COMPUTERIZED AUTOMATIC SYSTEMS

COMPREHENSIVE APPROACH TO PROTECTING DATA AND THE INFORMATION SYSTEM INTEGRITY

Ulyana Dzelendzyak, PhD, As.-Prof., Nazar Mashtaler, PhD Student
Lviv Polytechnic National University, Ukraine; e-mail: uliana.y.dzelendziak@lpnu.ua

https://doi.org/10.23939/istcmtm2024.03.

Abstract. The article discusses key information security principles, focusing on confidentiality, integrity, availability, traceability, and the DIE model (Distributed, Immutable, Ephemeral). Confidentiality emphasizes the importance of secrecy and controlling access to prevent sensitive information from misappropriation. Integrity ensures that data remains accurate and trustworthy, with measures to prevent unauthorized modifications. Availability highlights the necessity of reliable and timely access to data, even in the face of potential system failures or disasters, by implementing safeguards like backups. Traceability, or audit trails, ensures accountability by logging user actions, which is crucials for investigating suspicious activities or data loss.

The DIE model presents a modern approach to information security. Distributed systems minimize the impact of attacks by avoiding a single point of failure and incorporating redundancies. Immutable systems maintain unalterable logs to quickly identify and address anomalies, preventing malicious actors from covering up their actions. Ephemeral systems differentiate between essential, long-term "pets" and disposable "cattle," advocating for a flexible infrastructure that can easily adapt to new challenges and retire vulnerable legacy systems. This model enhances security by reducing the attack surface and ensuring that only necessary, secure systems are maintained.

Key words: Information security, confidentiality, data integrity, system availability, traceability, access control, data protection, distributed systems, immutability, and ephemeral infrastructure.

1. Introduction

In today's digital landscape, ensuring information and systems security is paramount. The article delves into fundamental information security principles, emphasizing the need for confidentiality, integrity, availability, and traceability. These principles are crucial for protecting sensitive data from unauthorized access, ensuring its accuracy and reliability, and maintaining consistent access to information. Additionally, the article introduces the DIE model—Distributed, Immutable, and Ephemeral—as a modern framework for enhancing security. By leveraging distributed systems, organizations can safeguard their assets and ensure robust security in an increasingly complex environment. To connect with protected resources and other services and systems, NET applications typically need connection strings, passwords, or other credentials that contain sensitive information. These sensitive pieces of information are called secrets. It's a best practice to exclude secrets in source code and not to store secrets in source control. Instead, it would be better to store the secrets in more secure locations [1]. A good approach would be to separate the secrets for accessing development and staging resources from the ones used for accessing production resources [1, 5, 11].

2. Drawbacks

While core information security principles and the DIE model offer substantial benefits, they are exploited with certain drawbacks. Implementing strong confidentiality measures can be complex and may hinder operational flexibility. Ensuring data integrity requires rigorous monitoring and control, which can be resource-intensive and may slow down system performance. Maintaining availability involves significant investment in redundant systems and backup solutions, potentially increasing costs. Traceability and auditing, while essential for accountability, can raise privacy concerns and lead to increased data management overhead. The DIE model's emphasis on distributed, immutable, and ephemeral systems might introduce challenges such as integration complexity, increased management of multiple components, and potential difficulties in adapting legacy systems to modern security practices.

3. Goal

To examine principles of information security that are confidentiality, integrity, availability, and traceability and to evaluate the effectiveness of the DIE model (Distributed, Immutable, Ephemeral) in enhancing security and their application in safeguarding data and systems, offering insights into their practical implementation and the associated benefits and challenges.

4. Integrating Foundational Principles and Modern Frameworks for Robust Information Security

In information security, four foundational principles are paramount: confidentiality, integrity, availability, and traceability are shown in Fig 1. Confiden-

tiality ensures that sensitive information remains protected from unauthorized access through effective access control measures. Integrity focuses on maintaining the accuracy and trustworthiness of data by preventing unauthorized modifications and ensuring proper monitoring and preservation. Availability emphasizes the importance of reliable and timely access to data, with safeguards in place to protect against data loss due to system failures, natural disasters, or cyberattacks. Finally, traceability, or audit trails, is crucial for accountability, providing detailed logs of user actions to monitor and prevent suspicious activities and data loss. These principles collectively form the backbone of a robust information security strategy [2].

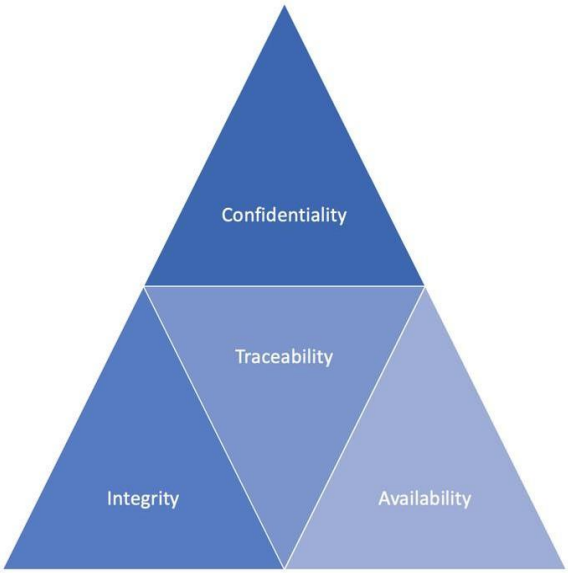

Fig. 1 Schema of information security

Confidentiality is the necessary level of designed secrecy when current information or documents are created, updated, or transmitted. Confidentiality depends on measures to prevent sensitive information from falling into the wrong hands. Access control is the key to confidentiality whether through sophisticated computer programs or old-fashioned locks and keys [11].

Integrity focuses on maintaining trustworthiness and ensuring the accuracy of the information. Data should not undergo any unauthorized changes throughout its lifecycle. Modifications require monitoring to ensure proper treatment and preservation of information. For example, how do you prevent unauthorized users from altering data? It is common to assign permissions only to a handful of people, enable version control, and store data backups to prevent accidental deletion of our information.

Availability concerns the systems and is significant for the reliability and timely access to data. Organizations have to put reasonable safeguards against data loss in unpredictable events such as system failure, natural disasters, or, as we have seen in recent cases, ransomware attacks. Having a backup copy of data ensures your business can continue to operate in such occurrences [2, 3, 12].

Traceability, known as audit trail, is a prerequisite for accountability. Traceability is ensured by providing a detailed log of the actions performed by a user who can be held responsible in occurrences such as:

- Suspicious activities from employees after business hours or on their last working day: Ex-employees tend to leave with company data on their previous work days. A solution that monitors and audits such suspicious activities could help prevent possible data leakage before it is too late.
- Loss of data: Employee activities like deleting important documents un/intentionally could be commonly seen in eDiscovery or legal investigations.

The DIE Model of Information Security: Distributed, Immutable and Ephemeral shown on Fig 2.

Distributed. A single target is far easier to attack than multiple ones. In the modern age of DDoS attacks, where bad actors take advantage of brute force and thousands of bots to disrupt a single entry point, companies must avoid putting all their (security) eggs in one basket. A distributed system limits the impact of any attack by confining it to a single space. It also has built-in redundancies, so if one system is compromised, it's possible to recover the system. There's never a single point of weakness to exploit.

Immutable. The first hint that there's a problem in security is an anomaly. The program doesn't behave as it should. So, the first thing bad actors do is eliminate evidence of the abnormality. Immutable logs make it impossible to cover up the anomalies. A change in a transparent system is quickly discovered, traced, and contained. This strategy is also good when security holes occur without malice, as the action that created this risk is traceable to a single action. Remedial measures are possible before exploitation.

Ephemeral. Administrators must break data and systems down to "pets and cattle." Pets are long-term necessities and cattle are dispensable. The more cattle, the better. A flexible network infrastructure made of cattle is also immutable. This means a temporary computing/logic layer over a persistent, secure data layer. It's safe to change and stop using legacy apps because they're no longer needed. Legacy programs are among the most vulnerable to attack—because bad actors have had time to learn their weaknesses—so it's best to eliminate them. An ephemeral infrastructure meets new challenges to protect indispensable assets [12].

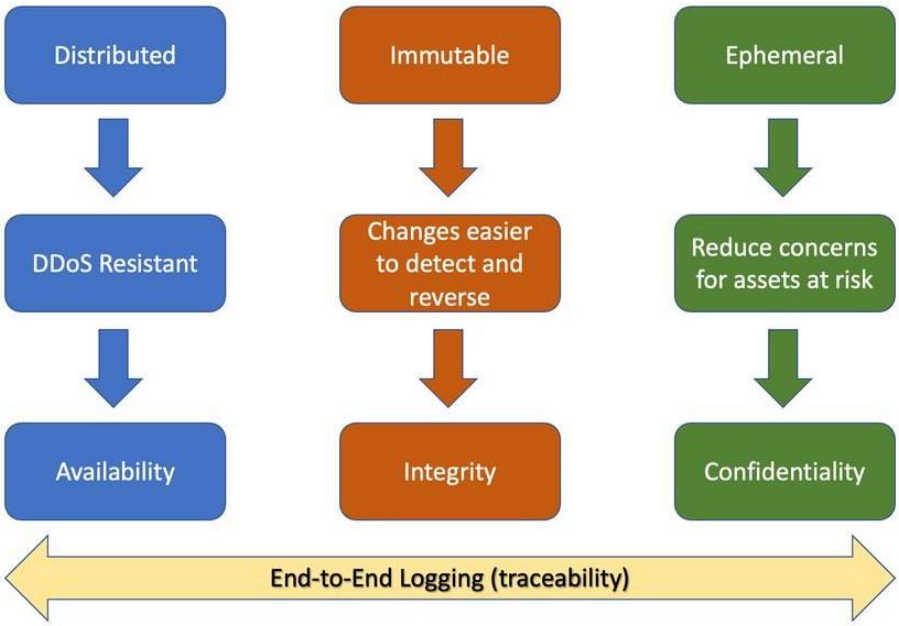

Fig. 2 Model of Information Security

Security Development Lifecycle. The Security Development Lifecycle (SDL) comprises a set of practices designed to ensure security and compliance requirements (Fig 3). Developed by Microsoft, the SDL aims to assist engineers in creating more secure software by minimizing the number and severity of vulnerabilities [4,6]. In a military context, the principles of confidentiality, integrity, availability, and traceability are crucial for ensuring the security and effectiveness of operations. Confidentiality safeguards classified information, such as intelligence and strategic plans, from unauthorized access. This is vital for maintaining operational security and preventing adversaries from exploiting sensitive data. Integrity ensures that information remains accurate and trustworthy preventing risks. Last would arise from tampered data, such as altered navigational coordinates or compromised mission orders. Availability is equally important, as military operations often depend on real-time access to critical systems and information; any disruption could hinder decision-making and jeopardize mission success. Finally, traceability provides accountability and enables thorough analysis by maintaining detailed logs of actions and decisions. This is essential for detecting unauthorized activities, investigating security breaches, and learning from operations to improve future strategies. Together, these principles form the foundation for the secure and effective functioning of military operations, ensuring that forces can respond swiftly and accurately to external and internal threats.

Secure Software Development Life Cycle (SSDLC)

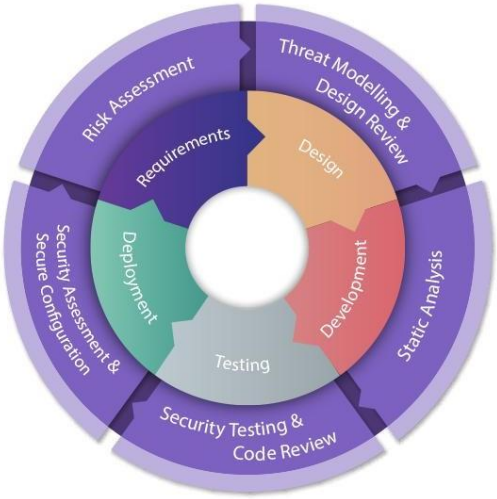

Fig. 3 SSDLC model

Training and Awareness. Security is a collective responsibility within an organization. Developers, DevOps engineers, and program and product managers must understand fundamental security principles and integrate security into software and services. This integration must balance business needs and user value. A comprehensive understanding of attackers' perspectives, objectives, and potential strategies enhances security awareness and elevates the collective level [1, 2].

Defining Security Requirements. The consideration of security and privacy is crucial in the development of secure applications and systems. Secu-

rity requirements must be continuously updated to reflect changes in functionality and the threat landscape, regardless of the development methodology. Factors influencing security requirements include legal and industry regulations, internal standards and coding practices, previous incidents, and known threats [4].

Metrics and Compliance Reporting. Defining minimum acceptable levels of security quality and holding engineering teams accountable is vital. The early definition helps teams understand risks, identify and fix security defects during development, and maintain standards throughout the project. Establishing a meaningful bug bar involves defining severity thresholds for security vulnerabilities and adhering to these thresholds. To effectively track key performance indicators (KPIs) and ensure the completion of security tasks, it is essential to utilize bug tracking or work tracking mechanisms. These mechanisms should allow for security defects and work items to be clearly labeled and marked with their appropriate severity, facilitating accurate tracking and reporting.

Threat Modelling. When dealing with high-security risk environments, it is crucial to engage in threat modeling. This methodology can be implemented at the component, application, or system scale, allowing development teams to consider, record, and deliberate on the security implications of their working environment. By following a systematic approach to potential threats, teams can efficiently and cost-effectively pinpoint vulnerabilities, evaluate risks, choose security measures, and implement suitable mitigations [7].

Design Requirements. SDL encompasses assurance activities that help engineers implement secure features. This process typically relies on security components such as cryptography, authentication, authorization, and logging. These features must be meticulously engineered to achieve the desired security outcomes.

Cryptography Standards. With the proliferation of mobile and cloud computing, it is critically important to ensure that all data, including security-sensitive information and management and control data, is protected from unintended disclosure or alteration during transmission or storage. Encryption is the primary method used to achieve this protection. Incorrect use of any aspect of cryptography can have catastrophic consequences, necessitating the development of clear encryption standards detailing every element of the encryption implementation. These standards have to be established by experts. A general guideline is to use only industry-vetted encryption libraries and ensure their implementation.

Managing Security Risks of Third-Party Components. The majority of modern software projects incorporate third-party components, both commercial and open-source. Maintaining an accurate inventory of third-party components and having a response plan for newly discovered vulnerabilities are essential stage in mitigating this risk. Depending on it, the type of used component, and the potential impact of a security vulnerability, additional validation seems to be necessary [8].

Using Approved Tools. Defining and publishing a list of approved tools, an associated security checks, such as compiler/linker options and warnings, is essential. Engineers should aim to use the latest versions of approved tools, including compiler versions, to benefit from new security analysis functionalities and protections.

Static Analysis Security Testing (SAST).

SAST involves analyzing the source code before compilation, providing a scalable method of security code review and ensuring compliance with secure coding policies. SAST is typically integrated into the commit pipeline to identify vulnerabilities when the software is built or packaged. Some tools integrate into the developer environment to detect and replace unsafe or banned functions with safer alternatives during active coding. The optimal frequency for performing SAST varies, and development teams should deploy multiple tactics to balance productivity with adequate security coverage.

Dynamic Analysis Security Testing (DAST). DAST involves performing runtime verification of fully compiled or packaged software to identify functionality issues that are apparent only when all components are integrated and running. DAST typically uses tools or prebuilt attacks to monitor application behavior for memory corruption, user privilege issues, and other critical security problems. Similarly to SAST, DAST does not have a one-size-fits-all solution [9]. While some tools, such as web application scanning tools, can be integrated into the continuous integration and delivery pipeline, others, such as fuzzing, require different approaches.

Penetration Testing. Penetration testing is a comprehensive security analysis performed by skilled security professionals simulating the actions of a hacker. The objective is to uncover potential vulnerabilities as the results of coding errors, system configuration faults, or operational deployment weaknesses. Penetration tests often find the variety of vulnerabilities and are typically performed together with automated and manual code reviews, providing a higher level of analysis than would otherwise be possible [10].

Establish a Standard Incident Response Process. Having an Incident Response Plan is crucial for helping to address new threats that can emerge over time. An incident response plan should be developed in

collaboration with dedicated Product Security Incident Response Team (PSIRT). It should outline who to contact in the event of a security emergency and establish the protocol for addressing security issues, including plans for code inherited from other groups within the organization and for third-party code. It's important to test the incident response plan before it's needed [9].

In a military context, integrating security principles becomes crucial for ensuring operational resilience and system integrity. These principles include training and awareness, defining security requirements, metrics and compliance reporting, threat modeling, design requirements, cryptography standards, management of third-party components, use of approved tools, static and dynamic analysis, penetration testing, and incident response.

Training and awareness help elevate the collective security posture, enabling personnel to anticipate and counter adversarial tactics. Continuous refinement of security requirements, informed by evolving threats and regulatory standards, ensures that military systems remain robust and compliant. Metrics and compliance reporting facilitate the early identification and remediation of vulnerabilities, while threat modeling allows for systematic risk assessment and mitigation across various system levels.

Cryptography and design requirements are fundamental for safeguarding sensitive communications, with the proper management of third-party components mitigating risks associated with external dependencies. The use of static and dynamic analysis, along with penetration testing, provides a rigorous evaluation of software security, ensuring that systems deployed in the field are resilient against sophisticated attacks. Finally, a well-defined incident response process is critical for enabling rapid and effective recovery from security breaches, thereby maintaining mission continuity and operational readiness.

These integrated security measures are essential for sustaining the strategic and tactical advantages necessary for military effectiveness.

Monitoring and Observability. In the realm of system management and performance analysis, the concepts of monitoring and observability play pivotal roles. Both are essential for maintaining robust and reliable systems. In this scientific discourse, we dissect these concepts, elucidating their nuances and implications. More details Cons are described in Table.

An alert serves as a notification intended for human consumption. It is dispatched to systems such as bug or ticket queues, email aliases, or pagers. However, paging a human is a costly endeavor. Interruptions disrupt workflow, whether at work or during personal time. Frequent alerts lead to noise, obscuring critical issues. Effective alerting systems strike a delicate balance between signal and noise.[2, 13].

Table. Monitoring vs Observability

Monitoring	Observability
Monitoring identifies what's going on	Observability explains why something is happening and provides actionable information.
Monitoring is designed around the collection of metrics and log files	Observable systems inherently provide data on their condition through instrumentation (telemetry)
Built around known entities - decide what to monitor	Collect and enriches monitoring data; adds other data sources and answers questions that could never have been asked before
Good for establishing overall health state. Poor performance is an unhealthy state	Designed for granular vision, context and debugging
Monitoring remains a key task for IT Operations, DevOps and SREs (Site Reliability Engineer)	Observability includes monitoring but prolongs it. It is both a result and a culture

Monitoring revolves around collecting metrics and log files. It focuses on established entities—known aspects of the system. Monitoring is adept at establishing overall health states, flagging poor performance as an unhealthy condition. It remains a core responsibility for IT Operations, DevOps, and Site Reliability Engineers [13-14].

Observability extends beyond monitoring. It embraces a culture and mindset. Observable systems inherently provide data through instrumentation (telemetry). Observability enriches monitoring by answering questions that were previously unasked. It thrives on granular vision, contextual understanding, and debugging prowess. The “why” behind system behavior drives observability [14].

5. Conclusions

In the constantly evolving field of information security, it is crucial along with modern frameworks to adopt a comprehensive approach that incorporates fundamental principles. This is essential to protect sensitive data and maintain the integrity of systems. The key principles of confidentiality, integrity, availability,

and traceability form the foundation for safeguarding information, ensuring its accuracy, enabling reliable access, and providing comprehensive accountability through robust audit mechanisms.

The incorporation of the DIE (Distributed, Immutable, Ephemeral) model introduces a modern paradigm in security architecture. This model leverages distributed systems to mitigate single points of failure, employs immutable logs to enhance transparency and traceability, and adopts an ephemeral infrastructure to phase out legacy vulnerabilities. As a result, it offers a resilient structure that addresses current and emerging threats.

The Security Development Lifecycle (SDL) further augments this security posture by integrating security considerations throughout the software development process. Through systematic training programs, the continuous refinement of security requirements, and the application of rigorous testing methodologies—such as Static Analysis Security Testing (SAST) and Dynamic Analysis Security Testing (DAST)—the SDL ensures that security is a foundational element in software design and implementation. Additionally, the SDL emphasizes the critical importance of managing third-party components, utilizing approved tools, and establishing a robust incident response framework, all of which contribute to a comprehensive security strategy.

After integrating these principles, models, and methodologies, organizations can build a highly resilient and secure information system architecture. This integrated approach is essential for minimizing vulnerabilities, ensuring regulatory compliance, and protecting data assets in an increasingly complex and adversarial digital landscape. The fundamental principles of confidentiality, integrity, availability, and traceability are vital for securing classified and mission-critical data. They ensure the accuracy of information essential for strategic decision-making and maintain access to systems under all conditions. The DIE model introduces a resilient security architecture that mitigates single points of failure, enhances transparency through immutable logs, and reduces vulnerabilities by eliminating legacy systems. These aspects are crucial in countering advanced and evolving threats. The SDL integrates security throughout the software development process, embedding rigorous testing methodologies and systematic management of third-party components to ensure the reliability and security of military systems. This approach is essential in minimizing vulnerabilities, ensuring compliance with stringent defense standards, and protecting critical data assets in an increasingly adversarial digital environment.

6. Gratitude

The authors express their gratitude to the staff of Department of Information and Measuring Technologies for help in their work.

7. Mutual claims of authors

The authors have no claims against each other.

References

[1] Mark G. Graff, Kenneth R. van Wyk, Secure Coding: Principles and Practices, O'Reilly Media, Inc., 2023. https://www.amazon.com/Secure-Coding-Principles-Mark-Graff/dp/0596002424

[2] Welcome to the OWASP Top 10 – 2021 OWASP 2022. [Online]. Available https://owasp.org/Top10/

[3] Paco Hope, Ben Walther, Web Security Testing Cookbook, O'Reilly Media, Inc., 2008. https://www.oreilly.com/library/view/web-security-testing/9780596514839/

[4] Secure coding guidelines, Microsoft 2021. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/security/secure-coding-guidelines

[5] Mark J. Price. C# 9 and .NET 5 – Modern Cross-Platform Development: Build intelligent apps, websites, and services with Blazor, ASP.NET Core, and Entity Framework Core using Visual Studio Code, 5th ed; Packt Publishing: 35 Livery Street Birmingham B3, 2PB, UK, 2020. https://www.amazon.com/NET-Cross-Platform-Development-intelligent-Framework/dp/180056810X

[6] Samuele Resca. Hands-On RESTful Web Services with ASP.NET Core 3 1st ed; Packt Publishing: 35 Livery Street Birmingham B3, 2PB, UK, 2019. https://www.amazon.com/Hands-RESTful-Services-ASP-NET-Core/dp/1789537614

[7] Secure development and deployment guidance, National Cyber Security Centre. [Online]. Available: https://www.ncsc.gov.uk/collection/developers-collection

[8] Adam Freeman. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages, 9th ed; Appres: London, UK, 2022. https://www.amazon.com/Pro-ASP-NET-Core-Cloud-Ready-Applications/dp/1484279565

[9] Cesar de la Torre, Bill Wagner, Mike Rousos, NET Microservices Architecture for Containerized .NET Applications, One Microsoft Way Redmond, Washington 98052-6399, 2022. https://learn.microsoft.com/en-us/dotnet/architecture/microservices/

[10] V. Samotyy, U. Dzelendzyak, N. Mashtaler, "A Comparative Study of Data Annotations and Fluent Validation in. NET", International Journal of Computing, Vol. 23, iss. 1, p. 72–77, 2024, doi: 10.47839/ijc.23.1.3437.

[11] Suliman Alazmi; Daniel Conte De Leon, "A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners", IEEE Access, Vol. 10, p. 33200 - 33219, 2022, doi: 10.1109/ACCESS.2022.3161522

[12] Andreas Dann, Henrik Plate, Ben Hermann, Serena Elisa Ponta, Eric Bodden, "Identifying Challenges for OSS Vulnerability Scanners - A Study & Test Suite", IEEE Transactions on Software Engineering, Vol. 48, p. 3613 - 3625, 2022, doi: 10.1109/ TSE.2021.3101739.

[13] Ishan Siddiqui, Ankit Pandey, Saurabh Jain, Hetang Kothadia, Renuka Agrawal, Neha Chankhore, "Comprehensive Monitoring and Observability with Jenkins and Grafana: A Review of Integration Strategies, Best Practices, and Emerging Trends", Comprehensive Monitoring and Observability with Jenkins and Grafana: A Review of Integration Strategies, Best Practices, and Emerging Trends , Ankara, Turkiye, 26-28 October 2023, doi: 10.1109/ISMSIT58785.2023.10304904.

[14] Muhammad Usman, Simone Ferlin, Anna Brunstrom, Javid Taheri, "A Survey on Observability of Distributed Edge & Container- Based Microservices", IEEE Access, Vol. 10, p. 86904 - 86919 , 2022, doi: 10.1109/ ACCESS.2022.3193102.