Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Security and Resilience in Modern Distributed Commerce Architectures

Shaibal Maji
University of the Cumberlands, USA

ARTICLE INFO

Received: 02 March 2026
Revised : 08 April 2026
Accepted:20 April 2026

ABSTRACT

Modern digital commerce platforms increasingly adopt distributed microservices-based architectures that fundamentally transform the security landscape through fragmented trust boundaries, multiple integration points, and complex failure modes. This article examines critical security challenges inherent in microservices architectures, including identity and authorization management across distributed systems, payment processing and transaction state management, data protection through tokenization, and internal trust boundaries requiring zero-trust principles. The article synthesizes findings from systematic literature reviews, empirical studies of open-source microservices systems, and practitioner surveys to demonstrate that security incidents in distributed commerce environments emerge primarily through subtle vulnerabilities during system degradation, partial outages, and cascading failures rather than direct external breaches. The article reveals that traditional perimeter-based security models prove insufficient for distributed microservices where services may number in the hundreds or thousands, requiring continuous identity validation, sophisticated token propagation mechanisms, and explicit state machines for transaction workflows. The article emphasizes that security and resilience must be treated as inseparable architectural concerns, with comprehensive strategies accounting for both the distributed nature of microservices and various failure modes that can compromise security guarantees. The article demonstrates that organizations must implement defense-in-depth approaches combining role-based access control, contextual authentication awareness, and zero-trust principles to maintain security integrity across independently deployed services maintained by different development teams with varying security expertise.

Keywords: Microservices Architecture, Distributed Systems Security, Zero-Trust Architecture, Transaction State Management, Identity Management

INTRODUCTION

The growing use of these architectures in the digital commerce platforms of the modern era, based on microservices and other techniques for modularizing and deploying at speed, does allow organizations to respond more quickly to market needs and to scale environments. But it also creates a new security landscape, no longer based on a single protected perimeter but on shared trust boundaries, multiple integration points, and different failure modes.

The increased use of API, cloud-native architecture, and service-oriented composition creates both increased opportunities and increased challenges for security. Moving from a monolithic application architecture to microservice architecture fundamentally changes the security model. As widely shown in microservices evolution literature, decomposing monolithic applications into smaller independent deployable services introduces new dimensions to security issues [1]. Every microservice behaves like a standalone entity having its own data storage, business logic, and communication interfaces, which multiplies the number of attack surfaces that an opponent can target. Because internal calls into the monolith have been replaced with inter-process communications over a network, those communications are subject to interception, tampering, and replay attacks. Therefore, security previously performed at the edges of the application now needs to be performed at each service boundary in an inter-service communication.

Security issues do not usually manifest as outside attacks but as indirect exposure that occurs when a system degrades, partially fails, or experiences a fault cascade. Research into security issues in microservices architecture

has found that the attack surface of microservices is larger due to factors such as increased interservice communication, distributed authentication/authorization infrastructure, and inconsistent security policies across a large number of independently developed services [2]. The service discovery, container orchestration, and API gateway layers that give microservices the flexibility they need can also expose potential security risks if they are not configured, managed, and monitored correctly. Authentication tokens must be transmitted between services, sessions tend to be stateless and separated, and authorization decisions are often distributed and difficult to enforce consistently.

This mandates treating security and resilience as architectural concerns. Resilience in distributed architecture introduces new failure modes not present in a monolithic architecture; failure of one service potentially leads to failure of other (dependent) services while running in degraded modes (where security concerns may arise). In the event of failure, such as service degradation, fallbacks and retry patterns may bypass security requirements that would otherwise be enforced. Elasticity and auto-scaling infrastructure, which are desirable features of commerce platforms, also introduce their own challenges. When services are added to the mesh, they need to be immediately authenticated and authorized before they can be trusted to handle sensitive data or financial transactions. Organizations must employ security controls that take into account both the distribution of microservices and the many failure modes that could break security guarantees, allowing security controls to continue working when individual services or communication channels are degraded.

Identity and Authorization in Distributed Systems

Identity management across distributed commerce platforms requires continuous validation rather than point-in-time authentication. When services operate independently across web, mobile, physical retail, and partner channels, authorization decisions become scattered across numerous components, each relying on transmitted identity context. This fragmentation introduces significant risk, particularly when authentication verification occurs only at initial entry points. Comprehensive research examining microservices architectures reveals that identity and access management represent critical challenges in distributed systems, where traditional centralized authentication models prove insufficient for handling the complexity of service-to-service communication patterns [3]. The decomposition of monolithic applications into independently deployable microservices fundamentally alters the authentication landscape, as each service boundary potentially requires separate identity validation mechanisms. In practice, this means that a single user transaction traversing multiple microservices may trigger authentication checks at each service interface, creating latency concerns that must be balanced against security requirements. The stateless nature of microservices, while enabling scalability and independent deployment, complicates session management and requires sophisticated token propagation mechanisms to maintain identity context across distributed service calls.

Static tokens and unverified identity claims create exploitable vulnerabilities, especially during partial system failures when fallback mechanisms may bypass standard security controls. Analysis of microservices architecture implementations in DevOps environments, based on a systematic mapping study of 47 primary studies conducted between January 2009 and July 2018, demonstrates that security emerges as one of the most negatively affected quality attributes when employing microservices in DevOps contexts [4]. The research identified 24 distinct problems with corresponding solutions regarding the implementation of microservices architecture in DevOps, with security challenges receiving particular attention from practitioners. Specific concerns included authentication token management, authorization policy enforcement across service boundaries, and the increased attack surface resulting from exposing multiple HTTP-based service endpoints. Industry reports referenced in the research indicate that 82% of organizations adopt microservices architecture to gain agility, 78% for scalability improvements, and 57% to enhance organizational performance, while 47% of organizations cited DevOps as their primary motivation for implementing microservices [4]. The study further reveals that microservices running via HTTP protocols and utilizing vulnerable third-party libraries or frameworks expose systems to potential attacks, particularly during degraded operational states when timeout mechanisms or circuit breaker fallback logic may compromise normal security validation processes. Among the 50 tools identified for supporting microservices-based systems in DevOps, security-focused solutions remain underrepresented compared to deployment and orchestration tools.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Effective identity enforcement demands contextual awareness throughout the session lifecycle, incorporating device signals, behavioral analysis, and continuous authorization checks. Identity services must maintain security guarantees even during degraded operation, implementing graceful degradation patterns that preserve authentication integrity while managing reduced functionality. The systematic examination of DevOps practices in microservices environments highlights that only a limited number of the 47 studies examined addressed problems and solutions related to specific quality attributes such as availability, reusability, reliability, maintainability, modularity, and portability in the security context [4]. Industry projections suggest that by the end of 2021, 80% of cloud-based applications would be developed using microservices architecture, with the worldwide DevOps market expected to grow to 5.6 billion dollars during the same period. This gap suggests that while organizations recognize security as paramount in distributed systems, practical guidance for implementing continuous authorization, context-aware authentication, and security-preserving degradation patterns remains underdeveloped. The challenge intensifies when considering that microservices-based systems must balance competing demands of performance, scalability, and security across numerous independently deployed services, each potentially maintained by different development teams with varying security expertise and operational practices.

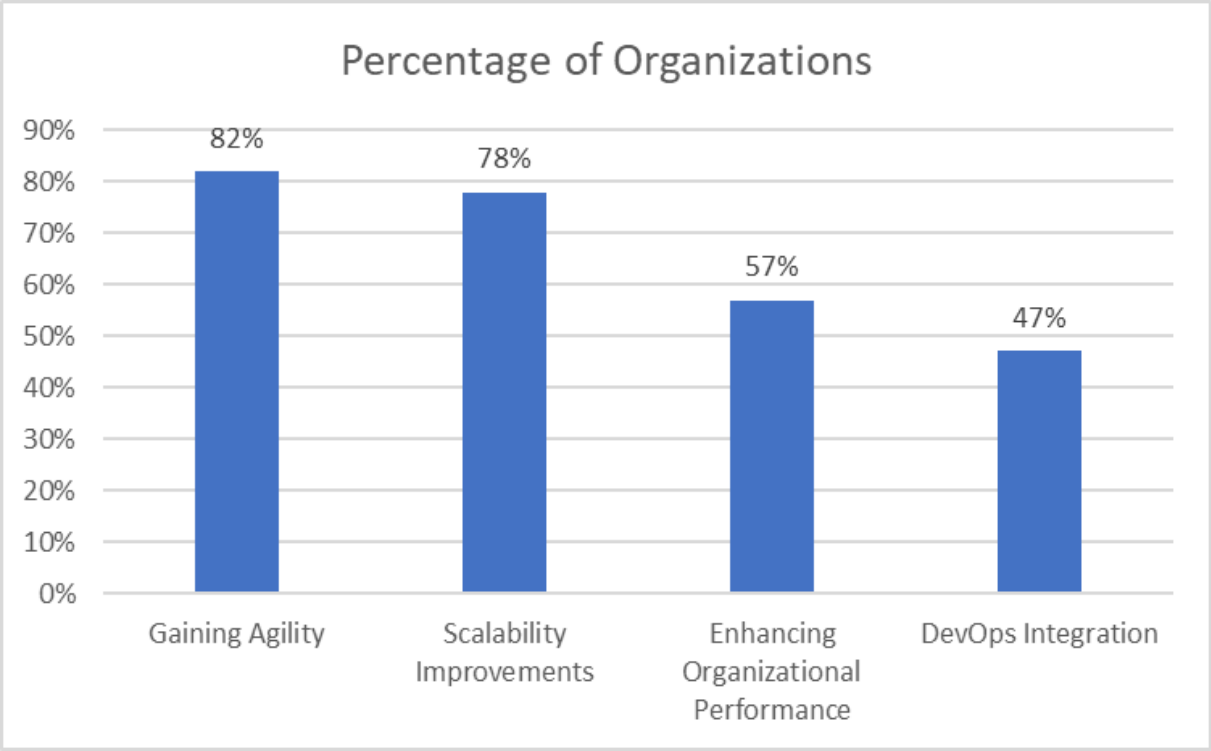

Fig 1: Organizational Motivations for Microservices Architecture Adoption [3, 4]

Payment Processing and Transaction State Management

Transaction State Management in Payment Processing

Payment systems, among the most valuable assets on commerce platforms, can suffer from transaction state ambiguity in distribution systems due to orchestration, asynchronous processing, third-party integration, and multi-service collaboration, where different services may not have a consistent view on the payment state. Also, an analysis of Saga for reactive microservices provides the understanding that transaction processing of distributed systems has substantially different characteristics than in monolithic systems. For instance, long-lived transactions can span multiple services and stay open for a long time [5]. These results show that customary two-phase commit (2PC) protocols can maintain strong ACID (atomic, consistent, isolated, and durable) guarantees but remain incompatible with the responsiveness requirement of reactive systems since they can be blocking. Today, locking protocols that guarantee consensus throughout the transaction are prohibitive for high-volume transactional workloads in terms of availability and user experience. Of four Java enterprise systems for sagas, when tested with

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

1000 and 10 threads, or 10,000 and 100 threads, it has been found that saga handling performance depends heavily on architectural choices [5].

Transaction Integrity Challenges

Retry logic, timeout handling, and idempotency windows also affect security; an attacker can exploit timing and state inconsistencies between retries to cause duplicate charges. Furthermore, in distributed systems, services may be designed to be asynchronous; that is, the service might intermingle requests without understanding their purpose with complete transaction semantics. Various performance tests on the saga sequencer implementations exhibited some severe problems regarding transaction safety. One implementation would deadlock the database after 5000 requests, and another implementation that retried 1000 requests in rapid succession from the same node would generate duplicate orders, for a total of 2000 orders being completed. In another experiment, 19,791 orders were successfully completed from 10,000 requests. A challenge both faced and reported with distributed payment processing was timeouts; for one framework, a fake one-second timeout was required for separate processing of a ship request and invoice response, impacting overall performance. With 10,000 requests and 100 threads, the time taken to process a request was between 22 seconds and 14 minutes and 5 seconds, depending on the implementation. The time taken to process all requests was between 3 minutes 56 seconds and 14 minutes 46 seconds, showing how critically important the architecture choice is. [5]

State Machine Approaches

Explicit state machines for payment workflows can improve user experience and correctness guarantees. By modeling transaction states, payment platforms can restrict unwanted states and reduce ambiguity in payment outcomes. State durability, strict idempotency, and transaction workflow observability bring the persistence of financial transactions despite heavy workloads or partial failures of the system. Saga pattern research has defined the requirements for compensating transactions to be idempotent and to semantically undo operations (beyond just changing the audit state) [5]. In comparisons of implementations, saga execution frameworks that use fluent APIs to declare and execute saga steps have resulted in superior performance, with one implementation processing all 10,000 requests with only 22 seconds of overhead for processing delay. The authors conclude that sagas based on BASE transaction models, ensuring availability over strong consistency via support for eventual consistency, are an advanced alternative to customary transaction processing for 21st-century distributed commerce applications requiring non-blocking transaction coordination. [5]

Study Component	Count/Value	Description
Test Scenario 1 - Requests	1,000	Performance testing with lower load
Test Scenario 1 - Threads	10	Concurrent threads for scenario 1
Test Scenario 2 - Requests	10,000	Performance testing with higher load
Test Scenario 2 - Threads	100	Concurrent threads for scenario 2

Table 1: Java-Based Enterprise Frameworks Analyzed for Saga Pattern Support [5]

Data Protection Through Tokenization

Tokenization is a simplistic form of data protection. Its effectiveness may be reduced in a distributed system if the token scope or lifetime accidentally exceeds its intended boundary. Tokens may also be sent to services with no legitimate reason to access sensitive information. In the context of microservice architecture, the defenses in depth paper observes that authentication and authorization require multiple security controls and that in a federated authentication using tokens, services must validate tokens across organizations and networks [6]. Role-based access control and defense in depth once again become valuable principles, as microservices run outside the context of a protected perimeter. Token management in particular becomes a concern with increasing number of services.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Each service may use a different permission and different access levels, giving rise to an exponential number of possible combinations of permissions. It is not uncommon to see organizations encountering tokens issued for specific purposes that gradually accrue broad permissions over time, as the use of the tokens expands due to developer actions, and only detected later, in large security audits [6].

Tokens should be short-lived, as granular as possible, bound to the intended recipient, and operationally isolated from sensitive operations, and services should not process tokens meant for data they do not require. Token usage provides early detection of incidents, compliance checking, and a way of observing a data flow perspective in distributed systems. A systematic literature review considered 3842 papers published between January 2014 and February 2022, and 85 studies were chosen for analysis. Security is one of the nine main challenge areas of microservices architectures [7]. The issues in these areas are the same for authentication, authorization, data protection, and secure inter-service communication. The systematic review process occurred using the IEEE Xplore (233 initial studies, 48 selected studies), ACM (755, 12), Springer (1619, 10), Science Direct (978, 11), and Wiley (174, 4) digital libraries and databases [7]. On an eight-question checklist, each using a three-point scale, the systematic review established that 82.3% of the primary studies included in the review reported good quality. Of the 54 included studies, 63.5% could be immediately applied. 68.2% of the included studies (58 primary papers) correctly indicated the validity of their results regarding security concerns. To identify the subcategories of security problems. Token management and data protection were identified to be the elements that need to be standardized in distributed service architectures. 59 papers with a score equal to or greater than six (69.4%) have presented relatively good evidence on security problems, and 11 of the studies were rated high quality [7]. These results suggest a trade-off between security and complexity in deciding how to use tokens, especially in microservices with hundreds or thousands of services deployed within a large-scale system.

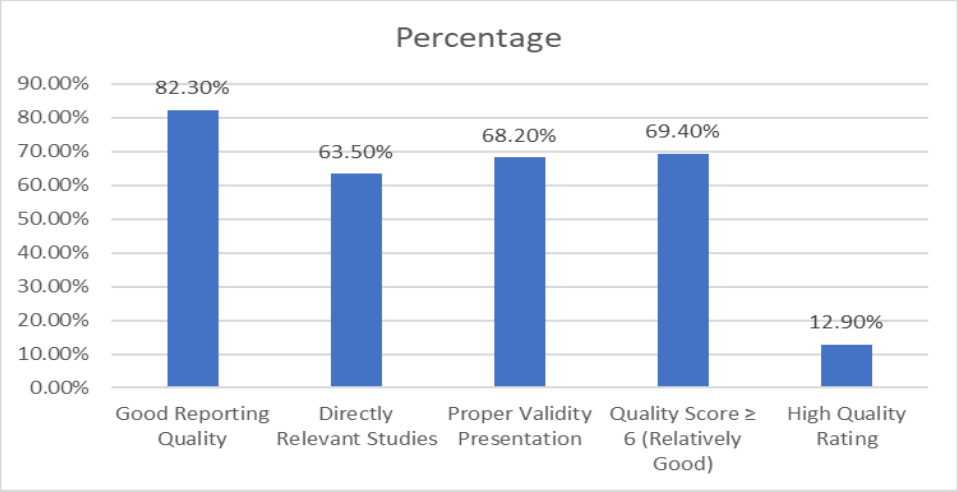

Fig 2: Quality Assessment Results for Primary Studies on Microservices Security [6, 7]

Internal Trust Boundaries and Zero-Trust Principles

Microservices architectures create a larger attack surface than perimeter security alone addresses, because lateral movement is possible on the implicit trust of the internal network after compromising a single service. Managing the vulnerability of dependencies and configurations is difficult. As the number of services grows within a microservice-based system, so do the chances of misconfiguration and unpatched libraries. Studies on the use of authentication and authorization in microservice-based architectures describe the management of trust boundaries as one of the main challenges when communicating across organizational and network boundaries [9]. The survey of authentication and authorization mechanisms concludes that perimeter-based models are not sufficient for microservices due to the fact that services can be deployed in a distributed system with hundreds or thousands of services. Analysis of existing authentication models indicates that microservices require advanced service-level identity propagation mechanisms, where security context is preserved throughout a chain of service invocations, and where every service-to-service transition may require separate authentication and authorization. This analysis

also reveals that services can establish trust links between them, without explicit configuration, leading to a situation where the compromise of a single service gives access to sensitive information anywhere in the system, unnoticed by the security system. This may happen if the services trust requests from internal networks [8].

Under the zero-trust policy for internal communication, every service request is intrinsically untrustworthy. Instead of merely authenticating the service, a central authority can be used to authenticate and authorize the request, regardless of whether it originates from trusted hosts. Transparent policy enforcement and strong service identity can be used to address perimeter security weaknesses. By investigating 2,641 issues from 15 platforms hosting open-source microservices projects on GitHub, interviewing 15 practitioners, and surveying 150 practitioners from 42 countries on six continents, it was found that security issues constitute 7.89% of all issues in microservices systems, and 213 security issues were identified [9]. Among them, authentication and authorization issues make up 4.55% (123 issues) of all issues. Access control issues account for 2.37% of all issues (64 issues), and issues with secure certificates or connections represent 1.59% (43 issues). In the causes of security issues, 5.55% are by bad security management (126 causes), with 55 coding-level security issues at 2.42%, 40 communication-level security issues at 1.76%, and 29 application-level security issues at 1.27% of causes. In a systematic literature review that screened 3842 papers and selected 85 of them as primary studies that were published from 2014 to 2022, according to [9], security issues exist on different levels, such as authentication, authorization, secure communication protocols, and trust boundaries of distributed services. Of the 150 surveyed practitioners, 18.67% answered that security problems happen very often, while 64.00% answered that they often happen. These data show the importance of addressing security problems in microservices production environments. The empirical results show that security solutions are often applied by fixing artifacts (54.54%, 1056 instances), by adding security features and artifacts (18.59%, 360 instances), and by modifying security-related components (10.95%, 212 instances) across the studied microservices systems [9].

Frequency of Security Issues	Percentage of Respondents
Experience Security Issues "Very Often"	18.67%
Experience Security Issues "Often"	64.00%

Table 4: Practitioner Survey Results on Security Issue Frequency [8, 9]

CONCLUSION

Empirical investigation of security and resilience in modern distributed commerce system architecture finds microservices-based systems expose substantially transformed security properties compared to monolithic systems. The investigation finds security challenges in multiple domains, including fragmentation of identity management requiring permanent authentication verification between services, ambiguity in transaction state in payment processing systems, tokenization complexity and configuration drift, and internal trust boundaries vulnerable to lateral movement upon service compromise. Systematic literature reviews, open-source microservices systems, and global practitioner surveys suggest that security breaches in microservices principally occur not due to external intrusion but mostly due to subtle vulnerabilities during system degradation or partial failure, such as fallback mechanisms that bypass existing security controls in the system. This suggests the inadequacy of perimeter-based security models in distributed microservices systems. This requires security measures such as continuous authorization, explicit transaction state machines, short-lived narrowly scoped tokens, and zero trust (the principle that internal networks should be considered hostile). Security and resilience should be treated as inseparable architectural concerns that need to be implemented in a consistent manner across multiple independently deployed services. Principles of automated policy enforcement, strong service identity, and graceful degradation of authentication integrity with reduced functionality are essential. The architecture and design of payment solutions, the general lack of security focus, and the prevalence of security challenges described by practitioners across countries and companies (many of which are common to many different industries) point to the need for standards, tools, and processes that work to build a secure infrastructure for a distributed commerce ecosystem with hundreds or thousands of services spread over multiple organizations and networks.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

REFERENCES

[1] Nicola Dragoni et al., "Microservices: How to Make Your Application Scale," Arxiv, 2017. Available: https://arxiv.org/pdf/1702.07149
[2] Luis Ferrera et al., "Security in Microservices Architectures," ResearchGate, January 2021. Available: https://www.researchgate.net/publication/349530577_Security_in_Microservices_Architectures
[3] Hulya Vural et al., "A Systematic Literature Review on Microservices," ResearchGate, July 2017. Available:https://www.researchgate.net/publication/318425527_A_Systematic_Literature_Review_on_Microservices
[4] Muhameed Waseem et al., "A Systematic Mapping Study on Microservices Architecture in DevOps," Sciencedirect, Dec. 2020. Available: https://www.sciencedirect.com/science/article/abs/pii/S0164121220302053
[5] Martin Štefanko et al., "The Saga Pattern in a Reactive Microservices Environment," Researchgate, January 2019. Available: https://www.researchgate.net/publication/335069774_The_Saga_Pattern_in_a_Reactive_Microservices_Environment
[6] Gaston Marquez et al., "Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping," ResearchGate, June 2019. Available: https://www.researchgate.net/publication/333805235_Security_Mechanisms_Used_in_Microservices-Based_Systems_A_Systematic_Mapping
[7] Mehmet Söylemez et al., "Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review," Researchgate, May 2022. Available: https://www.researchgate.net/publication/360933678_Challenges_and_Solution_Directions_of_Microservice_Architectures_A_Systematic_Literature_Review
[8] Alexander Barabanov et al., "Authentication and Authorization in Microservice-Based Systems: Survey of Architecture Patterns," ResearchGate, January 2020. Available: https://www.researchgate.net/publication/346523340_Authentication_and_Authorization_in_Microservice-Based_Systems_Survey_of_Architecture_Patterns
[9] Peng Liang et al., "Understanding the Issues, Their Causes, and Solutions in Microservices Systems: An Empirical Study," Aug. 2025. Available: https://arxiv.org/html/2302.01894v3