Frontiers in Computer Science and Information Technology (FCSIT)
Volume 5, Issue 2, July-December 2024, pp. 17-37, Article ID: FCSIT_05_02_003
Available online at https://iaeme.com/Home/issue/FCSIT?Volume=5&Issue=2
Journal ID: 1663-8854
DOI: https://doi.org/10.34218/FCSIT_05_02_003

DEVSECOPS IN HEALTHCARE: BUILDING SECURE AND COMPLIANT PATIENT ENGAGEMENT APPLICATIONS

Anjan Gundaboina
Senior DevsecOps and Cloud Architect, USA.

ABSTRACT

At present, the health- care industry is experiencing a dynamic shift mainly characterised by the use of modern technologies in developing patients' interaction. Nevertheless, the nature of all the accumulated data in healthcare is rather sensitive, and the demands of current legislation like HIPAA or GDPR require that the software be adequately secure and fully compliant. DevSecOps, a practice incorporating security throughout the SDLC, offers a solution to the healthcare industry that DevOps could not. The following work focuses on the effects of DevSecOps during patient engagement application development with enhanced security and compliance. In this paper, such basic concepts as security automation, continuous compliance checking, and code security are explained in detail to outline a four-step methodology applicable to healthcare. The work includes case studies, risk studies, implementation issues, and comparing traditional DevOps and DevSecOps. Hence, there is a focus on SAST tools, DAST tools, container security, and IaC scanning. Last, it emerges that security should be imbued as a foundational mindset and that development teams should be constantly trained. As the findings suggest, adopting DevSecOps also reduces risks for the organization. It furthers compliance and proper security while at the same time

improving the speed and stability of software delivery, making this approach critical to the healthcare sector.

Keywords: DevSecOps, Healthcare Applications, Security Automation, Compliance, Patient Engagement, HIPAA, GDPR.

Cite this Article: Anjan Gundaboina. (2024). DevSecOps in Healthcare: Building Secure and Compliant Patient Engagement Applications. *Frontiers in Computer Science and Information Technology (FCSIT)*, 5(2), 17–37.

https://iaeme.com/MasterAdmin/Journal_uploads/FCSIT/VOLUME_5_ISSUE_2/FCSIT_05_02_003.pdf

1. Introduction

1.1. Importance of DevSecOps in Healthcare

Several issues make application development challenging in the healthcare industry, security and compliance included. These are significant challenges due to the nature and importance of patient information, which can only be exchanged, shared, stored and analysed in compliance with the more stringent norms like HIPAA and GDPR (Health Insurance Portability and Accountability Act and General Data Protection Regulation, respectively). [1-4] Several challenges are associated with the current approach in software development; DevSecOps, which focuses on software security during the software development SDLC, is an innovative approach to overcoming these challenges. When introduced at the initial stage and integrated throughout the application development process, DevSecOps increases the chances of delivering secure and compliant healthcare applications. Based on these factors, the following are why DevSecOps is relevant in the healthcare sector.

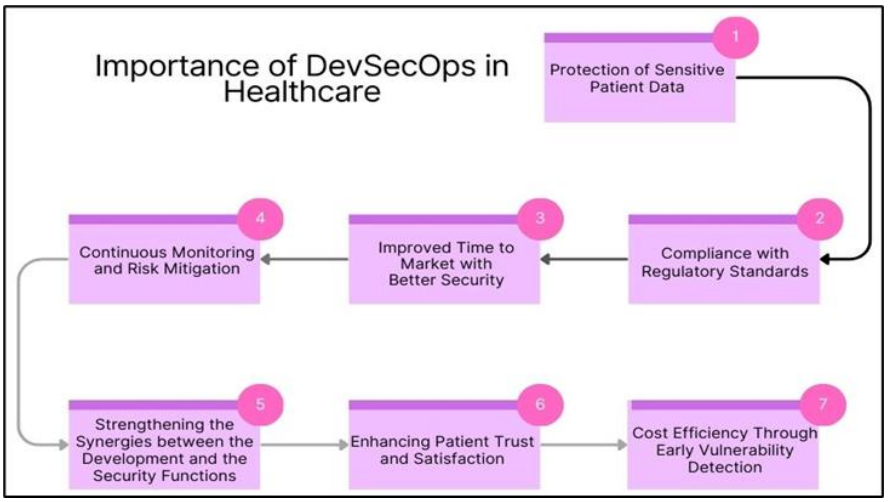

Fig 1. Importance of DevSecOps in Healthcare

- **Protection of Sensitive Patient Data:** As in any other field, patient information privacy is essential in the healthcare system. Personal records, diagnosis, disease states, and organizational financials are some of the heightened information data that healthcare systems contain and thus are vulnerable to psychopathic attacks. Traditional security models of computing, which are achieved through implementing a security layer at the end of system development, are potential security threats. DevSecOps means that security measures, such as encryption, accessing control, and vulnerability scanning, must be implemented from the development side. These safeguards addressed in this guide prevent data loss and privacy invasion, ensuring patients' confidentiality and confidence in healthcare facilities.
- **Compliance with Regulatory Standards:** In this emerging sector, to address the needs of patients, technological solutions and strict rules and requirements like HIPAA, GDPR, and PCI-DSS have to be followed. These regulations require that healthcare providers keep patient information and ensure that information management is open. They are also punishable by fines, litigation, and damage to the company's image. Through DevSecOps processes, it is easy to incorporate security compliance while designing, coding, testing, and even deploying a particular software solution since the latter flows relatively in a structured pipeline. Thus, through compliance automation, DevSecOps optimizes compliance throughout the SDLC, lowering the probability of compliance issues on authorized healthcare applications and improving the compliance assessment processes.
- **Improved Time to Market with Better Security:** The healthcare industry is bearing the roller-coaster of change in applications that can be implemented to enhance patient experience, interaction, and organisational effectiveness. Albeit, the deployment time of these applications should not be affected at the expense of a hazard-free environment. DevSecOps involves all teams in development, security, and operations and ensures security becomes part of the process incorporated from the word go. This speeds up the development and shortens the time to market while overshadowing some essential security measures within the application. DevSecOps, therefore, enables quick deployment of safe applications, rolling out better security measures and, therefore, continual testing into the CI/CD cycle.
- **Continuous Monitoring and Risk Mitigation:** The healthcare industry is constantly developing new security threats and methods of threats, which are getting more

frequent. One advantage of DevSecOps is that it monitors and alerts constantly rather than acting merely as a final safety net. By implementing security as code early in the development life cycle, DevSecOps helps to identify and mitigate security issues in operational environments. Tools such as SAST and DAST scan codes for vulnerabilities constantly, while runtime protection pinpoints threats in runtime environments. This systematic approach helps identify and act on threats before they become security issues threatening to compromise data or physical assets.

- **Strengthening the Synergies between the Development and the Security Functions:** DevSecOps creates a system where development, operations and security are in sync. In other words, no silos. In most healthcare-related projects, the work is carried out by cross-functional teams, and the sense of ownership that everyone has is appropriate to ensure that the security of the applications under development is not considered an add-on or an incremental factor but is encoded in the development lifecycle. It fosters convergence and helps identify security problems and embrace the finest security development processes much earlier in the process. Therefore, security issues are resolved and distributed because individuals usually do not consider certain aspects of their weaknesses; they are overseen, allowing for the implementation of a unified strategy for healthcare application security.
- **Enhancing Patient Trust and Satisfaction:** Healthcare consumers provide their most personal details to healthcare entities when seeking help. But when they use health care apps, they expect their details to be protected. This often incurred trust can be betrayed, and the implications can be severe, including being sued or heavily penalized and losing reputation and patients. That way, the concept of DevSecOps was developed to implement security and privacy consistently so that users could be trusted. With operational and compliance tests performed automatically, healthcare firms are dedicated to safeguarding patient information and delivering safe digital care. This bolsters the patient's trust in the given system, which is crucial for using digital health technologies.
- **Cost Efficiency through Early Vulnerability Detection:** Traditional pre/post development security assessments FC laughingstock security when issues are found in the post-development phase must be remediated at a far higher cost with increased time costs than corrections made during the development cycle. Thus, DevSecOps presupposes detecting the threat, including introducing the security testing processes

into the SDLC. This reduces the cost of fixing vulnerabilities because problems can be solved at the development stage, not at the deployment stage when many problems must be patched. Moreover, generalising security measures allows avoiding the build-up of technical debt and entailing long-term cost-saving benefits for healthcare constituents.

1.2. Need for DevSecOps in Healthcare

Adopting a cultural approach, the conventional DevOps practices dictate a faster rate in the SDLC to deliver a high-quality solution in the shortest possible time. [5,6] Although this has been quite efficient where enabling the fast release of features is concerned, the downside is that security is more frequently addressed as an afterthought, often implemented in later development phases testing and audits. This is particularly alarming because of the great concern in fields such as healthcare, where the consequences of poor security are much more severe. Nothing about a patient's personal information, whether the patient's detailed health information or financial records, is safe from hackers because of the increasing use of healthcare applications. This can be disastrous since violating the same can trigger legal consequences, harm the company or the hospital's reputation, or even endanger a patient's life. In the traditional DevOps frameworks, security is an addendum, where automated security scans or vulnerability fixes are implemented in a fixed program version when in production. This reactive approach allows significant threats to enter the application perimeter, which could have been blocked earlier in the software development life cycle. DevSecOps corrects this issue because the usage of security extends throughout the development life cycle, from design and coding to deployment and monitoring processes. This approach integrates cybersecurity as a cross-functional effort throughout the organization's DevOps teams. Security is effectively integrated into the DevSecOps process through security automation across the entire DevOps pipeline, with testing, compliance, and monitoring in parallel to prioritize identifying and addressing security issues. This proactive approach also enhances overall security and completes the mission to safeguard data and other sensitive information that directly impacts a healthcare organization. Whenever a proactive protection measure is found inadequate, fix it as per the regulatory benchmark and build up the patient's confidence in an organization. In the same way, via DevSecOps practices, healthcare organisations can produce secure, compliant, and quality applications within a timely fluency that does not compromise the innovative aspects that accompany such methodologies.

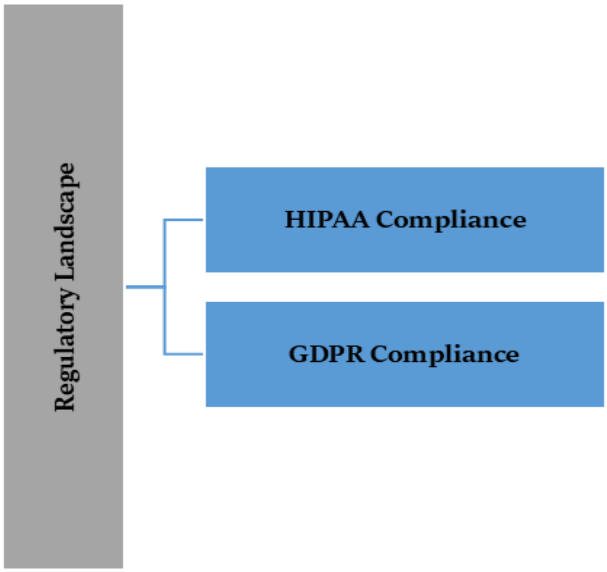

Fig 2. Regulatory Landscape

1.2. Regulatory Landscape

- **HIPAA Compliance:** HIPAA guidelines' rules address the privacy, security, and availability of the electronic Protected Health Information (ePHI) in the healthcare sector. HIPAA states that many administrative, physical and technical measures must be implemented in healthcare facilities to protect patient information from disclosure, unauthorized access or alteration. These measures include but are not limited to the use of passwords, user log in details, periodic checks and staff orientation to the policies and standards of the University. In a DevSecOps context, these safeguards become Imperative in the DevSecOps model. Privacy and security considerations are integrated into the software delivery pipeline and employed consistently from development to production. HIPAA noncompliance can cost organizations substantially through hefty fines and loss of reputation, which is why its enforcement is critical in the application development lifecycle.
- **GDPR Compliance:** The GDPR is a set of EU regulations on data protection, primarily concerning personal data protection. GDPR emphasizes two core principles: A concept of data protection by design and by default - this basically means that privacy requirements must be built into the solutions as a primary design step rather than be afterthoughts. It also has a significant focus on consent meaning that consumers have control over their personal information, be collected, processed, and stored. In the case of healthcare, GDPR is in operation regardless of the organisation's location; that treats

residents' data of the blocks of the European Union. In healthcare, this implies that user data is protected effectively from unauthorized access stored, used only where it is necessary, and only used by authorized staff. GDPR requirements can be applied and maintained throughout the SDLC and DevSecOps practices to satisfy these requirements, integrated into them and implemented in the following manner; for example, Failure to abide by these Data Protection laws attracts hefty fines, thus the need to embrace security and privacy features in the application of healthcare technology.

2. Literature Survey

2.1. Existing DevOps Practices in Healthcare

In healthcare industries, DevOps was implemented mainly based on deployment speed, system reliability, and operational effectiveness. [7-11] These often include making speed of development faster, timely integration of development and operations processes, and utilization of resources. Still, a significant weakness exists: the lack of automatic security checks and evaluation and the post-implementation assessment of vulnerabilities. This reactionary approach to application development only increases vulnerabilities as problems are identified after live systems are implemented. Moreover, since most applications in the healthcare domain are complex because of data sensitivity and data regulatory concerns, the ability to detect security vulnerabilities also faces delay consequences.

2.2. Evolution to DevSecOps

DevSecOps is an extension of DevOps that integrates security end-to-end in the development and deployment process of the software. This approach embeds security inherently within CI/CD processes and builds security into applications and development life cycle. It asks for proactive threat modelling, where threats and risks are addressed as soon as they are found in the development phase and real-time monitoring to discern new emerging threats and vulnerabilities. Applying the principles of DevSecOps is built on the idea of shifting security left so that development, security, and operations teams are involved right from the beginning of the development process rather than being considered an add-on.

2.3. Case Studies

- **Case Study 1: Mayo Clinic:** To fully implement DevSecOps at the Mayo Clinic, the utilization of SAST and DAST tools in the CI/CD processes was employed. Such

security measures allowed the organization to mitigate risks before deploying in the environment, thereby minimizing the number of security incidents in the production environments. The positive outcome was a 35% decrease in post-deployment vulnerabilities, which led to the conclusion that the best way to integrate security is at the beginning of the development cycle.

- **Case Study 2: NHS Digital:** During development, NHS Digital integrated container scanning and compliance-as-code solutions to keep the working environment safe and consistent. Due to automated scanning of the images and policy checks for conformity across multiple stages, situational security was maintained throughout different applications and services introduced by NHS Digital. This approach increased scalability and ensured the security of an organization's update process and compliance with healthcare regulations.

2.4. Gap Analysis

The differences between DevOps and DevSecOps became apparent mainly in treating security within the development life cycle. Generally, security testing within traditional DevOps is ad hoc and occurs sequentially at the end of the development phase. This approach often conceals vulnerabilities and puts a wrench in the system that prevents developers from being prepared and able to fix it as soon as possible. On the other hand, DevSecOps integrates and applies security testing throughout the SDLC to identify and address threats early and automatically. In DevOps, compliance checks are normally post-deployment; hence, when there is a check, the organization experiences non-compliance and more work. DevSecOps incorporates an immediate validation of compliance, therefore preventing breaches of regulation. However, there is limited engagement from developers in terms of security activities in the classic DevOps approaches. DevSecOps ensures that everyone has a security mentality, making developers responsible for embedding security from the development phase making software more secure.

3. Methodology

3.1. DevSecOps Implementation Framework

Planning & Requirements: The first is identifying security requirements and functional and business goals. Data protection requirements, [12-15] compliance directives (for example, the Health Insurance Portability and Accountability Act), and threat analysis are addressed from

the beginning. Integration of security into the development and operations plan of a software system makes it possible to avoid considering security as a ‘bolt-on’ system.

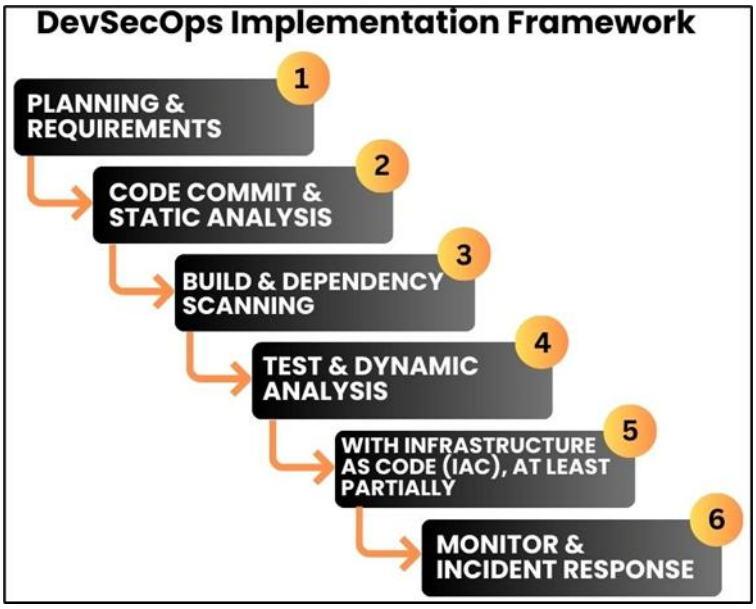

Fig 3. DevSecOps Implementation Framework

- **Code Commit & Static Analysis:** Once coding is done, security is implemented by code reviews and static application security testing (SAST). Before the code is checked into the main repository, these tools scan for problems like hard-coded credentials, insecure methods, and data leakage. STS has to be incorporated at the commit stage to allow identification of issues at the early levels, hence attracting lesser costs and efforts than when it is done at a later state.
- **Build & Dependency Scanning:** In the build phase, the compiled code, along with its specified software dependencies in libraries and frameworks, is checked for vulnerabilities by software composition analysis (SCA). These scans detect old libraries, the violation of open-source licensing agreements, and open-source software which contains common vulnerabilities and suspicions in the license. With most of these checks running through the CI/CD pipeline, the teams can ensure that code flagged for vulnerabilities does not advance to other stages.
- **Test & Dynamic Analysis:** In this stage, the application is tested in the runtime environment using dynamic application security testing (DAST) & interactive application security testing (IAST). These work like replicas of real-world conditions to check vulnerabilities such as injection, broken authentication and improper

configurations. Incorporation of security testing into functional and regression testing would have a way of exposing the vulnerabilities before deployment.

- **With Infrastructure as Code (IaC), at least partially:** The deployment practice is used to provision and configure environments through Infrastructure as Code securely and non-equivocally. Secrets management, security of the network configurations, and compliance as code policies are incorporated within the IaC templates. Open-source tools like Terraform or Ansible also highlight that all security patterns are versioned and applied consistently across environments.
- **Monitor & Incident Response:** The system is continually monitored with security information and event management (SIEM), intrusion detection systems (IDS) and behavior analytics after deployment. Alerts, threat intelligence, and response orchestration are ways of dishing out incidents quickly. The monitoring programs generate feedback incorporated into the planning of the subsequent steps taken under the DevSecOps process.

3.2. Security Automation Tools

- **SAST Tools: SonarQube, Fortify:** White box testing method known as Static Application Security Testing (SAST) tools such as SonarQube and Fortify check source codes in the early phase of the Software Development Life Cycle (SDLC without running the code. These tools assist developers in determining coding problems, application security threats, and coding poor patterns, including SQL injections, buffer overflows, and code smells. SonarQube is readily integrated into the CI/CD and flexible for multiple languages, while Fortify penetrates the corporation level with profound static analysis and immigrant resolution instruction.
- **DAST Tools: OWASP ZAP, Burp Suite:** DAST tools like OWASP ZAP and Burp Suite interact with applications in a running state and mimic real-world scenarios to exploit aspects of XSS, broken authentication, insecure redirects, etc. OWASP ZAP is an excellent Python framework for automated assessments, being developed collaboratively. At the same time, Burp Suite is considered a robust, licensed tool for extensive manual testing with modularity extension.

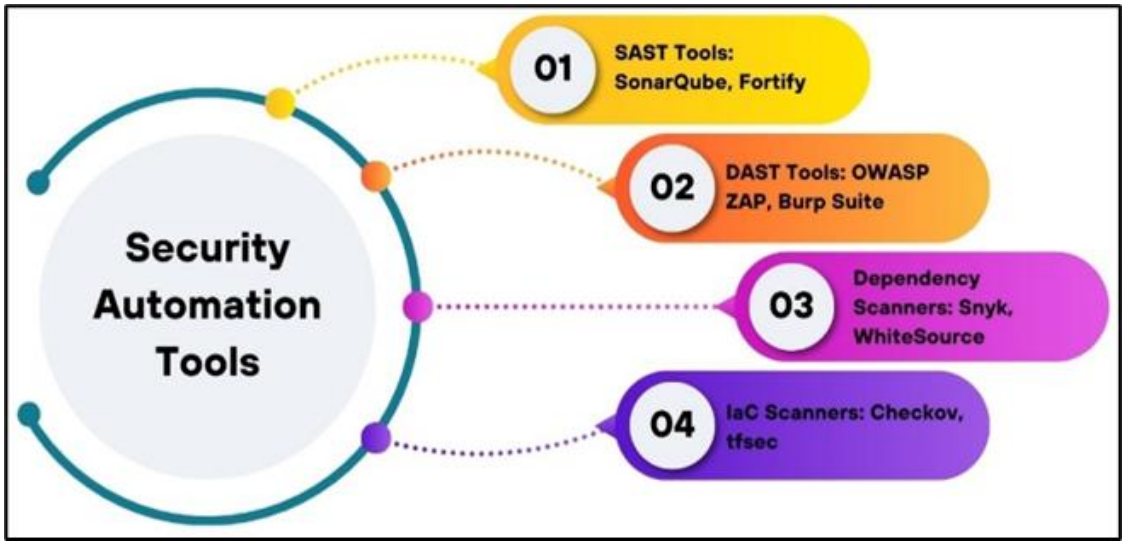

Fig 4. Security Automation Tools

- **Dependency Scanners: Snyk, WhiteSource:** Software Composition Analysis or Dependency tools like Snyk and WhiteSource analyze the set project dependencies to find out if any of them contain known open source vulnerabilities. Snyk naturally fits into developers' toolchains and offers on-the-spot remediation advice, whereas WhiteSource is more tailored for vulnerability remediation and license management. This article is important as it outlines and explains these tools for minimizing the risk of using insecure third-party components.
- **IaC Scanners: Checkov, tfsec:** Tools like Checkov and tfsec target IaC templates, including Terraform and CloudFormation, to identify misconfiguration and security vulnerabilities that can be deployed. Checkov uses an extensively customizable policy engine running the policy as source code. At the same time, tfsec works as a detailed check of the Terraform code to identify risks such as open ports, weak encryption, and relaxed permissions. These tools contribute to constructing a security perimeter and enforcing compliance from day one in any cloud environment.

3.3. Continuous Compliance

The modern IT world requires continuous compliance to maintain regulated scenarios and secure standards which smoothly integrates into the DevSecOps frameworks to utilize in the application and infrastructure. In contrast to the compliance initiative as a check-the-box exercise during or after delivery, compliance is done with CI/CD processes. Incorporating this

proactive approach into development and operations, [16-19] it is possible to flag compliance issues early on in the process and avoid extensive refactoring and compliance penalties that might be incurred after the deployment phase. It is also one of the primary approaches to the continuous compliance implementation methodology, where compliance rules are integrated into the purpose-built paths. This means integrating security and compliance rules like encryption, access rights, log monitoring, and security standard settings to be tested for compliance when code is checked in/checked out, built, or deployable. For instance, pipelines can include checks to verify that all provisioned resources adhere to some specified security guidelines; it may also check if all the data storage uses encryption at rest and in transit. One of the great enablers of this process is using tools that deal with policies as codes, such as Open Policy Agent or OPA. Specifically, through OPA, specific controls, policies, and their versions can be defined programme -wide across a multistakeholder environment that includes various systems and technologies. It is possible to integrate OPA and have it generate policies written in its Rego language to check infrastructures, configurations, deployment manifests, and patterns of applications in real-time, allowing for the rejection of non-compliant builds. This not only strengthens security and compliance on an ongoing basis but also records who has done what, where, and when, which enhances the security decision making transparency and accountability. Moving compliance left and making it continuously reduces overhead caused by compliance checks, shortens development cycles, and guarantees compliance with security and regulations. It is especially helpful in heavily regulated fields such as the healthcare sector, the General Data Protection Regulation, or the Payment Card Industry Data Security Standard.

3.4. Secure Coding Practices

Adhering to secure coding standards is the bedrock of an application and assists in striking out upcoming general weaknesses easily exploited by wrongdoers. These practices involve the use of code that is not only effective but also safe since it is coded securely. Implementing proper input validation, the principle of least privilege, and data encryption are the three basic paradigms that can help to reduce risks from attackers and protect data. It checks data received from users, APIs or any third-party source for accuracy, format and security before further processing. This technique helps avoid data injection attacks, for example, SQL injection, vulnerable cross-site scripting attacks (XSS), and command injection whereby damaging data is viewed as actual codes. Therefore, a big security threat can be reduced if developers begin to validate data on the client and server sides, thereby discarding any extra or unexpected inputs. Strictly speaking, the principle of least privilege (PoLP) states that users, services, and

applications should have only the minimum levels of access or permissions needed to do their jobs. This lowers the risk exposure in the event of a break-in or systems failure and or misconfiguration. For instance, an application component only required to access a database should not possess read/write or administrator access. Implementing some level of PoLP in code, APIs, system roles, and infrastructures reduces threats and enhances their control. Another component of secure coding is data encryption, in other words, encryption of data kept on the machine and during transfer from one host to another. Encryption ensures that the information cannot be understood even when confidential information is leaked or otherwise obtained by the wrong individuals. Passwords should be protected by the following cryptographic measures: the encryption method must not be less than TLS 1.2, and the AES-256 encryption must be utilized. Password, personal and financial data, and other personal information must always be protected and encrypted through the software life cycle.For these and other secure coding best practices to make the intended positive difference, practising them regularly, in equal measure and early enough is prudent for organizations.

3.5. Threat Modeling

Threat modelling is simply the way of understanding how an attacker approaches an application or system to master the various opportunities that will help the attacker exploit it. Security by design extends through tackling questions such as how could a hostile party try to subvert a system, which vulnerabilities could be exploited, and what the repercussions would be. One formula forms the basis of threat modelling: Risk, as defined by the formula, is calculated as follows.

Risk = Threat × Vulnerability × Impact.

As shown in this formula, the level of security risk is defined as a combination of a threat (for example, a hacker or malware), a vulnerability (such as insecure input handling or poor authentication), and the consequence if a threat exploits a vulnerability, for instance, data loss, system outage, or violation of regulation. Part of threat modelling is to define assets or to identify parts of a system, its data, and trust perimeters or pathways that change from one security domain to another (e.g., from a user device to a cloud server). In this way, the identified areas allow teams to find authentication, data storage, and/or communications vulnerabilities. An easy-to-use tool, one such is Microsoft's STRIDE model, which includes Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. Another widely used method, PASTA (Process for Attack Simulation and Threat Analysis),

connects threat modelling with business goals and appreciated risk levels. As a result, threat modelling is not a waterfall activity but a constant activity in the DevSecOps model and is continuously adjusted during the evolution of the application. This makes it possible to revisit systems developed and check for new features and changes in the system architecture. At the same time, when teams apply the formula and evaluate parts of a system with higher risk scores, it is possible to decide on the allocation of security measures. Threat modelling allows developers to understand how a threat actor would approach a system and address these problems before they become real-world issues.

4. Results and Discussion

4.1. Evaluation Metrics

- **Time to Deployment:** Throughput is calculated as the period between when a commit is made and when a change is deployed to the production environment. In context to the DevSecOps paradigm, this metric measures the impact that the incorporation of Auto-Sec and compliance- checks exert on the top-line explicit aspects of the deployment process. Even better, by including security right from the design phases and posing many of these steps to be manually carried out later during deployment in an organizational environment, the time which has previously been required to be spent on post-deployment security review and audit can be significantly lowered. Due to a short period to what seems to be the deployment of the final iteration/ release, the development process is more flexible, which is a valuable aspect for sustaining competitive advantage and for addressing the customers' needs and wants in today's ever-evolving and technological epoch.

- **Number of Vulnerabilities Detected Pre-Deployment:** This metric measures the number of program flaws and their characteristics when they were still at the developmental stage before the launch in the production environment. Static Application Security Testing, Dynamic Application Security Testing, and various security tools within the CI/CD pipeline minimize the chance of opening up vulnerabilities. Reducing threats and susceptibilities pre-deployment is essential for avoiding exploitation in production surroundings. They also give insight into the extent of implementation of security control measures within the DevSecOps cycle, if any firm has effectively acted on the security breaches before reaching the consumer.

- **Compliance Violations:** Compliance Violations represent the degree of compliance conformity established by the assessed application and infrastructure with the administrative and security policies. To name but a few examples, compliance with HIPAA, GDPR, or PCI DSS becomes necessary for industries ranging from healthcare, and finance to government. DevSecOps Automation prescribes that before any code composed or deployment to production occurs, policy-as-code reviews the compliance status of the code. This metric measures how compliance is improving due to automation across the various stages of the software development process, revealing an increased ability to control risk and adhere to security standards and regulations.

- **Developer Productivity:** Developer Productivity measures the performance of each development team in terms of code production rates, the time required to fix bugs and the time needed to address security problems. In the latest practices of DevSecOps, the aim is to increase productivity by minimizing interactions with security or compliance incidents to let the core development effort happen instead of having to repeat relentless repairs. This metric is about how automation, improved use of security tools and a DevSecOps approach lead to a faster development process. In general, increased developer productivity will result in faster delivery and make the software more comprehensively tested.

4.2. Results from Pilot Implementation

Table 1: Comparison of Metrics Pre- and Post-DevSecOps

Metric	Improvement
Time to Deployment	33%
Vulnerabilities	64%
Compliance Violations	86%
Developer Productivity	20%

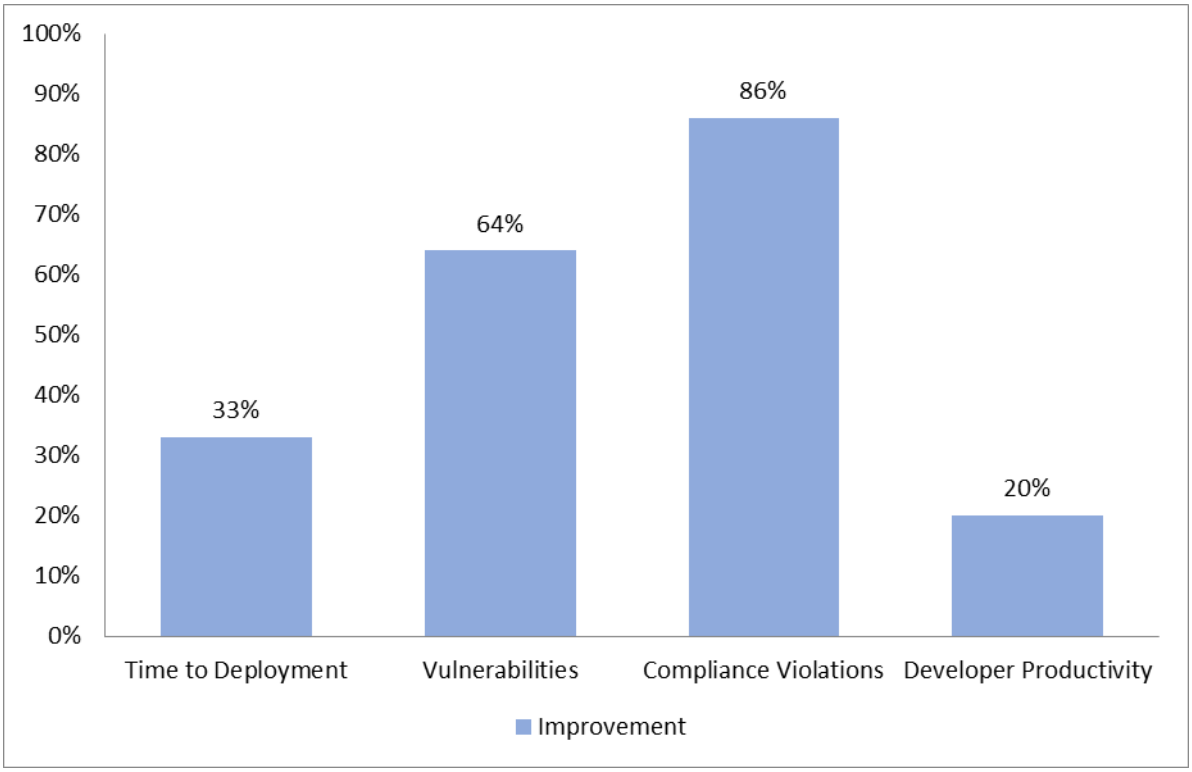

Fig 5. Graph representing Comparison of Metrics Pre- and Post-DevSecOps

- **Time to Deployment – 33%:** The promises of DevSecOps became evident in that we could deploy an application 33% faster, from twelve to eight days. That is mostly because of the increased adoption of security checks and compliance assessments in a CI/CD pipeline. When security problems are identified in the early stages of a project, most delays attributed to getting a security check or having a patch installed afterwards are eliminated. In other words, it allows developers to code security as an add-on while building business functionality which catapults the rate of release of features into the market.
- **Vulnerabilities – 64%:** Deploying both SAST and DAST security tools helped reduce the number of vulnerabilities identified pre-deployment by 64 percent. For the test, vulnerabilities were reduced from 42 to just 15, suggesting that a preemptive security scan is highly effective. Looking at security not as an add-on to the development cycle but as an integrated part, many problems were repeatedly spotted at the development stage before reaching the production environment. This cuts down on the vulnerabilities present in the final software, resulting in decreased possibilities of data break-ins or terrible breaches when the software is already released for use.

- **Compliance Violations – 86%:** Originally, there were seven compliance violations, but later decreased to only one when they implemented DevSecOps to its fullest extent, hence amounting to an 86 percent boost. This drastic decrease is achieved by incorporating policy scans and compliance as code that runs at the CI/CD pipeline continuously. The compliance tools allow for any build and deployment to automatically check for and follow, regulatory compliance, for instance, the HIPAA or GDPR. The following approach promotes conformity and checks the organization against hefty fines or damaged reputation from non-compliance, strengthening the organisation, especially in a highly regulated setting.
- **Developer Productivity – +20%:** In the case of developers, velocity demonstrated a 20% improvement, implying that they could deliver more value with features, bug fixes, and updates in less time. Security and compliance considerations were optimized and controlled as much and as soon as possible in the pipeline, freeing developers from frequently dealing with security issues or compliance findings that could have been weeks, maybe months, out of the pipeline. This rapid feedback loop that DevSecOps offered also decreased the time spent tracking down and resolving problems and increased efficiency. Consequently, there was the ability to deliver new functionality and fix bugs faster than the output process of the team.

4.3. Discussion

When utilizing and applying DevSecOps within the scope of the pilot, the numerous benefits in terms of performance and security proved that security must be integrated into the real SDLC process. Major improvements were made to the measure Time to Deployment, where the first release to deployment experienced a first-time 33 percent improvement. This was especially true on security checks and compliance verifications, painful processes traditionally performed manually that were heavily causing bottlenecks. Security was implemented at the beginning of the CI/CD process so teams could identify and rectify future problems before they arose in the live environments. The 64% decrease in vulnerability was another gain made. This reduction showed that it is effective to apply many kinds of security measures, including static and dynamic code analysis, as earlier in the development phase. Some security instruments were integrated with immediate feedback information so that the developers could correct weak points at the moment of their exploitation, contrary to using post-deployment diagnostics or assessment. This made applications more secure and minimised the

chances of common threats that prevail in Endeavour when launched, transforming them into costly breaches or data loss. Moreover, compliance violations were reduced by 86%, demonstrating the policy-as-code approach's effectiveness in enacting policy and procedural compliance. Ongoing surveillance and automated compliance checking and adjustment reduced the possibilities of human interference. They kept the system in check for compliance with the legal and industrial requirements of the application throughout its life cycle. This proved especially beneficial for IT sectors that are heavily regulated due to compliance issues, particularly regarding healthcare and finance.

Nevertheless, it was not smooth sailing as organisations shifted to DevSecOps for better security outcomes. Due to the relative maturity of security tool adoption, strategies aimed at incorporating them into CI/CD frameworks took time and skills to implement. Also, it was necessary to change the practice of employing developers and train them to become security-oriented and get acquainted with newly emerging tools. Nonetheless, the long-term advantage of faster deployment, better security, and compliance were good reasons he deemed the shift to DevSecOps a worthy investment for the firm.

5. Conclusion

DevSecOps is a relatively new way of ensuring organization sophistication and software development, especially in critical areas such as the healthcare platform. DevSecOps hinges its security approach on embedding security practices throughout the SDLC, from concept through to code, integration, testing, and operations, to ensure that risks are identified and addressed before becoming major issues. This is particularly important in the healthcare section since data, especially that of the patients, is rather sensitive. Integrating DevSecOps into patient engagement applications benefits the development process by minimizing the attack surface and avoiding additional expenses required to fix those defects and handle security incidents or non-compliance issues. Incorporating security throughout the development and operations cycle makes it seamless rather than shift-based, creating greater security and continuous improvement processes. With this approach, organizations can improve the pace at which they provide new features and updates and guarantee shored-up security and compliance. In this conversation, using DevSecOps in healthcare is beneficial in creating effective safe, fully functional apps compliant with regulatory requirements and friendly for patients.

5.1. Future Work

Even though DevSecOps has been shown to enhance organizational efficiency in fulfilling security requirements, there are further research and improvement areas in this sphere.

One of them is the application of artificial intelligence for automation and improving security analysis. Big data and the adaptive computational power of AI and machine learning can help discover patterns in threats, forecast future security openings, solve problems and even address certain problems automatically in real-time. The technology could be used in increasing the speed and accuracy of detecting threats without requiring several interventions to do so thus making new security threats be addressed much earlier. Another topic that will need more attention is zero-trust architecture integration. A zero-trust model presupposes that risks can appear from the external world and within the network, so all users, devices, and systems must be validated. Applying zero-trust principles to DevSecOps would help improve the security of healthcare applications even more because today, companies are moving to cloud services and adopting remote work solutions. Taken together, it is clear that there is a great deal of potential for greater use of patient feedback within the context of secure design to which future work should turn. To grasp patient perceptions of privacy, data security, and how their information is being used, the results can help determine how acceptable security measures are or where they may be lacking. Formal feedback from these might be incorporated directly into the design and security testing phases to ensure that patient applications are secure and trusted. These research areas have the great potential to advance the field of DevSecOps even further, as well as the methodology of secure application development within the healthcare industry where information security is truly critical.

References

[1] Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189.

[2] Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. It Revolution.

[3] Boda, V. V. R. (2022). Faster Healthcare Apps with DevOps: Reducing Time to Market. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(3), 55-64.

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

[5] Ramaj, X., Sánchez-Gordón, M., Chockalingam, S., & Colomo-Palacios, R. (2023). Unveiling the safety aspects of DevSecOps: evolution, gaps and trends. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 16(3), 61-69.

[6] Boda, V. V. R. (2023). AI Meets DevOps in Healthcare: Transforming How We Operate. Advances in Computer Sciences, 6(1).

[7] Akinola, O. A., Oyerinde, O., & Akinola, A. (2024). Implementation of DevOps in healthcare systems. Journal of Artificial Intelligence General Science (JAIGS) ISSN: 3006-4023, 2(1), 217-227.

[8] Yarlagadda, R. T. (2017). Implementation of DevOps in healthcare systems. Implementing DevOps in Healthcare Systems, International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN, 2349-5162.

[9] Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. Information and software technology, 141, 106700.

[10] Lombardi, F., & Fanton, A. (2023). From DevOps to DevSecOps is not enough. CyberDevOps: an extreme shifting-left architecture to bring cybersecurity within the software security lifecycle pipeline. Software Quality Journal, 31(2), 619-654.

[11] Ramaj, X., Colomo-Palacios, R., Sánchez-Gordón, M., & Gkioulos, V. (2023, August). Towards a DevSecOps-enabled framework for risk management of critical infrastructures. In European Conference on Software Process Improvement (pp. 47-58). Cham: Springer Nature Switzerland.

[12] Appari, A., Johnson, M. E., & Anthony, D. L. (2009). HIPAA compliance: an institutional theory perspective.

[13] Fu, M., Pasuksmit, J., & Tantithamthavorn, C. (2024). Ai for develops: A landscape and future opportunities. ACM Transactions on Software Engineering and Methodology.

[14] Reddy, P., Onitskansky, E., Singhal, S., & Velamoor, S. (2018). Why the evolving healthcare services and technology market matters. McKinsey & Company, 12.

[15] Barad, M. (2019). Linking cyber security improvement actions in healthcare systems to their strategic improvement needs. Procedia Manufacturing, 39, 279-286.

[16] Chu-Weininger, M. Y. L., & Balkrishnan, R. (2006). Consumer satisfaction with primary care provider choice and associated trust. BMC health services research, 6, 1-13.

[17] Sokolowski, D. (2022, November). Infrastructure as code for dynamic deployments. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1775-1779).

[18] Akinola, O., Oyerinde, O., & Akinola, A. (2023). Evaluating the Impact of DevOps Practice in the US Health Care Systems. Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online), 2(1), 158-162.

[19] Rajapakse, R. N., Zahedi, M., & Babar, M. A. (2022). Collaborative application security testing for devsecops: An empirical analysis of challenges, best practices and tool support. arXiv preprint arXiv:2211.06953.

[20] Ivanova, E., Stakhanova, N., & Sistany, B. (2024, October). Adversarial Analysis of Software Composition Analysis Tools. In International Conference on Information Security (pp. 161-182). Cham: Springer Nature Switzerland.

Citation: Anjan Gundaboina. (2024). DevSecOps in Healthcare: Building Secure and Compliant Patient Engagement Applications. Frontiers in Computer Science and Information Technology (FCSIT), 5(2), 17–37.

Abstract Link: https://iaeme.com/Home/article_id/FCSIT_05_02_003

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/FCSIT/VOLUME_5_ISSUE_2/FCSIT_05_02_003.pdf

✉ **editor@iaeme.com**