The Role of Chaos Engineering in DevOps for Software Robustness

Nihar Ajay Mhatre, Mugdha Shailendra Kulkarni, Fatima Ali

Symbiosis Centre for Information Technology, Symbiosis International (Deemed University), Pune, India

Corresponding author: Nihar Ajay Mhatre, Email: nmhatre2541@gmail.com

This research paper explores the evolution of software development methodologies, beginning with DevOps, a collaborative approach that integrates development and operations to streamline workflows and enhance software delivery. Method: Building upon DevOps, the paper explores the emergence of DevSecOps, an extended paradigm that integrates security throughout the development lifecycle. Within the realm of DevSecOps, the paper further examines the role of Site Reliability Engineering (SRE), emphasizing the critical intersection of security and reliability. Site Reliability Engineering (SRE) principles, rooted in Google's operational expertise, focus on maintaining scalable and highly reliable systems. Results: Expanding on these foundations, the research investigates the incorporation of Chaos Engineering into DevSecOps practices. Chaos Engineering, involving deliberate and controlled experiments to uncover vulnerabilities, is introduced as a proactive measure to increase resiliency and antifragility in software systems. Conclusion: By systematically injecting faults and simulating real-world disruptions, organizations can fortify their systems against unforeseen challenges, contributing to developing more robust and secure software ecosystems in the ever-evolving technological landscape.

Keywords: DevOps, DevSecOps, Chaos Engineering, Site Reliability Engineering, CI/CD.

2024. In Mukesh Saraswat & Rajani Kumari (eds.), *Applied Intelligence and Computing*, 9–17. Computing & Intelligent Systems, SCRS, India. DOI: https://doi.org/10.56155/978-81-955020-9-7-2

1 Introduction

The collaborative DevOps methodology, which replaced traditional approaches, represented a paradigm shift in the rapidly evolving software development industry. The software delivery industry was revolutionised by eliminating barriers, promoting collaboration, and automating processes by merging operations and development, or DevOps. Organisationshave become more cognizant of the need for security concerns as a critical part of the development lifecycle, which is a result of the advantages of improved agility and faster time-to-market. This realization led to the evolution of DevSecOps, an expanded paradigm that seamlessly integrates security across the software development process. According to DevSecOps, security should be addressed frequently and early on since it considers security an essential rather than an afterthought [1].

Building on the foundations of DevSecOps, this paper examines the vital role of Site Reliability Engineering (SRE) inside the DevSecOps framework. SRE, which bridges the gap between development and operations, is a discipline created by Google that focuses primarily on system performance, scalability, and reliability. The goals of DevSecOps align with SRE ideas, which heavily emphasise automation, monitoring, and incident response to maintain highly reliable and secure systems. Digital infrastructures are built robustly thanks to the strategic approach of strengthening software ecosystems against potential vulnerabilities and disruptions by integrating SRE principles inside DevSecOps [2].
The focus of the inquiry gradually moves to Chaos Engineering, a disruptive force essential to the DevSecOps paradigm. Chaos Engineering provides a controlled way of experimentation that enables one to examine how a system responds in unfavourable conditions by intentionally introducing faults and interruptions. Organizations can proactively uncover vulnerabilities by intentionally creating chaos, assessing the effect on security measures, and enhancing incident response procedures. DevSecOps and Chaos Engineering work together to improve the ability to build resilient systems that can develop and adapt when faced with challenges. Purposefully inducing controlled chaos measures the ability of a system to adapt, develop, and harden [3].

This in-depth analysis aims to clarify the complex links between DevSecOps, Site Reliability Engineering, and Chaos Engineering. In an era of dynamic dangers in digital landscapes, this all-encompassing approach reflects a strategic change towards proactive security and reliability solutions. To foster a culture of resilience, adaptability, and continuous improvement, the research attempts to demonstrate how companies may successfully integrate these approaches into their software development lifecycles. By understanding and leveraging the relationships between these paradigms, organizations can effectively handle the complexities of modern software development and ensure that security and reliability are essential and deeply ingrained in their digital activities [3] [22].

2 Review of Literature

System testing can be done dynamically with the help of chaos engineering. The main concept is to examine how a system behaves under difficult conditions to identify flaws before they manifest in real-world scenarios. Chaos engineering was developed in response to the inherent complexity of modern networked systems. It accepts that mistakes are inevitable and that planning and anticipating them is important. Chaotic engineering is a new field that has emerged to support cloud resiliency. Chaotic engineering is the process of evaluating distributed systems to boost confidence in their ability to withstand disruptions in production. Chaotic engineering requires experiments to validate or invalidate hypotheses [4, 5].

3 Methodology

3.1 Preliminary Research

In this preliminary research, a search string containing the keywords “DevOps,” “DevSecOps”, “Chaos Engineering”, “Site Reliability Engineering”, “CI/CD”, “Observability”, and “implementation” was conducted on IEEE Xplore, Scopus, and Google Scholar databases. Leveraging these sources, we aim to unearth a wealth of literature that sheds light on the intricate interplay between these methodologies. The goal is to identify critical insights, emerging trends, and gaps within the discourse on how these practices collectively contribute to bolstering system resiliency and nurturing antifragility. The collection of works deals with DevSecOps and Chaos Engineering from peer-reviewed journals in the application of Systematic Review. Research notes were made once a relevant subset was found.

The inclusion criteria were limited to the following keywords: “DevOps”, “DevSecOps”, “Challenges”, “Implementation”, and “Security”. A paper was not considered for evaluation if it did not directly relate to DevOps, Security, or a DevOps component like continuous integration.

For the Exclusion criteria, if the paper's title did not contain the relevant keywords or the content was unrelated to the software systems, the paper was not considered for the review [4].

3.2 Selection of Work(data) using PRISMA Approach

To execute this Literature Review, PRISMA was used to report the required references in this literature review. This approach provides a transparent and structured framework, ensuring a rigorous selection process for relevant studies. Beginning with an exhaustive search, our methodology involves carefully screening and evaluating identified articles based on predefined inclusion and exclusion criteria. This stringent process will help us filter out studies that align most closely with the focal points of our research. By adopting PRISMA, we aim to uphold methodological integrity, minimize bias, and extract meaningful insights from the selected literature, contributing to a nuanced understanding of how the amalgamation of DevSecOps and Chaos Engineering augments resiliency and antifragility in contemporary software development practices. (see Figure 1)

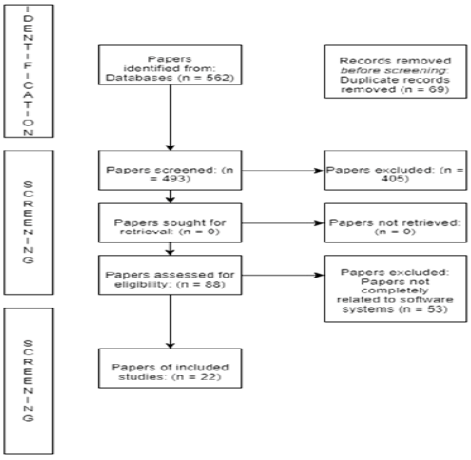

Figure 1. Flow Diagram of the PRISMA approach

Steps used in the Prisma approach:

- Title and Abstract Screening for the Keywords Searched.
- Full-Text Assessment.
- Data Extraction of the relevant Research References.
- Quality Assessment.
- Shortlist of the papers considered.
- (Total papers considered for this study after using PRISMA: 22)

Using the PRISMA methodology in the study, we expect a well-organized and reliable overview of the existing DevSecOps and Chaos Engineering research. We carefully select relevant studies from respected databases, ensuring they meet specific criteria. This method ensures that we include high-quality research and maintain a high standard of evidence. As we go through the data extraction and analysis process, the approach would help identify the findings clearly and transparently. Ultimately, this systematic review aims to provide a straightforward and evidence-based understanding of how combining DevSecOps and Chaos Engineering can make software systems more resilient and antifragile. The results will not only contribute to our current knowledge but will also guide future research and practical applications in the field of software development and security.

4 Discussion

The systematic "chaotic engineering" approach aims to increase system dependability and resilience through carefully monitored experimentation. The first important step is to thoroughly understand the system from beginning to end. Following an understanding of the system, the next step in the chaos engineering process is to develop hypotheses and educated predictions about how the system would behave under controlled disruptions. A thorough experiment plan is then created, outlining the precise chaos scenarios that will be shown. After purposefully disrupting the system during the chaotic experiments, the outcomes are carefully examined. The results are used to support or refute the original theories. The process is repeated as the explosion radius, which indicates the degree of chaos introduction, expands and confidence in the system's resilience increases. This iterative process encourages ongoing development.
Lastly, as the system matures, chaos engineering can be included in production environments to support the resilience and ongoing evolution of the system. Organisations are ensured to systematically identify vulnerabilities, fortify their systems, and prepare for issues in the actual world by using this systematic methodology [5] [6].

In Site Reliability Engineering (SRE), chaos engineering is essential because it introduces controlled disruptions into a system to highlight flaws, enhance fault tolerance, and foster resilience [18]. Chaos experiments provide a proactive approach to dependability by teaching SREs about a system's capacity and limits through deliberate stress testing and simulated failures. This iterative approach improves system design and architecture by taking lessons from both successful and bad cases [6]. Automation technologies are frequently used to mimic interruptions, which promotes the creation of reliable monitoring and alerting systems. Ultimately, chaos engineering helps organisations change their culture by reducing the likelihood of unanticipated production-related incidents and fostering confidence in a system's capacity to handle erratic, real-world conditions [12] [17].

4.1 Observability

Observability in DevSecOps refers to gaining comprehensive insights into the security aspects of the software development and delivery process. It involves monitoring and analysing various elements such as code repositories, builds, deployments, and runtime environments to ensure security measures are effectively implemented [9][15].

4.2 Case Study Implementation of Chaos Engineering

Netflix created Chaos Engineering, a field that involves controlled trials on a distributed system to increase confidence in its capacity to endure chaotic conditions in production and improve system resilience and reliability. Chaos Engineering has become a key component of Netflix's methodology, utilising ideas including creating hypotheses around steady-state behaviour, altering real-world occurrences, conducting experiments in production, and automating trials for continuous testing. (see Figure. 2) Netflix engineers intentionally introduce errors and simulate real-world disruptions to find weaknesses, guarantee gradual system deterioration, and enhance overall system resilience. Based on Netflix's experience, the following guidelines advance the subject of chaotic engineering by understanding system behaviour and encouraging a proactive approach to dependability. To provide ongoing service even in the face of unforeseen problems, the process entails identifying quantifiable steady states, formulating hypotheses, adding real-world factors, and methodically questioning system behaviour. Chaos engineering may become more widely used in various fields, which will need case studies, improved tools, and investigation of event injection models to develop further and expand this novel strategy [6].

To foster the need for availability and Resilience, Netflix introduced the Simian Army. Simian Army comprises a suite of tools deliberately designed to cause malfunctions and interruptions in Netflix's cloud-based infrastructure. By putting the system to the test in a realistic setting, the Simian Army enables engineers to find and fix flaws before they become severe problems in real-world situations [20].

Essential elements of the Simian Army are different "monkeys," each with a distinct function in emulating various failure scenarios. By erasing virtual machine instances randomly, the Chaos Monkey forces the system to adjust and bounce back from unforeseen setbacks. Latency Monkey creates delays in replicating real-world network latency situations to assess the system's performance in challenging circumstances. Conformity Monkey focuses on finding instances of not complying with best practices and security regulations to maintain a standardised and secure environment. Doctor Monkey actively monitors and resolves sick instances in the production environment to maintain the system's general health. Janitor Monkey performs a cleanup function by locating and eliminating zombie instances and underutilised resources. Security Monkey is committed to examining policy infractions and flaws, offering crucial information for preserving environmental security.Last, Chaos Gorilla tests the system's resilience to significant disruptions by mimicking the outage of an entire Amazon availability zone. Collectively, these primates constitute an all-encompassing toolkit for chaotic engineering, cultivating an anticipatory approach to system evaluation and ongoing enhancement within Netflix's operational framework [21].

Businesses across all sectors, particularly the IT industry, must change their business strategies to increase productivity and high-quality deliverables in response to technological advancements. Customers expect software programs to be more accessible and user-friendly to meet their evolving needs. Because this might result in numerous errors, it is difficult for software companies to produce and deploy their products frequently enough to meet the expectations of their customers. Customers will be unhappy, and substantial losses in money, time, and resources will result [7].

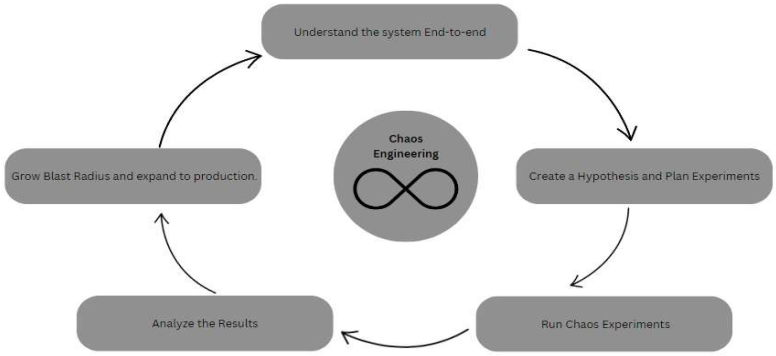

Figure 2. - Chaos Engineering Cycle

Chaotic engineering and continuous integration and deployment (CI/CD) strengthen software development and delivery pipelines. Organisations can proactively evaluate their systems' behaviour when faced with challenging circumstances by introducing defects, failures, or interruptions into the CI/CD processes. By implementing this approach, teams may find vulnerabilities, improve fault tolerance, and ensure the CI/CD pipeline can manage unforeseen difficulties gracefully. In continuous integration and delivery (CI/CD), chaos engineering entails designing controlled experiments that emulate real-world situations. This enables development and operations teams to verify the resilience and dependability of their automation workflows. Ultimately, CI/CD processes that use chaotic engineering concepts help create more trustworthy, resilient, and adaptive software delivery pipelines [8].

To execute the Chaos attack (Gremlin attacks) in CI/CD, we can use Gremlin API. The following steps can be followed to execute the attack.

To post attacks, use CURL.

Block and poll our observability tools (like Datadog or New Relic) and the Gremlin API to find out about the current state of our environment and the attack's progress while it's ongoing.

We fail the build if the observability tool's API returns results higher than our predetermined threshold. We fail the build if the attack enters one of the failure scenarios [16].

5 Implications

5.1 Chaos in Operation

Chaos in operations refers to deliberately introducing controlled disruptions or unpredictable events within an organisational framework to assess the system's resilience, identify vulnerabilities, and improve overall operational robustness. This practice, often associated with disciplines like Chaos Engineering, is employed to proactively test how well operations can adapt and recover from unexpected challenges, such as system failures or sudden increases in demand. Organisations gain valuable insights by intentionally creating chaos scenarios and analysing the system's responses,

enabling them to enhance their operational procedures, fortify critical processes, and foster a culture of adaptability and continuous improvement in the face of unforeseen challenges [9].

5.2 Chaos in the Cloud

According to Basiri et al., chaos engineering is the practice of testing a distributed cloud system to increase confidence in its ability to survive chaotic conditions in production. Experiments to support or refute theories are important to chaotic engineering. In this context, a hypothesis is a system's anticipated or presumptive behaviour under conditions. In chaotic engineering experiments, theories are examined by introducing disturbances, such as malfunctions, in authentic scenarios and monitoring the system's behaviour. The observed behaviour is fresh knowledge since it provides insights into how the system will fail or withstand (confirm or disprove the defined hypotheses) [13]. On the other hand, modern chaotic engineering approaches concentrate on availability experiments, in which theories are based on availability characteristics like latency. We think security-oriented conjectures are likewise feasible and highly advantageous to security experts. Moreover, resilience is essential to security (confidentiality and integrity) and availability. Consequently, the groundwork for these linkages is established in the following subsections [10] [11].

6 Role of Chaos in SRE

Chaos Engineering plays a vital role in Site Reliability Engineering (SRE) by introducing controlled disruptions into a system, aiming to identify weaknesses, improve fault tolerance, and build resilience [18]. Through intentional stress testing and simulated failures, chaos experiments help SREs understand a system's limits and capacity, fostering a proactive mindset toward reliability. This iterative process involves learning from both successful and unsuccessful scenarios, driving improvements in system design and architecture [6]. Automation tools are often utilized to simulate disruptions, encouraging the development of robust monitoring and alerting systems. Ultimately, chaos engineering contributes to a cultural shift within organizations, instilling confidence in a system's ability to handle real-world, unpredictable conditions and mitigating the risk of unexpected outages in production [12] [17].

7 Challenges in Chaos Engineering

Implementing chaos engineering in organizations presents several challenges that need careful consideration. First, there is frequently cultural opposition since teams may be reluctant to completely embrace the technique out of concern of purposefully introducing mistakes. Resource limitations are another major issue because conducting chaotic experiments well requires a lot of staff, time, and equipment. Because of the complexity of implementation, especially in large, distributed systems, careful preparation and specialised knowledge are required to guarantee the safety and control of experiments. Additionally, there is an inherent risk in chaos engineering because ill-thought-out experiments may cause unintentional outages or system deterioration, therefore it is important to carefully weigh the risks and rewards. It can be challenging to gauge how chaotic engineering affects system resilience and reliability; successful measurement and monitoring are necessary. Finally, in order to ensure that chaotic engineering enhances rather than interferes with present workflows, integrating it with current DevOps and security approaches necessitates team alignment and collaboration. These difficulties highlight the necessity of careful planning, effective resource allocation, and open communication when implementing chaos engineering techniques inside a company [8, 14].

8 Conclusion

This literature analysis shows how DevOps evolved into DevSecOps, emphasising integrating Chaos Engineering to improve resilience and antifragility. Software development underwent a paradigm shift byadopting collaborative DevOps over traditional techniques, emphasising automation and agility. DevSecOps, an expanded paradigm that seamlessly integrates security throughout the development lifecycle, arose from recognisingsecurity's vital role. This paradigm is further strengthened by Site Dependability Engineering (SRE), which is based on Google's experience and emphasises system performance and dependability through automation and monitoring. A proactive experimentation method is introduced by incorporating Chaos Engineering, whereby errors are purposefully introduced to reveal vulnerabilities and improve system robustness. As we go through the data extraction and analysis process, the approach would help identify the findings clearly and transparently. Ultimately, this systematic review aims to provide a straightforward and evidence-based understanding of how combining DevSecOps and Chaos Engineering can make software systems more resilient and antifragile. The results will not only contribute to our current knowledge but will also guide future research and practical applications in the field of software development and security.

Reference

[1] X. Ramaj, "A DevSecOps-enabled Framework for Risk Management of Critical Infrastructures," 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Pittsburgh, PA, USA, 2022, pp. 242-244, doi: 10.1145/3510454.3517053.

[2] Sandeep Madamanchi, Google Cloud for DevOps Engineers: A practical guide to SRE and achieving Google's Professional Cloud DevOps Engineer certification, Packt Publishing, 2021.

[3] A. Basiri et al., "Chaos Engineering," in IEEE Software, vol. 33, no. 3, pp. 35-41, May-June 2016, doi: 10.1109/MS.2016.60.

[4] D. Anjaria, Mugdha Kulkarni, “Effective DevSecOps Implementation: A Systematic Literature Review”, Cardiometry, 2022.

[5] M. Arsecularatne and R. Wickramarachchi, "The Adoptability of Chaos Engineering with DevOps to Stimulate the Software Delivery Performance: A Systematic Literature Review," 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), Lonavla, India, 2023, pp. 1-5, doi: 10.1109/I2CT57861.2023.10126414.

[6] H. Tucker, L. Hochstein, N. Jones, A. Basiri and C. Rosenthal, "The Business Case for Chaos Engineering," in IEEE Cloud Computing, vol. 5, no. 3, pp. 45-54, May./Jun. 2018, doi: 10.1109/MCC.2018.032591616.

[7] D. M. Shawky, "Traditional vs Agile development a comparison using chaos theory," 2014 9th International Conference on Software Paradigm Trends (ICSOFT-PT), Vienna, Austria, 2014, pp. 109-114.

[8] M. Shahin, M. Ali Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," in IEEE Access, vol. 5, pp. 3909-3943, 2017, doi: 10.1109/ACCESS.2017.2685629.

[9] P. Dedousis, G. Stergiopoulos, G. Arampatzis and D. Gritzalis, "Enhancing Operational Resilience of Critical Infrastructure Processes Through Chaos Engineering," in IEEE Access, vol. 11, pp. 106172-106189, 2023, doi: 10.1109/ACCESS.2023.3316028.

[10] S. De, "A Study on Chaos Engineering for Improving Cloud Software Quality and Reliability," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 2021, pp. 289-294, doi: 10.1109/CENTCON52345.2021.9688292.

[11] K. A. Torkura, M. I. H. Sukmana, F. Cheng and C. Meinel, "CloudStrike: Chaos Engineering for Security and Resiliency in Cloud Infrastructure," in IEEE Access, vol. 8, pp. 123044-123060, 2020, doi: 10.1109/ACCESS.2020.3007338.

[12] Deep Manishkumar Dave, ” Impact of Site Reliability Engineering on Manufacturing Operations: Improving Efficiency and Reducing Downtime”, International Journal of Scientific and Research Publications, Volume 13, Issue 11, November 2023, ISSN 2250-3153, DOI: 10.29322/IJSRP.13.11.2023.p14312

[13] F. Poltronieri, M. Tortonesi and C. Stefanelli, "A Chaos Engineering Approach for Improving the Resiliency of IT Services Configurations," NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022, pp. 1-6, doi: 10.1109/NOMS54207.2022.9789887.

[14] E. Zio, "Some Challenges and Opportunities in Reliability Engineering," in IEEE Transactions on Reliability, vol. 65, no. 4, pp. 1769-1782, Dec. 2016, doi: 10.1109/TR.2016.2591504.

[15] I. Siddiqui, A. Pandey, S. Jain, H. Kothadia, R. Agrawal and N. Chankhore, "Comprehensive Monitoring and Observability with Jenkins and Grafana: A Review of Integration Strategies, Best Practices, and Emerging Trends," 2023 7th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkiye, 2023, pp. 1-5, doi: 10.1109/ISMSIT58785.2023.10304904.

[16] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter and V. Sekar, "Gremlin: Systematic Resilience Testing of Microservices," 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), Nara, Japan, 2016, pp. 57-66, doi: 10.1109/ICDCS.2016.11.

[17] Srikarthick Vijaykumar, "Site Reliability Engineering (SRE)", EJIAR, vol. 2, no. 2, pp. 6–11, Apr. 2023.a

[18] Petersson, L, "An Empirical Investigation of the Acceptance of Chaos Engineering (Dissertation)." 2022.

[19] "Damian A. Tamburri, Marcello M. Bersani, Raffaela Mirandola, Giorgio Pea" DevOps Service Observability By-Design: Experimenting with Model-View-Controller

[20] Service-Oriented and Cloud Computing, 2018, Volume 11116

[21] Bailey, T., Marchione, P., Swartz, P., Salih, R., Clark, M. R., & Denz, R. (2022, May). Measuring the resiliency of systems using chaos engineering experiments. In Disruptive Technologies in Information Sciences VI (Vol. 12117, pp. 20-32). SPIE.

[22] Monge Solano, I., & Matók, E. (2020). Developing for Resilience: Introducing a Chaos Engineering tool.

[23] S. Shariah and A. Ferworn, "Securing APIs and Chaos Engineering," 2021 IEEE Conference on Communications and Network Security (CNS), Tempe, AZ, USA, 2021, pp. 290-294, doi: 10.1109/CNS53000.2021.9705049.