 2019 | ScienceDirect | 2 |
| P19 | On evaluating commercial Cloud services: A systematic review |  | ScienceDirect | 3 |
| P20 | Automated identification of security discussions in microservices systems: Industrial surveys and experiments | 2021 | ScienceDirect | 3 |
| P21 | Amazon Smart sales Ticketing System | 2016 | ScienceDirect | 1 |
| P22 | An intelligent Edge-IoT platform for monitoring livestock and crops in a dairy farming scenario | 2020 | ScienceDirect | 1 |
| P23 | Protecting critical infrastructure against cascading effects: The PRECINCT approach | 2024 | ScienceDirect | 2 |
| P24 | A secure and distributed message oriented middleware for smart building applications | 2018 | ScienceDirect | 2 |
| P25 | AutoLog: Anomaly detection by deep autoencoding of system logs | 2022 | ScienceDirect | 1 |
| P26 | Systems and Application Security | 2016 | Willey | 1 |
| P27 | A systematic survey on security and privacy issues of medicine supply chain: Taxonomy, framework, and research challenges | 2024 | Willey | 2 |
| P28 | Cyber threat intelligence for critical infrastructure security | 2023 | Willey | 2 |
| P29 | Microservice transition and its granularity problem: A systematic mapping study | 2020 | Willey | 2 |
| P30 | Towards a Modular Architecture for Industrial HMIs | 2020 | IEEE | 3 |
| P31 | Dispute Resistance Multilayered RFID Partial Ownership Transfer With Blockchain | 2022 | IEEE | 2.5 |
| P32 | Security threat classification for outsourced IT projects | 2017 | IEEE | 3 |
| P33 | A Joint Optimization Approach to Security-as-a-Service Allocation and Cyber Insurance Management | 2015 | IEEE | 4 |
| P34 | vepRisk - A Web Based Analysis Tool for Public Security Data | 2017 | IEEE | 4 |
| P35 | A Semantic Approach to Cloud Security and Compliance | 2015 | IEEE | 3 |
| P36 | A framework for implementing micro frontend architecture | 2022 | ACM | 3.5 |
| P37 | A threat-driven approach to modeling and verifying secure software | 2005 | ACM | 2 |
| P38 | Remote Internet voting: developing a secure and efficient frontend | 2013 | Springer | 2 |
| P39 | Motivations, Benefits, and Issues for Adopting Micro-Frontends: A Multivocal Literature Review | 2020 | ScienceDirect | 3 |
| P40 | A survey on security issues in service delivery models of cloud computing | 2011 | ScienceDirect | 3 |
| P41 | Security as a service (SecaaS)—An overview | 2015 | ScienceDirect | 2 |
| P42 | Classification of Security Threats in Information Systems | 2014 | ScienceDirect | 3 |
| P43 | Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers | 2019 | Willey | 2 |
| P44 | Micro-frontend architecture base | 2023 | Spie.Digital | 3 |

Following the guidelines of Hannousse & Yahiouche (2021), we employed Word Clouds to analyze the relevance of our result set of primary studies. The figure 7 highlights that the most commonly used words are "security", "micro-frontends”, "microservices", "architecture", and "challenges". Conversely, terms such as "threats," "vulnerabilities," and "risks" are less frequently used in the titles and abstracts.

*Figure* 7*. Research keywords*

## MFA Security Threats (SRQ1)

Micro-frontend architecture, an emerging paradigm in software engineering, introduces new security threats and vulnerabilities. These threats can originate from insiders (i.e., internal attacks) or outsiders (i.e., external attacks). To properly secure micro-frontend-based systems, all threats, regardless of their origin, must be detected and prevented using either existing mitigation techniques or innovative solutions. This study aims to identify the focus of existing efforts concerning the source of threats (internal, external, or both). Figure 8 depicts the distribution of identified and selected studies based on the addressed source of threats. The findings indicate that 60% of primary studies focus on external attacks, 20% address internal attacks, and only 10% cover each infrastructure and software of threats. This demonstrates a clear research bias towards external attacks.

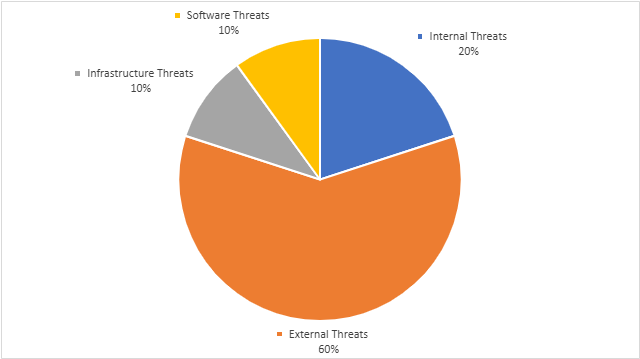


Figure 8. MFE security threats probation

The primary security threats associated with micro-frontends can be categorized into four main areas: user-based attacks, data attacks, infrastructure attacks, and software attacks. Each category encompasses specific threats that can compromise the security and integrity of micro-frontend architectures.

* User-based attacks:
  + Malicious User Actions: This includes actions taken by users with malicious intent to exploit the system, such as privilege escalation, unauthorized data access, and launching phishing attacks.
  + Inadvertent Insider Actions: Mistakes or unintentional actions by legitimate users that can lead to security breaches, such as accidental data leakage or misconfiguration of security settings.
* Data attacks:
  + Sensitive Data Exposure: Unauthorized access or leakage of sensitive information, such as personal data, payment information, or confidential business data.
  + Data Manipulation: Unauthorized alteration of data, which can lead to integrity issues, including tampering with transaction records or modifying user data.
* Infrastructure attacks:
  + Unauthorized Access: Gaining unauthorized access to the infrastructure, potentially compromising multiple micro-frontends and the overall system.
  + Denial of Service (DoS) Attacks: Overloading the system to make services unavailable to legitimate users, disrupting the functionality of micro-frontends.
  + Compromising Monitors: Attacking monitoring systems that oversee the health and performance of micro-frontends, leading to undetected issues.
  + Compromising Message Brokers: Attacking the message brokers that facilitate communication between micro-frontends, leading to interception or alteration of messages.
* Software attacks:
  + Code Injection: Introducing malicious code into the application through various vectors, such as input fields, which can execute unauthorized actions.
  + Code Transformation: Modifying the existing code to include malicious functionality, often through supply chain attacks or exploiting vulnerabilities in the deployment process.

Table 8 shows the set of micro-frontend security threats addressed by primary studies, grouped by category. The results reveal that unauthorized access, sensitive data exposure, and compromising individual micro-frontends are the most frequently addressed threats by contemporary studies. Additionally, infrastructure attacks are the most diverse but are less frequently addressed in the selected studies.

Table 8. MFA threats grouped categories

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Security Threats | Studies | Percentage |
| User-based Attacks | Malicious User Actions | P1, P3, P5, P7, P9, P11, P13, P15, P31, P33, P35, P38, P39 | 29.55% |
| Inadvertent Insider Actions | P2, P4, P6, P8, P38, P39 | 13.64% |
| Data Attacks | Sensitive Data Exposure | P1, P2, P3, P5, P6, P7, P8, P9, P23, P24, P27, P28, P33, P36, P37, P40, P41 | 38.64% |
| Data Manipulation | P11, P12, P13, P14, P15, P16, P17, P25, P26, P35, P37, P43 | 27.27% |
| Infrastructure Attacks | Unauthorized Access | P1, P6, P12, P13, P16, P20, P21, P25, P26, P31, P32, P35, P38, P39, P40, P42, P44 | 38.64% |
| DoS Attacks | P4, P9, P14, P19, P24, P29, P33, P34 | 18.18% |
| Compromising Monitors | P10, P18, P27, P30 | 9.09% |
| Compromising Message Brokers | P3, P28 | 4.55% |
| Software Attacks | Code Injection | P4, P10, P16, P18, P19, P20, P22, P23, P25, P29, P34, P36, P38, P40, P41, P43 | 36.36% |
| Code Transformation | P5, P8, P28 | 6.82% |

The analysis of security threats across various categories in micro-frontends for cloud-based applications reveals distinct focus areas among contemporary studies. User-based attacks are primarily concerned with malicious user actions (29.55%) and inadvertent insider actions (13.64%), indicating significant attention towards mitigating risks associated with user behavior.

Data attacks are predominantly focused on sensitive data exposure, with 38.64% of studies addressing this issue, highlighting its critical importance. Data manipulation is also a significant concern, covered in 27.27% of studies, emphasizing the need for data integrity and protection mechanisms.

Infrastructure attacks show a high prevalence of studies on unauthorized access (38.64%), pointing to the necessity of robust access control mechanisms. Denial of Service (DoS) attacks are addressed in 18.18% of studies, reflecting concerns over availability and performance. Compromising monitors, and message brokers are lesser-studied threats but remain crucial, with 9.09%, and 4.55% of studies respectively, indicating a need for comprehensive security measures across all infrastructure components.

Software attacks such as code injection (36.36%) and code transformation (6.82%) are also significant areas of focus, underscoring the importance of secure coding practices and runtime protection.

Overall, the distribution of studies across these categories illustrates a balanced approach to addressing both external and internal threats, ensuring data protection, infrastructure security, and software integrity in micro-frontend architectures.

## MFA Security Mechanisms (SRQ2)

In the context of micro-frontend architecture, the security mechanisms proposed in primary studies can be categorized as follows:

* General Protection Measures: These include the use of general security techniques to mitigate known threats in MFA, or a set of guidelines on choosing appropriate languages and technologies.
* Framework-based Solutions: These are architectural frameworks for MFA that incorporate specific modules to handle certain security aspects and mechanisms such as authorization, continuous monitoring, and diagnosis.
* Technique-based Solutions: These are newly designed or adopted techniques from other domains to mitigate or prevent some security threats in MFA.
* Tool-based Solutions: These are newly developed tools that implement security measures.
* Algorithm-based Solutions: These are new algorithms designed for the detection or prevention of security threats.
* Protocol-based Solutions: These are new protocols designed for the protection of communications among the different MFA architectural elements.

Proposed solutions for securing micro-frontend architectures can be classified into proposals for enforcing authentication and authorization, access control, auditing, and security best practices:

* Authentication & Authorization:
  + Single Sign-On (SSO): Techniques used to verify the identity of users requiring access to MFA resources and data.
  + OAuth and OpenID Connect: Protocols for secure token-based authorization and authentication.
* Access Control:
  + Role-Based Access Control (RBAC): Restricts access based on the roles assigned to users.
  + Attribute-Based Access Control (ABAC): Uses attributes of users and resources to make access decisions.
* Auditing:
  + Activity Logs: Keeps detailed logs of user activities and interactions with micro-frontends.
  + Audit Trails: Provides a record of all security-relevant events, useful for detecting and investigating security incidents.
* Security Best Practices:
  + Principle of Least Privilege: Limits access rights for users to the bare minimum necessary.
  + Regular Security Assessments: Conducts vulnerability assessments and penetration testing to identify and mitigate security weaknesses.

Table 9. MFA security mechanism per category

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Mechanism | Studies | Percentage |
| Authentication & Authorization | Single Sign-On (SSO) | P28, P18, P6, P11, P15, P5, P1, P21, P35, P40 | 22.73% |
| OAuth and OpenID Connect | P2, P7, P10, P12, P17, P20, P22, P25, P27, P29, P34, P39 | 27.27% |
| Access Control | Role-Based Access Control | P3, P8, P13, P16, P26, P39, P40 | 15.91% |
| Attribute-Based Access Control | P4, P9, P14, P19, P24 | 11.36% |
| Auditing | Activity Logs | P3, P8, P13, P16, P23, P26, P33, P35, P38, P40, P41 | 25.00% |
| Audit Trails | P2, P10, P17, P5, P20, P25, P29, P30, P34, P36, P37 | 25.00% |
| Security Best Practices | Principle of Least Privilege | P28, P18, P6, P11, P15, P1, P21, P31, P33, P38, P41, P44 | 27.27% |
| Regular Security Assessments | P4, P7, P9, P12, P14, P19, P22, P24, P27, P32, P33, P34, P41, P42, P43, P44 | 36.36% |

The distribution of security threats in the provided data reveals several key insights into the vulnerabilities within security frameworks. Single Sign-On (SSO), representing 22.73% of the threats, highlights its dual nature as both a convenient authentication method and a significant target for attacks due to its centralized mechanism. OAuth and OpenID Connect, accounting for 27.27% of the threats, underscore their critical role in secure authorization, emphasizing the necessity for robust implementation to prevent misuse.

Role-Based Access Control (RBAC), with 15.91%, and Attribute-Based Access Control (ABAC) at 11.36%, reflect the vulnerabilities inherent in access control models, where misuse or insufficient management can lead to security breaches. The data associated with Activity Logs and Audit Trails, each at 25%, highlights their importance in tracking user actions and detecting anomalies, suggesting a need for robust protection against tampering and effective analysis to identify suspicious activities.

The Principle of Least Privilege (PoLP), also at 27.27%, indicates that violations of this fundamental security principle are common and can lead to significant breaches. Regular Security Assessments, with the highest representation at 36.36%, emphasize the critical need for ongoing evaluations to identify and mitigate vulnerabilities proactively.

In summary, the analysis points to OAuth and OpenID Connect, PoLP, and Regular Security Assessments as high-risk areas requiring focused security measures. Meanwhile, SSO and Activity Logs are substantial concerns, and RBAC, ABAC, and Audit Trails, although representing lower percentages, still demand significant attention. This underscores the importance of a comprehensive, multi-faceted approach to security, addressing potential vulnerabilities across various systems and practices.

### Architectural layered approach (SRQ3)

The adoption of micro-frontend architecture introduces security vulnerabilities across different architectural layers. To address these challenges, security measures must be implemented at each layer within the cloud-based architecture. We identify the following layers and their associated security considerations:

1. Application Layer:

* Individual micro-app of MFA. However, they are susceptible to blockage or compromise through malicious code injection.
* Security measures should prioritize adopting trusted micro-app and safeguarding them against internal and external threats.

1. Composition Layer:

* Interconnections among microservices can break down, and compromising a single microservice may impact the entire system.
* Insecure configuration options for individual microservices, their locations, and interconnections necessitate security measures at this level.

1. API Layer:

* Fine-tuned attacks on APIs can circumvent traditional security provided by API gateways.
* Malicious users may gain unauthorized access, potentially compromising assets.
* Rigorous security measures at API gateways are essential to mitigate these vulnerabilities.

1. Deployment Layer:

* Containers housing microservices can also harbor vulnerabilities.
* Unauthorized access or vulnerabilities from untrusted image sources pose risks.
* Security measures at the deployment level are crucial.

1. Soft-Infrastructure Layer:

* Infrastructure vulnerabilities affect all software entities on the network, including monitors, registries, message brokers, and load balancers.
* Techniques introduced at this level ensure the security of diverse software network entities and their configurations.

1. Hard-Infrastructure Layer:

* Hardware components are susceptible to intentional or unintentional attacks.
* Bugs or backdoors introduced during manufacturing can be exploited.
* Implementing error and backdoor detection mechanisms is essential.

Table 10 summarizes the distribution of proposed solutions across application layers in primary studies. Notably, solutions focus heavily on soft infrastructure and API gateways, while composition and hard-infrastructure layers receive less attention.

Table 10. Distribution of proposed solutions for different layers

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Description | Number of Studies | Percentage |
| Application Layer | Security measures for individual micro-app to prevent compromise and malicious code injection. | P10, P15, P20, P1, P25, P30, P31, P32, P33, P34, P35, P36, P38, P39, P44 | 34.09% |
| Composition Layer | Measures to secure interconnections among microservices and address insecure configuration options. | P28, P22, P37, P40, P43 | 11.36% |
| API Layer | Rigorous security at API gateways to prevent fine-tuned attacks and unauthorized access. | P3, P7, P12, P17, P23, P27, P40, P43 | 18.18% |
| Deployment Layer | Security measures for containers housing microservices. | P4, P9, P14 | 6.82% |
| Soft-Infrastructure Layer | Techniques to safeguard diverse software network entities (monitors, registries, etc.). | P2, P18, P6, P8, P11, P13, P16, P24, P26, P21, P29, P33, P34, P35, P36, P38, P40, P41, P42, P43 | 45.45% |
| Hard-Infrastructure Layer | Detecting vulnerabilities in hardware components introduced during manufacturing. | P5, P19, P41, P42 | 9.09% |

The distribution of studies across different application layers in micro-frontends reveals a focused approach to addressing specific security concerns within each layer.

The deployment layer, addressed by 6.82% of the papers, suggests that while deployment security is crucial, it might not be as extensively covered in the literature. This layer likely deals with configuration management, container security, and secure deployment pipelines. The soft-infrastructure layer, with the highest coverage at 45.45%, indicates a strong focus on securing the underlying software infrastructure. This includes virtual machines, container orchestration systems, and other middleware components, addressing issues such as security configurations, software updates, and third-party software vulnerabilities.

Finally, the hard-infrastructure layer, discussed in 9.09% of the papers, highlights the importance of securing physical and hardware-related components, such as servers and network devices. Although less frequently addressed, this layer remains critical to the overall security posture.

### Verification and validation techniques (SRQ4)

Verification and validation are two crucial aspects of ensuring the security and overall quality of micro-frontends in cloud-based architectures. For validating the proposed solutions for micro-frontends in a cloud-based architecture, we have identified several verification and validation approaches as outlined below:

For verification mechanisms:

* Static code analysis: It detects security vulnerabilities in the source code without executing the program via tools like SonaQube, Checkmarx or Fortify.
* Penetration testing: Simulate attacks to identify and exploit vulnerabilities in the application.
* Automated testing: Integrate common automated testing tools such as Selenium, Cypress security tests into the CI/CD pipeline to catch vulnerabilities early.
* Security audits: This approach performs comprehensive review of security policies, procedures, and controls via manual code review, compliance checks.

For validation mechanisms:

* Real-Time monitoring: This involves continuously monitoring application and infrastructure for suspicious activities and anomalies.
* Incident response simulations: Using case studies to validate the feasibility of the proposed solution.
* Vulnerability scanning: Regularly scan applications and infrastructure for known vulnerabilities.

Table 11. Verification and validation approaches

|  |  |  |
| --- | --- | --- |
| Category | Studies | Percentage |
| Verification |  |  |
| Static code analysis | P28, P9, P14, P17, P1, P23 | 13.64% |
| Penetration testing | P2, P18, P8, P12, P15, P5, P25, P27, P21, P32, P37, P40 | 27.27% |
| Automated testing | P3, P6, P10, P13, P19, P22, P26, P31, P34, P43 | 22.73% |
| Security audits | P4, P7, P11, P16, P20, P24, P29, P30, P33, P36, P38, P39, P42, P44 | 31.82% |
| Validation |  |  |
| Real-Time monitoring | P4, P7, P10, P11, P13, P14, P16, P19, P20, P1, P25, P26, P21, P30, P34, P38 | 36.36% |
| Incident response simulations | P2, P18, P8, P5, P23, P33, P37, P40 | 18.18% |
| Vulnerability scanning | P28, P3, P6, P9, P12, P15, P17, P22, P24, P27, P29, P31, P32, P39, P42, P43, P44 | 38.64% |

The analysis of the verification methods in the research articles provides valuable insights into the emphasis placed on different security verification techniques within micro frontend architectures.

Penetration testing is the most frequently addressed verification method, covered in 27.27% of the papers. This suggests a strong focus on actively probing systems for vulnerabilities through simulated attacks, which is essential for identifying weaknesses that could be exploited by malicious actors. The prevalence of penetration testing highlights its importance in ensuring robust security defenses. Static code analysis is covered in 13.64% of the papers, indicating a significant emphasis on analyzing code for potential security vulnerabilities without executing the program. This method is crucial for identifying issues such as code quality and adherence to security best practices early in the development process. Automated testing, addressed by 22.73% of the papers, further underscores the importance of integrating security checks into the development pipeline, enabling continuous validation of security measures as code changes. Security audits, the most extensively covered verification method at 31.82%, reflect the critical need for comprehensive evaluations of security practices, configurations, and policies. These audits ensure that security measures are correctly implemented and maintained over time, providing a systematic approach to managing security risks.

In terms of validation methods, real-time monitoring emerges as the most frequently addressed, with 36.36% of the papers covering this technique. Real-time monitoring is essential for continuously observing system behavior and detecting anomalies that could indicate security breaches. Its prevalence highlights the importance of proactive security measures in maintaining the integrity of micro frontend architectures. Incident response simulations, covered by 18.18% of the papers, emphasize the need for preparedness in the face of security incidents. These simulations help organizations test and refine their response strategies, ensuring they can effectively mitigate the impact of security breaches when they occur. Vulnerability scanning, the most extensively covered validation method at 38.64%, highlights the critical role of identifying and addressing security vulnerabilities. Regular vulnerability scanning enables organizations to discover and remediate potential weaknesses before they can be exploited by attackers, underscoring its importance in maintaining a secure environment.

## Analyze Results from Grey Literature

The analysis of grey literature research articles reveals additional insights into various aspects of security threats, and solution types which is depicted in Table 13.

Table 11. Verification and validation approaches

|  |  |  |
| --- | --- | --- |
| ID | Title | Year |
| GL1 | A Cyber Security and Digital Transactions to Educated the Micro, Small and Medium Business Community | 2023 |
| GL2 | Static-Analysis-Based Solutions to Security Challenges in Cloud-Native Systems: Systematic Mapping Study | 2023 |
| GL3 | Revolutionizing Frontend Development: Embracing Micro UI Architecture with Cloud Integration | 2024 |
| GL4 | Research and application of micro frontends. iop conference series materials science and engineering | 2019 |
| GL5 | Implementing Micro Frontends Using Signal-based Web Components | 2022 |
| GL6 | SIT: A Lightweight Encryption Algorithm for Secure Internet of Things | 2017 |
| GL7 | Fine-Grained Support of Security Services for Resource Constrained Internet of Things | 2016 |
| GL8 | Mitigation of Database Security Threats in Transaction Processing System | 2023 |
| GL9 | Security Threat and Vulnerability Assessment and Measurement in Secure Software Development | 2022 |
| GL10 | Impact of Security assessment for more secure software – A Tactics and Multi-Dimensional Perspective | 2024 |

Similar to the findings of primary studies, the grey literature predominantly focuses on external threats over internal threats, as evidenced by study GL2. This trend suggests that external security breaches are considered the most critical challenge. However, studies GL8 and GL9 highlight that many significant security threats originate from internal errors, such as malicious user actions or sensitive data exposure. This contrast indicates a potential research gap, emphasizing the need for more attention on internal threats to ensure comprehensive security coverage.

Different solution types are explored across the grey literature. Studies GL1 and GL3 propose new methodologies to implement micro UIs that enhance cloud integration, scalability, and security against external vulnerabilities via different cloud services. Study GL5 mentions modern tools like open-source security platforms and code lint tools to prevent system errors from developers. Additionally, study GL6 highlights the use of iframes as an optimal solution for micro-frontend applications to mitigate risks from users' inadvertent actions.

In summary, while grey literature mirrors primary studies by emphasizing external threats, it also brings to light significant internal vulnerabilities that require focused research and solutions. The diverse range of solutions, from methodologies and tools to regular security practices, underscores the need for a multifaceted approach to address the complex landscape of security threats in micro-frontend and cloud-based architectures. This approach is essential for developing robust systems capable of withstanding both internal and external security challenges.

# VALIDATION

In this section we describe construct, internal, external and conclusion validity threats and how we mitigated their effects on the obtained results.

* **Construct Validity:**

The construct validity of our study focuses on the operational measures that are studied to represent the goal and how we investigated while considering the research questions. The identification of primary research from the papers available in the literature is also reflected in it. The core aspect of designing the search strategy was the research topic, which guided us thorough the selection of the search query. Finally, we made two iterations of snowballing to feature additional research for consideration. Finally, we devised a set of stringent inclusion and exclusion criteria to ensure the inclusion of excellent papers, where only peer-reviewed journal and conference papers were accepted for their completeness and adequate findings.

* **Internal Validity:**

We acknowledge that our search process may not be comprehensive. As described in Section 3, we have used five scholar databases. We have not considered other scholar databases such as Scopus, which may include relevant micro-frontends publications. Our use of seven search strings may also not be comprehensive, as the search strings may leave out security-related publications during our search process.

* **Conclusion Validity:**

We apply a set of inclusion criteria to select which publications are related to security of micro-frontends. We acknowledge that the process of selecting these publications can be subjective, with the potential of missing related publications. We mitigate the subjectivity by using two raters who individually determined which publications are related to IaC. We apply qualitative analysis to determine the topics that are being discussed in IaC-related publications. We determine these topics by extracting qualitative codes and following the guidelines of qualitative analysis [58]. We acknowledge the process of generating topics can be subjective. We mitigate this limitation by using two qualitative raters.

* **External Validity:**

Our analysis is dependent on our set of 29 publications collected on May 2024. Furthermore, we have used certain scholar databases, which may not include all relevant publications for our paper. Due to the above-mentioned issues, generalizability of our findings can be limiting. We mitigate this threat by using five scholar databases.

# CONCLUSION AND FUTURE WORK

This section provides a summary of your research findings, emphasizes the impact of micro-frontends on cloud architectures (particularly scalability and security), and explores potential limitations and future research avenues.

## Conclusion

In this study, we conducted a systematic mapping of securing micro-frontends, focusing on threats, applicable platforms, and validation techniques of security proposals. We constructed a mind map to depict the relationships among dependent variables of micro-frontend security criteria. This comprehensive approach allows for a clearer understanding of how various security threats interact with different layers of micro-frontend architecture and how these can be effectively mitigated through appropriate validation techniques and security measures.

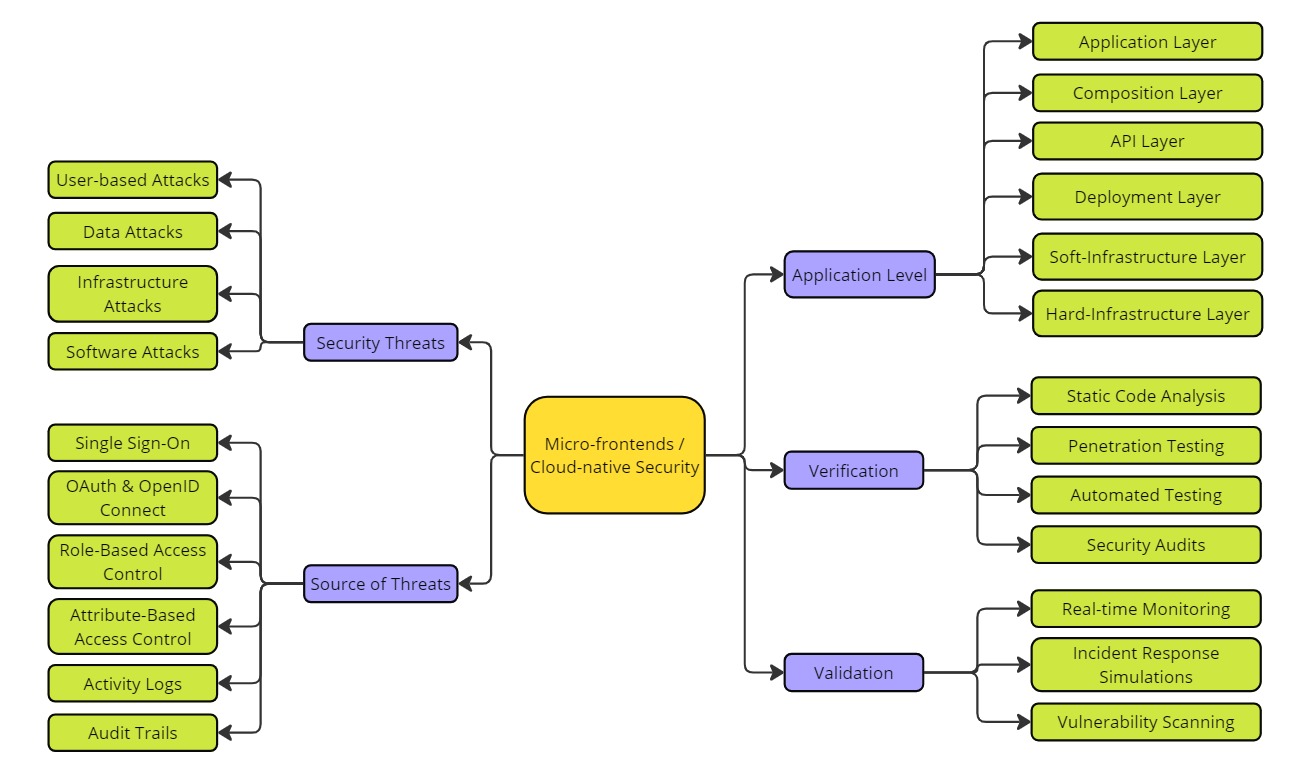


Figure 9. Criteria of MFE security

The study examined 29 papers published since 2016. The results revealed that unauthorized access, sensitive data exposure, and compromising individual micro-frontends are the most addressed threats in contemporary studies. Additionally, the results showed that auditing, enforcing access control, and prevention-based solutions are the most proposed security mechanisms.

Our analysis found that most proposed solutions are applicable at the application layer of micro-frontend architectures. Specifically, 34.78% of the papers proposed security solutions that are applicable across different platforms, with the same proportion noted for cloud-based solutions. Verification and validation methods predominantly relied on performance analysis and case studies.

We also proposed and made available an ontology summarizing and gathering the retrieved results. This ontology serves as a guide for developers, highlighting recognized threats and security mechanisms specific to micro-frontends. Interestingly, most addressed threats are well-known from other architectural styles, with few unique to micro-frontends. Specifically, compromising individual micro-frontends can lead to significant cascading failures within the system. Continuous monitoring has become popular among designers to preemptively address potential threats, while encryption remains the most widely used technique to protect sensitive data.

We observed an unbalanced research focus on external attacks and prevention techniques. Therefore, we advocate for more studies on internal attacks and proposing mitigation techniques. Additionally, further research is needed to address vulnerabilities at the individual micro-frontend and communication layers.

## **Limitations**

In this section, we will examine the potential areas of research that necessitate further investigation within the field. These research gaps have been identified through an analysis of the mapping results, indicating the need for additional scrutiny and exploration.

### Lack of Appropriate Solutions to Emerging [To be update]

Solutions that are cloud-focused and platform-independent are found at higher rates, 34.78% and 28.26% respectively. The interest in cloud computing is understandable due to the various facilities provided to companies when they adopt micro-frontends for developing their applications. Adopting micro-frontends for developing applications in the cloud allows companies to integrate existing legacy systems, grow with demands, and use up-to-date and intuitive interfaces (Author et al., Year).

Solutions provided for IoT applications are also garnering more attention due to the specificity and the growing market needs for these applications. However, more attention should be paid to 5G platforms. These are emerging technologies that require specific attention (Author et al., Year).

### Absence of Appropriate Comparison Techniques [To be update]

The research highlighted the use of multiple validation techniques, which can be attributed to the nature of the proposed solutions. For instance, formal verification is employed when there is a need to describe the formal specification of systems and their properties (e.g., p20) (Author et al., Year). On the other hand, performance analysis plays a crucial role in assessing the suitability of proposed solutions to specific environments and platforms, and it is also significant for decision-makers. Although performance analysis is applicable to various proposals and has been adopted at a higher rate (39.13%), it has not been adopted by all studies. It’s important to note that the diversity and incompatibility of validation techniques make the comparison of proposed solutions more challenging. The problem could be alleviated if all studies adopted performance analysis (Author et al., Year).

## **Future Work**

The future impact of micro-frontends on cloud-based architecture presents several promising areas for further research, particularly regarding performance, cost-effectiveness, and scalability.

* **Performance:**  
  Future research could explore the extent to which micro-frontends enhance performance in cloud-based architectures. This involves assessing load times, responsiveness, and overall user experience improvements compared to traditional monolithic frontends. Detailed benchmarking studies across various application types and cloud environments could provide a comprehensive understanding of performance gains. Additionally, analyzing how micro-frontends affect latency and resource utilization during high-traffic periods would offer valuable insights into their performance benefits.
* **Cost-Effectiveness:**  
  Investigating the cost-effectiveness of micro-frontends in cloud environments is another crucial area for future work. Research could focus on quantifying potential cost savings achieved through more granular resource allocation and dynamic scaling capabilities. Comparing the operational costs of maintaining multiple independent micro-frontends versus a single monolithic frontend would help clarify the financial implications. Furthermore, examining the long-term cost benefits related to reduced infrastructure investments and operational efficiency could provide a deeper understanding of the economic advantages.
* **Scalability:**  
  The scalability of cloud-based architectures with micro-frontends is a significant area for exploration. Future studies could examine how micro-frontends facilitate both horizontal and vertical scaling, allowing applications to handle varying loads more efficiently. Case studies of organizations that have successfully implemented micro-frontends to enhance scalability could offer practical insights and best practices. Additionally, analyzing the role of different cloud service providers in supporting the deployment and scaling of micro-frontends would be beneficial for understanding their scalability potential.
* **Integration:**  
  The integration of micro-frontends with existing cloud-based systems and other microservices is an important consideration for future research. Studies could investigate strategies for seamless integration, focusing on overcoming interoperability challenges. Understanding how micro-frontends interact with other cloud-native technologies, such as containerization and serverless computing, could lead to more cohesive and efficient cloud architectures. Research could also explore the impact of micro-frontends on continuous integration and continuous deployment (CI/CD) pipelines.

# REFERENCES

Agarwal, S., Kant, R., & Shankar, R. (2022). Exploring sustainability balanced scorecard for performance evaluation of humanitarian organizations. *Cleaner Logistics and Supply Chain*, *3*. <https://doi.org/10.1016/j.clscn.2021.100026>

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58. <https://doi.org/10.1145/1721654.1721672>

’Berns, S., & ’Kratzke, N. (2018). Microservices and their impact on the development of cloud-native applications. *Journal of Cloud Computing: Advances, Systems and Applications*.

Cam Jackson. (2019, June 19). *Micro Frontends*. <Https://Martinfowler.Com/Articles/Micro-Frontends.Html>.

Debski, A., Szczepanik, B., Malawski, M., Spahr, S., & Muthig, D. (2018). A scalable, reactive architecture for cloud applications. *IEEE Software*, *35*(2), 62–71. <https://doi.org/10.1109/MS.2017.265095722>

Guimarães, C., Gomes, P., & Costa, A. (2020). Maintainability in micro-frontend architectures. *Software Maintenance and Evolution*.

Hannousse, A., & Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study. In *Computer Science Review* (Vol. 41). Elsevier Ireland Ltd. <https://doi.org/10.1016/j.cosrev.2021.100415>

Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernandez, E. B. (2013). An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, *4*(1), 1–13. <https://doi.org/10.1186/1869-0238-4-5>

Kerimovs, A. (2023). *Scalability and Performance of Microservices Architectures*.

Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). *Evidence-Based Software Engineering and Systematic Reviews*. Chapman and Hall/CRC. <https://doi.org/10.1201/b19467>

Kitchenham, Brereton, & Budgen. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*.

Li, C., Tang, J., & Luo, Y. (2020). Elastic edge cloud resource management based on horizontal and vertical scaling. *Journal of Supercomputing*, *76*(10), 7707–7732. <https://doi.org/10.1007/s11227-020-03192-3>

Mannisto, J., Tuovinen, A. P., & Raatikainen, M. (2023). Experiences on a Frameworkless Micro-Frontend Architecture in a Small Organization. *Proceedings - IEEE 20th International Conference on Software Architecture Companion, ICSA-C 2023*, 61–67. <https://doi.org/10.1109/ICSA-C57050.2023.00025>

Mark Richards. (2016). *Microservices vs. Service-Oriented Architecture*. O’Reilly Media, Inc.

Martin Fowler. (2015). *Microservices: a definition of this new architectural term*. <Https://Martinfowler.Com/Articles/Microservices.Html>.

Mell, P. M., & Grance, T. (2011). *The NIST definition of cloud computing*. <https://doi.org/10.6028/NIST.SP.800-145>

Pereira, R., Sousa, P., & Fernandes, J. (2021). Independent deployment of micro-frontends. *Proceedings of the 2021 International Conference on Web Engineering*.

Perlin, R., Ebling, D., Maran, V., Descovi, G., & MacHado, A. (2023). An Approach to Follow Microservices Principles in Frontend. *17th IEEE International Conference on Application of Information and Communication Technologies, AICT 2023 - Proceedings*. <https://doi.org/10.1109/AICT59525.2023.10313208>

Rashid Dar, A. (2016). Survey On Scalability In Cloud Environment. In *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* (Vol. 5, Issue 7).

Sam Newman. (2021). *Building Microservices: Designing Fine-Grained Systems* (2nd ed.). O’Reilly Media.

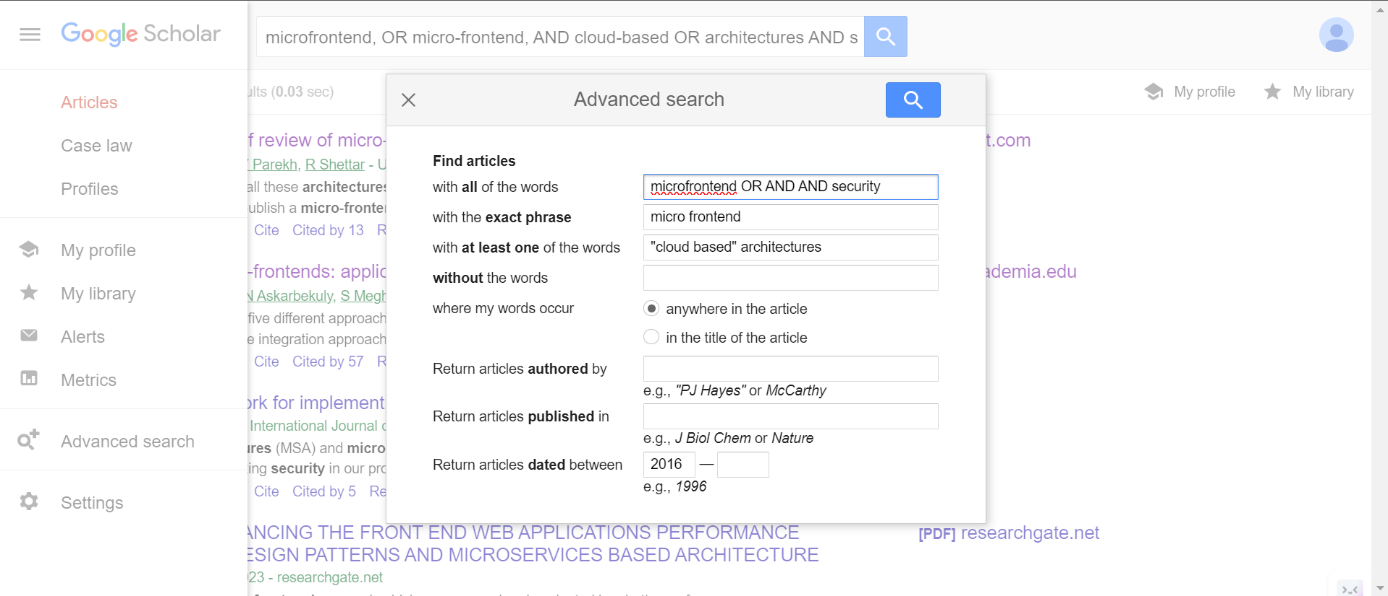
Santos, D., & Costa, E. (2020). Scalability in micro-frontend architectures. *Journal of Software Engineering Research and Development*.

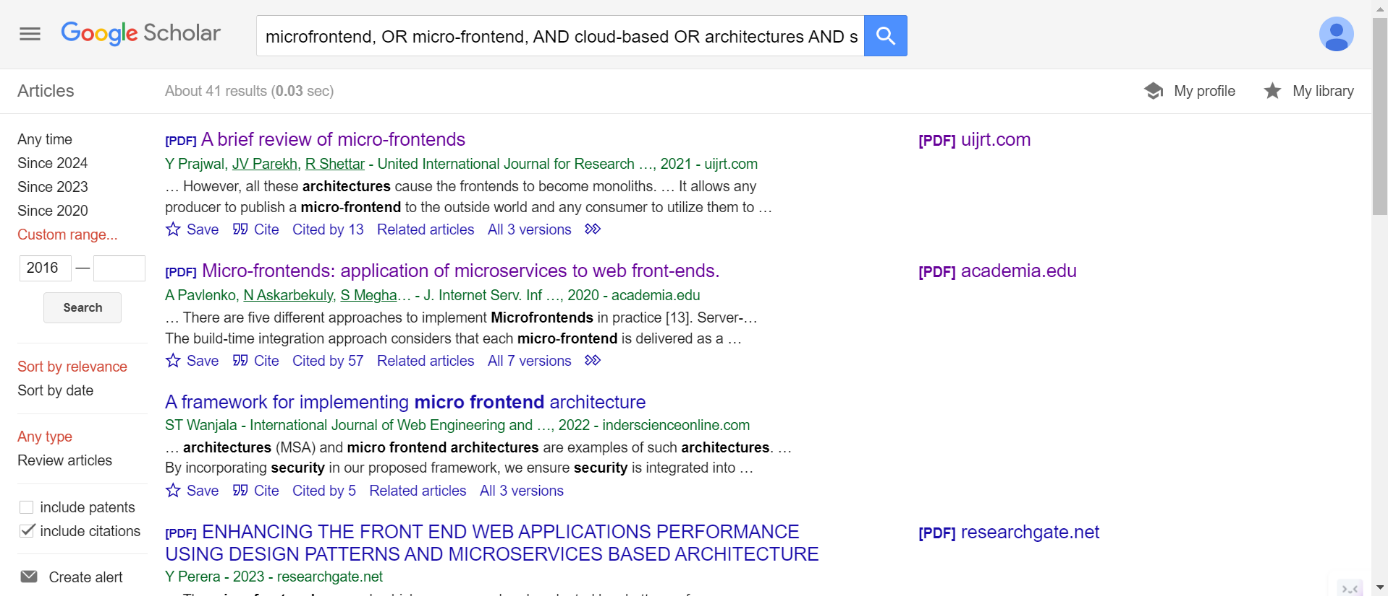
Thatikonda, V. K. (2023). Assessing the Impact of Microservices Architecture on Software Maintainability and Scalability. *European Journal of Theoretical and Applied Sciences*, *1*(4), 782–787. <https://doi.org/10.59324/ejtas.2023.1(4).71>

Yarygina, T., & Bagge, A. H. (2018). Overcoming Security Challenges in Microservice Architectures. *Proceedings - 12th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2018 and 9th International Workshop on Joint Cloud Computing, JCC 2018*, 11–20. <https://doi.org/10.1109/SOSE.2018.00011>

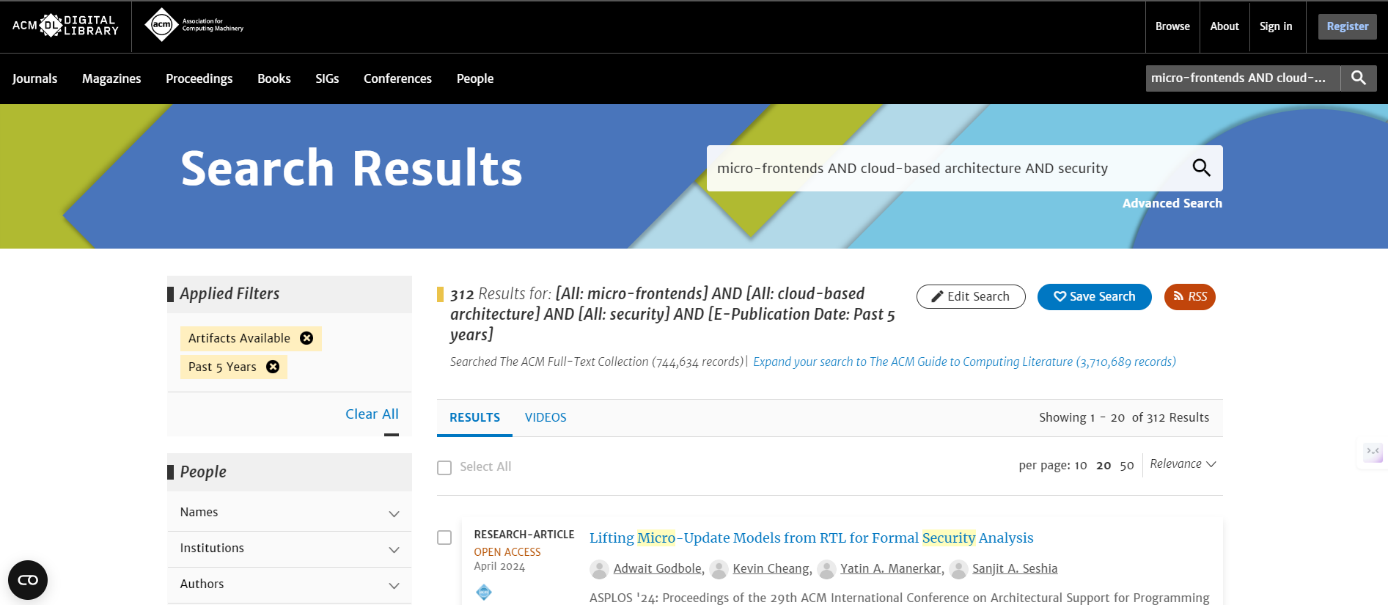
# APPENDIX A

Google scholar

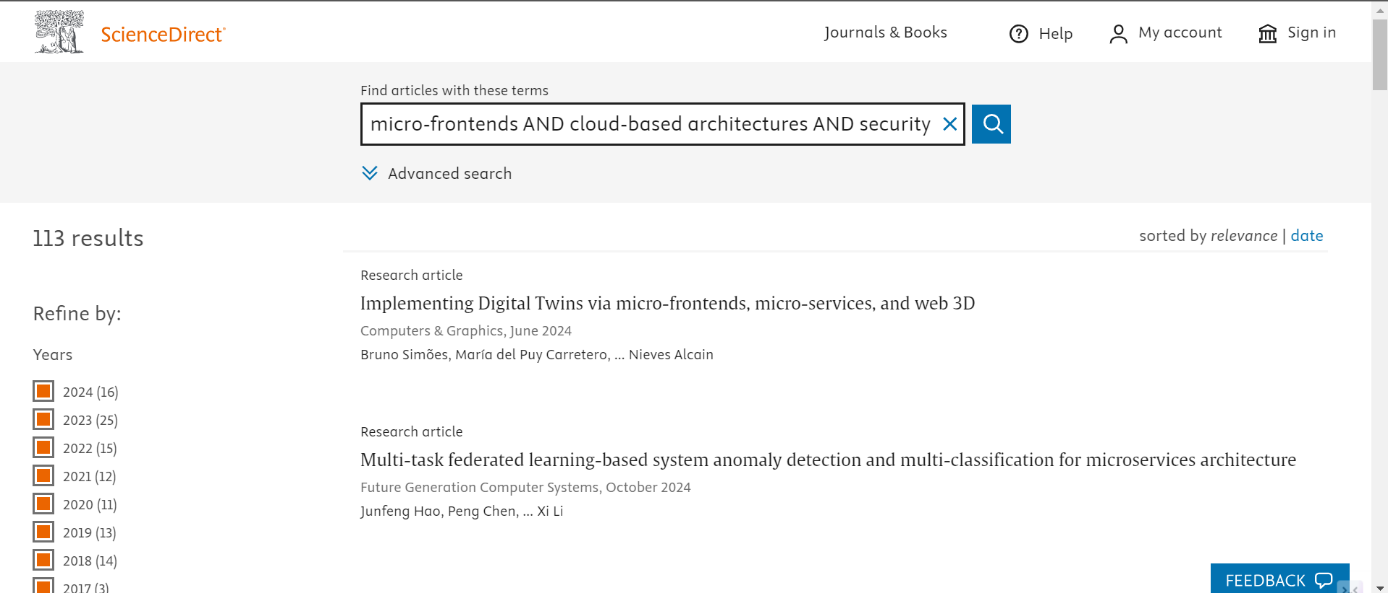




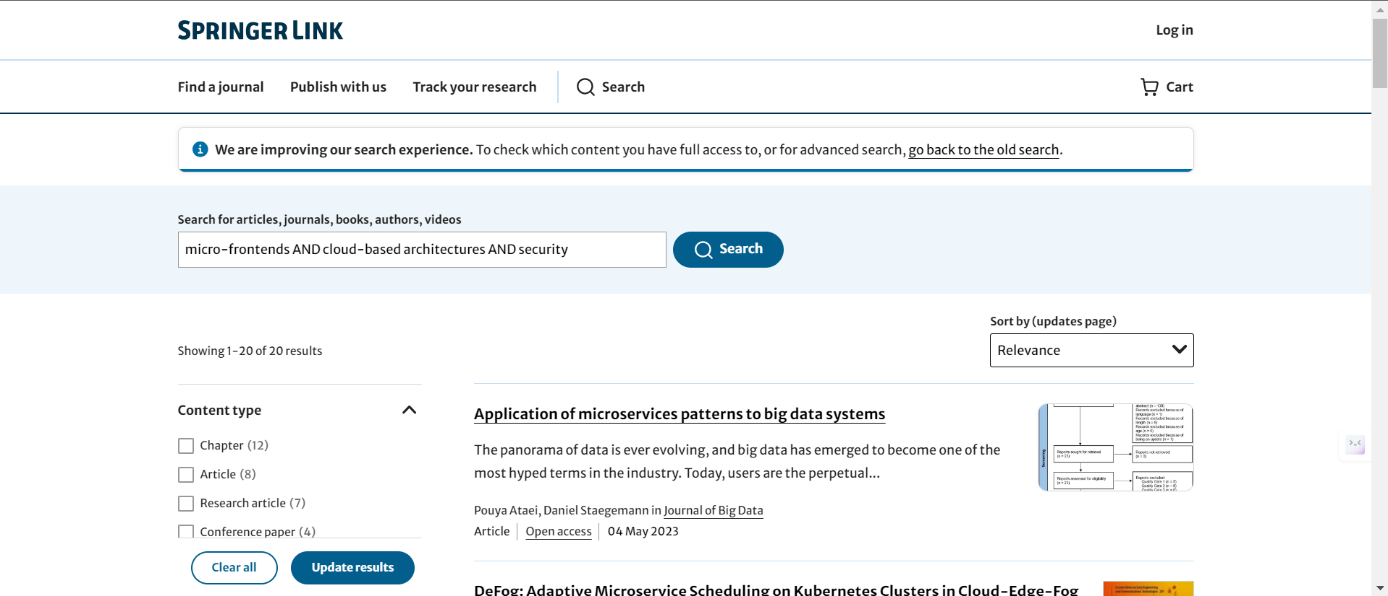
ACM Digital Library:



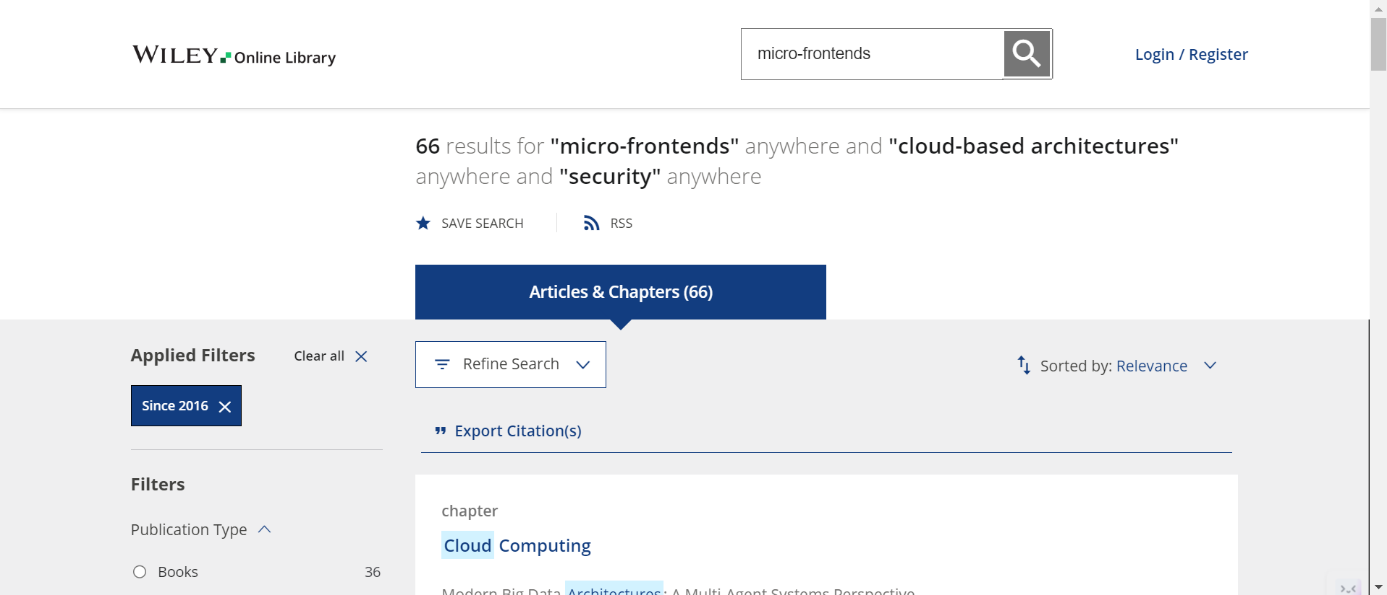
ScienceDirect:



SpringerLink:



Wiley Online Library:



# APPENDIX B