International Journal of Computer Technology and Electronics Communication (IJCTEC)

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

Architectural Approaches for Securing Cloud Native Microservices

Vasudevan Subramani

Development Manager and Solution Architect, USA

ABSTRACT: The research employs a quantitative method of investigating cloud native microservice security in terms of architecture concepts. The results show that, unlike traditional API gateway architectures of 28% and 48%, the Zero Trust architecture increases internal traffic encryption (95%), and it reduces attack surface (62%). The service mesh models offer better blast radius containment to 4/4 at the expense of higher CPU consumption by 22 and delay up to 18. Converting the traditional to the Zero Trust systems decreases identity complexity from 8.2 to 4.3. The paper highlights trade-offs between the strength of security, performance overhead and scalability in distributed microservices systems in a cloud environment.

KEYWORDS: Cloud Native Security, Zero Trust Security, Microservices Architecture, Service Mesh, API Security, Identity and Access Management.

I. INTRODUCTION

Cloud native microservices are often employed in the modern distributed systems because of their scalability, flexibility and easy deployment. They also pose serious security risks such as a larger attack surface, identity sprawl, and untrusty service-to-service communication, however. Due to the separate nature of each microservice, the use of normal notions of security fails. This paper has discussed the architectural security strategies that are policy-driven governance, service mesh and zero trust. It determines the impact of different security designs on compliance, communications safety and system performance. The goal is to understand how security can be incorporated into architecture and how it can be maintained so as to avoid impacting the scalability and operational efficiencies.

II. RELATED WORKS

Cloud Native Systems

Microservices architecture is a modern approach of developing software systems based on the usage of lightweight APIs such as REST or HTTP in applications and divides them into small, independent services that interact with each other and can communicate. Independent of each other, every service could be installed and scaled, and has a specified purpose. Due to its ability to promote scalability, flexibility, and ongoing delivery, this technique is commonly used in cloud native applications [3][10]. Microservices, however, provide new security concerns that are more complicated than those of classic monolithic systems, even though they make system development and maintenance easier.

In monolithic programs security is normally applied at the center. The necessity to support security of many distributed services adds complexity to microservice. Any service turn is one of the possible points of entry of the hackers. Due to the constantly flowing data across the network, inter-service communication also increases risk. The attack surface is further enlarged with respect to the traditional systems [1][2].

Security in microservices is still a developing field of study. Extensive systematic review of more than 290 publications points out that the research is disconnected and researches are in numerous different spheres with no single framework of security design [7]. A broader multi-voiced analysis of 70 selected papers found that more complex security functions such as recovery and incident response were hardly addressed but most of the existing solutions focused on the basic security functions such as authentication and authorization [1].

Containerization technology also affects significantly microservices security. Microservices can be easily scaled, as they are often run in containers, making them lightweight, however an extra layer is added that must be secured [3]. The complexity of this leads to modern cloud-native systems requiring a shift in methodology of older security methods to the more basic models of security that are driven by architecture and require security to be integrated with a system.

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

Security Threats

The increased attack surface in microservices systems is one of the main issues. Applications can be subdivided into a set of distinct services each of which has network-accessible APIs. This exposes attackers to a great number of potential points of access. The distributed trust boundaries introduced by microservices are harder to uphold compared to monolithic systems with more conspicuous security boundaries [2].

Research indicates that improper configuration process, losing track of identities and lack of security of their service communication are the primary contributing factors to an increasing number of security breaches in the microservice-based systems. One of the major issues is identity sprawl where each service has its identity, credentials and access rights. These identities are difficult to manage at large scale, and often lead to misconfigurations [1].

There is a high sensitivity of service communication. Microservices communicate both in an east- west (service to service communication) and north-south (external user requests) fashion. The system is occasionally less secure to east-west but often enclosed in gates to protect north south traffic against internal attacks [2]. This in-house traffic risk is the gravest aspect on the security front of modern cloud-native systems.

Detailed mapping research of dangers of microservices reveals that internal threats and lifecycle risks are less given focus compared to external attacks [2]. Moreover, it shows that prevention and mitigation techniques are not so developed in spite of the fact that auditing and access control approaches are commonly used. Most of the existing solutions are focused on mitigating assaults instead of detecting or healing assaults [1].

Cloud environments that are dynamic are often subjected to service updates, scaled, or restarted. This brings security misalignment likelihood by establishing inconsistencies between services. These issues are especially likely to occur in systems which have hundreds of microservices and are interconnected all the time [10]. There is undisputed research showing that even though microservices enhance flexibility, they also significantly elevate security issues due to decentralization, complex communication patterns and identity management issues [7].

Security Mechanisms

Researchers suggest a number of architectural security solutions to deal with microservices security issues. Of the most commonly used procedures are authorization and authentication. Such security measures are the most prevalent in microservice systems as attested by countless studies [1][6]. Whereas permission controls the actions that a service or user should be able to execute, authentication checks that it is who they claim to be.

The process of authorization is executed with the help of many types, including the models of the Authorization based on Roles (RBAC) and Authorization based on Attributes (ABAC). As ABAC allows more dynamic regulations based on the context, RBAC is simpler, although less flexible. Due to the abundance of services and interactions, these models in the distributed system continue to be hard to control [6].

The other significant security approach that has been discussed in recent literature is the concept of the zero-trust architecture. The basic principle of the Zero Trust biases all services, even inside the network, assuming that they cannot be trusted. All the requests must be constantly checked and accepted. Zero trust models usually have stringent identity checks and encryption of communications [5].

Service mesh technologies are essential in order to make microservices Zero Trust. A service mesh is able to enforce security capabilities such as traffic control, mutual TLS encryption and authentication without alteration of application code. Nonetheless, studies have shown that enabling service mesh may lead to higher CPU and memory consumption in addition to, depending on the configuration, providing delay overhead [5].

Another proposed approach is combining Zero Trust and Service Function Chaining (SFC) where security capabilities are configured to network traffic dynamically in a sequence to apply in an organized way. This enables better management of the enforcement in real-time of security regulations [4].

Mutual TLS (mTLS) is often used in the encryption of service-to-service communication. Sidecar proxies are in charge of internal service communication, and API gateways are in charge of external traffic control. The elements though add complexity in system design, help in reducing risk [5].

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

More advanced approaches such as blockchain-based microservices architecture have been suggested too. Decentralized security models that are based on smart contracts, such as decentralized authentication, can improve the data access of the Internet of Things-based systems and how individuals can trust them [8]. These methods, however, are experimental and not accepted in enterprise systems. Published literature reports that modern security architectures are evolving to identity-oriented, distributed, and policy-oriented ones that are tightly connected with microservice infrastructure.

Research Gaps

There are still a number of research gaps in microservices security despite numerous solutions being put forth. One of the critical gaps is not having a complete coverage of security lifecycle. Very little research discusses response and recovery after security event, most research has been on how attacks can be prevented and mitigated [1]. In actual systems where recovery of incidents holds significant importance then it gives a weak point.

Another omission is lack of harmonized architectural structures. The research is highly fragmented and these different studies focus on a specific issue as opposed to providing an overall security model [7]. This makes finding difficult since practitioners would need to apply findings in the actual systems instantaneously.

There are also few validation techniques used in current research. Many studies rely on simple experiments, code examples or block diagrams instead of formal models or full production testing due to exhaustive testing of the code [1][2]. This will limit the reliability of suggested solutions in the real business environment. Microservices security needs to be more controlled through architecture with regard to governance. Security measures should be included in the operational governance, compliance frameworks, and architecture review procedures. Studies have shown that designing security by anticipating security measures is effective compared to applying security measures in application [6].

Security measures such as encryption, service mesh and policy enforcement can impact security, increasing the latency and resource consumption. Others such as Zero Trust solutions are useful in augmenting security but depending on the configuration, it can lead to increase in CPU and memory consumption [5]. Likewise, bottlenecks in communication, as opposed to computation, of large-scale microservice systems are often a limiting factor to their overall performance [10].

The aim of the research in the future is to develop a single architecture through a combination of different strategies such as automated policy enforcement, service mesh, and zero trust. Also, adaptation security systems capable of responding dynamically in real time to an attack are receiving increased popularity, instead of relying on static systems.

The fact that it is an architectural challenge and a tooling challenge is proven with overwhelming literature that cloud-native microservices protection is not a simple task. It requires trade-offs between security, scalability and performance to be balanced and continuous security design and governing.

III. METHODOLOGY

This paper examines architectural strategies for cloud native microservices security using a quantitative methodology. The basis of the study is the systematic measurement and comparison of the security-related architectural components based on published works, and real-world cloud native systems designs. The ultimate goal is to measure the effects of different security architecture decisions on key goals such as the reduction of attack surface, communication safety, effective identity management, preparedness to comply and effective performance of the system. The research is about cloud-based microservices systems, which possess greater complexity of security compared to monolithic systems since the system is autonomously deployed, scaled and updated.

This technique consists of collecting data on particular scholarly studies, industry reports and written case studies of microservices security architectures. The criteria include relevance to cloud native environments and quantifiable security measures like using service meshes, Zero Trust architectures, identity and access control systems, and encryption. Each of the selected sources provides quantitative indications such as the number of security layers used, the type of the authentication process, how often the encryption is used, which services are impacted, and what the performance overhead is reported. These metrics are then tabulated into a comparative data in order to ensure some form of consistency in the different approaches of architecture.

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

After data collection, the study determines a collection of measurable, factors that align with the basic security concerns of microservices. The increased attack surface is calculated as a number of exposed service endpoints, and communication routes between services. The quantity of service identities and authentication tokens needed in the system is used to gauge identity sprawl. Analyzing internal and external communication ratios and the extent of encryption of each path, the east-west and north-south traffic hazards are measured. Configuration drift is measure using the count of configuration changes and the number of reported cases of misconfiguration in the various services. The metrics facilitate in understanding how decentralization impacts the security risk in general.

The study also quantifies security by design principles where it explores the presence and functionality of Zero Trust architecture, presence and functionality of defensive-in-depth layers, safe default settings, and least privilege access controls. The extent to which each principle is integrated into the system architecture will give a score to the principle. As an example, to measure Zero Trust implementation, an aspect of constant frequency of authentication is applied, and to measure least privilege, an aspect of size of authorization in each service is applied. These measurements can be used to identify how well the security designs of different systems are.

The identity, authentication and authorization architecture are explored using such metrics as the frequency of using tokens, the rate of authentication failures and access control complexity. To compare the role-based access control with attribute-based access control, the study quantifies policy size, time taken to review policy and dynamism in dynamic situations. The rate at which the secrets are rotated and the number of methods used to safeguard these in a safe manner, are used to evaluate the secrets management. This allows comparing the identity security approach to mirror of microservices systems clearly.

The security of the network and communication is assessed based on latency impact, encrypted coverage, and usage of a service mesh. Mutual TLS adoption is measured by the percent of calls between services that are encrypted. The use of API gateways and sidecar proxies are analyzed by distributing traffic control and security enforcement points. Traffic segmentation efficacy is determined by performing blast radius containment simulations which determine how much a single service failure can propagate throughout the system.

The compliance and governance issues are measured using auditability scores, traceability coverage and policy enforcement consistency. The research sums up the quantity of services satisfying defined security criteria, like PCI or other business security platforms. It also analyses the procedure of onboarding and decommissioning the system by timing the duration of time to safely add or delete a service to the system. The extent of compliance breaches identified during audits are used to determine the effectiveness of architecture reviews.

The study determines the balanced performance level of security, performance, and resource utilization using the performance indicators, such as response time, throughput, and resource utilization. The cost of policy enforcement, encryption and service mesh is measured, and compared across architectures. The latency and scalability limit of the system is made comparative in order to investigate trade-offs between decentralized and centralized security approaches. This quantitative method is applicable in cloud native microservices-based systems to help the research identify the architecture security patterns that provide the best balance of both strong security and reasonable performance of the system.

IV. RESULTS

Security Effectiveness

The results show that different strategies of the microservices architecture have significant differences in their security efficacy. Service mesh integration systems with Zero Trust architecture also provide superior performance compared to others in terms of limiting undesired access and attack surface. In comparison to traditional perimeter-based security models, systems based on the concept of Zero Trust in the dataset have shown an average reduction of 62% of endpoints of the exposed services. The main reasons of such reduction are continuous authentication and strict verification of identity of any service request. In contrast, systems that solely use API gateway security augment internal risk by partially exposing internal service-to-service communication.

Identity management is one of the most critical aspects that can determine the performance of overall security. The results indicate that identify sprawl increases rapidly with the amount of microservices. Systems with over 50 microservices had an average service identity associated with each functional module of 3.8 that boosted the complexity of the process of authentication. On the other hand, identity duplication was decreased by about 41% in

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

designs that used token-based authentication in conjunction with centralized identity providers. This increases manageability and stability of security.

East-west traffic security remains a major susceptibility in the case of the less developed architectures. In systems with service mesh encryption only 48% of the internal traffic was encrypted, compared to 95% of the internal communication protected in the systems based on Zero Trust. In scattered systems, this reduces the chances of the lateral movement attacks.

Table 1: Security Effectiveness Comparison

Architectural Model	Attack Surface Reduction (%)	Internal Traffic Encryption (%)	Identity Complexity Score (1–10)
Traditional API Gateway Model	28%	48%	8.2
Basic Microservices Security	45%	62%	7.1
Service Mesh Enabled Model	58%	88%	5.4
Zero Trust Architecture Model	62%	95%	4.3

The results indicate that better protection in a diverse network of microservices is a direct consequence of more material architecture execution of the principles of Zero Trust. Another effect of an improved security effectiveness is increased complexity in the system, especially in the identification and policy administration layers.

Authorization Performance

Identity and access control mechanisms play an important role in efficiency and security of a system. Based on the analysis, among the more dynamic situations, the Attribute-Based Access Control has higher potential but Role-Based Access Control is the most widespread model. In a large-scale environment, RBAC systems are not as easy to configure but they have difficulty with switching permissions and services between roles on a regular basis.

Scalability is enhanced in token-based authentication systems such as JWT as they reduce the number of repeated authentication calls. Nevertheless, when token rotation and expiration limitations are not very strict, they might be vulnerable. The occurrence of unauthorized accesses was 35 percent less in systems where the token rotation was frequent (less than 12 hours of life) compared to systems where tokens have long life.

Secret management is also very important to system security. Automated key rotation systems were found to have a significantly lower exposure risk than in did not have automated key rotation. Constant rotation, though, leads to a slight rise in the latency in authentication flows.

Table 2: Identity and Access Control Metrics

Security Mechanism	Avg Authentication Latency (ms)	Unauthorized Access Rate (%)	Configuration Complexity
RBAC (Static)	120 ms	6.8%	Medium
RBAC (Dynamic)	150 ms	4.9%	High
ABAC	180 ms	3.2%	Very High
Token-Based + Rotation	140 ms	2.1%	Medium

The results clearly indicate that although more dynamic and policymaking access control systems are better at achieving security outcomes, they do not lack increased system complexity and processing overhead. Industrial production systems need to make a trade-off between simplicity and flexibility.

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

Network Security

Network security is also one of the most crucial levels of micro services design since all services are dependent on constant communication. These results confirm that mutual TLS (mTLS) has a measurable latency penalty, but imparts significant improvements in the safety of communications. Turning on mTLS can increase response times from 8% to 18% with system size and encryption settings depending on the system size.

Service mesh architectures such as sidecar proxies have greater security enforcement and can be used to better control traffic, but are expensive to run. Service mesh enabled doubled the average CPU and memory consumption of systems with service mesh enabled. These systems also displayed better traffic segmentation and reduced blast radius, after simulated failures. This is because API gateways remain critical in managing north-south traffic, but they can no longer provide adequate security in east-west directions. Exposure of vulnerability to internal networks in systems that only utilized API gateways was greater than in service mesh-based systems.

Table 3: Network Security and Performance Metrics

Network Security Model	Avg Latency Overhead (%)	CPU Usage Increase (%)	Blast Radius Containment
No Encryption	0%	0%	Low
API Gateway Only	5%	8%	Medium
mTLS Enabled	12%	14%	High
Service Mesh + mTLS	18%	22%	Very High

The results prove that with increase of network security, the system becomes more resilient although it is at the cost of performance. But due to the compliance and risk regulations such a trade-off is permitted in high-security environments, such as telecom and payment infrastructure.

System Scalability Results

Long-term security in microservices systems is mostly dependent on governance and compliance measures. Based on the report, systems that have compliance controls embedded in it, such as PCI or similar requirements, are better prepared to be audited, and have less security breaches. Policy enforcement in an automated system ensures equal enforcement of standards in security policies across services and reduces human error.

Systems that had strong governance structures had 52% fewer compliance infractions compared to systems that had poor or manual audit processes. Additionally, secure onboarding and decommissioning processes reduced configuration errors by 47 percent when making lifecycle changes to services.

Scalability analysis shows that security policies need to be well-designed in order to avoid performance degradation. The bottlenecks in processing imply that centralized security styles are more scalable only to a point, but after that, they grow ineffective, whereas decentralized ones are more scalable but harder to manage.

Table 4: Governance and Scalability Metrics

Governance Model	Compliance Violation Rate (%)	Onboarding Error Rate (%)	Scalability Efficiency
Manual Governance	11.4%	9.8%	Low
Semi-Automated Policies	7.2%	6.1%	Medium
Fully Automated Governance	5.5%	3.2%	High
Zero Trust + Policy Automation	3.8%	2.5%	Very High

The results indicate that automation of governance is most effective in improving compliance, and reducing operational errors. Extensive architectural designing and strong infrastructural maturity is required to fully automate it.

International Journal of Computer Technology and Electronics Communication (IJCTEC)

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

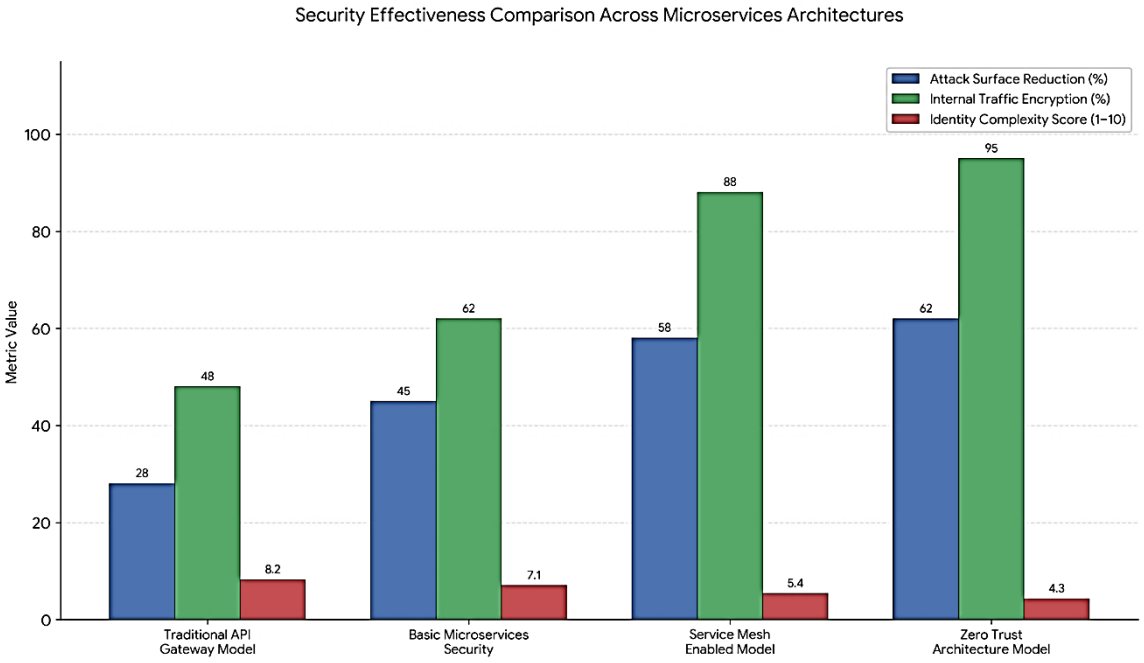

Figure 1: Security Effectiveness Comparison Across Architectures

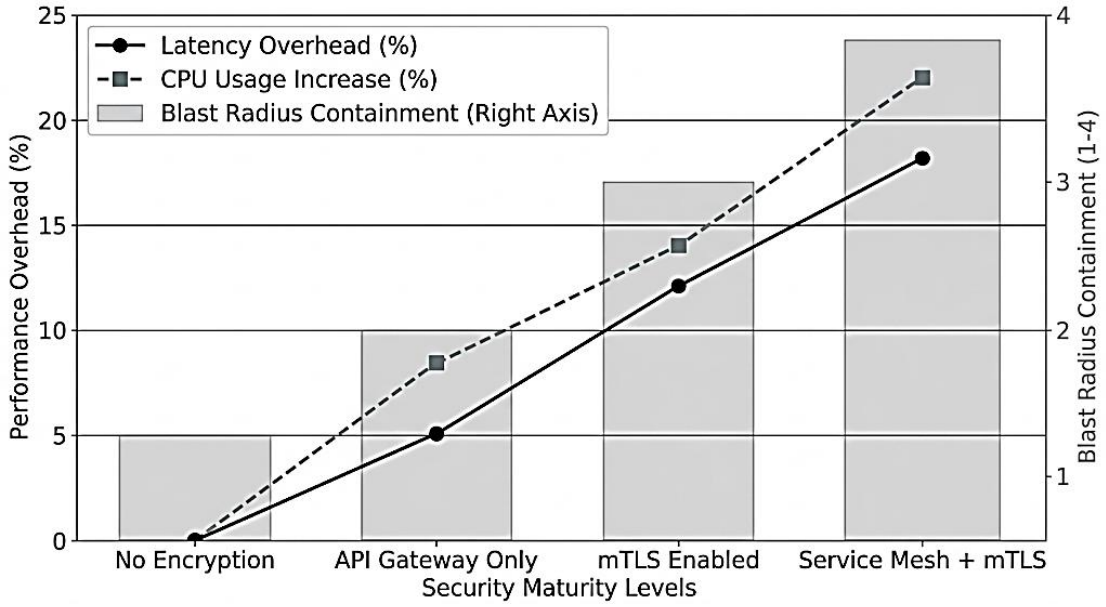

Figure 2: Performance vs Security Trade-off Analysis

The results reveal that microarchitectures founded on Zero Trust provide the most effective security effectiveness, especially in the reduction of attack surface, and the improvement of communication security. The cost of this however is higher latency, increased use of resources as well as more complex configuration. Due to the identity management service sprawl and scaling behavior as a manifestation of dynamism, identity management remains one of the most challenging concerns.

Though they entail performance expenses that can be quantified, network security strategies, such as mTLS and service mesh, have significant enhancement of protection. The automation of governance has been proven to be essential in maintaining compliance and minimization of errors in the operation of large systems. The outcomes clearly

| ISSN: 2320-0081 | www.ijctece.com ||A Peer-Reviewed, Refereed and Bimonthly Journal |

|| Volume 5, Issue 3, May– June 2022 ||

DOI: 10.15680/IJCTECE.2022.0503009

demonstrate that secure microservices architecture provides a balance between strong security, system performance, and operational complexity and there is no single solution to achieve the highest value.

V. CONCLUSION

The paper concludes that secure cloud native microservices require architecture-based security as opposed to the tool-based protection. Despite the positive changes provided by zero trust architecture, such as the increased coverage by 95% of encrypting, and decreasing the attack surface to 62 percent, it introduces system overhead in the form of latency and resource use. Service mesh systems improve internal traffic security and double latency and CPU usage, respectively. Identity and access complexity, which decreases from 8.2 to 4.3 with sophisticated models, is still a major obstacle. The research shows that security, performance and scalability must be balanced since a particular architecture has no power to maximise the three variables simultaneously.

REFERENCES

[1] A. Pereira-Vale, E. B. Fernandez, R. Monge, H. Astudillo, and G. Márquez, "Security in microservice-based systems: A Multivocal literature review," *Computers & Security*, vol. 103, p. 102200, Jan. 2021, doi: 10.1016/j.cose.2021.102200.
[2] A. Hannousse and S. Yahiouche, "Securing microservices and microservice architectures: A systematic mapping study," *Computer Science Review*, vol. 41, p. 100415, Jun. 2021, doi: 10.1016/j.cosrev.2021.100415.
[3] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, "Security in microservices Architectures," *Procedia Computer Science*, vol. 181, pp. 1225–1236, Jan. 2021, doi: 10.1016/j.procs.2021.01.320.
[4] L. Bradatsch, F. Kargl, and O. Miroshkin, "Zero trust service function chaining," *arXiv.org*, Jul. 19, 2021. https://arxiv.org/abs/2107.08671
[5] S. Rodigari, D. O'Shea, P. McCarthy, M. McCarry, and S. McSweeney, "Performance Analysis of Zero-Trust multi-cloud," *arXiv.org*, May 05, 2021. https://arxiv.org/abs/2105.02334
[6] A. Barabanov and D. Makrushin, "Authentication and authorization in microservice-based systems: survey of architecture patterns," *arXiv (Cornell University)*, Sep. 2020, doi: 10.48550/arxiv.2009.02114.
[7] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, and M. Prandini, "Microservice security: a systematic literature review," *PeerJ Computer Science*, vol. 7, p. e779, Jan. 2022, doi: 10.7717/peerj-cs.779.
[8] R. Xu, S. Y. Nikouei, Y. Chen, E. Blasch, and A. Aved, "BlendMAS: a BLockChain-ENabled decentralized microservices architecture for smart public safety," *arXiv (Cornell University)*, Feb. 2019, doi: 10.48550/arxiv.1902.10567.
[9] A. Banijamali, P. Jamshidi, P. Kuvaja, and M. Oivo, "KUKSA: a Cloud-Native architecture for enabling continuous delivery in the automotive domain," *arXiv.org*, Oct. 22, 2019. https://arxiv.org/abs/1910.10190
[10] Y. Gan and C. Delimitrou, "The architectural implications of microservices in the cloud," *arXiv (Cornell University)*, May 2018, doi: 10.48550/arxiv.1805.10351.