Estrategias para implementar el enfoque devsecops en el ciclo de vida del desarrollo de software ágil

Strategies for implementing the devsecops Approach into the agile software development lifecycle

[1]Diana Patricia Gamba Alarcón

[1]*Universidad Pedagógica y Tecnológica de Colombia*

Recibido: 03/15/2023 Aprobado: 30/06/2023

Resumen

El objetivo es generar un procedimiento para la implementación de DevSecOps en las primeras etapas del ciclo de vida de desarrollo de proyectos de software que apliquen metodologías ágiles; para crear entregas rápidas y seguras para minimizar costos, ahorrar tiempo y generar productos de mayor calidad. La necesidad surge porque, en el campo de la ingeniería de software, es evidente que la seguridad no se tiene en cuenta durante todo el ciclo de vida del desarrollo sino que suele dejarse para la última etapa, generando retrasos en la entrega de los proyectos al usuario final.

Palabras clave: Devsecops, ágil, seguridad de desplazamiento a la izquierda, sdlc, calidad, ic/dc.

Abstract

The aim is to generate a procedure for implementing DevSecOps in the early stages of the development life cycle for software projects that apply agile methodologies; to create fast and secure deliveries to minimize costs, save time and generate higher-quality products. The need arises because, in the field of software engineering, it is evident that security is not taken into account during the entire development life cycle but is usually left for the last stage, generating delays in the delivery of projects to the end user.

Key words: *Devsecops, agile, shift left security, sdlc, quality, ic/dc.*

Citación: Gamba Alarcón, D. P. . (2023). Estrategias para implementar el enfoque Devsecops en el ciclo de vida de desarrollo de software ágil. Publicaciones E Investigación, 17(2). https://doi.org/10.22490/25394088.6720

[1] dipagal_45@hotmail.com - https://orcid.org/0000-0002-8864-3689

https://doi.org/10.22490/25394088.6720

1. Devsecops approach into the agile software development lifecycle

Currently, in agile software development, the market demands continuous integrations and deliveries so that companies can be more productive and increase their competitiveness. This requires short reaction times in the development of new products as well as new versions and updates, which has led to implementing methodologies involving all the areas and processes affected to increase the value of the service.

DevOps meets these expectations because it allows for reducing the maximum time between each delivery. It also aims to carry out a different way in which the company works together, that is to say, an organizational methodology that considerably impacts the productivity and efficiency of software development. By applying it, the development and operations teams integrate both tasks. In this way, possible problems in operations can be foreseen from development, and the operations team benefits from an early stage of the knowledge and novelties that arise in the product.

In this model, security is left to the end (after the development stage), which means that the planned standards are not achieved. In many cases, companies have to choose between a high level of security, which requires a significant investment of time, or short release cycles, thus preceding security. This has a high impact on the final product, which is why DevOps evolves, and DevSecOps has emerged, offering a solution that combines the advantages of a high level of security and short product release cycles.

To implement it, security requirements must be integrated from the programming phase. Consequently, excellent communication between the security team, the development team, and the operations team is of paramount importance. The interdisciplinary scope of the process is key to an exemplary implementation in the life cycle of a software project.

This approach arose approximately in 2020, and in the literature, it is not easy to find a guide that allows implementing it in software projects. From here arises the need to delve deeper into this area and propose a procedure to implement DevSecOps in the development life cycle of the agile methodology in software projects.

The following concepts are used to understand the incursion of the DevSecOps approach:

Agile software development encompasses an approach to decision-making in software projects.

Software projects: refers to software engineering methods based on incremental and iterative development, where solutions and requirements evolve quickly in the timeline according to the project needs.

Software development life cycle: These are the stages defined in software engineering to verify the development of an application to corroborate that it meets the designed requirements.

Continuous delivery: For (Mishra & Otaiwi, 2020) frequency and quality. DevOps is a mixture of different developments and operations to its multitudinous ramifications in software development industries, DevOps have attracted the interest of many researchers. There are considerable literature surveys on this critical innovation in software development, yet, little attention has been given to DevOps impact on software quality. This research is aimed at analyzing the implications of DevOps features on software quality. DevOps can also be referred to a change in organization cultures aimed at removal of gaps between the development and operations of an organization. The adoption of DevOps in an organization provides many benefits including quality but also brings challenges to an organization. This study presents systematic mapping of the impact of DevOps on software quality. The results of this study provide a better understanding of DevOps on software quality for both professionals and researchers working in this area. The study shows research was mainly focused in automation, culture, continuous delivery, fast feedback of DevOps. There is need of further research in many areas of DevOps (for instance: measurement, development of metrics of different stages to assess its performance, culture, practices toward ensuring quality assurance, and quality factors such as usability, efficiency, software maintainability and portability implies automating and streamlining the deployment processes. This allows the organization to develop and deliver high-quality products efficiently and reduce time considerably.

Continuous integration and development (CI/CD): This method frequently delivers applications to end users by applying automation in development. The objective is to incorporate new code, monitoring its impact on the entire development lifecycle, from the integration stage to distribution and deployment. Here plays a vital role in adhering to operations and development teams with agile methodology.

Software engineering methods are techniques or tasks performed orderly, systematically, and structured to create high-quality and affordable software. It includes requirements analysis, design, code construction, testing, and maintenance.

Software security: Non-functional attribute, according to (Portal ISO 25000, n.d.), is the ability to protect data and information so unauthorized systems or persons cannot read or modify them.

Software product quality: according to (Portal ISO 25000, n.d.), it can be understood as the level at which the product meets the user requirements, providing value. It considers requirements such as functionality, performance, maintainability, and security.

DevSecOps approach (IBM - Deutschland | IBM, n.d.) It is the security integration in each phase of the software development lifecycle, from initial design to integration through testing, implementation, and software delivery. This approach represents a natural and necessary evolution in how development organizations approach security.

Shift Left Security: (Aqua, n.d.) "refers to the efforts of a DevOps team to ensure application security early in the development lifecycle, like a part of an organizational pattern known as DevSecOps (a collaboration between development, security, and operations). Shift left means moving a process to the left in the traditional linear representation of the software development life cycle (SDLC). There are two common themes of left-shifting initiatives in DevOps: security and testing."

DevOps: "Focuses on rapid software development and delivery using agile practices to improve collaboration between development teams and operations teams to reduce inconsistencies between development, operations, and releases." (Akbar *et al.*, 2022)

2. SDLC

As mentioned in (Ruparelia, 2010) the development lifecycle model, called "agile", focuses on dividing the project scope into smaller sub-projects, face-to-face communication, and little documentation during development. This results in smaller deliverables in shorter time intervals and having several versions of the software to make small incremental changes in a productive environment. For this model, it is optional to emphasize proper process-oriented steps.

This model is currently applied in the software development life cycle (SDLC), which is very important because it is an organized way of transforming an idea into a functional application. In this cycle, different activities are implemented, such as team management, end-user experience, procedure design, and implementation of security policies. Planning is directed at fulfilling the objectives outlined in the previously defined stages.

There are seven stages defined in SDLC, which will be mentioned below, and the purpose of each will be briefly explained. First, there is the planning stage and the most important one that defines; eighty percent of the project's success. In this stage, the end user's needs are identified, limiting the scope of the problem and its possible solutions. For this, aspects such as cost, execution time, amount of resources, the specialty of the resources, and tools to be used, among others, must be taken into account.

The next stage is the analysis; the team will focus on defining the functional requirements needed to provide the appropriate solution to the problem and scope defined in the previous stage. It is also necessary to establish who will be responsible for each stage of the project and the schedule to be carried out.

Next is the design stage, where the characteristics and specifications needed to meet the requirements proposed in the analysis stage must be defined. Here you must specify the technical specifications, such as connections between applications (cloud, database, network, others) and information of the core business of the company that requires the application, to correctly interpret what is needed for the specific flow and connections that are necessary for proper operation when implementing the new code.

The fourth stage is development, where the developer begins to design the solution using a flowchart and implements it in source code. Additionally, the developer must perform unit tests to verify that the code is running correctly.

The next stage, called "testing and integration", includes quality assurance (QA) resources to determine whether the design proposed and implemented in the previous step achieved the objective presented at the beginning. Here, as many errors as possible are found so that they can be solved before putting the application into a production environment. It should be noted that the ideal is to reduce errors and risks as much as possible and, if possible, eliminate them; for this, different types of tests are also implemented to validate the connection between components and systems if required by the type of application being tested. The tests can be repeated as often as necessary to verify that the faults are solved. Finally, the end user must accept or approve the tests performed on the basis that they meet their needs.

The sixth stage is called implementation. In the projects that apply the CI/DI continuous integration and continuous development, the delivery is ready when the program code is completed (the previously defined part). It is deployed to the production environment. This means that the development is installed and coexists with the other applications; this is done by bringing the necessary components to the new environment. This is usually done when there is less activity in the application. In a productive environment, the refined project should be reflected in the changes implemented up to that moment.

In the last stage, called "maintenance," the application's performance should be monitored, and if the end-user so decides, adjustments should be made to the system to improve its behavior. Additionally, remove or replace obsolete hardware, implement updates to comply with new standards, and apply new security policies to counteract new threats.

3. Security in the SDLC

Security activities must be implemented from the beginning of the life cycle and throughout the process to mitigate risks and errors in a software development project. This is to have quality and reliability in the final product. Another critical point to consider is continuous monitoring, which allows identifying problems quickly and applying security measures.

Another critical point to rescue, as mentioned by Field [7]: is that security should be crucial for top management. All company personnel should then be trained to understand the importance of this quality attribute in software. In this way, global awareness is created and not specific as it is currently managed, meaning that only a group of people belonging to the project is aware of it. The objective of the special training is to identify threats, errors, and defects and to determine the security measures to be taken to protect the software. General and specific training creates a security culture within the organization with sufficient seriousness in its tasks and activities. This is successful only if every human resource is aware of the importance of security and knows how to play his role correctly in the fulfillment of secure software development.

Procedure for incorporating the DevSecOps approach to the agile software development life cycle

It is proposed to incorporate the following activities in each stage to add security to the scheme being worked on in the DevOps methodology. But first it is important to mention the following: currently, vulnerability assessment is performed only before the application is deployed in the cloud environment. If the application is updated or any code is changed by the customer, the application's vulnerability is not reassessed and migrated directly to the cloud again. Therefore, the application could be in a vulnerable state after updating and when the vulnerabilities are not checked and fixed, there are chances that the application is under threat. In one of the research (Sharon Solomon, 2015) it was found that data breach causes more than $7.2 million and it takes 80 days to detect these issues. (Vijayakumar & Arun, 2019)

Shifting left means integrating testing and security activities into every relevant stage of development, from design to production. The goals of this shift are simple: Build security best practices into your process from start to finish. Detect potential issues as early in the lifecycle as possible. (Apisec, 2022)

4. Planning

At this stage, the most important for determining the success of the project, the security risks associated with the requirement must be thoroughly evaluated. First, each threat must be detailed, then each one must be classified, on a scale defined in the project (for example from 1 to 10, this being the highest impact), the vulnerability and the impact that would occur if these threats were to materialize during the project.

One method commonly used for risk analysis in projects is the "risk matrix"; an example of how to implement it is shown in Table 1. This defines two scales, one, the probability of the risk occurring, which can be classified as: frequent, probable, occasional, possible, improbable, and the other scale, the impact, which can be classified as: negligible, minor, moderate, major and catastrophic. Once the scales have been defined, each risk listed is classified according to the level of impact it may have and the probability of its occurrence. These values are represented in a table where one scale is placed in the rows and the other in the columns; it can be represented by a specific symbol such as an x or by colors, as long as it is clear in which box each risk is classified.

It is important to keep in mind that when defining each scale, the number of parameters must be equal, which means that it should not happen that 4 scales are defined for probability and 5 for impact. Always respecting the 1:1 scale

5. Analysis

Here it is necessary to evaluate, determine and design the action plan to be carried out when these threats emerge. Thus, it can be seen that the definition made in the previous stage is very important since it is the key to identify the most critical ones.

First, the 4 types of risks found in the matrix are defined: extreme risk, high risk, tolerable risk and acceptable risk. Then they are placed in the matrix bearing in mind the impact/probability ratio for each type of risk defined.

Table 2 shows a classification of the risks analyzed in the matrix that allows understanding the criticality of each one.

For acceptable type risks it is important to monitor and review them periodically as they do not indicate an alarm as long as they continue to behave in the same way. For tolerable and high risks, it is necessary to keep them in mind and pay attention to them so that they do not bring more serious consequences to the project, and for extreme risks, it is necessary to define strong action plans and controls to mitigate the probability of occurrence and, if they do occur, to reduce the impact.

On the other hand, according to (Diaz Diaz, 2014) non-functional requirements must be specified, such as: user administration and authentication, data confidentiality, cryptography, application availability, data integrity, session and session variable management, client code execution, data privacy.

6. Design

For extreme risks, it is necessary to define the resources involved, the preventive work plans focused on these risks, how to document the findings that are found, how to record the implementations that are being made, define the periodic updates that will be made. On the other hand, it is also important to design or establish the rules for writing code correctly, that is to say, that it is easy to understand by several resources, not only by the person who wrote it; a developer with experience in security is necessary so that when writing the code the policies are implemented and not waiting until the source code is finalized. This goes hand in hand with the developers implementing software security best practices, which is part of their integrity as professionals.

Finally, at this stage it is essential to properly identify threats in order to generate correct security requirements focus on relevant threats and report on them in the subsequent stages of the SDLC.

7. Development

Exists diferent ways to add the security in the continuous Integration and continuous Deployment (CI/CD), for example in the cloud, Vijayakumar and Arun mentioned the common problem in the vulneravilities of the aplications is when the code is change after deployed into the cloud environment. They propose a system that notifies when the code has a change and concluded that implementing Pub/Sub model or AWS SQS (Simple Queue Service) Service) would be make the systems more scalable (Vijayakumar & Arun, 2019) . Each of these is listed below

Pub/Sub allows services to communicate asynchronously, with latencies on the order of 100 milliseconds. Pub/Sub is used for streaming analytics and data integration pipelines to ingest and distribute data. It's equally effective as a messaging-oriented middleware for service integration or as a queue to parallelize tasks. Pub/Sub enables you to create systems of event producers and consumers, called publishers and subscribers. Publishers communicate with subscribers asynchronously by broadcasting events, rather than by synchronous remote procedure calls (RPCs). Publishers send events to the Pub/Sub service, without regard to how or when these events are to be processed. Pub/Sub then delivers events to all the services that react to them. (Google Cloud, n.d.-b)

Amazon Simple Queue Service (SQS) lets you send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available (Amazon Web Services, n.d.)

Another alternative to implement security in software is through containers, these are packages of software that contain all of the necessary elements to run in any environment. In this way, containers virtualize the operating system and run anywhere, from a private data center to the public cloud or even on a developer's personal laptop (Google Cloud, n.d.-a)

Some of the most commonly used containers are docker and kubernetes; Docker is a containerization platform and runtime and Kubernetes is a platform for running and managing containers from many container runtimes. Kubernetes supports numerous container runtimes, including Docker. (Atlassian, n.d.)

While Docker is a container runtime, Kubernetes is a platform for running and managing containers from many container runtimes. Kubernetes supports numerous container runtimes including Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface). A good metaphor is Kubernetes as an "operating system" and Docker containers are "apps" that you install on the "operating system".(Atlassian, n.d.)

8. Testing and integration

At this stage, the software quality tests must be implemented in the software quality tests, the security tests, in addition to those that the quality analyst or tester considers necessary to cover all the test scenarios in order to validate the good behavior of the application according to the initial requirements.

This type of test is implemented in order to detect errors and vulnerabilities. Here it should be noted that quality assurance tests should not only be implemented at this stage, there are also tests called static tests that are tests that do not involve the execution of the component or system being tested (ISTQB, 2018), that is, they are tests in which it is evaluated that the requirements and design of improvement plans are correctly created without even having the source code as input.

The most important thing when implementing dynamic tests (tests that involve the execution of the component or system under test (ISTQB, 2018)) and/or static tests, is to implement them in early stages of the sdlc. The feedback from the quality analysts should be taken into account from the very beginning when starting the definitions in the planning stage.

At this stage, non-functional tests should be implemented in order to evaluate the product's quality characteristic: safety. By performing these tests, problems such as vulnerabilities, failures in data integration, availability, loss and theft of information are detected. (Menejías García *et al.*, 2021)

In security testing, the following phases are carried out as mentioned (Cynoteck, 2020) first a vulnerability scan is performed with automated software, then the security scan is performed in order to find the weakness of the system and find answers to reduce the threat. This is followed by penetration testing also called ethical hacking, the purpose is to find security weaknesses that a hacker could exploit. With all the information obtained up to this point, a risk assessment is performed, i.e. the security risks are listed and controls and measures are created to reduce the risk. Then a security audit is performed, using tools that allow the detection of security imperfections, such as line-by-line code examination. After this, the ethical hacking is done, exposing the application being tested to different threats to discover its weak points and finally the position is evaluated, that is to say, a diagnosis of the security of the application is given based on all the findings found.

9. Implementation

It will be reviewed from the operational perspective unifying it with security.

As mentioned (Redhat, 2021) Kubernetes streamlines and unifies workflows across application development and operations teams. On the same structure that all teams like operations, development and qa work, it can be complemented by implementing a native kubernetes security platform.

A great advantage of this is that it reduces money and time, for example in the learning curve of new team members, and also allows early analysis and correction of errors.

The disadvantage that is evident today in teams implementing devops and also security but with different tools is that too many configuration issues arise.

By implementing all devsecops activities in containers, i.e. creating and shipping the applications in these; you get fewer interfaces, tools and models to deploy. This makes the final application easier to understand from code and allows identifying potential risks since its component structure allows it.

Another interesting point is the kubernetes manifests that allow devops to define the resources needed by a specific application. Policies are applied to these resources, reducing complexity and improving security at the macro level.

Kubernetes is particularly useful for DevOps teams since it offers service discovery, load balancing within the cluster, automated rollouts and rollbacks, self-healing of containers that fail, and configuration management. Plus, Kubernetes is a critical tool for building robust DevOps CI/CD pipelines.(Atlassian, n.d.)

10. Maintenance

Here it is important to perform quality tests either because there is a change in the application while it is deployed in production or because it is desired to update a component due to a change in technology.

It is also important to be alert at this stage, since over time new threats may arise that were not detected in the past because they did not exist. It is important to be clear that as technology evolves, so do threats. This is done by performing periodic reviews of both the technological infrastructure and the business logic and improvements proposed by the quality team. This process of application security testing must be constant, and process and product improvement must be maintained in order to achieve a high level of stability and standardization that allows providing a better service. (Diaz Diaz, 2014)

11. Conclusions

Early monitoring has, as a consequence, the early detection of errors and, therefore, the application of corrective actions soon after.

Developing applications with a high level of security will allow them to control critical points, make decisions about safety, reinforce security policies, and provide accurate information about future estimates.

For a project to have reasonable security policies applied, it must first understand its role and take seriously the activities involved in putting it into practice correctly. Additionally, the company plays an essential role since it must be aware that the corporate culture encourages and includes good safety practices as well as the way of approaching it; it also influences the knowledge about safety and the experience it has with previous incidents in safety issues.

In order to adequately mitigate the security risks that may occur in the project, it is important to define and implement effective strategies to properly manage the risks during each stage of the software development life cycle.

The quality assurance of the product is implicitly in each stage of the sdlc, since when managing the security risks, with the periodic reviews and preventive plans it is being assured that it fulfills the necessary measures of quality.

By implementing security policies using the native kubernetes application, the risk of implementing security in a tool other than the one used by the operations and development teams is mitigated.

The most important thing when implementing dynamic and/or static tests is to implement them in early stages of the sdlc. The feedback from the quality analysts should be taken into account from the very beginning when starting the definitions in the planning stage. It is important to focus on security in requirements and design, i.e. in the early stages of the SDLC.

The process of application security testing during the maintenance stage must be constant in order to improve both the process and the product, so that a high level of stability and standardization is achieved to provide a better service.|

References

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*(October 2021), 106894. https://doi.org/10.1016/j.infsof.2022.106894

Amazon Web Services, I. (n.d.). *Amazon SQS*. https://aws.amazon.com/sqs/?nc1=h_ls

Apisec. (2022). *Shift Left Security: The Ultimate Guide*. https://www.apisec.ai/blog/shift-left-security#:~:text=Shifting left means integrating testing,in the lifecycle as possible

Aqua. (n.d.). *What is Shift Left Testing & Security?* https://www.aquasec.com/cloud-native-academy/devsecops/shift-left-devops/

Atlassian. (n.d.). *Kubernetes vs. Docker*. https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker

Cynoteck. (2020). *An Introduction to Security Testing*. https://cynoteck.com/blog-post/introduction-to-security-testing/

Diaz Diaz, S. M. (2014). *Pruebas de seguridad en aplicaciones web como imperativo en la calidad de desarrollo del software*. 8. https://www.unab.edu.co/sites/default/files/MemoriasGrabadas/papers/capitulo7_paper_13.pdf

Google Cloud. (n.d.-a). *What are Containers?* https://cloud.google.com/learn/what-are-containers

Google Cloud. (n.d.-b). *What is Pub/Sub?* https://cloud.google.com/pubsub/docs/overview

IBM - Deutschland | IBM. (n.d.). *What is DevSecOps?* https://www.ibm.com/cloud/learn/devsecops

ISTQB. (2018). *Certified Tester Foundation Level Syllabus International Software Testing Qualifications Board*. 96. https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018-syllabus.html

Menejías García, R., Hidalgo Reyes, N. H., Marín Díaz, A. & Trujillo Casañola, Y. (2021). Procedure for evaluating security of software products. *Revista Cubana de Ciencias Informáticas*, *15*, 333–349. https://www.redalyc.org/journal/3783/378370462020/html/

Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. *Computer Science Review*, *38*, 100308. https://doi.org/10.1016/j.cosrev.2020.100308

Portal ISO 25000. (n.d.). *ISO 25010*. https://iso25000.com/index.php/normas-iso-25000/iso- 25010

Redhat. (2021). *Advantages of Kubernetes-native security*. https://www.redhat.com/en/topics/containers/advantages-of-kubernetes-native-security

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8–13. https://doi.org/10.1145/1764810.1764814

Vijayakumar, K., & Arun, C. (2019). Continuous security assessment of cloud based applications using distributed hashing algorithm in SDLC. *Cluster Computing*, *22*(s5), 10789–10800. https://doi.org/10.1007/s10586-017-1176-x

Appendix

Table 1
Risk matrix

Impact/Probability	Frequently	Probably	Occasionally	Possible	Unlikely
Insignificant	risk 3				
Lower	risk 4			risk 2	
Moderated		risk 1			risk 6
More			risk 7		
Catastrophic				risk 8	risk 5

Table 2
Ranking of risks analyzed in the matrix

Impact/probability	Frequently	Probably	Occasionally	Possible	Unlikely
Insignificant	high	acceptable	acceptable	acceptable	acceptable
Lower	high	tolerable	tolerable	acceptable	acceptable
Moderated	extreme	high	tolerable	tolerable	acceptable
More	extreme	extreme	high	tolerable	tolerable
Catastrophic	extreme	extreme	extreme	high	high