 e-ISSN No. 2584-1025

DevSecOps: Improving Software Development Lifecycle

Name: Ruchi Ranka, Sankalp Paranjpe, Rohit Pachlor
Affiliation: MIT ADT University, MIT ADT University, MIT ADT University
Email id: ruchiranka2103@gmail.com, sankalpparanjpe.sp@gmail.com, rohit.pachlor@gmail.com

***Abstract*—In the rapidly evolving landscape of software development, the seamless integration of development, IT operations, and security teams is vital to ensuring the creation of secure and robust applications. Unfortunately, organizations currently face a significant challenge in effectively bridging the gap between these teams during the application development process. This disconnect results in critical security threats that pose substantial risks to the organization's applications, data, and customers. This review paper examines existing literature to assist emerging researchers and students in revitalizing research papers.**

***Index Terms*—Software Development Life Cycle (SDLC), Secure SDLC, Continuous Integration and Continuous Deployment (CI/CD) Pipelines, Risk Assessment and Mitigation, Threat Modeling, Secure Coding, Software Security.**

INTRODUCTION

DevSecOps is a methodology where security is implemented in each stage of software development, ensuring a robust defense against potential threats. This proactive approach significantly reduces the potential risks and vulnerabilities, making the software resilient against attacks. Furthermore, as development, security, and operations teams are integrated, DevSecOps promotes collaboration and communication among different stakeholders. This synergy ensures that security concerns are addressed from the very beginning of the software development cycle. This incorporation of security concerns along with the software development life cycle is called the Secure Software Development Life Cycle. In order to understand the different phases of secure SDLC as described in, a basic understanding of SDLC and knowledge of security terminologies are required.

A. Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a process used to design, develop, test, and deploy software applications. It provides a structured approach for creating, planning, testing, deploying, and maintaining information systems and software projects. SDLC aims to produce software that meets customer demands, is completed within time and cost estimates, and satisfies the needs of stakeholders involved in the project. Following are the stages involved in the SDLC-

1) Planning: This is the first phase and it involves defining the scope, goals, and requirements of the project. In this, project managers work closely with stakeholders to gather information, assess feasibility, and create a project plan for developing the project. They outline resources, time frames, and a rough cost estimate during this phase.

2) Analysis: This is the second phase and it involves the analysis of the project's technical, operational, and financial aspects. It determines if the proposed software or hardware application is practical and achievable within the given constraints of requirements.

3) Design: This is the third phase and it involves the system architects and developers to create a detailed technical design based on the requirements gathered. This design serves as a blueprint for the actual development and coding process. It includes high-level and low-level design specifications, database structures, and overall system architecture.

4) Implementation: This is the fourth phase and it involves the developers to start coding according to the design specifications. They follow coding standards, guidelines, and best practices to ensure the code is readable, maintainable, and efficient. Continuous collaboration and communication are crucial in this phase to address any issues promptly. Technologies such as HTML, CSS, JavaScript, .NET, Flutter, Java, Spring Boot, Python, Django, PHP, AWS, Azure, GCP, CMS, etc. are used.

5) Testing: This is the fifth stage, when the processes for validation and quality assurance are involved. It consists of acceptability testing, system testing, integration testing, and unit testing. Testing guarantees that the program works as planned and satisfies the client's needs.

6) Deployment: This is the sixth phase and it involves the software being deployed to a production environment. Deployment can be done in phases or all at once, depending on the project requirements. Deployment activities include data migration, configuration setup, and user training.

7) Maintenance: This is the sixth phase, which involves developers and support teams addressing user-reported issues, updating the software to adapt to changing environments, and adding new features or enhancements after the software has been deployed in the production environment.

B. Security Terminologies

- **Vulnerability:** A weakness in the system that can be exploited.
- **Attack:** The type of an attack.

Fig. 1: Phases of Software Development Life Cycle

• **Attack Vector:** The path that an attacker can take to exploit a vulnerability in the system.
• **Attack Surface:** Anything that can be obtained, used, or attacked by a threat actor.
• **Risk:** The probability of occurrence of Attack.

$$Risk = Impact * Likelihood \quad (1)$$

B. Secure Software Development Life Cycle

The Secure Software Development Life Cycle (Secure SDLC) is an improved approach to the standard Software Development Life Cycle (SDLC) that combines security measures and best practices across the whole software development process. Secure SDLC strives to discover and eliminate security vulnerabilities early in the development life cycle, decreasing the risk of security breaches and enabling the creation of robust, secure software products. Following are the stages involved in the SDLC-

1) Requirements Gathering and Analysis: This is the first phase and it involves security requirements to be identified alongside functional requirements. Security experts collaborate with stakeholders to determine potential threats and establish security objectives.

2) Threat Modeling: This is the second phase and it involves Threat Modelling. Threat modeling is a structured approach to identifying, evaluating, and mitigating security risks in software applications or systems. It helps to visualize the possibility of threats and vulnerabilities in the system or application software. It is used by software developers, architects, and security professionals to understand potential threats, vulnerabilities, and possible attack vectors that could compromise the security of a system or application software.

Phases of Threat Modelling:

• **Identifying Organization's assets:** This is the stage where the system or application software is identified, along with its assets, components, and boundaries.

• **Identifying Threats:** This stage identifies potential threats or risks to the system. These can include both hardware and Software vulnerabilities.

• **Assessing Vulnerabilities:** This stage determines the vulnerabilities or weaknesses in the application that the identified threats could exploit. This can involve analyzing the system architecture, code reviews, or penetration testing. It assigns a Common Vulnerability Scoring System (CVSS) score and an Exploit Prediction Scoring System (EPSS) score.

• **Iteration and Updation:** Regularly new Common Vulnerability Exposures (CVEs) are assigned, and threat modeling should be a continuous process. Continuous iteration and updating are necessary. There are several methodologies and frameworks available for conducting threat modeling, such as OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation), DREAD (Damage, Reproducibility, Exploitability, Affected Users, Discoverability), and STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). These methods provide structured approaches to systematically analyze and address security risks in software systems. Excluding threat modeling from the Software Development Life Cycle (SDLC) poses significant risks to the security integrity of the developed software. It can be likened to constructing a building without a foundation, leaving the structure vulnerable to unseen vulnerabilities. Without a meticulous threat analysis, the software remains susceptible to a plethora of potential security breaches, ranging from unauthorized access to data manipulation and exploitation. This exclusion essentially creates an open door for cyber threats. Integrating threat modeling into the SDLC is imperative, serving as a crucial preemptive measure akin to fortifying the digital infrastructure, and ensuring robust protection against evolving cybersecurity challenges.

3) Secure Coding and Implementation: This is the fourth phase and it involves the developers writing secure code based on secure coding guidelines and best practices. They use secure APIs, input validation techniques, and encryption to protect against common vulnerabilities like SQL Injection and Cross-Site Scripting (XSS). Code reviews and static analysis tools are employed to identify and rectify security issues in the source code.

4) Security Testing: This is the fifth phase and it involves DevSecOps engineers and security engineers to perform security testing at various levels, including static analysis, dynamic analysis, and penetration testing. Static analysis tools like SonarQube scan the source code for vulnerabilities, dynamic analysis tools such as OWASP ZAP test the software during runtime, and penetration testing simulates real-world attacks to uncover potential weaknesses.

5) Deployment: This is the sixth phase and security measures continue during deployment. Secure configurations, strong authentication, and encryption of data in transit and at rest are to be ensured. Proper access controls are to be enforced, and security patches are to be promptly applied to the software and its dependencies to prevent known vulnerabilities from being exploited.

6) Logging, Monitoring, and Response: This is the seventh phase and once the software is deployed, continuous logging and monitoring is essential. Security logs are analyzed, and any suspicious activities are investigated promptly. Incident response plans are in place to address security breaches, ensuring a rapid and effective response to any security incidents that may occur.

II. OUR CONTRIBUTION

This review paper provides a comprehensive examination of the current landscape of DevSecOps and DevOps practices, focusing on the security challenges encountered and the solutions proposed in the field. Its purpose is to aid software developers and researchers in establishing more secure and efficient software development processes. The study delves deeply into understanding the inherent security challenges in these domains, the diverse research methods employed to navigate these challenges, potential security threats that can compromise the software development process, and solutions

proposed to overcome these obstacles. The paper thoroughly investigates the issues faced when implementing DevSecOps and DevOps in software development processes, encompassing both theoretical challenges and practical hurdles confronted by organizations striving to enhance the security and efficiency of their development processes. Additionally, it conducts a detailed analysis to comprehend the various methodologies and techniques utilized to tackle the challenges of secure software development. The evolving landscape of potential security threats targeting the software development process is outlined in the literature survey. Furthermore, the exploration of solutions proposed in the reviewed literature illuminates best practices, tools, and strategies suggested to bolster the security of the software development life cycle, ensuring the integration of security considerations at every stage.

III. LITERATURE SURVEY

1) Title: DevOps methodology in modern software development.
Authors: Adin Jahiac, Nevzudin Buzadija.
Publication: Quantum Journal of Engineering, Science and Technology.
Publisher: Polytechnic Faculty, University of Zenica, Zenica, Bosnia and Herzegovina.
Year: 2023
Description: The paper discusses various key concepts in DevOps, including cloud computing, virtualization, containerization, and infrastructure as code. It explains the need for an internal developer platform (IDP) as the main focus of DevOps engineers. The authors also delve into the challenges faced during the continuous integration and continuous deployment processes and discuss the use of virtual machines and the emergence of CI/CD pipelines as solutions to address these challenges. The paper provides insights into the classification and terminology associated with continuous integration, continuous delivery, and continuous deployment. It highlights the importance of collaboration, automation, continuous development, resolving client needs, and abstraction in DevOps. [1]

2) • Title: An Enhanced CICD Pipeline: A DevSecOps Approach.
• Authors: Olumide Bashiru Abiola, Olusola Gbenga Olufemi.
• Publication: International Journal of Computer Applications.
• Publisher: ResearchGate
• Year: 2023
• DOI: 10.5120/ijca2023922594
• Description: The paper discusses the implementation of a DevSecOps approach in the Continuous Integration/Continuous Delivery (CI/CD) pipeline. It addresses the challenges faced in software development, including the rise in cyber-attacks and the need for effective security measures in the CI/CD pipeline. The paper also mentions the increasing threat of cyber-attacks and the need to combat them. The paper also highlights the necessity for enterprises to embrace a DevSecOps approach in order to guarantee the security, speed, and quality of software releases. It also discusses the usage of AWS CI/CD technologies and their potential to improve the security of the pipeline.[2]

3) Title: DevSecOps: Design Science Research of the DevSecOps Technology Stack, Definitions, Concepts, and Improvements Identified Thereof.
• Authors: Christopher E Otwell
• Publication: ProQuest
• Publisher: Colorado Technical University
• Year: 2022
• Description: The study utilizes the Design Science Research (DSR) methodology to explore known DevSecOps implementations and address the gaps in the existing literature. It focuses on the impact of software factories on security within automated CI/CD factory software deployment systems. It provides a conceptual framework, background information, and references to seminal literature on DevSecOps. The study identifies the need for a specific Fig. 2: Phases of Secure Software Development Cycle Parameters SDLC Secure SDLC Focused on Overall software development process. Integrating security measures and practices throughout. Security Activities included Not inherently specific to security. Includes threat modeling, secure design, coding, and testing. Action on security risks Vulnerabilities is addressed later. Proactive identification and mitigation of security risks. Incorporation of Security Typically addressed later in the process. Integration at every phase, from planning to deployment. Education and Awareness General best practices; security education varies. Emphasizes specific security training for developers. Continuous Improvement Focuses on general process improvement. Learns from incidents, promotes continuous security upgrades. TABLE I: Comparing SDLC and Secure SDLC definition and practical roadmap for implementing DevSecOps and proposes a comprehensive technology stack based on the DevSecOps Reference Design framework. Overall, this study aims to standardize and improve the definition of DevSecOps technology stacks through DSR.[3]

4) Title: Definition of a DevSecOps Operating Model for software development in a large Enterprise.
• Authors: Valentina Tortoriello.
• Year: 2022
• Description: The study discusses the need for technology companies to adopt agile methodologies like DevOps in response to the rapidly evolving IT landscape. It highlights DevOps as a means to accelerate software development and release, through automation with quality maintenance and security. It acknowledges the challenge of seamlessly integrating security into DevOps and proposes a DevSecOps approach that emphasizes cross-team collaboration. It also includes research, conducted within Vodafone Italia's Cyber Security team, that aims to enhance the company's setup, tools, and processes to align with DevSecOps principles. The outcome is a DevSecOps Operating Model, defining roles and responsibilities among various teams to strengthen product security. [4]

5) Title: Challenges and solutions when adopting DevSecOps: A systematic review.
• Authors: Roshan N. Rajapakse, Mansooreh Zahedi, M. Ali Babar, Haifeng Shen.
• Publication: ScienceDirect.
• Publisher: Elsevier B.V.
• Year: 2021
• DOI: 10.1016/j.infsof.2021.106700
• Description: The paper conducts a systematic literature review on DevSecOps adoption challenges and provides solutions based on 54 selected papers. Challenges encompass tool selection, security issues, configuration management, analysis tools limitations, container vulnerabilities, and more. Solutions proposed include tool standardization, clear documentation, best practices, cloud solutions, Interactive Application Security Testing (IAST) tools, vulnerability assessment, security champions, culture change, product-specific vulnerability evaluation, and improved inter-team communication. The methodology involves snowballing, data extraction, thematic analysis, and gap identification. The study emphasizes the importance of tool selection, collaboration, automation, and continuous security assessment for successful DevSecOps implementation.[5]

6) Title: Best Practices for Ensuring Security in DevOps: A Case Study Approach.
• Authors: Rajavi Desai, T N Nisha. • Publisher: IOP Publishing Ltd.

• Year: 2021
• DOI: 10.1088/1742-6596/1964/4/042045
• Description: Using a case study methodology, the paper aims to uncover the best strategies for guaranteeing security in DevOps. A literature review was conducted, as well as interviews with three organizations that have used DevSecOps. The interviews provided insights into the implementation of DevSecOps and the challenges faced by the organizations. The challenges identified are due to collaboration and integration of DevSecOps implementation. To resolve these challenges, the paper suggests implementing training programs to educate employees about security and their contributions, integrating security tools and systems with the original systems, and taking preventive measures such as securing codes, databases, networks, and physical environments.[6]

7) • Title: Security as Culture: A Systematic Literature Review of DevSecOps.
• Authors: Mary Sanchez-Gordon, Ricardo Colomo-´ Palacios.
• Publisher: Association for Computing Machinery (ACM).
• Year: 2020
• DOI: 10.1145/3387940
• Description: The paper highlights the limited empirical research on DevSecOps culture, indicating that it is an important topic that requires further investigation. It also discusses the difficulties in finding employees who understand DevSecOps both technically and procedurally, as well as the necessity of transparency in the DevSecOps model. The study uses a standard Systematic Literature Review (SLR) methodology, which entails inclusion and exclusion criteria, search protocols, and a research question. The paper provides insights into the cultural aspects of DevSecOps and serves as a starting point for further research. The paper emphasizes the importance of collaboration, continuous improvement, communication, trust, and leadership in fostering a strong DevSecOps culture. [7]

8) • Title: Putting the Sec in DevSecOps: Using Social Practice Theory to Improve Secure Software Development.
• Authors: Debi Ashenden, Gail Ollis.
• Publisher: Association for Computing Machinery (ACM).
• Year: 2020
• DOI: 10.1145/3442167
• Description: The paper addresses the challenge of blending cybersecurity with modern software development practices like open-source, agile, DevOps, and DevSecOps. Rather than solely focusing on individual training, it advocates for understanding the collective behaviors of software developers. It utilizes Social Practice Theory (SPT) to offer recommendations for aligning cybersecurity with software development. The study involves a quick analysis of software development practices and interviews with key informants to inform interventions for secure software development. The paper concludes by proposing workshops for collaboratively shaping security practices within software development and it also suggests avenues for future research. [8]

9) Title: Automating Security Tests for Web Applications in Continuous Integration and Deployment Environment.
• Authors: Abdollah Shajadi.
• Publication: Theseus.
• Publisher: Oulu University of Applied Sciences.
• Year: 2018
• Description: The paper addresses the challenges of traditional penetration testing reports, which are often lengthy and difficult for developers to navigate, leading to an abundance of false positives and unrelated issues. The paper introduces the Skinner which is a command line tool written in Python that serves as a proof of concept for a Dynamic Application Security Testing (DAST) solution that automates the security testing process. The methodology used in the paper involves integrating security testing tools into the existing DevOps infrastructure and making them automated. The solutions provided in the paper aim to make security testing more agile and integrated into the software development process. The paper also suggests future developments, such as incorporating additional security tools like fuzzers and analyzers, to enhance the sophistication of scans.[9]

10) Title: Aspects of Enhancing Security in Software Development Life Cycle.
• Authors: Anuradha Sharma, Praveen Kumar Misra. • Publication: Research India Publications.
• Year: 2017
• Description: The paper suggests the need for proactive security measures, risk analysis, and continuous monitoring to ensure the development of secure software applications. It also mentions the use of threat modeling, architectural risk analysis, and neural network approaches to enhance security. The paper emphasizes the importance of continuous monitoring and analysis of risks throughout the software life cycle. It implies that information that is learned by comprehending exploits and attacks ought to be delivered back into the development organization. [10]

11) • Title: Developer-Driven Threat Modeling.
• Authors: Danny Dhillon.
• Publisher: IEEE
• Year: 2011 • DOI: 10.1109/MSP.2011.47
• Description: The paper addresses various threat modeling experiences, including main problems encountered and lessons acquired. The paper presents a developer-driven threat modeling approach that aims to address security flaws and reduce risk in software systems. The methodology used in this paper involves creating annotated data flow diagrams, identifying and analyzing threats, and providing actionable mitigation strategies. The paper underscores the advantages of employing threat modeling, including the ability to detect architectural security vulnerabilities at an early stage. [11]

12) Title: Integrating Risk assessment and Threat modeling within SDLC process.
• Authors: Maheshwari V, Prasanna M.
• Publisher: Institute of Electrical and Electronics Engineers (IEEE).
• Year: 2011
• DOI: 10.1109/INVENTIVE.2016.7823275
• Description: The paper explains the risk management process, which includes risk identification, analysis, planning, and monitoring, and its integration into the SDLC. The threat modeling methodology was utilized in this paper, which entails identifying potential threats and risks, analyzing their impact, and determining countermeasures to minimize them. The paper discusses the STRIDE threat model methodology (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). The relevance of incorporating risk assessment and threat modeling into SDLC is emphasized in the paper. [12]

13) Title: Threat Modeling within the SDLC.
• Authors: Tony UcedaVelez and Marco M. Morana. ´
• Publisher: John Wiley & Sons, Inc.
• Year: 2015
• Description: The paper discusses the integration of threat modeling into different types of Software Development Life Cycles, including Waterfall and Agile methodologies to enhance application security. It highlights the significance of creating a thorough security test plan that includes both manual test cases and automated techniques for detecting potential security risks. The paper also emphasizes the need for secure configuration, installation, and operation of applications during the deployment stage. Additionally, it discusses the role of threat modeling in change management, which is to assess the impact of proposed changes on the security of the application and maintain security during subsequent releases. [13]

14) • Title: Introducing development teams to threat modeling in SDLC.
• Authors: Izar Tarandach, Matthew J. Coles.
• Publisher: TechTarget.
• Year: 2015
• Description: The study states that while threat modeling is a mature concept, its consistent application within organizations is not well-established. The authors anticipate that the industry will evolve with more people adopting threat modeling, leading to improved system security as practices become more refined and accessible. The study explores two approaches to automate threat modeling: threat modeling with code and threat modeling from code, each with distinct goals and considerations. It emphasizes that automation tools can serve as facilitators for threat modeling conversations, allowing organizations to focus on understanding and applying threat modeling principles.[14]

15) Title: Threat Modeling: from Software Security to Cyber Risk Management.
• Authors: Ronan Mouchoux.
• Publisher: Xrator
• Year: 2023
• Description: The article discusses the importance of cyber risk management and the potential consequences of cyber-attacks. It highlights the need for clear terminology and semantic models to identify threats and anticipate attack opportunities. The article explores the utility of threat analysis and the integration of cyber threat intelligence into the threat modeling and risk management processes. The methodology used by the article is Cyber Threat Intelligence, which is the discipline of documenting and normalizing adversaries and threat events that add value to cyber threat modeling. [15]

16) Title: Systematic Literature Review on Security Risks and Its Practices in Secure Software Development.
• Authors: Rafiq Ahmad Khan, Siffat Ullah Khan, Habib Ullah Khan, Muhammad Ilyas.
• Publication: Institute of Electrical and Electronics Engineers (IEEE).
• Publisher: Qatar National Library, Doha, Qatar.
• Year: 2022
• DOI: 10.1109/ACCESS.2022.3140181
• Description: In the article, the authors have conducted a Systematic Literature Review (SLR) in order to identify the best practices and security risks in Secure Software Development. In this literature survey, they have identified 145 security risks. In addition, they have also highlighted 424 security practices that are helpful in managing security aspects in Software Development Life Cycle. The identified security risks and practices can be incorporated into each phase of the SDLC to enhance the security of SDLC. The objective of this article is to use the findings of survey to aid in software development organizations in enhancing the level and efficiency of security of their software products. [16]

17) Title: Risk Assessment, Threat Modeling and Security Testing in SDLC.
• Authors: Alya Hannah Ahmad Kamal, Caryn Chuah Yi Yen, Gan Jia Hui, Pang Sze Ling, Fatima-tuzZahra.
• Publisher: Taylor's University, Selangor.
• Year: 2012
• Description: The paper discusses the importance of integrating security principles such as risk asssessment, threat modeling, secure code review, and security testing into the software development life cycle. The paper proposes the use of the Secure Software Development Life Cycle methodology as a solution to ensure a more secure software development process.

IV. LEARNINGS

Analyzing these research papers on different parameters such as use of threat modeling, discussion of security and its automation, security risks' assessment and use of best practices to incorporate security in the software development life cycle and DevSecOps, yields insights into security trends, common issues and challenges, and threats, effective solutions, and emerging methodologies. These papers shed light on how organizations approach shared responsibility, leverage automation, secure their CI/CD pipelines, and foster a DevSecOps culture. These findings provide a holistic view of security practices in the entire software development lifecycle.

I. V. CONCLUSION AND FUTURE WORK

In summary, the research discussed here underscores the critical evolution of software development practices, specifically highlighting the imperative integration of security measures. The insights provided emphasize the collaborative and automated nature of DevOps, positioning it as a cornerstone in contemporary software development. Additionally, the urgent need for a DevSecOps approach is underscored, emphasizing the seamless integration of security protocols within CI/CD pipelines to combat escalating cyber threats effectively.

Building upon the foundational findings derived from the review of these research papers, future work would revolve around the development of a comprehensive and adaptable framework that harmonizes threat modeling and infuses security best practices throughout the software development lifecycle. This framework should prioritize a shared responsibility model, advanced automation, and robust CI/CD pipeline security.

Fig. 2: Phases of Secure Software Development Cycle

Parameters	SDLC	Secure SDLC
Focused on	Overall software development process.	Integrating security measures and practices throughout.
Security Activities included	Not inherently specific to security.	Includes threat modeling, secure design, coding, and testing.
Action on security risks	Vulnerabilities are addressed later.	Proactive identification and mitigation of security risks.
Incorporation of Security	Typically addressed later in the process.	Integration at every phase, from planning to deployment.
Education and Awareness	General best practices; security education varies.	Emphasizes specific security training for developers.
Continuous Improvement	Focuses on general process improvement.	Learns from incidents, promotes continuous security upgrades.

TABLE I: Comparing SDLC and Secure SDLC

Paper/Parameters	Paper-1	Paper-2	Paper-3	Paper-4	Paper-5	Paper-6
Security Discussion	✗	✓	✓	✓	✓	✓
Security Automation	✗	✗	✓	✗	✓	✓
Logging and Monitoring	✗	✓	✗	✗	✗	✗
CI/CD	✓	✗	✓	✗	✗	✗
Large Enterprise focused	✗	✗	✗	✗	✓	✗
Large Enterprise focused	✗	✗	✗	✗	✗	✗
Infrastructure as a Code	✓	✗	✗	✗	✗	✗
Source code analysis	✓	✗	✓	✗	✓	✓
Threat Modeling	✗	✓	✗	✓	✗	✓
Shared Responsibility Model	✗	✗	✓	✗	✓	✗
Application focus	DevOps in Software Development	Risk Assessment using various methodologies and techniques of threat modeling	CI/CD	Threat modeling, identifying security flaws, and mitigating risks in software systems	DevSecOps implementation	Data Extraction and Synthesis, Software Security Risks

TABLE II: Comparison of relevant parameters in different papers

VI. REFERENCES

[1] Jahic, Adin & Buzaija, Nevzudin. (2023). DEVOPS METHODOLOGY IN MODERN SOFTWARE DEVELOP-MENT. 4. 2716-6341.

[2] Bashiru, Olumide & Olufemi, Olusola. (2023). An Enhanced CICD Pipeline: A DevSecOps Approach. International Journal of Computer Applications. 184. 8-13. 10.5120/ijca2023922594.

[3] DevSecOps: Design Science Research of The DevSecOps Technology Stack, Definitions, Concepts, And Improvements Identified Thereof - ProQuest". Proquest.Com, 2023

[4] F. Floris, "POLITECNICO DI TORINO Definition of a DevSecOps Operating Model for software development in a large Enterprise Company Tutor," 2021. Available: https://webthesis.biblio.polito.it/secure/23649/1/tesi.pdf

[5] Roshan N. Rajapakse a b et al. (2021) Challenges and solutions when adopting DevSecOps: A systematic review, Information and Software Technology.

[6] Desai1, R. and Nisha1, T.N. (2021) *IOPscience*, *Journal of Physics: Conference Series*.

[7] M. Sanchez-Gord´ on and R. Colomo-Palacios, "Security´ as Culture," Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, Jun. 2020, doi: https://doi.org/10.1145/3387940.3392233.

[8] D. Ashenden and G. Ollis, "Putting the Sec in DevSecOps: Using Social Practice Theory to Improve Secure Software Development," New Security Paradigms Workshop 2020, Oct. 2020, doi: https://doi.org/10.1145/3442167.3442178

[9] Automating security tests for web applications in continuous security testing. Available at: https://www.theseus.fi/bitstream/handle/10024/166541/Abdoll ah Shajadi Automated Security Testing.pdf

[10] Misra, Anuradha. (2017). Aspects of Enhancing Security in Software Development Life Cycle. International Journal for Computational Methods in Engineering Science and Mechanics. 1. 203-210.

[11] D. Dhillon, "Developer-Driven Threat Modeling: Lessons Learned in the Trenches," in IEEE Security & Privacy, vol. 9, no. 4, pp. 41-47, July-Aug. 2011, doi: 10.1109/MSP.2011.47.

[12] V. Maheshwari and M. Prasanna, "Integrating risk assessment and threat modeling within SDLC process," 2016 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2016, pp. 1-5, doi: 10.1109/INVENTIVE.2016.7823275.

[13] Ucedavelez, Tony & Morana, Marco. (2015). Threat Modeling and Risk Management. 10.1002/9781118988374.ch5.

[14] R. A. Khan, S. U. Khan, H. U. Khan and M. Ilyas, "Systematic Literature Review on Security Risks and its Practices in Secure Software Development," in IEEE Access, vol. 10, pp. 5456-5481, 2022, doi: 10.1109/ACCESS.2022.3140181.

[15] Kamal, Alya & Yen, Caryn & Hui, Gan & Sze Ling, Pang & Tuz Zahra, Fatima. (2020). Risk Assessment, Threat Modeling and Security Testing in SDLC.

[16] Jahic, A. and Buzaija, N. (no date) DEVOPS method-´ ology in modern software development, Quantum Journal of Engineering, Science and Technology. Available at: https://www.qjoest.com/index.php/qjoest/article/view/81

[17] Bashiru, Olumide & Olufemi, Olusola. (2023). An Enhanced CICD Pipeline: A DevSecOps Approach. International Journal of Computer Applications. 184. 8-13. 10.5120/ijca2023922594.

[18] A. Ibrahim, A. H. Yousef and W. Medhat, "DevSecOps: A Security Model for Infrastructure as Code Over the Cloud," 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 2022, pp. 284-288, doi: 10.1109/MIUCC55081.2022.9781709.

[19] A. M. Putra and H. Kabetta, "Implementation of DevSecOps by Integrating Static and Dynamic Security Testing in CI/CD Pipelines," 2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM), Laguboti, North Sumatra, Indonesia, 2022, pp. 1-6, doi: 10.1109/ICOSNIKOM56551.2022.10034883.

[20] T. Chen and H. Suo, "Design and Practice of Security Architecture via DevSecOps Technology," 2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2022, pp. 310-313, doi: 10.1109/ICSESS54813.2022.9930212.

[21] N. Chavan, N. Bharambe, S. Deshmukh, D. Ahire, and A. R. Jain, "Implementing DevSecOps pipeline for an enterprise - IJARSCT," International Journal of Advanced Research in Science, Communication and Technology, https://ijarsct.co.in/A3883.pdf