The American Journal of Engineering and Technology
ISSN 2689-0984 | Open Access

TYPE Original Research
PAGE NO. 240-247
DOI 10.37547/tajet/Volume07Issue06-25

Check for updates

The Role of Chaos Engineering in DevSecOps Strengthening Security and Compliance in Agile

Dhanasekar Elumalai
Fidelity investments, USA

OPEN ACCESS

SUBMITED 17 April 2025
ACCEPTED 24 May 2025
PUBLISHED 30 June 2025
VOLUME Vol.07 Issue 06 2025

CITATION

Dhanasekar Elumalai. (2025). The Role of Chaos Engineering in DevSecOps Strengthening Security and Compliance in Agile. The American Journal of Engineering and Technology, 7(06), 240–247. https://doi.org/10.37547/tajet/Volume07Issue06-25.

Abstract: Recently, the DevSecOps practice has improved companies' agile development of secure software, reducing problems and improving return on investment. However, overreliance on security tools and traditional security techniques can facilitate the implementation of vulnerabilities in different stages of the software lifecycle. The evolution, principles, and importance of DevSecOps in contemporary software engineering. DevSecOps arises from the recognition that traditional security measures often lag the rapid pace of DevOps development cycles, leading to vulnerabilities and breaches. By integrating security early and continuously throughout the software development lifecycle, DevSecOps aims to proactively identify and mitigate risks without impeding the agility and speed of DevOps practices. The core principles of DevSecOps, emphasizing automation, collaboration, and cultural transformation. Automation streamlines security processes, enabling the automated testing and validation of code for vulnerabilities. Collaboration fosters communication and shared responsibility among developers, operations, and security teams, breaking down silos and promoting a collective approach to security. Cultural transformation involves cultivating a security-first mindset across the organization, where security is not an inherent part of the development process. The importance of DevSecOps cannot be overstated in today's digital landscape, where cyber threats are omnipresent, and the cost of security breaches is staggering. By integrating security into every stage of the DevOps pipeline, organizations can enhance their resilience to cyber-attacks, comply with regulatory requirements, and build trust with customers. DevSecOps represents a holistic approach to software

development that prioritizes security without compromising speed or innovation. Embracing DevSecOps principles is imperative for organizations seeking to stay ahead in complex and hostile digital environment.

KEYWORDS

DevSecOps; Security; Practices; Emergence; DevOps; Review.

INTRODUCTION

DevSecOps is a methodology that integrates security practices into the DevOps process, ensuring security measures are implemented and maintained throughout the entire software development lifecycle. It emphasizes collaboration, automation, and cultural transformation to enable faster delivery of secure and reliable software. The DevOps pipeline is set of practices and tools used to automate the software delivery process, from code development to deployment and operations. It typically includes stages such as coding, building, testing, deployment, monitoring, and feedback. The goal of the DevOps pipeline is to streamline and accelerate the delivery of high-quality software by breaking down silos between development and operations teams and promoting continuous integration and continuous delivery (CI/CD) practices. Traditionally, security has been treated as separate phase in software development lifecycle, added as afterthought by separate security team. This approach can lead to security vulnerabilities and risks being overlooked too late in development process, resulting in costly and time-consuming security breaches. With increasing frequency and sophistication of cyber threats, organizations no longer afford to treat security as isolated concern. DevSecOps acknowledges importance of security in software development and address vulnerabilities and security risks early and continuously throughout the DevOps pipeline. By integrating security in every stage of development process, DevSecOps helps organizations mitigate security threats, comply with regulatory requirements, and build trust with customers.

1. Principles of DevSecOps

DevSecOps emerged as response to limitations of traditional security approaches in fast-paced DevOps environments. The evolution of DevSecOps be traced back to early efforts to integrate security into DevOps pipeline, security-focused DevOps tools and practices. Key milestones include publication of influential papers and frameworks advocating DevSecOps principles, as the development of specialized security tools and technologies to support DevSecOps practices. There is significant increase in adoption of DevSecOps practices across industries, driven by factors such as rise of cloud computing, the proliferation of cyber threats, and growing regulatory pressure ensuring security of software systems. Organizations are increasingly recognizing benefits of DevSecOps in improving security posture, faster time-to-market, and greater operational efficiency. As a result, DevSecOps is becoming increasingly mainstream, with many organizations integrating security into DevOps pipelines and a culture of security-first development [1].

Automation a crucial role in DevSecOps enabling rapid and consistent testing and validation of security measures throughout development process. Automated security tests help identify vulnerabilities early in development lifecycle, allowing teams address promptly and reduce risk of security breaches. These include static code analysis tools for scanning code for potential security issues, dynamic application security testing (DAST) tools simulating real-world attacks, and container security tools scanning container images for vulnerabilities. DevSecOps promotes collaboration and communication between traditionally siloed development, operations, and security teams. Breaking down these silos, teams share insights, expertise, and responsibilities, leading to better alignment and coordination addressing security concerns. Cross functional collaboration involves bringing individuals together from different disciplines, as development, operations, security, and quality assurance, working together towards common security goals. This collaboration fosters understanding of security requirements and challenges across teams, leading to effective security measures. DevSecOps requires a cultural shift prioritizing security throughout the software development lifecycle. This involves a mindset where security is considered fundamental aspect of software development, rather a separate concern. In DevSecOps, security is everyone's responsibility, not responsibility of security team. Fostering a culture of shared responsibility involves empowering developers,

operators, and other stakeholders taking ownership of security tasks and make security-conscious decisions in respective roles.

2. Integrating Security Practices into DevOps Pipeline

Providing developers with training and resources on secure coding practices helping to write code that is resilient to security threats. This includes educating developers on common vulnerabilities, secure coding guidelines, and best practices for writing secure code. Establishing and enforcing secure coding guidelines and standards ensures consistency and adherence to security best practices across development teams. These guidelines cover areas as input validation, authentication, authorization, and data encryption. Integrating security testing into CI/CD pipelines allowing for automated and continuous validation of security controls throughout the software delivery process. This includes running security tests as static code analysis, dynamic application security testing (DAST), and dependency scanning the automated build and deployment process. Static analysis, dynamic analysis, penetration testing, etc.

Security testing encompasses various techniques and methodologies for identifying and assessing security vulnerabilities in software applications [4]. This includes static analysis, examining code for security flaws executing it, dynamic analysis, which tests application runtime environment, and penetration testing, simulating real-world attacks uncovering vulnerabilities. Infrastructure as Code involves managing and provisioning infrastructure using code and automation tools. By implementing security controls directly in infrastructure code, the configuration files and scripts, organizations ensure security measures are consistently applied across environments. Enabling organizations to define and deploy infrastructure configurations in repeatable and consistent manner, reducing risk of configuration drift and misconfigurations leading security vulnerabilities. By treating infrastructure as code, organizations apply same versioning, testing, and review processes to infrastructure changes as they do for application code.

DevSecOps emphasizes integration of security practices throughout software development lifecycle, enabling teams to detect and address security vulnerabilities early on. By incorporating automated security testing and continuous monitoring into development pipeline, DevSecOps facilitates proactive identification of vulnerabilities, allowing organizations fixing issues before they being exploited by attackers. This proactive approach helps overall security posture of software systems, reducing the risk of potential security breaches.

3. **Reduced attack and likelihood of breaches**:

Through implementation of DevSecOps practices, organizations minimize their attack surface potential points of entry for cyber threats. Integrating security measures such secure coding practices, vulnerability scanning, and access controls into development process, DevSecOps reduces likelihood of successful attacks. This reduction in attack surface area, coupled with proactive identification and mitigation of vulnerabilities, lowers risk of security breaches and data compromises. DevSecOps assists organizations meeting regulatory standards and compliance mandates embedding security into development process. Regulations such as GDPR, HIPAA, PCI DSS, and other organizations implement specific security measures protecting sensitive data and ensure privacy and confidentiality. DevSecOps facilitates compliance with regulations by automating security controls, conducting regular security assessments, and documenting security measures, thereby reducing risk of non-compliance and associated penalties. By incorporating security into every stage of software development lifecycle, DevSecOps enables organizations demonstrate their commitment to security best practices. This includes industry recognized security frameworks NIST Cybersecurity Framework or CIS Controls adhering to established security guidelines and standards [4].

By demonstrating these best practices, organizations build trust with customers, partners, and regulatory bodies, showcasing dedication protecting sensitive information and mitigating security risks. DevSecOps play crucial role building trust with customers and stakeholders ensuring security and reliability of software products and services. Prioritizing security throughout development process, organizations demonstrate their commitment to protecting customer data and sensitive information. This fosters trust and confidence in organization's ability delivering secure and dependable

solutions, enhancing customer satisfaction and loyalty. In the security incident, organizations have implemented DevSecOps practices to better equip responding effectively, minimizing impact on reputation and brand. Proactively identifying and addressing vulnerabilities, organizations reduce likelihood and severity of security incidents. Additionally, maintaining transparent communication and demonstrating commitment to remediation and improvement, organizations mitigate reputational damage and rebuild trust with customers and stakeholders, ultimately preserving their reputation and credibility in market.

How can an organization successfully integrate a DevOps environment and a chaos engineering methodology to increase enterprise availability through increased resilience and reliability? The central question to a broad and general question addition sub-questions could be posed. The following sub-questions relate aspects of integration of DevOps or DevSecOps environment with chaos engineering methodology supporting gathering data:

1. How does the implementation of chaos engineering methodologies augment the DevOps/DevSecOps environment providing available, robust, and reliable systems?

2. What is the intersecting and overlapping areas between chaos engineering and DevOps/DevSecOps that provide the most synergy to improve system availability

Several research methods including questioning, data collection, data analysis, interpretation, and validation study focused on understanding how integration of distinct methodologies of chaos engineering and DevOps could improve software system resiliency and reliability. Although DevOps is relatively new methodology, is far more mature in development and industry adoption than chaos engineering, allowing identifying practices leading to improved system reliability, resilience, and availability. The evaluation of DevOps best practices from a chaos engineering perspective, together with examination of current processes and methodologies in place, provided perspective to effective method and criteria for selection of semi-structured interview questions elaborating the conceptual framework. Both DevOps and chaos engineering disciplines have processes and tools, to purpose the synergetic elements between DevOps and chaos engineering leading to greater systematic availability and robustness either methodology acting independently.

4. The Importance of Chaos Engineering in DevSecOps

Chaos engineering promotes a security-focused culture enhancing collaboration across development, operations, and security teams throughout the software development lifecycle (SDLC). This proactive approach ensures that security prioritized at every stage, thus reducing vulnerabilities and risks in DevSecOps [6].

Advantages of integrating Chaos engineering in Devsecops

4.1. Proactive Vulnerability Management: By continuously testing systems under various failure scenarios, organizations can identify and remediate security vulnerabilities before they can be exploited during a real incident.

4.2. Enhanced Compliance: DevSecOps emphasizes regulatory compliance, ensuring that chaos testing adheres to necessary security benchmarks and standards.

4.3. Continuous Monitoring: Real-time insights from chaos tests enable ongoing monitoring of security controls' effectiveness, helping organizations adapt and respond to emerging threats promptly.

5.1.1. Collaboration and Knowledge Sharing: By bringing together development, security, and operations teams, DevSecOps fosters a culture of collaboration and knowledge transfer, allowing for the sharing of best practices in chaos testing [2].

Tools and Technologies for DevSecOps in Chaos Testing

Implementing the checks outlined above requires the right set of tools. Below are some commonly used technologies that can assist senior SREs integrating DevSecOps practices into chaos testing:

1. Chaos Testing Tools

- **Gremlin:** An advanced chaos engineering platform that allows users to create and run chaos experiments with ease. It includes built-in safety features to manage risks effectively.
- **Chaos Monkey:** Part of the Simian Army, Chaos Monkey randomly terminates instances ensuring that applications can tolerate instance failures.
- **Litmus:** An open-source chaos engineering platform provides ability creating and managing chaos experiments across various environments.

Security Tools

- **OWASP ZAP:** An open-source DAST tool for identifying vulnerabilities in applications during chaos tests. It can be integrated into CI/CD pipeline to automate assessments [4].
- **Snyk:** A developer-centric tool focused identifying and fixing vulnerabilities within open-source libraries and containers.
- **Aqua Security:** Focuses on cloud-native security, providing scanning and security measures for containers and serverless applications.

5.1.2. Monitoring and Logging Tools

- **Splunk:** A powerful tool for monitoring, logging, and analyzing machine-generated data, helping teams detect threats and anomalies during chaos engineering experiments.
- **ELK Stack (Elasticsearch, Logstash, Kibana):** A popular open-source solution for collecting, storing, and analyzing log data from different sources.
- **Prometheus:** A monitoring system designed for reliability and scalability, Prometheus collects metrics and offers powerful querying capabilities.

Automation Tools

- **Jenkins:** A widely used CI/CD tool allowing for automation of deployment processes, including running security checks before chaos experiments [5].
- **GitHub Actions:** Automate workflows right in GitHub repositories, easily integrating testing and security processes into the version control system.
- **Terraform:** Infrastructure as code tool that allows for consistent and secure configuration, enabling automated deployment of infrastructure environments for chaos tests.

Major cloud providers, such as AWS, also offer chaos engineering tools, including AWS Fault Injection Simulator and AWS Resilience Hub. Additionally, the integration of security orchestration tools for chaos testing enhances the ability simulating real-world cyber threats, further strengthening the resilience of systems in unpredictable conditions.

These tools primarily aim to use chaos engineering to avert availability failures. However, despite its potential advantages, the security sector has yet to fully embrace chaos engineering, even though its principles could offer significant benefits for enhancing cybersecurity.

As organizations continue to realize value of proactive cybersecurity, many are turning to Managed Security Chaos Engineering services**.** These services provide structured, expert-led approach to simulating real-world security threats, helping organizations uncover vulnerabilities before they become a problem.

By utilizing Managed Security Chaos Engineering services, businesses ensure that their systems are tested thoroughly and consistently, resulting in improved resilience and a more robust security framework.

5. ChaosSecOps in Action on AWS

An e-commerce platform on AWS experiences performance slowdowns and is potential security vulnerabilities. The platform handles large volume of transactions and stores sensitive customer data. They want to improve the resilience and security of their system using ChaosSecOps [3].

1 Data Flow:

"Product Listing" Request The following describes the successful flow of a user request to view a list of products. It assumes no security issues are detected, and no errors occur.

1. Internet (User clicks 'View Products'): The process begins with a user action: clicking a link or button on the e-commerce website to view products (e.g., browsing a category, performing a search). This generates an HTTP (or HTTPS) request.

2. AWS WAF (Checks for malicious requests): The user's request first encounters the AWS Web Application Firewall. The WAF analyzes the request, looking for patterns that indicate malicious activity (e.g., SQL injection attempts, cross-site scripting payloads, known attack signatures). In this successful flow, the WAF determines the request is legitimate and allows it to pass.

3. API Gateway (Routes to Lambda_PL): The request, now cleared by the WAF, reaches the AWS API Gateway. The API Gateway acts as a reverse proxy. Based on the request's URL, HTTP method, and potentially other headers, it determines the appropriate backend service to handle the request. In this flow, the API Gateway identifies the request as one for [3] product listing and routes it to the Lambda_PL function (the Lambda function specifically designed for handling product listing requests).

4. AWS Lambda (Product Listing - Lambda_PL) (Prepares DB query): The Lambda_PL function is invoked by the API Gateway. The request data (e.g., query parameters like category, keywords, filters) is passed to the Lambda function as input. The Lambda function's code contains the logic to process this request. It prepares a SQL query to retrieve the relevant product information from the database. This query will likely include WHERE clauses based on the user's request (e.g., WHERE category = 'shoes' AND color = 'red'). The query is not executed within the Lambda function itself; rather the Lambda constructs the query string.

5. AWS RDS (PostgreSQL - Read Replica) (Executes query, returns data): The Lambda_PL function establishes a connection to the Read Replica of the AWS RDS PostgreSQL database. Using a Read Replica for read-only operations like product listing is a best practice for scalability and performance. The Lambda function sends the prepared SQL query to the Read Replica. The Read Replica executes the query against the product catalog data. The Read Replica returns the results of the query (a set of product data matching the criteria) to the Lambda_PL function.

6. Lambda_PL (Formats data): The Lambda_PL function receives the raw data from the database. It then formats this data into a suitable response format for the client (usually JSON). This might involve structuring the data, adding additional fields, or converting data types. The Lambda_PL function returns the formatted JSON response to the API Gateway.

7. API Gateway --> WAF: The API Gateway sends the response back towards the user. The response passes back through the WAF.

8. WAF --> Internet (User receives product listing): The WAF, having already vetted the initial request, typically allows the response to pass through without modification (unless specific response filtering rules are configured, which is less common). The response goes to the internet. The user's web browser receives the JSON response. The browser's JavaScript code (or the mobile app) renders this JSON data into a visually appealing product listing (images, names, prices, etc.). The user sees the list of products.

9. AWS CloudWatch: Throughout the entire process, CloudWatch is continuously collecting metrics and logs from all the involved AWS services (WAF, API Gateway, Lambda, RDS). This data is crucial for monitoring performance, identifying bottlenecks, and detecting errors.

7. Management Services Flow: Supporting Processes

1. IAM: Each AWS service (Lambda, API Gateway, RDS) operates with specific IAM roles that grant it the necessary permissions to perform its tasks (e.g., Lambda has permission to read from RDS).

2. Guard Duty: Provides real-time threat detection - monitors for suspicious activities and unusual patterns across the environment.

3. CloudTrail: Records all AWS API activity - creates an

audit trail of who did what and when for compliance and security analysis.

4. CI/CD Pipeline: Automates deployment - handles code testing, security validation, and controlled releases, including pre-deployment chaos testing to verify system resilience.

These background services don't handle user requests directly but form the operational foundation that keeps the main application secure, compliant, and regularly updated.

8. DevOps Toolchain

CI/CD: AWS Code Pipeline: Orchestrates the build, test, and deployment process. Used to integrate chaos experiments as a stage.

AWS Code Build: Compiles the code for the Lambda_PL function (and other components, even if not directly used in this flow).

AWS Code Deploy: Deploys the Lambda_PL function (and other components) to the appropriate AWS environments (staging, production).

Configuration Management:

AWS CloudFormation: Defines and provisions the AWS infrastructure (Lambda, RDS, API Gateway, WAF, etc.) as code. Crucial for ensuring consistent environments for testing.

Ansible: (Optional - include if used for any configuration management tasks within instances, e.g., configuring the PostgreSQL database).

Monitoring: AWS CloudWatch: Collects metrics and logs from all the relevant AWS services (Lambda, RDS, API Gateway, WAF).

Prometheus: (Optional - include if used for additional monitoring, especially if you have a hybrid environment or use it with Kubernetes).

Grafana: (Optional - include if used for visualizing Prometheus metrics).

Security Scanning: AWS Inspector: Automated security assessments for AWS resources (used to establish baseline security posture).

SonarQube: Static code analysis for the Lambda_PL code (and other codebases). Integrated into Code Build. OWASP ZAP: Dynamic application security testing, used to simulate attacks against the WAF (in the staging environment).

Chaos Engineering: Gremlin: Used for conducting the chaos experiments (RDS failover, Lambda resource exhaustion, WAF testing).

Continuous Integration: Integrated the RDS failover and Lambda resource exhaustion experiments into the Code Pipeline as post-deployment steps in the staging environment.

9. CONCLUSION

DevSecOps represents holistic approach to software development integrating security practices seamlessly into the DevOps pipeline [1]. Prioritizing security, fostering collaboration across teams, and embracing automation and cultural transformation, organizations enhance their security posture and resilience to cyber threats. In today's digital landscape, cyber threats are increasingly sophisticated and regulatory requirements becoming more stringent, embracing DevSecOps principles and practices essential for organizations to protect sensitive data, ensure regulatory compliance, and maintain trust and credibility with customers and stakeholders. Organizations encourage to invest in training, tools, and technologies effectively implement DevSecOps and stay ahead of emerging threats and regulatory requirements. By integrating security into every aspect of software development lifecycle and fostering culture of security awareness and collaboration, organizations can build robust and secure software solutions meeting the demands of today's dynamic and evolving threat landscape.

REFERENCES

Gupta, Subrat. The Art of DevOps Engineering. Subrat Gupta, 2024.

Runsewe, Oluwayemisi, Olajide Soji Osundare, Samuel Olaoluwa, and Lucy Anthony Akwawa Folorunsho. "End-to-End systems development in agile environments: Best practices and case studies from the financial sector." International Journal of Engineering Research

and Development 20, no. 08 (2024): 522-529.

Merkow, Mark. Secure, resilient, and agile software development. Auerbach Publications, 2019.

Abrahams, T.O., Ewuga, S.K., Kaggwa, S., Uwaoma, P.U., Hassan, A.O. and Dawodu, S.O., 2023. Review of strategic alignment: Accounting and cybersecurity for data confidentiality and financial security

Akbar, M.A., Smolander, K., Mahmood, S. and Alsanad, A., 2022. Toward successful DevSecOps in software development organizations: A decision-making framework. Information and Software Technology, 147, p.106894.

Rosenthal, C., Jones, N., & Allspaw, J. (2020). Chaos Engineering: System Resiliency in Practice. O'Reilly Media.