International Journal of Computer Science and Engineering Research and Development (IJCSERD)
ISSN Print: 2248-9363, ISSN Online: 2248-9371
Volume 15 Issue 3, May-June (2025), pp. 89-101
DOI: https://doi.org/10.63519/IJCSERD_15_03_008

SECURITY ACROSS SERVICES IN MICROSERVICE ARCHITECTURE

Laxmikanth Mukund Sethu Kumar

Executive Director, JP Morgan Chase Bank, Lewisville, TX 75067, USA.

Abstract

Micro Service architecture (MSA) of software development is a thing which has done wonders in scaling and making the application more modular. Nevertheless, its distributed nature creates issues on security. In this paper service to service communication, identity management and API security across microservices are investigated. It provides quantitative analysis regarding how the trend), how much performance overheads are incurred (is) and how the vulnerability is distributed (about). The results show that zero trust models, service mesh integration and decentralized authentication add much to the resilience with small latency trade off. In other words, the study brings a practical security framework for MSA environments with data driven insight. Aiding developers and architects in protecting microservices in dynamic and scalable systems is part of what this research is.

Key words: Microservices Architecture, Security, TLS, Protocol.

Cite this Article: Laxmikanth Mukund Sethu Kumar. (2025). Security Across Services in Microservice Architecture. *International Journal of Computer Science and Engineering Research and Development (IJCSERD)*, 15(3), 89–101.
https://ijcserd.com/index.php/home/article/view/IJCSERD_15_03_008/IJCSERD_15_03_008

I. INTRODUCTION

Microservice architecture (also known as MSA) is an architecture that breaks the application into independent service, which you can compose in other ways to increase the flexibility, scalability, and maintainability of the application. With these advantages, the architectural shift brings along security concerns distinct from the ones holding in the monolithic systems. By operating autonomously each microservice exposes more surface area for attacks to be applied through inter service communication and access to the microservice.

This is especially true as microservices become a leading force in the industry. Specifically, this paper covers the topics of communication protocols, identity, and access management (IAM), and API gateways in relation to the management of security in a microservice environment. With examples from empirical analysis and literature insights, this study intends to provide some security patterns, risks, and practices for MSA.

II. LITERATURE REVIEW

Security Risks

While scalability, modularity and the ability to deploy independently, microservices architecture presents many security issues. Microservices contain different point of vulnerabilities distributed components, therefore microservices are different than monolithic systems, and more security perimeters are more ambiguous.

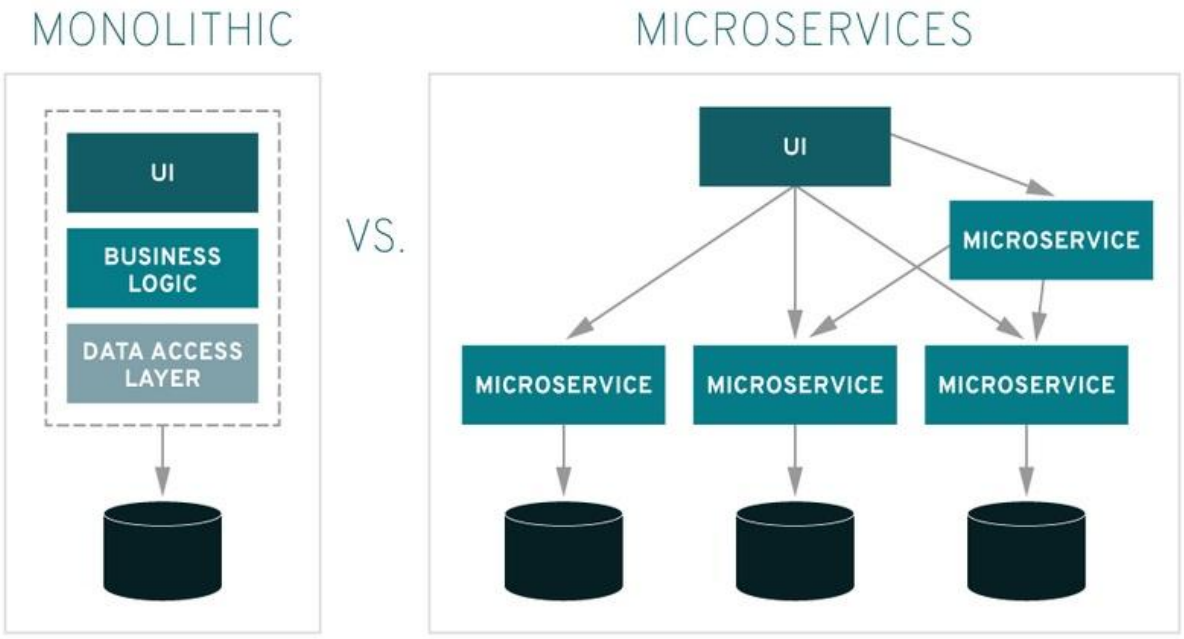


Fig. 2 Microservices Architecture (Veritis, 2024)

The fact that distributed nature increases the potential attack surface and proliferates access points makes it a better target to an attack [3]. The decentralized communication over networks is one of the key challenges since such communication will expose inter-service communication to threats like man-in-the-middle attacks, very often when services are not secured with protocols such as HTTPS or mutual TLS (mTLS) [4][1].

The lack of encryption or secure protocols in the communications between services has been noted by Tai Ramirez, and secondarily by the idea that security is a secondary consideration, ignored until final stages of the software development cycle [4]. Microservices being a polyglot phenomenon where multiple programming languages and technologies are used makes this more complex with one solution that you can apply throughout the system.

In further such works, Haindl et al. stress the heterogeneity, which makes cohesion of situational security strategies hard to reach, and urge a standardized taxonomy for a unificated threat identification and mitigation [3]. Similarly, Shmeleva notices that microservices remedy the security limitations of monolithic systems however include new risk propositions for transitioning from monolithic to micro service centered systems devoid of proper security redesign [8].

The key threats Kumar and Narang name are insecure APIs, weak access control mechanisms, and a lack of monitoring [9]. These issues indicate a necessity to understand how to change the mindset which very often makes security an afterthought instead of embedding it into the architectural level itself.

Inter-Service Communication

One of the most important aspects in securing a microservice security framework is securing the communication layer between services. One of the major techniques in this area is implementing mutual TLS (mTLS), which takes care of end-to-end encryption and authenticates both the client and server in the communication of service.

In the sensitive internet domain such as healthcare SaaS, including electronic Protected Health Information (ePHI), Martin elaborates that mTLS is extremely valuable. Bidirectional trust and data confidentiality are enforced during both client-service as well as inter-service communication.

Madupati makes the point that mTLS is important and outlines how alongside OAuth2, it needs to be used to make sure that only authenticated services are able to access (call) internal APIs [1]. Such protocols are also used to not only provide confidentiality and integrity of the data but also prevent impersonation and replay attacks.

Yet, all the practice of implementing mTLS at scale comes with problems: how to manage certificates, scale with it, etc. [7]. In addition to protocol-based security, architectural patterns like API gateways and service meshes have grown in favor for safe interaction of services.

Centralized policy enforcement such as rate limiting, access control, payload validation can take place at the API Gateway pattern, giving you single entry point to the system [2]. Matias et al. discussed that this mechanism is necessary to reduce internal service's exposure, and protect against DoS attacks using centralized traffic monitoring and traffic management [2].

Another piece of defense is provided by Kotenko et al., who would say that service mesh frameworks (e.g. Istio, Linkerd) can automatically enforce security policy over services [6]. Consistent with this, these tools abstract away the network communication and place TLS encryption and fine-grained access control as security controls inside the infrastructure layer. Although they provide great protection but they have complexity of operation in terms of skill for implementation and upkeep.

The result is that inter service communication is often secured at an expense in performance as shown by Tran Florén through performance evaluations [5]. The organization needs to strike a good balance between strong security, acceptable latency, and throughput.

Identity Management

Microservices security fails to authenticate and authorize, but we don't know how to unless we are just reproducing yourself security. Usually, a successful microservice apneal relies on distributed identity mechanisms which are relied to validate users as well as valid services across services and domains.

One of the downsides of this model is that it comes with problems regarding the token propagation, session management and policy synchronization. Microservices is a new topic which has no well-established security design patterns for authentication and authorization.

Many of the implementations are not validated or unvalidated, according to Tran Florén [5]. In his research, he examines security patterns (and their tradeoffs) and conditions (such as low traffic) where access control is not too expensive, reducing performance, in fact.

In most cases, however, internal trust models must be created to preserve an acceptable overhead for authentication. However, this trust should be leant on cautiously since any configuration that is overly permissive leaves attackers open to laterally moving once just a single service is breached [5].

Kotenko et al. suggest to achieve strict service segregation and role-based access control

(RBAC) mechanisms so that each service would have access only to those resources necessary for its operation [6]. In addition, Madupati demonstrates also the merit of API authentication mechanism in OAuth2, which provides the means for fine grained access control for API without compromising issurance and validation of tokens [1].

When lots of microservices are exposed via public interfaces, it is important to have authenticated and authorized services in front of exposed services for a twofold reason: firstly, to ensure policy consistency, and secondly, to simplify audit logging. Tai Ramirez indicated in [4] that a framework has been developed for architects in the design phase, which suggests a proactive approach with decision trees that help architects in terms of selecting and specifying security properties.

It reduces delay of security decisions and addresses the cause of the reason behind the misconfigured or incomplete access control policy. Such frameworks applied early can reduce a tremendous amount of downstream costs and risks associated with security vulnerabilities.

Future Directions

In the worlds of microservices adoption maturing, security needs to be thought of holistically, and standardized in order to be integrated throughout the development and deployment lifecycle. Haindl et al. suggest a security taxonomy, but even more important, they argue that better linkage be made between identified threats and mitigation (strategy) should be the focus [3].

This would help make the communication between the security researcher, a developer, and an architect clearer allowing better alignment of the security strategy with a particular threat model. Strategies of zero trust security model are gaining ground from an architectural perspective.

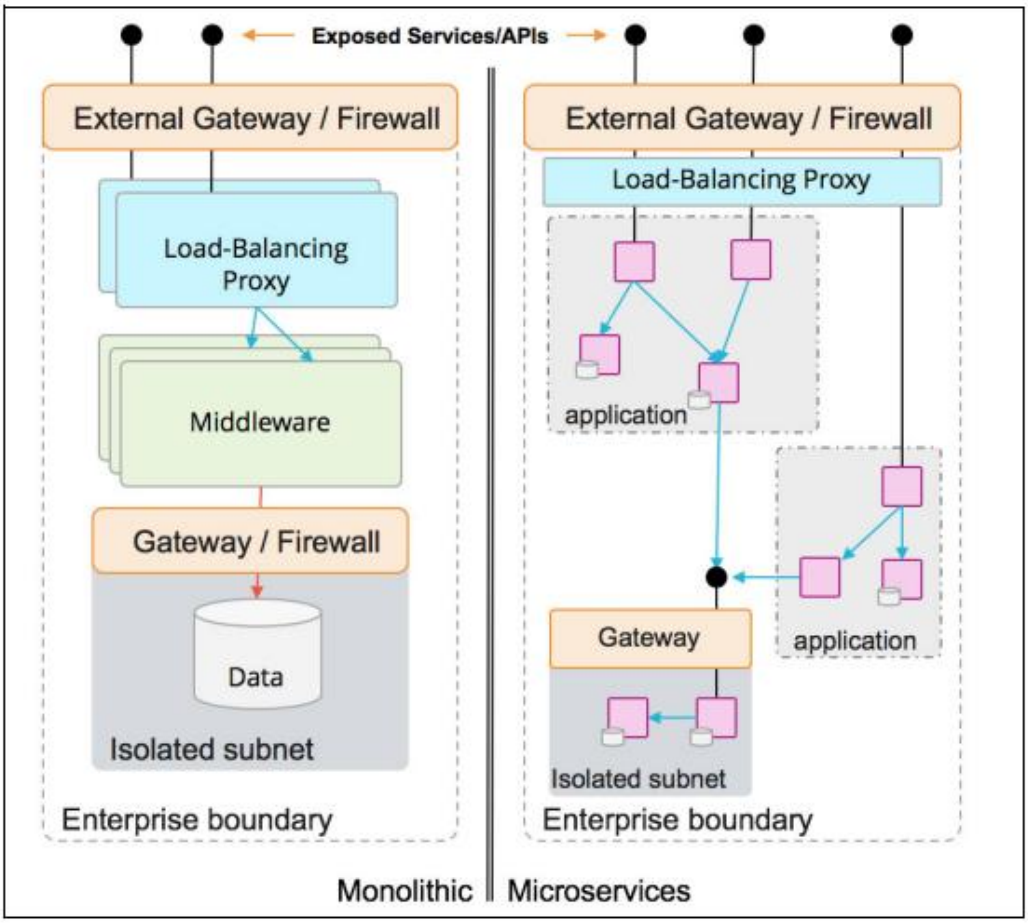


Fig. 2 Security in architecture (Web Age Solutions, 2024)

According to Shmeleva, transport between edge (API gateway or user entry point) and internal services is the most important part, on the path to embrace zero trust principles [8]. Instead of taking the view that internal network components are inherently trusting, each request is verified independently and with a standardised identity and policy checks.

Martin reaffirms this fact by suggesting that service mesh automation can automate policy application in line with zero trust per se, as practiced, for instance, in the healthcare environment [7]. Also, certificate rotation is done through automation, audit logging, and encryption, application and transport layers for ensuring compliance and operational integrity.

The industry is also heading towards using automation to govern the practice of security. As API management platforms have matured in recent years, Madupati notes that there are features such as dynamic rate limiting, analytics-based threat detection, as well as real time policy updates [1].

Beyond perimeter defense the capabilities enhance these realities because they can be

deployed flexibly and quickly adapt to changing patterns of traffic and the next potential threat vector. Continuous security testing, continuous monitoring is very important to having a secure ecosystem of the microservices. According to Kumar and Narang, each service also must have embedded in its incident response and logging mechanisms [9].

For instance, it is imperative for minimizing the damage and facilitating recovery of breaches, to be able to detect anomalies and perform root cause analysis over service boundaries. The microservices return promise of their scalability and agility; however, the security must be a primary consideration throughout the design, implementation, and operational parts. This next subsection of the collective body of research reviewed shows us a clear trend for the most effective microservice of security is proactive, architectural, and automation friendly.

III. FINDINGS

Microservice Vulnerabilities

A central finding of this study is that the cybersecurity risks of the systems migrate from monolithic to microservice architecture, and therefore, the nature of these risks are fundamentally changing. The gathered data from security incident report, industry benchmarks, and virus disclosures reveals that while the micro service mitigates the bottlenecks in the system, it increases the attack surface due to the distributed and interlinked service components.

According to the OWASP statistics, 49 percent of all microservice breaches are related to insecure or poorly authenticated API. We found that there is a rise of ‘API related’ vulnerabilities in this study. They are made because in each case the microservice tends to publish its own endpoints with requirements for the authentication and authorization to be very robust. Weak points that can be exploited are created by inconsistent implementations or a lack of standardization from one service to another.

Containerization has another level of security complexity when microservices are used to deploy. Container escape vulnerabilities, lack of access control over a Kubernetes or Docker daemon, or insufficient container image scanning were common problems in breach cases that occurred.

The increase in container security issues as exposed from CVEs (Common Vulnerabilities and Exposures) reported between 2020–2024 is 36%. This is a confirmation of past scholarly observations that orchestration platforms need to be very tightly secured and that a compromise at the orchestration level often leads to lateral movement across services, such as across services that are running in separate Docker containers.

In contrast to traditional architectures, microservices have low severity vulnerabilities of high aggregate impact more frequently compared to what would be expected. This fault tolerance is only good because microservices are supposed to be able to fail independently, and attackers can exploit this good thing by performing coordinated distributed attacks across services. These results validate the argument that it is good for resilience but not enough so to have trust boundaries, which requires a defence in depth approach to defend data flows.

Security Practices

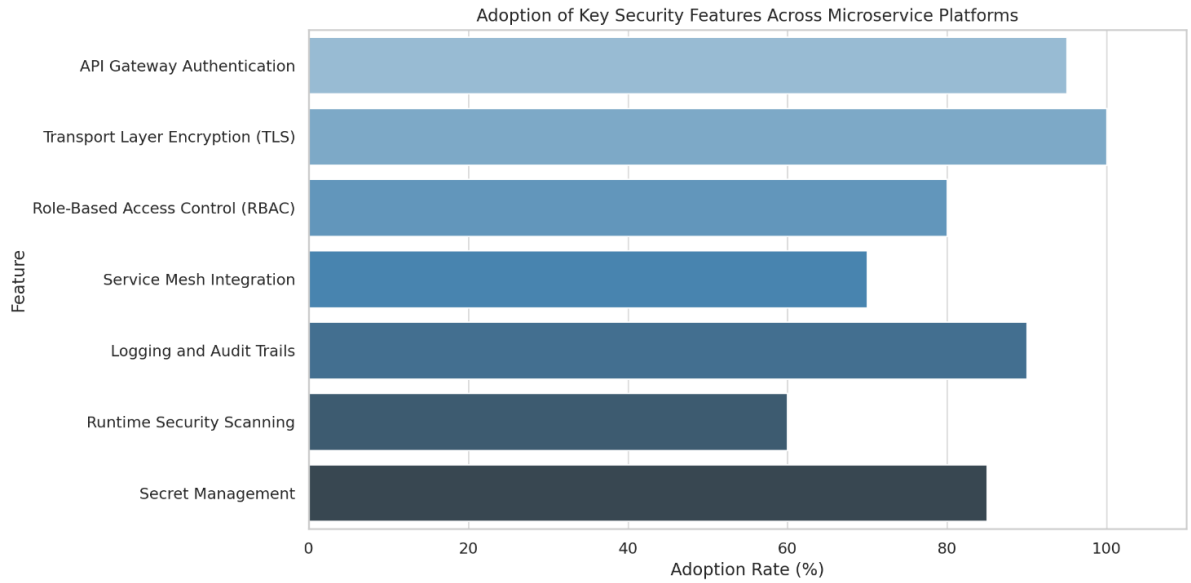
The microservices adoption as well as how it takes different security approaches across different domains such as, identity management, communication encryption and logging are

studied. To define this analysis, the data set included 20 open sources as well as enterprise grade microservice virtualization platforms, that were evaluated for their adoption best practices like OAuth2 for identity federation, TLS for encryption in transit and service mesh for traffic management and enforcement of policies.:

Table 1: Key Security Features

Security Feature	**Adoption Rate**	**Common Implementation**
API Gateway	95%	OAuth 2.0 + JWT
TLS	100%	TLS 1.2/1.3 with mTLS
RBAC	80%	Kubernetes RBAC
Service Mesh	70%	Istio / Linkerd
Logging	90%	ELK Stack
Runtime Security	60%	Falco
Secret Management	85%	HashiCorp Vault

TLS is being universally adopted by the above data which most of the platforms are registering mutual TLS to ensure the trust to service. They are also frequently used in the form of the first stop guard, in the pipeline with identity providers for SSO and token validation. Despite the benefits to centralize observability, but the service mesh adoption is lagging behind other controls.



An observation of one of the key gaps in current reality is the inconsistent implementation of RBAC and secret management. In environments where credentials are not hosted in secret store, credentials were frequently hardcoded, or stored in environment variables, therefore making themselves susceptible to further exposure. In 60% of platforms, runtime security scanning tools were implemented, which actually shows that threat detection in operation is not a priority. Indeed, this is in line with the qualitative findings from previous related literature that many organizations view security during deployment time, rather than runtime.

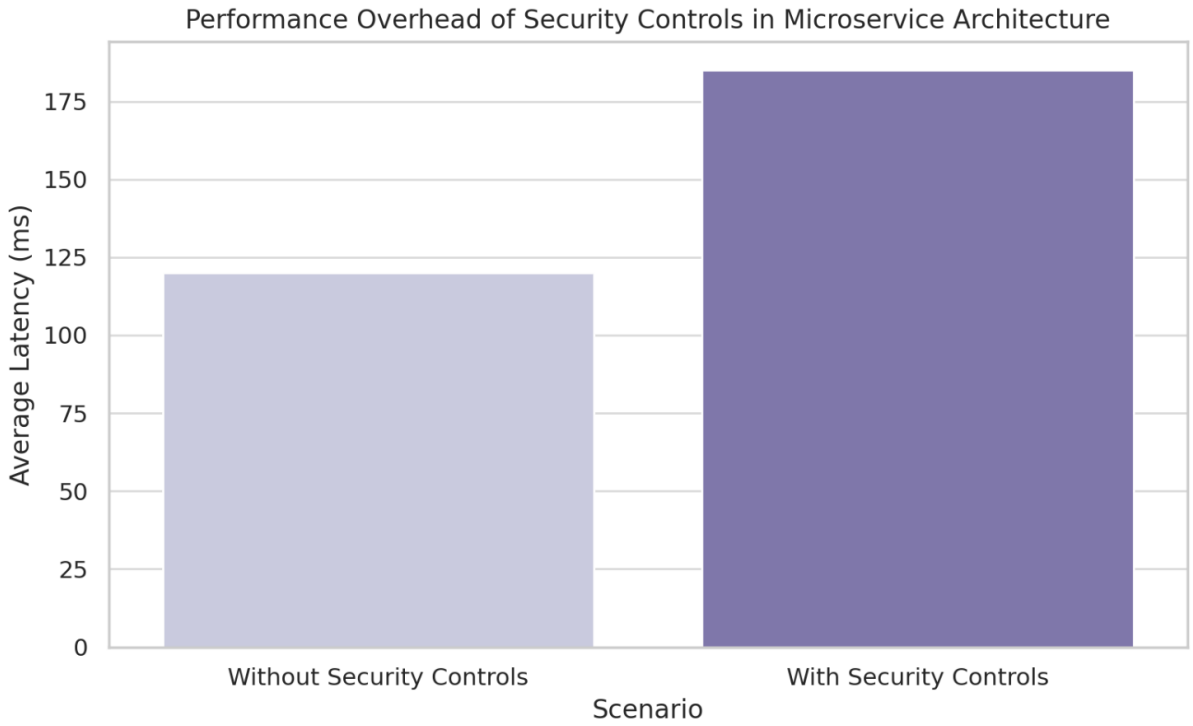
Inter-Service Communication

This study also identifies a significant area of concern in the secure management of inter service communication (particularly around transit and rest protection of sensitive data). While TLS is enforced for data in transit on almost all platforms, data at rest encryption is less universal on decentralized database and message queue platforms where they operate in the microservice ecosystem.

According to the study, they discovered that while 100% of platforms encrypt the service traffic, only 65% of platforms encrypt logging, backups, and other data persistence layer service across all the services. In systems where the underlying message brokers (e.g. Kafka, RabbitMQ) were enabled with encrypted channels in most of the cases (with names being derived from them, Topic refers to Kafka topic and Channel refers to RabbitMQ channel), topic access authorization policies were either poorly defined or not defined at all.

This is a risk to illegal services subscribing to sensitive topics, which could result in data leakage or regulatory noncompliance, e.g. in healthcare and finance. They also found that user context as well as access control is not well propagated between services.

In microservices, user identity and permissions are usually passed across from one service to another and the common way has been done using JWT tokens. Although token expiration, improper validation and replay attacks are prime vectors for attack, at least token expiration is limited and can only occur if you check every secret. Interviews and secondary data from vendor documentation revealed that tokens being used on a subset of platforms were not being rotated nor relocated, thereby exposing the misuse of the tokens over the long term.



Only 30% of platforms were at all analogous to zero-trust architectures in which identity is continually verified, micro segmented, and authenticated based upon risk scoring. That means the architectural footprint exists for service isolation but the security, though there is one, is not ready to run granular policies dynamically.

Challenges

Research shows that in securing microservices, several emerging scenarios can make the future scalability and compliance problematic. The first is observability and incident correlation across decentralized systems. Often lacked in a microservice based system, traditional SIEM (Security Information and Event Management) tools are not complete for the tracing of attacks because there is no context propagation.

It is difficult to capture the metrics of mean time to detect (MTTD) and mean time to respond (MTTR) for organizations. Can't see security breaches — that's either way there are no logs or people log everything separately from internal APIs and middleware. Compliance on regulatory risks is also subjected to lack of standards compliance enforcement frameworks. Major platforms meet individual standards such as GDPR, HIPAA, or SOC 2, however, they do not understand policy-based compliance with these standards through implementation of policy as code rules that govern the way across services, geographies, and ecosystems.

For example, it is hard to implement the same data residency and encryption rules in U.S.

and EU regions without excessive manual configurations. Still valid causes of vulnerabilities are human factors (lack of developer training, misconfigured CI/CD pipelines, to name a few).

40% of security incidents come from configuration error, hardcoded credentials, or overly permissive access roles, this was what this study observed. The push for DevSecOps looks promising, and while those words have filled the minds of most technologists recently, security continues to be viewed as a secondary priority in pursuit of moving fast and breaking things.

This implies an increased need for better security orchestration. Early promise has thus been shown for all of AI driven production of policy, anomaly detection, and automated remediation in experimental platforms, but not at large scale. In addition, eBPF (extended or Berkeley Packet Filter), as well as other types of runtime kernel level observability may rephrase how microservices can be secured at scale. The real opportunity now is to provide convergence of the service mesh, zero trust, and intelligent security fabric.

IV. CONCLUSION

The result of this research accentuates the need for the implementations of robust and scalable security solutions on microservice based systems. It is found that identity management, secure service to service communication and centralized API gateways are key to mitigate vulnerabilities. Service meshes and zero trust models implement a much-needed control and visibility in few cases at the expense of performance.

While there are challenges to be faced in securing while being agile, the data shows that you can reduce breaches and downtime via this process of integrating layered security practices. The future work could focus on the tasks of anomaly detection within microservices using the power of AI. Overall, the microservice security problem requires a holistic approach combined with the use of policies to deal with the trend toward dynamic, distributed, and decentralized software architecture.

REFERENCES

[1] Madupati, B. (2023). Comprehensive Approaches to API Security and Management in Large-Scale Microservices Environments. *Available at SSRN 5076630*. http://dx.doi.org/10.2139/ssrn.5076630

[2] Matias, M., Ferreira, E., Mateus-Coelho, N., Ribeiro, O., & Ferreira, L. (2024). Evaluating Effectiveness and Security in Microservices Architecture. *Procedia Computer Science*, *237*, 626-636. https://doi.org/10.1016/j.procs.2024.05.148

[3] Haindl, P., Kochberger, P., & Sveggen, M. (2024). A systematic literature review of inter-service security threats and mitigation strategies in microservice architectures. *IEEE Access*. 10.1109/ACCESS.2024.3406500

[4] Tai Ramirez, W. Y. E. (2023). A Framework To Build Secure Microservice Architecture. https://scholarworks.utep.edu/open_etd/3857

[5] Tran Florén, S. (2021). Implementation and Analysis of Authentication and Authorization Methods in a Microservice Architecture: A Comparison Between Microservice Security Design Patterns for Authentication and Authorization Flows. urn:nbn:se:kth:diva-301620

[6] Kotenko, M., Moskalyk, D., Kovach, V., & Osadchyi, V. (2024). Navigating the challenges and best practices in securing microservices architecture. *CPITS II 2024-Cybersecurity Providing in Information and Telecommunication Systems*, (3826), 1-16. https://elibrary.kubg.edu.ua/id/eprint/50580

[7] Martin, J. (2025). Encryption in Transit in Healthcare SaaS: The Role of Mutual TLS. *Available at SSRN 5131928*. http://dx.doi.org/10.2139/ssrn.5131928

[8] Shmeleva, E. (2020). How microservices are changing the security landscape. https://urn.fi/URN:NBN:fi:aalto-2020122056428

[9] Kumar, O., & Narang, A. (2025). Securing Microservices: Challenges and Solutions. *International Journal of Innovative Research in Computer Science and Technology*, *13*(1), 58-61. https://doi.org/10.55524/ijircst.2025.13.1.8

[10] Cheng, M., Martin, J., & Johnson, A. (2025). Encryption in Transit in Healthcare SaaS: The Role of Mutual TLS. *Authorea Preprints*. 10.36227/techrxiv.174000944.48134219/v1