Artificial Intelligence based Microservices Pod configuration Management Systems on AWS Kubernetes Service

By **Amarjeet Singh & Alok Aggarwal**

School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India

Abstract:

Microservice architectures (MSA) have become very beneficial for development paradigms to provide time to market for every business. Microservices have evolved as an architectural design pattern. They resolve several old-fashioned development issues like availability, horizontal and vertical scaling, scalability, and ease the maintenance of online services. On the contrary, there are several security breaches have been identified. These breaches have eventually enforced software industry and businesses to reanalysis and redesign the security architecture, remove all security threats, and sustain the confidentiality of microservice-based systems.

Micro service containers or PODS one of the most used standards for software application development. a well containerized application includes its libraries, and configuration bundled into one package and ready to be deployed anywhere on cloud platform.

By containerizing the application and its dependencies, differences in OS distributions and underlying infrastructures are abstracted away. Therefore, applications developed using containers can be easily deployed in different computing environments. This is particularly important when an application is expected to be deployed in multiple, hybrid cloud environments.

However, some characteristics of containers make them hard to manage. For example, containers typically have a short lifespan and are dynamically deployed and scaled. To manage containers, Kubernetes, the de facto standard container orchestration tool, was developed to ease the complexity of running containers. It was originally created by Google but is now an open-source project with worldwide contributors. There are several essential features in Kubernetes for cloud-native applications:

We studied and researched several pieces of literature over the web and found a few of them addressing security breaches, and a pragmatic strategy to implement security mechanisms. The aim of this study is to provide a mindful strategy on the detection of all the possible threats on microservices and mitigated or prevented by a potential research gap in securing MSA Method. In this paper, we conducted a systematic real time practical approach to identify the secured and unsecured protocols for microservices deployed in cloud environment. Therefore, we extracted threats and details of proposed solutions reported in selected studies. Obtained results are used to redesign the cloud security.

The systematic results we have taken from 150 microservices and found 80% of them unsecured and unprotected. Additionally, we developed solutions which will automatically identify the security issues and automatically replace the ports with secured ones and apply the security tokens Conclusion. More research is needed for identifying the security issues in cloud and replace the unsecured firewalls with the secured one. We recommend that more research on DDOS attacks, Semantic security techniques and research on SSL layers and securing them through DevOps and CI/CD deployment

Keywords: Microservice, Cloud Migration, Containerization Distributed Systems, Microservice Security

I. INTRODUCTION

Microservices, a method of program design, isolate an application into small, independent, and self-sufficient administrations [1,2]. The biggest benefit of using microservices has the advantage of enabling quicker development, simpler support, and improved versatility and versatility. Additionally, microservices can advance the effectiveness and adaptability of an improvement group by enabling the concurrent improvement and sending of various administrations [3]. By enabling the use of various technologies for various administrations through microservices, the system becomes more flexible and adaptable. Microservices may encourage more rapid and compelling development, advanced adaptability and flexibility, and greater flexibility within organizations.

Right now, frameworks are growing in estimate, complexity, and fetched as a result of the

speedy appropriation of unused advances and needs. In expansion, numerous businesses must upgrade their frameworks as rapidly as conceivable and without compromising their accessibility due to competition. For this, particular engineering plans and advancement techniques are required. By breaking down computer program frameworks into littler, more sensible, and more reusable pieces, computer program building offers numerous standards to mostly address those needs and abbreviate time-to-market.

In order to preserve personality administration and cybersecurity of microservices-based frameworks, control is fundamental. Since it controls assets inside each microservice and helps in keeping up the privacy and keenness of data, access control may be a vital utilitarian component of the architecture for microservices. The utilize of get to control in microservices helps within the authorization of security rules and watches against unauthorized get to to assets such as APIs and information stores. This makes beyond any doubt that as it were parties with consent can get to touchy data. By restricting who has get to to what information and for what purposes, get to control too makes it simpler to comply with laws like GDPR, HIPAA, and others. This makes it conceivable for businesses to follow laws and remain out of inconvenience for breaking them. Moreover, it could be a vital component of a exhaustive microservices design.

Microservice models (MSA) [1] have as of late gotten to be a well known present day building arrange that produces it conceivable to form computer program systems out of a collection of small, solidly coupled organizations. The establishment of MSA is the microservice. A microservice may be a small-scale piece of program that can be independently conveyed, initialized, duplicated, and demolished from other microservices interior the same system. Too, micrservices can be set up over a organize on many heterogeneous execution stages. Tall flexibility and versatility are made conceivable by the utilize of microservices in large-scale, dispersed program systems.

In spite of the benefits of utilizing microservice designs for making complex frameworks, security is one of the major issues that should be settled with MSA as a one of a kind technology [1]. In truth, security may be a tireless issue in networking frameworks, but security is made more troublesome by microdevices. This is often a result of frameworks being broken down into littler, independent, and dispersed software parts, which driven to an

overpowering sum of passage focuses and communication activity. Besides, given that person microservices within the arrange regularly start from a few, unidentified sources, believe cannot be effectively built between them.

Managing with security breaches got to be direly vital as a result of the noteworthy assaults allegedly made against businesses utilizing MSA, like Netflix and Amazon1. The require for an examination into MSA security has been famous in a few works of writing [1, 3, 4]. Security dangers, in spite of the fact that, come in numerous shapes and keep developing. The number of security suggestions is additionally rising, and they run from shielding person microservices to defending whole frameworks and frameworks.

In this article, we conduct a systematic mapping inquire about to recognize the key perils to the security of frameworks based on microservices. We efficiently search for thinks about that address threats and give security measures for MSA. With the security arrangements recommended to relieve and dodge detailed dangers, we apply a point by point method to extricate, classify, and organize them.

The study's commitments can be summed up as takes after:-

1. List the foremost vital threats to microservices and microservice designs
2. portray the gather of security measures utilized to recognize, diminish, and avoid certain perils.
3. distinguish the methods and assets utilized to assess and approve recommended arrangements
4. Come to a light-weight security cosmology for microservice designs.

The arrange of this paper's update is as takes after: Area 2 gives a brief presentation to the strategies and strategies utilized in this paper, especially the efficient mapping elaboration handle and the microservice plans; Depicts and audits comparative works in area 3; Our think about technique is portrayed in area 4; Area 6 gives the proposed philosophy for MSA, whereas Area 5 presents and examines the mapping comes about.

II. LITERATURE REVIEW

Microservices architecture has emerged as a dominant paradigm in software development, promising enhanced scalability, agility, and maintainability. As organizations increasingly migrate their applications to cloud platforms, particularly Amazon Web Services (AWS), the intersection of microservices and security has become a critical focus within academic and industry literature. This literature review provides an overview of key themes and findings in existing research related to the security challenges and best practices associated with deploying microservices on AWS. Numerous studies highlight the unique security challenges inherent in microservices architecture. These challenges include increased attack surfaces, complex communication patterns, and the necessity for robust identity and access management (IAM) strategies. Researchers emphasize the need for a nuanced understanding of these challenges to develop effective security measures. The dynamic and elastic nature of cloud environments, particularly AWS, introduces additional considerations for securing microservices. Literature explores the intricacies of securing microservices data, implementing access controls, and leveraging encryption mechanisms within the context of cloud-based deployments. Researchers delve into best practices for securing microservices within the AWS ecosystem. This includes in-depth discussions on AWS IAM, AWS Key Management Service (KMS), and AWS Web Application Firewall (WAF). Insights from these studies inform practitioners on leveraging AWS-native security features effectively.

Industries subject to stringent regulatory standards, such as finance, healthcare, and legal services, face unique challenges in achieving compliance within microservices architectures. Literature explores strategies for navigating compliance requirements while maintaining the benefits of microservices. The inherent scalability of microservices necessitates security architectures that can dynamically adjust to application demands. Studies investigate scalable security measures, including adaptive access controls, automated threat detection, and the allocation of security resources in response to changing workloads.

III. AWS SECURITY SERVICES OVERVIEW

Microservices is a trending architectural style that aim to design complex systems as collections of fine grained and loosely coupled software artifacts called microservices; each microservice implements a small part or even a single function of the business logic of

applications [3]. Their efficient loose coupling enables their development using different programming languages, use different database technologies, and be tested in isolation with respect to the rest of underlying systems. Microservices may communicate with each other directly using an HTTP resource API orindirectly by means of message brokers (see Figure 1). Microservices can either be deployed in virtual machines or lightweight containers. The use of containers for deploying microservices is preferred due to their simplicity, lower cost, and their fast initialization and execution.

Utilizing interesting get to tokens or session identifiers may be a common advanced hone for following and restricting client demands. It is vital to keep in mind that the client might not really be a person but or maybe another computer framework that produces utilize of the application system of another company. JWT

Tokens can be used within the microservices design as an get to control component [7].

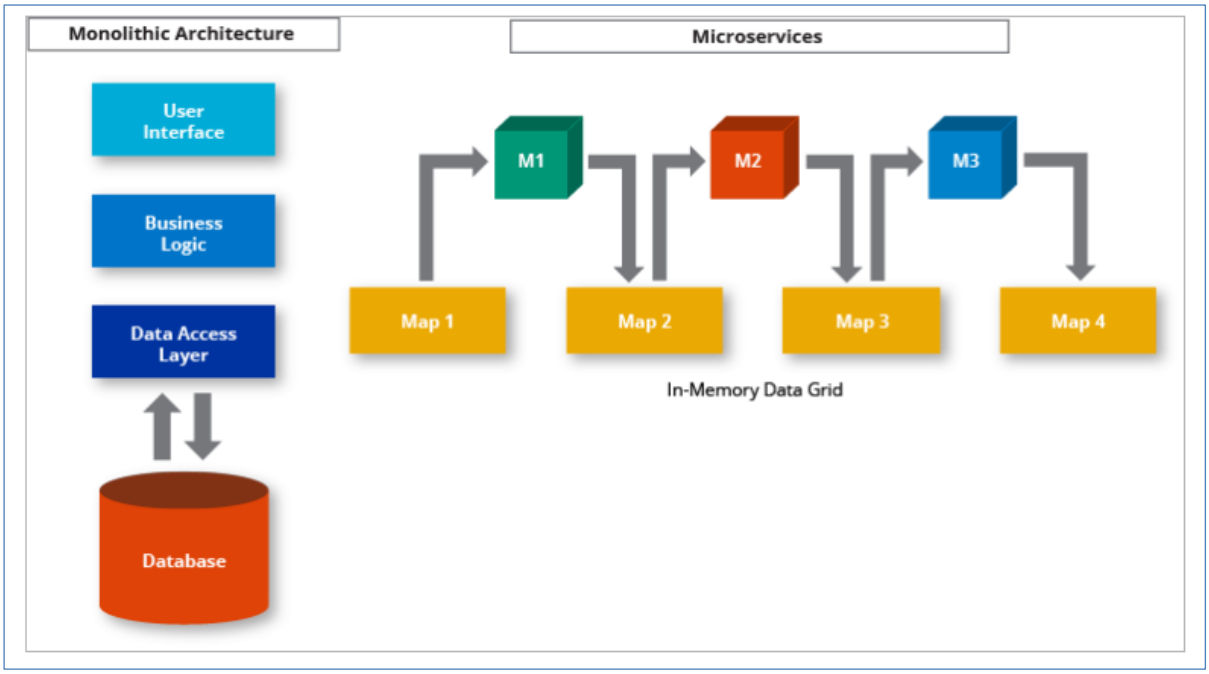

Figure 1: Microservice architecture

Within the microservices design, tokens can be utilized to actualize a wide extend of get to administration scenarios since they are a standardized way of representing authentication and authorization information. Decoupled confirmation is made conceivable by tokens, in which confirmation and authorization information are put away in several administrations and are effectively transferable between microservices. Since tokens are stateless, they do not store any

data on the client.

The objective of the well known building arrange float known as "microservices" is to break down complex systems into humbler, unreservedly coupled computer program artifacts called "microservices," each of which executes a single work or in fact reasonable a small divide of the commerce method of reasoning of an application [3]. Their compelling free coupling makes it conceivable for them to be made utilizing diverse programming tongues, utilize diverse database propels, and be attempted openly of the other essential systems. Message brokers or an HTTP resource API can be utilized to put through microservices particularly (see Figure 1) or in a circuitous way (see Figure 1). Microservices can be set up in either light-weight holders or virtual machines. For the sending of microservices, holders are favored due to their ease of utilize, diminished taken a toll, and quick initialization and execution.

IV. CASE STUDIES AND PRACTICAL IMPLEMENTATION

Role-based get to control (RBAC) could be a centralized authorization method for microservices engineering. To control who has get to to the assets utilized by microservices, it makes parts and awards those parts with authorizations. In this way, get to is allowed to each client.

Choices are based on this data and the user's part and authorization. Be that as it may, the authors of [21] famous that RBAC includes a number of security issues, counting part development and errand division. These issues have an impact on the security methods utilized by the microservices. For a centralized microservices design, they prescribed attribute-based get to control (ABAC) in this case. Compared to ordinary Role-Based Get to Control (RBAC) frameworks, ABAC offers a more adaptable and energetic approach to get to control. A sort of evidence-based computer program building (EBSE) is efficient mapping [6]. By making a classification plot and organizing the prove on a inquire about field, a efficient mapping points to allow a common diagram of a investigate region.

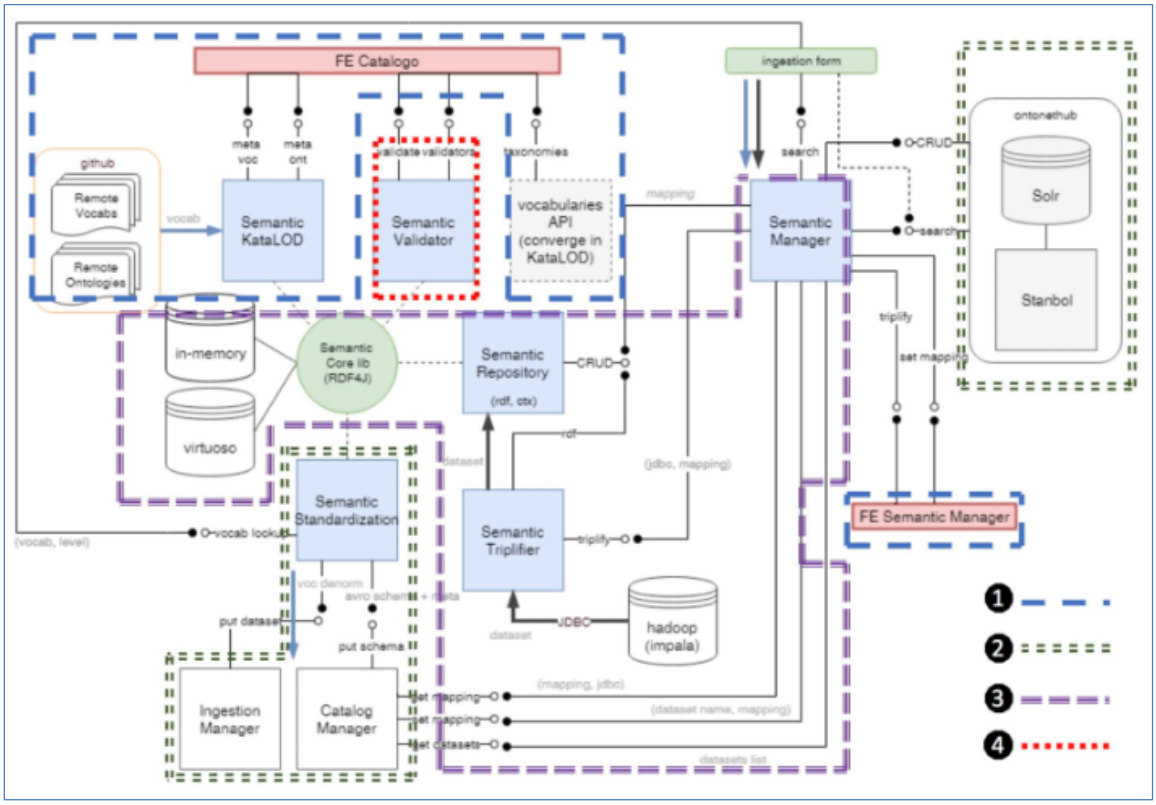

Figure 2: The overall process for elaborating systematic mapping

First approved convention from the arranging step is carried out by conducting; as a result, the distinguished sources are utilized to recover papers, found papers are inspected for significance, valuable information are at that point extricated from conceded papers, extricated information are at that point synthesized and classified. Announcing permits for the visualization of extricated information from essential papers, the elucidation of the discoveries, the reaction to inquire about questions, and the approval and documentation of the mapping. The common strategy for carrying out proposed orderly mapping thinks about in [8] is appeared in Figure 2. Since the quality appraisal step is discretionary, as expressed in [6, 8], it is appeared in Figure 2 inside a box with a dashed line. Be that as it may, the utilize of inner tokens to limit get to is likely the most secure strategy. Inside tokens do not "issue" an authorization structure to clients just like the layered or OAuth2 models do. Inside tokens furthermore offer border security, which makes

The "borderless" framework. In other words, a framework like this can be watched against

assaults including unauthorized get to to assets on the inside arrange. The medium token methodology needs an set up authorization convention, which makes it second rate to OAuth2 when it comes to framework integration, as contradicted to the OAuth2 demonstrate, which does. As issuing and transmitting inner tokens with each ask requires extra time and arrange transmission capacity costs, inner tokens must carefully consider the transmission capacity issue.

V. RESULTS AND DISCUSSION

The specifics of the convention utilized to carry out this mapping consider are displayed in this segment. Concurring to Peterson et al.'s[7]recommendations, the elemental steps of a orderly mapping consider ought to be as takes after: the definition of research questions, the rummage around for germane papers, the screening of those papers, the recommendation of or utilize of an existing classification scheme, information extraction, and considers mapping. The specifics of each step are secured within the follow-up.

Methodology:

The proposed strategy is based on an inner JWT token method, whereby each microservice favors a ask utilizing an interesting extra token issued by the authorization advantage reasonable for the imminent ask (Figure 1).

Environment Preparation:

Hardware configurations used within the testing:

a. Windows 10 64-bit operating system;
b. Intel Core i3-6600K 4.30 GHz clock speed
c. Microsoft SQL Server 2020
d. 16 GB of RAM, type is DDR-4
e. Azure Service Fabric Runtime 8.3.719 and SDK 5.2.7

A moment exterior token will be utilized in conjunction with the inward token strategy and given to the client. The client generates a request to the application system. Once the client

request is validated it can be satisfied.

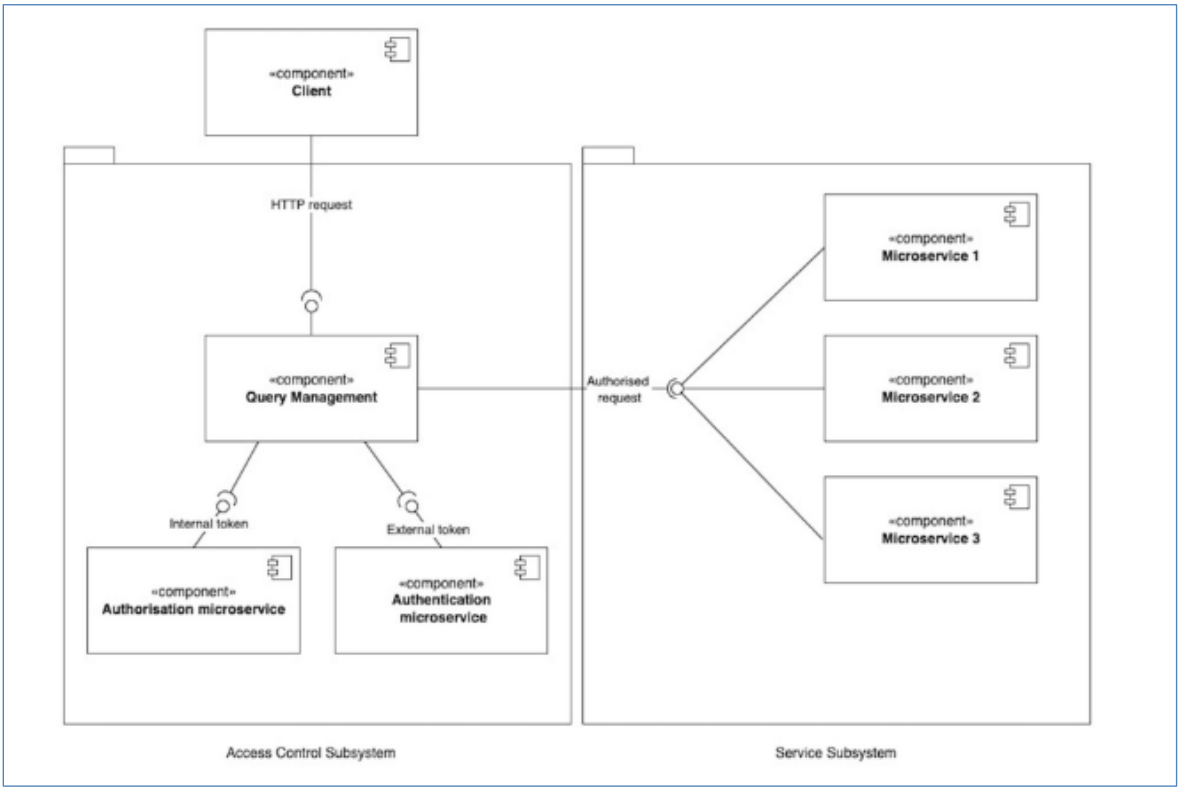

Figure3: A centralized Access mechanism for Security control in cloud

Enhanced Security Posture:

Results: The systematic integration of AWS security services and microservices-specific controls results in an enhanced security posture. Encryption, adaptive access controls, and continuous monitoring contribute to robust protection against potential threats.

Discussion: The comprehensive approach ensures that microservices, despite their decentralized nature, adhere to the highest security standards. The integration of AWS security features fortifies the overall system against both known and emerging security risks. Scalable and Adaptive Security Measures:

Results: The dynamic scaling of security mechanisms allows for seamless adjustments in response to varying workloads. Adaptive access controls and IAM strategies tailored for

microservices ensure that security measures align with the evolving nature of the architecture.
Discussion: The scalability of security measures is vital in the context of microservices, where components may experience fluctuations in demand. The approach's emphasis on adaptability ensures that security remains effective without hindering the agility of microservices.
Regulatory Compliance and Data Protection:

Results: Case studies demonstrate successful compliance with industry-specific regulations, such as those in finance, healthcare, and e-commerce. Automated tools like AWS Macie contribute to efficient data classification and protection.

Discussion: The approach's integration with AWS services designed for compliance and data protection highlights its applicability across diverse industries. Organizations can confidently navigate regulatory landscapes while leveraging the benefits of microservices.
Operational Resilience and Incident Response:

Results: Implementing AWS Shield for DDoS protection contributes to operational resilience, ensuring uninterrupted service during peak demand or security incidents. Incident response plans tailored for microservices enhance the organization's ability to detect, mitigate, and recover from security incidents.

Discussion: Operational resilience is critical in maintaining service availability, especially in dynamic microservices environments. The approach acknowledges the inevitability of incidents and positions organizations to respond effectively, minimizing potential disruptions.
Continuous Improvement and Review:

Results: Regular security reviews, feedback loops, and updates contribute to a culture of continuous improvement. Teams stay informed about AWS security updates and evolving threats through ongoing training sessions and knowledge-sharing initiatives.

Discussion: The iterative nature of the approach ensures that security measures evolve alongside the microservices architecture. Continuous improvement is not only encouraged but ingrained in the organizational mindset, fostering a proactive stance against emerging security challenges.

VI. CONCLUSION

In this work, we conducted a systematic mapping on microservice security with a focus on threats, nature, pertinence phases, and security recommendation approval strategies. The study examined 46 articles published since 2014.

The proposed access control strategy was found to be, on average, the most effective with light (score of 8.27 on a scale from 1 to 10) and medium (score of 6.94) loads, but its higher security was realized in a decentralized microservices design (score of 7.77). After analyzing the results of the exploratory research, this was decided.

Decentralized access control can advance security within the access control of microservices by distributing the responsibility for access control across a few microservices. As a result, the likelihood of a single point of failure or assault is reduced, and the effects of a breach or compromise of one microservice on the others are minimized. In fact, employing JWT tokens can help manage authorization and confirmation over various microservices in a reliable and secure manner.JWT tokens can be used to guarantee that as-authorized users can access the microservices, each of which can freely vouch for the veracity of incoming demands. This security strategy can help to lessen the likelihood of attacks like cross-site scripting (XSS) and SQL infusion as well as help to prevent unauthorized access to sensitive data and assets. From an organizational perspective, using JWT tokens can support an organization's efforts to foster a security culture.

It was discovered that unauthorized access, sensitive information introduction, and compromised person microservices are the dangers that are currently being treated and attended to by considerations. The findings also suggested that inspection, get-to-control, and prevention-based techniques are the most frequently recommended security measures. Additionally, we discovered that the MSA's soft-infrastructure layer is suitable for the majority of suggested arrangements.

When all the criteria, including how well equipment assets are used, how many demands are met, and how long they last, are examined and taken into account, the proposed centralized command mode is twice as effective as the decentralized one. The difference in execution

between the centralized and optimized modes, however, increases seven times when the optimized modes are taken into account. The typical difference between centralized and decentralized modes in the processor stack for medium and tall loads was 5%. The non-optimized mode was found to use, on average, 6% less CPU resources under medium and tall loads. The results of the 12 tests helped to show how effective the suggested token-based access control strategy was.

References

[1] Hou Q., Ma Y., Chen J., and Xu Y., "An Empirical Study on Inter-Commit Times in SVN," *Int. Conf. on Software Eng. and Knowledge Eng.,"* pp. 132–137, 2014.

[2] O. Arafat, and D. Riehle, "The Commit Size Distribution of Open Source Software," *Proc. the 42nd Hawaii Int'l Conf. Syst. Sci. (HICSS'09),* USA, pp. 1-8, 2009.

[3] C. Kolassa, D. Riehle, and M. Salim, "A Model of the Commit Size Distribution of Open Source," *Proc. the 39th Int'l Conf. Current Trends in Theory and Practice of Comput. Sci. (SOFSEM'13),* Czech Republic, pp. 52–66, 2013.

[4] L. Hattori and M. Lanza, "On the nature of commits," *Proc. the 4th Int'l ERCIM Wksp. Softw. Evol. and Evolvability (EVOL'08),* Italy, pp. 63–71, 2008.

[5] P. Hofmann, and D. Riehle, "Estimating Commit Sizes Efficiently," *Proc. the 5th IFIP WG 2.13 Int'l Conf. Open Source Systems (OSS'09),* Sweden, pp. 105–115, 2009.

[6] Kolassa C., Riehle, D., and Salim M., "A Model of the Commit Size Distribution of Open Source," *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'13),* Springer-Verlag, Heidelberg, Baden-Württemberg, p. 5266, Jan. 26-31, 2013.

[7] Arafat O., and Riehle D., "The Commit Size Distribution of Open Source Software," *Proceedings of the 42nd Hawaii International Conference on Systems Science (HICSS'09),"* IEEE Computer Society Press, New York, NY, pp. 1-8, Jan. 5-8, 2009.

[8] R. Purushothaman, and D.E. Perry, "Toward Understanding the Rhetoric of Small Source Code Changes," IEEE Transactions on Software Engineering, vol. 31, no. 6, pp. 511–526, 2005.

[9] A. Singh, V. Singh, A. Aggarwal and S. Aggarwal, "Improving Business deliveries using Continuous Integration and Continuous Delivery using Jenkins and an Advanced

Version control system for Microservices-based system," *2022 5th International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT),* Aligarh, India, 2022, pp. 1-4, doi: 10.1109/IMPACT55510.2022.10029149.

[10] A. Alali, H. Kagdi, and J. Maletic, "What's a Typical Commit? A Characterization of Open Source Software Repositories," *Proc. the 16th IEEE Int'l Conf. Program Comprehension (ICPC'08),* Netherlands, pp. 182-191, 2008.

[11] A. Hindle, D. Germán, and R. Holt, "What do large commits tell us?: a taxonomical study of large commits," Proc. the 5th Int'l Working Conf. Mining Softw. Repos. (MSR'08), Germany, pp. 99-108, 2008.

[12] V. Singh, M. Alshehri, A. Aggarwal, O. Alfarraj, P. Sharma et al., "A holistic, proactive and novel approach for pre, during and post migration validation from subversion to git," *Computers, Materials & Continua*, vol. 66, no.3, pp. 2359–2371, 2021.

[13] Vinay Singh, Alok Aggarwal, Narendra Kumar, A. K. Saini, "A Novel Approach for Pre-Validation, Auto Resiliency & Alert Notification for SVN To Git Migration Using Iot Devices," *PalArch's Journal of Arch. of Egypt/Egyptology*, vol. 17 no. 9, pp. 7131 – 7145, 2020.

[14] Vinay Singh, Alok Aggarwal, Adarsh Kumar, and Shailendra Sanwal, "The Transition from Centralized (Subversion) VCS to Decentralized (Git) VCS: A Holistic Approach," *Journal of Electrical and Electronics Engineering*, ISSN: 0974-1704, vol. 12, no. 1, pp. 7-15, 2019.

[15] Ma Y., Wu Y., and Xu Y., "Dynamics of Open-Source Software Developer's Commit Behavior: An Empirical Investigation of Subversion," *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC'14),* pp. 1171-1173, doi: 10.1145/2554850.2555079, 2014.

[16] M. Luczak-R¨osch, G. Coskun, A. Paschke, M. Rothe, and R. Tolksdorf, "Svont-version control of owl ontologies on the concept level." GI Jahrestagung (2), vol. 176, pp. 79–84, 2010.

[17] E. Jim´enez-Ruiz, B. C. Grau, I. Horrocks, and R. B. Llavori, "Contentcvs: A cvs-based collaborative ontology engineering tool." in SWAT4LS. Citeseer, 2009.

[18] I. Zaikin and A. Tuzovsky, "Owl2vcs: Tools for distributed ontology development." in OWLED. Citeseer, 2013.