e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:03/March-2024 Impact Factor- 7.868 www.irjmets.com

COMPREHENSIVE FRAMEWORK DEVELOPMENT FOR MICROSERVICES SECURITY: DESIGN PRINCIPLES AND EXTENSIBILITY

Manish Kumar*1, Dr. Murugan R*2

*1 2nd Year MCA (ISMS), School Of Computer Science And IT, Jain (Deemed To Be) University, Bangalore, India.

*2 Programme Head-MCA, School Of Computer Science And IT, Jain (Deemed To Be) University, Bangalore, India.

https://www.doi.org/10.56726/IRJMETS49976

ABSTRACT

The increasing prevalence of cloud computing and the shift towards microservices-based applications necessitate scalable solutions, posing challenges in both software creation and, more significantly, scalable system development. Microservices, in contrast to monolithic applications, represent a collection of independently deployable services. The study highlights quality attributes like performance, scalability, security, and maintainability as pivotal in contrasting monolithic and microservice applications. The research underscores a growing emphasis on quality-driven migration to microservices within the academic community.

In the realm of web application development, the last decade has witnessed a shift towards Service-Oriented Architecture (SOA) and the rise of Software as a Service (SaaS) and Serverless providers embracing DevOps for microservices' creation, maintenance, and scalability. Despite this trend, security remains a critical yet often underestimated aspect. This thesis reviews state-of-the-art security recommendations for microservices, aiming to integrate security seamlessly into the Software Development Lifecycle (SDLC). The research emphasizes enhancing security awareness and simplifying security measure integration. A small case study illustrates how dynamic startups can lead in adopting high cybersecurity standards.

The core of the research delves into Microservices Architecture (MSA), emphasizing its evolution from Monolithic Architecture and its intrinsic connection with container-based deployment. The study elucidates that container, due to their independence from embedded operating systems, serve as a natural compute platform for MSA. Microservices, represented as small, independent processes communicating through lightweight mechanisms, exhibit a symbiotic relationship with containers. The focus is on dissecting larger applications into discrete services, with security being a central concern in the cost-effective era.

In the context of security challenges in Microservice Architectures (MSA), the paper conducts a systematic mapping to categorize threats and proposed security solutions. The study identifies a research gap, emphasizing a need for more comprehensive approaches to secure MSA. While external attacks receive significant attention, internal threats, mitigation techniques, and considerations across communication and deployment layers require further exploration. The research advocates for a more holistic understanding of security in MSA, encouraging future studies to address these gaps.

Keywords: Microservices, Docker, Kubernetes, Software As A Service, Microservice Architecture

I. INTRODUCTION

The landscape of modern software development has undergone a transformative shift with the emergence of cloud computing and the pervasive adoption of microservices-based architectures. As organizations transition from traditional monolithic applications to more agile and scalable systems, the complexities and challenges associated with developing secure and scalable solutions have become increasingly evident. Microservices architectures, characterized by a collection of independently deployable services, offer a promising paradigm for enhancing flexibility and scalability in software development.

Recognizing the need for a comprehensive security framework tailored to the nuances of microservices, the objective is to provide companies and industries with a versatile framework that facilitates a seamless migration to microservices while establishing a methodological approach for evaluating the adoption's effectiveness. Through a systematic literature review spanning the years 2020 to 2023, the study critically examines 48 selected research papers, shedding light on the contrasting quality attributes of monolithic and

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:03/March-2024 Impact Factor- 7.868 www.irjmets.com

microservice applications.

Furthermore, in the rapidly evolving landscape of web application development, the last decade has witnessed a growing inclination towards Service-Oriented Architecture (SOA) and the rise of Software as a Service (SaaS) and Serverless providers. While these trends underscore the importance of DevOps tools for the creation, maintenance, and scalability of microservices, the crucial element of security often takes a backseat. This research aims to bridge this gap by reviewing state-of-the-art security recommendations for microservice architectures and proposing enhancements that seamlessly integrate security into the Software Development Lifecycle (SDL).

Delving deeper into the core of microservices, this study explores Microservices Architecture (MSA) and its inherent connection with container-based deployment. Microservices, characterized as small, independent processes communicating through lightweight mechanisms, are intricately linked with containers, which serve as a natural computer platform for this architecture. The focus on dissecting larger applications into discrete services prompts a crucial examination of the security aspects in this cost-effective era.

Acknowledging the critical role of security in Microservice Architectures (MSA), the research delves into a systematic mapping to categorize threats and proposed security solutions. By synthesizing information from a wide array of studies, the paper identifies existing research gaps and emphasizes the need for a more holistic understanding of security in MSA. This introduction sets the stage for a comprehensive exploration of the challenges, solutions, and future directions in ensuring the security and scalability of microservices-based applications.

II. BACKGROUND OF MICROSERVICES

The background materials delve into the architectural shift from monolithic structures to the increasingly prevalent microservices paradigm in software development. Monolithic architecture, depicted as a unified and solid application, is scrutinized for its challenges in handling growing complexity, leading to limitations in scalability, maintenance, and deployment efficiency. The narrative emphasizes that even major corporations like Amazon and eBay initially adopted monolithic approaches but faced issues as their applications expanded.

Contrasting monolithic setups, microservices are introduced as autonomous, fine-grained units that communicate through well-defined interfaces. This architectural style is celebrated for its ability to enhance agility and scalability by compartmentalizing business activities into independent code bases. The inherent advantages of microservices include fault tolerance, allowing an application to function even if some microservices fail, and horizontal scaling, enabling the scaling of specific services under heavy loads without affecting the entire system.

The detailed comparison between monolithic and microservices architectures highlights key distinctions. While monoliths operate as a single process with shared code bases, microservices can be developed independently and scaled autonomously. The comparison extends to various aspects, including componentization, elasticity, storage mechanisms, and technology diversity.

The discussion then shifts towards the migration process, offering insights into decomposing monolithic applications into microservices. Techniques such as analyzing source code, logs, and execution traces are presented as effective methods during the migration phase. The advantages of microservices, particularly when hosted in containers, are elucidated. This includes the ability to roll out updates swiftly, from feature releases to security patches, without disrupting the entire system.

Decoupling capabilities emerge as a crucial strategy during migration, allowing for a gradual transition. The narrative underscores the significance of considering factors like scalability, frequent feature releases, fault tolerance, and technology diversity when deciding to adopt microservices. Moreover, it explores the application of microservices in modernizing outdated systems, breaking down legacy features into network-accessible, independent components.

The migration framework is outlined, covering stages such as identifying motivation reasons, metrics, decision-making, and finalizing migration. It is emphasized that selecting an appropriate migration solution, be it refactoring, rebuilding, or creating a new application, depends on the unique architectural requirements of each system. The overall theme underscores the industry's pursuit of scalable, agile, and fault-tolerant software architectures, with microservices emerging as a compelling solution to these contemporary challenges.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:03/March-2024 Impact Factor- 7.868 www.irjmets.com

III. SECURITY SMELLS AND REFACTORING

The study undertakes a systematic exploration of security concerns in microservices architectures, delineating a taxonomy encompassing various security smells and corresponding refactoring. Rooted in the ISO/IEC 25010 standard, the taxonomy discerns confidentiality, integrity, and authenticity as pivotal security properties. The investigation methodically identifies specific security smells within these properties, elucidating critical issues such as insufficient access control, publicly accessible microservices, unnecessary privileges, and more.

Crucially, the study advocates for proactive mitigation strategies, or refactoring, tailored to each security smell. For instance, OAuth 2.0 emerges as a potent solution to fortify access control in scenarios of insufficient access control. The introduction of an API Gateway stands out as an effective countermeasure against publicly accessible microservices, consolidating security enforcement in a centralized manner.

Addressing the "own crypto code" smell, the study underscores the risks associated with custom encryption solutions, recommending reliance on established encryption technologies. Encryption emerges as a recurrent theme, extending to the mitigation of non-encrypted data exposure through the practice of encrypting sensitive data at rest.

The study delves into the pervasive issue of hardcoded secrets, proposing the encryption of secrets at rest as a safeguard. To fortify service-to-service communications, the study advocates the adoption of Mutual TLS, providing bidirectional encryption and bolstering data transfer security.

Unauthenticated traffic garners attention, prompting the study to endorse Mutual TLS and OpenID Connect as remedies. The complex challenge of multiple user authentication is addressed through the implementation of Single Sign-On, streamlining user authentication processes and mitigating associated security risks.

The study discerns a centralized authorization smell, emphasizing the importance of transitioning to decentralized authorization mechanisms. The adoption of JSON Web Token (JWT) emerges as a preferred avenue, offering a secure means of passing claims and data between microservices while maintaining integrity through cryptographic signatures.

IV. CONTAINER-DOCKER

Microservices, defined as small, independently testable, and deployable applications that scale based on demand, adhere to the single responsibility principle. This principle emphasizes that each microservice should offer a simple, well-defined, and atomic service. In contrast to monolithic systems, microservices advocate the segmentation of applications into smaller, separately maintained fragments. These fragments can collaborate and can be easily shut down when no longer needed, enabling efficient maintenance and scalability.

The microservices approach, as highlighted by Johannes Thönes and discussed in "Building Microservices" by Newman, facilitates rapid changes and faster deployments. The integration of Continuous Integration and Deployment (CI/CD) amplifies microservices' potential, ensuring more effective software delivery. Unlike monolithic applications where small changes require releasing the entire application, microservices permit handling errors more effectively due to service isolation and straightforward rollback options.

Scaling monolithic applications is challenging as they are viewed as single entities requiring specific resource allocations. Microservices address this by allowing the use of less powerful hardware and instantiating resources on demand. This approach extends to organizational benefits, fostering smaller and more productive teams.

Defining the granularity of microservices is not trivial, but generally, each service should stay within 2000 lines of code. Jon Eaves suggests using the time required to write a service from scratch as a metric, with an estimated two weeks considered appropriate. Microservices enable a high level of technology heterogeneity, promoting the use of diverse tools and languages tailored to each service's needs.

The decoupled nature of microservices allows for maintaining a lightweight stack, adopting new technologies seamlessly, and facilitating communication through well-defined APIs. However, challenges arise, including increased complexity as the number of connected pieces grows, making security measures against overall application attacks more intricate.

Containers, exemplified by Docker, offer a practical representation of microservices concepts, providing OS-level virtualization. Docker images, representing containers, are constructed either through Dockerfiles or runtime

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:03/March-2024 Impact Factor- 7.868 www.irjmets.com

changes, organized into registries. Kubernetes, an open-source project initiated by Google, serves as a container orchestrator, enhancing container potential by efficiently coordinating multiple services.

The DIE paradigm, emphasizing the short lifespan of pods in Kubernetes, aligns with microservices' immutable software concept. This paradigm encourages developers to maintain lean, clean, and easily replaceable services. Distroless containers, pioneered by Google, aim to reduce attack surfaces by minimizing dependencies, resulting in smaller, more secure images. This approach, coupled with multi-stage builds, enhances security and optimization, making distroless containers an effective mitigation against potential attacks, especially in Kubernetes clusters.

V. CUTTING EDGE APPROACH FOR SECURE DEVELOPMENT

In the realm of secure development for microservices environments, container clustering orchestration, particularly using Kubernetes, has gained significant popularity. However, the 2020 Snyk report reveals a concerning trend wherein 30% of users admitted to neglecting Kubernetes (K8s) manifest reviews. This underscores the critical need for specific tools and measures to bolster security in container clusters.

Kubernetes inherently incorporates security measures, managing IP addresses and implementing controls for secure communication between nodes. Nevertheless, it provides only foundational security features, leaving room for advanced security monitoring and compliance enforcement. To address this gap, various tools can be employed. Aqua Security solutions, endorsed by the Centre of Internet Security (CIS), offer tools like Aqua kube-bench, aligning with the CIS Kubernetes Benchmark, and kube-hunter for enhanced analysis and penetration testing capabilities.

Additionally, periodic container scans are crucial, especially considering the potential vulnerabilities found in container images labelled as "latest." Open-source tools like Clair can be instrumental in performing static analysis based on known vulnerability signatures. Moreover, creating a hacking testing environment with projects like WebGoat or KubernetesGoat allows for practical vulnerability exploration, contributing to enhanced security awareness.

General proposals for security improvement extend beyond microservices environments. Periodic vulnerability scans, conducted through tools like Nessus or OpenVAS, help organizations continuously audit systems. Interactive Application Security Testing (IAST) combines the strengths of Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST), providing a hybrid solution with more accurate results. Incorporating code review exercises, such as those offered by SecureCodeWarrior, becomes integral to reducing security vulnerabilities during the development process.

Furthermore, automation plays a key role in penetration testing, where certain tasks can be automated to expedite the process. Integrating penetration tests into Continuous Integration/Continuous Deployment (CI/CD) pipelines or scheduling them periodically helps ensure ongoing security checks. Dashboards, facilitated by tools like Kibana or Grafana, offer a visually intuitive representation of security insights, aiding developers, and security specialists. Additionally, centralized log management, supported by services like AWS or tools like Logstash, GrayLog, or FluentD, consolidates logs from various entities, enabling more comprehensive analysis and threat detection.

VI. CURRENTLY ADOPTED SECURITY METHODS AND PROCESSES

In recent years, the growing awareness of security has yet to translate into a widespread commitment of resources by companies to ensure robust security in their systems and processes. Unfortunately, security is often viewed as a resource-draining task with intangible results. However, understanding that security is not a patch but an integral part of the design phase is crucial for an effective and cost-efficient ecosystem.

The 2020 Snyk report indicates a growing trend of security awareness in companies, but practical technical improvements are lacking. The reality often falls short of expectations, with 26% of respondents in the 2020 Skyn report admitting a complete lack of security practices.

Examining the Software Development Life Cycle (SDLC) structure, comparisons reveal discrepancies between theoretical models and actual implementations. In the requirements phase, ISO 27001 adoption statistics in German firms show slower growth than expected. Factors include reliance on certifications of commercial partners and the avoidance of certification costs by some firms. The study suggests a potential shift towards

mandatory adoption in the future.

In the design phase, improvements are seen in the distribution of security responsibilities among developers, security teams, and operations, but there's a lack of programs to foster this culture. The selection of tools and packages in development sees positive trends, with developers evaluating community support, commit frequency, ratings, downloads, and known vulnerabilities.

In the implementation phase, the usage of linters (tools for identifying issues in code) is surprisingly low, indicating a gap in adopting essential security tools. The adoption rates of HTTP security headers for enforcing security measures in web applications are also suboptimal, exposing vulnerabilities.

During the test phase, manual vulnerability reviews and the use of audit tools are prevalent, but there's room for improvement. Continuous Integration (CI) practices are widely adopted, but the percentage decreases as the process moves toward production. Static Application Security Testing (SAST) is the most commonly integrated security control in pipelines, while Dynamic Application Security Testing (DAST) and dependency scanner tools see lower adoption rates.

In the deploy phase, the ability to detect and fix vulnerabilities is crucial. The 2020 Snyk report reveals varying response times, with only 1% of flaws fixed within a day, and the majority of vulnerabilities taking more than 20 days to be addressed. The urgency to patch vulnerabilities seems to be influenced by their severity.

In summary, there is a clear discrepancy between the idealized security models discussed in earlier chapters and the current state of security practices in companies. The adoption and implementation of security measures are crucial across all phases of the SDLC to build a robust and resilient security posture.

VII. CONCLUSION

In conclusion, the landscape of modern software development is undergoing a significant transformation, marked by the increasing prevalence of cloud computing and the widespread adoption of microservices-based architectures. This research has sought to address the challenges and opportunities presented by this shift, particularly in the context of security, scalability, and quality attributes. Through a Systematic Literature Review (SLR), key insights were gleaned from 48 research papers spanning 2020 to 2023, revealing a growing emphasis on quality-driven migration to microservices within the academic community.

Quality attributes such as performance, scalability, security, and maintainability were identified as pivotal considerations in contrasting monolithic and microservice applications. The study underscores the need for a holistic approach to security, emphasizing that it should be an integral part of the Software Development Lifecycle (SDL). A case study further illustrates how dynamic startups can lead in adopting high cybersecurity standards.

The core exploration delves into Microservices Architecture (MSA), emphasizing its evolution from Monolithic Architecture and its symbiotic relationship with container-based deployment. The research addresses security challenges in MSA through a systematic mapping of threats and proposed security solutions, identifying a need for more comprehensive approaches, especially regarding internal threats, communication layers, and deployment considerations.

The study sheds light on the cutting-edge approach for secure development, recognizing the rising popularity of container clustering orchestration, particularly using Kubernetes. However, it highlights a concerning trend of neglecting Kubernetes manifest reviews and underscores the importance of specific tools and measures to bolster security in container clusters.

Examining currently adopted security methods and processes reveals a gap between awareness and practical implementation. While there is a growing trend in security awareness, technical improvements are lacking, and security is often viewed as a resource-draining task. Discrepancies are observed in each phase of the Software Development Life Cycle (SDLC), from requirements to deployment, highlighting the need for a more robust and holistic security posture.

In essence, this research advocates for a paradigm shift in how security is approached in the age of microservices. It calls for a proactive integration of security measures throughout the SDLC, leveraging cutting-edge technologies, and fostering a culture of security awareness. As the industry continues to evolve, embracing these principles will be crucial in building resilient, scalable, and secure software ecosystems.

VIII. REFERENCES

[1] 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.

[2] S. Hussein, M. Lahami, and M. Torjmen, "Assessing the quality of microservice and monolithic architectures: systematic literature review," Nov. 2023, doi: 10.21203/rs.3.rs-3497708/v1.

[3] A. Hannousse and S. Yahiouche, "SECURING MICROSERVICES AND MICROSERVICE ARCHITECTURES: A SYSTEMATIC MAPPING STUDY A PREPRINT," 2020. Available: https://arxiv.org/pdf/2003.07262.pdf.

[4] G. Orazi and P. Sainio, "Enhancing and integration of security testing in the development of a microservices environment," 2020. [Online]. Available:
https://www.utupub.fi/bitstream/handle/10024/150701/orazi_master_thesis.pdf?sequence=1.

[5] F. Ponce, J. Soldani, H. Astudillo, and A. Brogi, "Smells and refactorings for microservices security: A multivocal literature review," Journal of Systems and Software, p. 111393, Jun. 2022, doi: 10. 1016/j. jss.2022. 111393.

[6] S. Behrens and B. Payne, "Starting the Avalanche: Application DDoS In Microservice Architectures," The Netflix Tech Blog, https://netflixtechblog.com/starting-the-avalanche-640e69b14a06, 2017.

[7] E. Boersma, "Top 10 security traps to avoid when migrating from a monolith to microservices," Sqreen, https://blog.sqreen.com/top-10-security-traps-to-avoid-when-migrating-from-a-monolith-to-microservices/, 2019.

[8] J. Bogner et al., "Towards a Collaborative Repository for the Documentation of Service-Based Antipatterns and Bad Smells," in 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), IEEE Computer Society, United States, 2019, pp. 95–101, doi: 10.1109/ICSA-C.2019.00025.

[9] CollabNet and VersionOne, "13th annual state of agile report," 2018. Available:
https://stateofagile.com/#ufh-i-613553418-13th-annual-state-of-agile-report/7027494.

[10] CollabNet and VersionOne, "1st annual state of agile report," 2007. Available: https://stateofagile.com/