Privacy-Based Deployments: The Role of DevPrivOps in 6G Mobile Networks

Catarina Silva, Vitor A. Cunha, João P. Barraca, and Paulo Salvador

The authors propose the integration of DevPrivOps, a privacy engineering approach, in 6G software development to address the privacy challenges.

Abstract

The advent of 6G brings unprecedented opportunities for connectivity and software applications. However, it also poses significant privacy challenges. In this article, we propose the integration of DevPrivOps, a privacy engineering approach, in 6G software development to address these challenges. DevPrivOps emphasizes continuous integration, delivery, and deployment while incorporating privacy concerns from the early stages of development. This article outlines the principles and components of DevPrivOps in the context of 6G, including privacy quantification and intelligent prediction of privacy breaches. By integrating privacy engineering in 6G software development, DevPrivOps aims to enhance the privacy level of software, prioritize privacy considerations, ensure compliance with data privacy regulations, promote transparency, optimize performance, and inform decision-making processes. The proposed methodology addresses gaps in previous DevPrivOps concepts and highlights the need for generic solutions that can quantify privacy in various software domains.

Introduction

Software has become a critical component in the operations of most companies, as well as in daily human activities. The advent of the new generation of cellular communication technology, 6G, brings the ideal support required for generating and storing vast amounts of data, thereby benefiting Artificial Intelligence (AI) technology. With this trend, software must be sufficiently equipped to accommodate the demands of this technology, which highlights the relevance of delivering reliable, useful, private, and secure software products.

Software development involves two crucial factors: speed and quality. Speed is critical for responding to consumer needs and receiving market feedback quickly. While software quality refers to how well a software system or product meets stated and implied requirements (ISO/IEC 9126:2001, revised in ISO/IEC 25010:2011) [1]. To achieve both fast development speed and high product quality, software engineers adopt a Development and Operations (DevOps) culture. DevOps emphasizes collaboration between development and operations teams and breaks down the development process into different cycles to enable a fast delivery to customers. It involves a set of methods that aim to enhance communication and collaboration between developers and operations to deliver software and services rapidly, reliably, and with higher quality.

Privacy concerns are often overlooked during the software development life cycle, underscoring the need to promote Privacy by Design (PbD) and privacy by default strategies. DevPrivOps emerged as a specialized branch of the DevOps culture that incorporates privacy requirements into the software development life cycle. However, due to the vast range of issues that privacy encompasses, such as economic, psychological, and legal considerations, not all software engineers possess the necessary knowledge to address privacy concerns effectively.

We advocate for privacy quantification approaches adoption as a compelling means to prioritize privacy needs in software development. By leveraging metrics-based evaluation methods, it is possible to improve product quality, mitigate the negative impact of privacy concerns on society, and maintain fast development speed. Moreover, learning abilities improve these approaches by deciding when re-quantify or predict possible privacy breaches. The inclusion of privacy quantification into the DevPrivOps process helps organizations on ensuring that privacy risks are identified and addressed throughout the Software Development Life Cycle (SDLC). Privacy quantification approaches can be applied throughout the various stages of the software development life cycle, offering developers several benefits. These approaches enable developers to easily verify the level of privacy protection provided by the developed software, require no non-technical knowledge, and help reduce the likelihood of data breaches by applying appropriate methodologies. Privacy quantification in DevPrivOps provides a structured approach to identifying and managing privacy risks in software development.

The proposed approach includes reactive and proactive functionalities to evaluate the privacy level at different stages of software development. Moreover, it also considers Machine Learning (ML) models usage to estimate when a privacy re-quantification is required. Finally, the proposed approach considers local and distributed tests to reduce the possibility of privacy breaches in the software itself and when combining data with other available software.

ISSN: 0163-6804
Digital Object Identifier: 10.1109/MCOM.004.2300405

The authors are with Universidade de Aveiro, Portugal

The article is organized as follows: The next section presents the current background about privacy engineering. We then describe the key principles of DevOps. Following that, we identify the current state-of-the-art about DevPrivOps. We then introduce our proposed approach. Following that, we describe the main benefits in 6G scenarios. The final section presents the main conclusions.

Privacy Engineering Empowering Software

Privacy engineering is a systematic effort to include privacy requirements in the software planning [2]. This emerging area focuses on providing developers with the ability to embed privacy preservation solutions during the software development. It includes techniques, procedures, or methods related to privacy preservation to provide guidance or rules to achieve privacy goals and to support developers in carrying out engineering tasks related to privacy and, more broadly, to data management compliance.

Privacy Engineering Methodologies (PEMs) and respective possible evaluations have been proposed to guide privacy evaluation and consider different stakeholders' impact, data identification, and annotation about how data is used and communicated. Some other methodologies focus on privacy modeling and quantification.

Including privacy during the early stages of the software development brings advantages to end-users and developers [3] such as reduced costs, improved user trust, legal compliance, competitive advantage, and ethical responsibility. It is generally aligned with other good practices such as security by the design, and the much desired, left shift in the SDLC.

By avoiding the increased risk of data breaches and addressing privacy concerns earlier, companies can save costs, as well as increase users' trust in the system and, consequently, privacy-based systems' integration into society and daily activities. The increasing perception over the privacy of users, as many countries are working on privacy legislation and data protection regulation, imposing that companies must implement privacy controls during software development, also motivates this new approach. Additionally, formal and informal groups are working on standards to propose best practices for software design, with focus on privacy. Therefore, companies can ensure legal compliance and avoid fines or other potential legal action. Privacy can also contribute to companies making a distinction by attracting customers who value their privacy, which is much relevant in areas such as Internet of Things (IoT) [4]. Finally, companies have an ethical responsibility to protect their users' data and privacy. Incorporating this responsibility into the development process, companies are demonstrating their commitment to ethical practices.

Privacy-based deployment emphasizes approaches where privacy is given utmost priority during the software development. This approach involves incorporating privacy into the software design and development process, adopting methods of PbD, and ensuring that privacy is maintained throughout the SDLC, by taking necessary measures to protect user data and privacy.

The Key Principles of DevOps

The DevOps methodology is a software development approach that enables the frequent and iterative release of software in short cycles. In this approach, development teams work to continuously deploy new versions of the software into production, while operations teams focus on maintaining software stability and addressing other non-functional requirements. The continuous DevOps SDLC consists of several stages, such as: Planning, Coding, Building, Testing, Releasing, Deploying, Operating, and Monitoring.

DevOps promotes closer collaboration between developers and operators, being the commonly used strategy for service management. However, it is equally important to include academic researchers in this collaborative effort. Academic researchers can provide valuable insights to discussions among engineers and managers, as well as contribute to educating the next generation of software engineers on DevOps principles and practices. The essence of DevOps revolves around collaboration, and the involvement of academic researchers in this collaboration is crucial to address the key objectives of reducing risks and costs, ensuring regulatory compliance, and improving product quality and customer satisfaction [5].

Continuous Delivery (CD) helps to achieve this due to the short feedback cycle. However, some questions can be considered: How can organizations perform a cultural shift that implies more responsibilities for developers? How do developers acquire operations skills? It is essential to continuously deliver new versions; however, it is also necessary for each new version to consider conformance with the desired outcomes, including performance, availability, scalability, resilience, and reliability. Adequate quantitative assessment metrics help evaluates during the feedback cycle. Monitoring, for example, the performance level helps improve the final product [5].

To ensure desirable outcomes and software quality is crucial to identify quality attributes through the relationship between DevOps features and software quality. These quality attributes include flexibility, testability, usability, efficiency, maintainability, portability, reliability, security, reusability, and interoperability [1]. Activities such as the OWASP Application Security Verification Standard (ASVS), enable the development of generic, security driven, layered and verifiable tests/gates, meeting well discussed requirements.

DevOps features that are related to the software quality are automation, measurement, sharing, and quality assurance. Automation in the development process increases deployment rates, enabling quality deliveries with shorter cycle times. Measurement plays a crucial role during development by checking performance metrics, ensuring consistent results, and contributing to quality. Most notable, requirements definition can also be highlighted, as a stronger approach toward privacy, during the early stages of the project, will lead to a generally better product [6].

Highlighting collaboration in process, tools, goals, and development also contributes to quality by improving communication. Finally, DevOps is linked to quality assurance and enables the production of better software, aligned with the actual use cases.

Automation in the development process increases deployment rates, enabling quality deliveries with shorter cycle times. Measurement plays a crucial role during development by checking performance metrics, ensuring consistent results, and contributing to quality.

Creating specific cases of DevOps allows organizations to adjust DevOps principles to the unique needs of their technology stacks, business operations, and industry regulations, ultimately leading to more efficient and effective software development and operations.

Although adopting DevOps principles has numerous advantages, as described, the development of specific cases of DevOps has been motivated by the necessity to apply these principles to specific technology stacks and business operations.

For instance, NetDevOps [7] (also known as DevNetOps, NetOps, or Super NetOps) is an approach that integrates DevOps best practices into network engineering and operations. This involves deploying infrastructure based on version control, consistent provisioning, and parallel automation. In contrast, SP-DevOps [8] is specifically applied to Service Provider (SP)s and relies on repeatable and reliable processes. It emphasizes the importance of automated service verification as an integral part of the deployment processes.

Creating specific cases of DevOps allows organizations to adjust DevOps principles to the unique needs of their technology stacks, business operations, and industry regulations, ultimately leading to more efficient and effective software development and operations.

DevSecOps

The paradigm of DevSecOps refers to security integration as principles and practices to consider in the DevOps life cycle. The main idea is to keep security as a key focus throughout the DevOps cycle [9]. This approach addresses the need for security in the software development process from the beginning rather than being an afterthought or a separate process. Organizations can identify and address security issues earlier in the development cycle, reducing the risk of security vulnerabilities and breaches.

Furthermore, regulatory requirements and compliance obligations are increasingly demanding security to be a fundamental aspect of software development. DevSecOps provides an effective approach to meet these requirements while ensuring the continuous delivery of high-quality software.

Organizations are increasingly aware of security concerns such as data breaches and leaks and are taking them into account during software development, particularly for safety-critical systems. The DevSecOps culture characterizes by collaboration, knowledge sharing, fast and continuous feedback, continuous improvement, automation, commitment, and communication, similar to the DevOps culture. DevSecOps also emphasizes responsibility (ownership and accountability for security aspects), trust (identifying the right people at the right time to make better decisions), and transparency (ensuring teams understand all dependencies and associated software risks) [10].

DevSecOps integrates security practices into the DevOps pipeline. In contrast, privacy concerns entail a distinct focus beyond these security aspects. Privacy is primarily concerned with the intricacies of personal data acquisition, processing, storage, and dissemination.

Furthermore, DevSecOps does not focus on adhering to data minimization principles, which is essential in privacy-centric software development. The privacy framework underscores the importance of selectively collecting data for specific purposes and retaining it for the minimal required duration.

Moreover, DevSecOps does not address critical elements to ensure well-informed user consent in software usage, data flow transparency, and data portability. These aspects are essential in the context of privacy-conscious software development.

DevPrivOps: Privacy-Enabled DevOps

Effectively implementing PbD and Data Protection by Design (DPbD) requires a Data Protection Impact Assessment (DPIA). This task involves assessing privacy risks and managing them with appropriate countermeasures. The overall risk involves various factors, such as the impact of data breaches, the nature and value of the data sets, and the characteristics of the data subjects. Many approaches have been proposed to estimate privacy risk through individual factors, but they mainly focus on static realization during design and then development.

Sion *et al.* [11] introduced the term DevPrivOps to address this issue. DevPrivOps integrates the concept of privacy and data protection risk into continuous development and integration practices. The authors also proposed an overlay of enabling systems on the DevOps life cycle to enable more responsive and continuous privacy risk assessment and management. This enablers are a Privacy Risk and Compliance Dashboard that is a dashboard offered to developers and operators that provides a continuous, run-time inspection view on the system, users, processed information, and overall compliance status; Self-Service for Data Subject Rights allow data subjects to manage their risk and to exercise their data subject rights; Incident Response and Reporting to integrate incident and intrusion detection systems.

While the proposal of DevPrivOps was a valuable contribution that integrated privacy into the software development life cycle and bridged the gap between legal strategies such as PbD or DPbD and software engineering, there are some limitations to this approach.

One limitation is the lack of guidance on implementing the proposed enablers or which technologies to use. Additionally, the approach is reactive, meaning that incident response and reporting only occur after a problem has occurred. It is also unclear how the data protection impact assessment accounts for the sensitivity of different data types. Different scenarios require varying levels of data protection since they involve collecting and processing different types of data with varying sensitivity levels.

Grunewald [12] extended this previous work, illustrating how to ensure privacy in cloud-native architectures and identifying tools for privacy-friendly and agile development of large-scale service infrastructures.

Privacy Quantification in DevPrivOps

The advent of new technologies has increased the risk of data breaches, making it imperative to provide appropriate methods and tools for software developers and end-users to assess the privacy level of systems and understand the privacy impact of their usage.

Privacy quantification approaches use quantitative methods to determine the level of privacy offered by software. By utilizing mathematical metrics, these approaches consider various factors as privacy indicators that depend on the

specific application scenario. Weighted parameters can then be applied to produce a final value that corresponds to an evaluation on a privacy scale, which is easily understandable by users independently of their literacy levels. This scale enables quick identification of the software privacy risk, allowing more informed decisions about privacy [13].

Privacy quantification has a positive impact on software development, particularly in the early stages. By incorporating privacy considerations early on, developers can achieve better results. This approach does not require developers to be familiar with all the different privacy indicators for a specific scenario. Instead, they can rely on the results of the quantification to measure the effort needed to improve the final product in terms of privacy. Privacy quantification can serve as a bridge between developers and privacy standards and norms, facilitating compliance with these requirements. It can also help developers produce software that is more privacy-friendly, enhancing user trust and confidence in the application.

In DevPrivOps, developers can utilize privacy quantification strategies as part of a PEM. This emphasizes the importance of integrating privacy considerations throughout the software development life cycle and its evaluation. Benefits of this approach include risk assessment, compliance, transparency, performance, and decision-making.

We propose the DevPrivOps lifecycle based on privacy quantification, autonomous monitoring, and a combination of multiple software analyses. Figure 1 depicts the new lifecycle as an enhancement to the most popular DevOps lifecycle.

To incorporate privacy considerations into software development from the early stages is essential to initiate the collection of privacy indicators during the Privacy Indicators stage. To address the gaps identified in previous DevPrivOps concepts is crucial to have generic solutions capable of quantifying privacy across various software domains, such as medical applications, smart environment management applications, or location-based services. This requires DevPrivOps to extract privacy indicators for different domains and consider a wide range of data sources for different application scenarios. Thus, these indicators are fundamental to ensure that the software meets the necessary level of privacy requirements, and is appropriately configured for the specific application scenario. For example, for social network scenarios, Hughes- Roberts [14] considered concern as a privacy indicator of intentions and actions. Once the privacy specifications for the given scenario are understood, the implementation of Privacy Enhancing Technology (PET) becomes crucial during the software development process. Therefore, the *PET Implementation* stage is included to ensure the integration of PET (Fig. 2). This can follow many approaches, with the ones based on privacy assurance level, similar to what was adopted in OWASP ASVS, present a good potential outcome.

Considering the SDLC, it is expected that software with a version is ready for testing. Consequently, the subsequent stage will be the *Privacy Knowledge Extraction* stage, which requires dynamic and autonomous tools to extract parameters from the code and infrastructure. This extraction

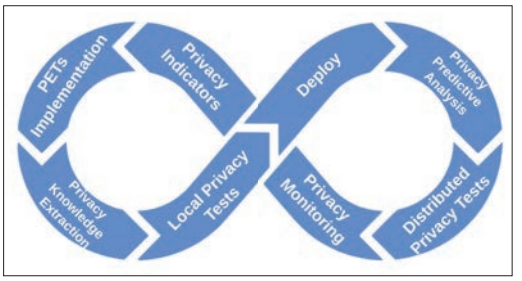

FIGURE 1. Proposed DevPrivOps Lifecycle.

aims to evaluate the current privacy status of the software, encompassing aspects such as code dependencies, communication with third parties, network protocols used, default configurations, data modulation in database schemas, data management (in use, or in transit), and the privacy controls.

After gathering these privacy parameters and considering the privacy indicators, the following stage, *Local Privacy Tests*, involves the utilization of an autonomous privacy quantification framework to assess the privacy risks associated with the current software, providing a score. This stage focuses solely on the specific software scenario and local tests, as its purpose is to understand the risks inherent in the software under development and testing. The individual requirements can be used to check if the software meets an assurance level, while the score allows setting overall quality gates for the privacy dimension of the SDLC.

Given that the software development process encompasses a *Deploy* stage, it is important to incorporate the evaluation of privacy risks in this stage as well, which consider deployment decisions, such as location, additional controls, and operational processes. To enable this monitoring it is crucial to extract the relevant features during the stage to initiate predictive analysis in the subsequent *Privacy Predictive Analysis* stage. In the *Privacy Predictive Analysis* stage, we emphasize the need to employ an autonomous model that leverages learning capabilities or statistical models. By utilizing such a model, we aim to autonomously predict the likelihood of privacy breaches, and anticipate potential issues before they occur. We envision that this stage can consider existing, community and industry landscape updates, produced by official entities such as MITRE, NIST, ENISA, or OWASP, in the reports concerning vulnerabilities leading to leaks. Furthermore, the inclusion of the privacy attack model is a key aspect of ML algorithms in the predictive analysis realm. This inclusion is of central significance, as it works as a preventive measure against complications arising from the automated analysis and the potential introduction of malicious software.

To further enhance privacy and achieve a more comprehensive privacy assessment is essential to account for data aggregation from multiple software systems and evaluate the associated privacy risks. These include assessing privacy risks from combining various systems/services during the *Distributed Privacy Tests* stage. For instance, consider a scenario where software systems do not process personal or sensitive data. However, combining multiple data sources makes predicting personal or sensitive data a possibility. Another

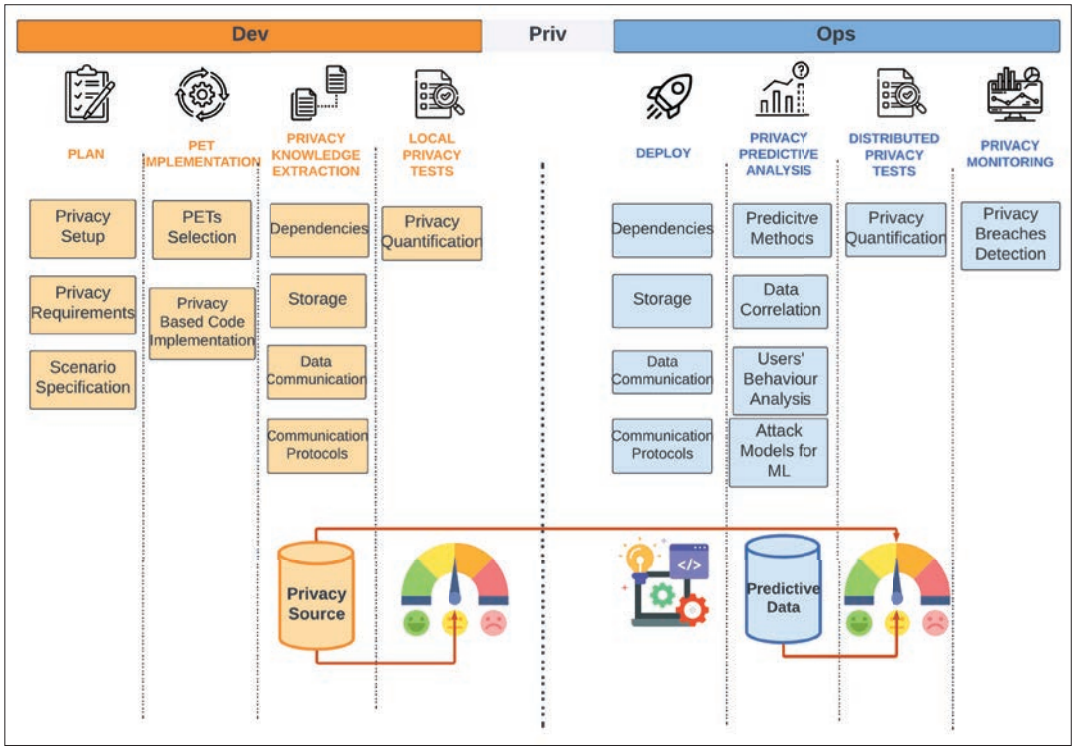

FIGURE 2. Privacy in Software Development.

aspect to consider is user behavior. While user actions may differ, this factor is not typically addressed during software development. However, during the *Distributed Privacy Tests* stage, it becomes crucial to consider the potential for privacy breaches by collecting non-essential data about user behavior that could be utilized to infer personal or sensitive information. Ensuring a robust level of privacy and conducting realistic privacy analyses in the software under consideration requires continuous privacy monitoring, considered in the last stage, *Privacy Monitoring*. This stage is significant for continuously detecting privacy breaches.

In this way, we promote for the need to have both reactive and proactive DevPrivOps. In addition to technologies responsible for incident response and reporting, DevPrivOps should be able to autonomously predict possible privacy breaches. One potential approach is to use ML algorithms to analyze systems and predict potential privacy breaches. These ML algorithms can also help detect when re-quantification is necessary, that is, when updates to systems may impact users' privacy. Re-quantification can also be achieved by utilizing predictive analysis, which involves using statistical algorithms to analyze system data and forecast future behavior based on historical patterns.

Upon the privacy level re-quantification, the quantification parameters (privacy indicators and the metrics employed) can be flexibly adapted according to the current software conditions. Consequently, this adaptability allows the evaluation of any change or new version of software according to the new configurations.

Generally, by incorporating generic privacy quantification solutions and utilizing reactive and proactive DevPrivOps, we can better address privacy concerns in various software domains and ensure that privacy is considered throughout the development life cycle. Despite the numerous advantages of DevPrivOps inclusion into software development, potential challenges can be identified. Two significant challenges include scalability constraints and computational complexity. Within the DevPrivOps methodology, privacy quantification models introduce additional overhead when analyzing data flows and identifying sensitive data points. Comprehensive data analysis, particularly when managing extensive datasets, can impose significant computational needs. Correspondingly, the execution of privacy impact assessments across several projects and services often entails intricate computations and modeling. In addition, evaluating the performance of privacy-aware applications under different loads, especially in scenarios that involve encryption, anonymization, or data access restrictions, can be computationally intensive.

We also advocate for users' right to decide about their privacy. Therefore, our proposed DevPrivOps approach emphasizes the need to involve users in the decision-making process by ensuring informed consent and transparent data flow. Privacy quantification in DevPrivOps informs users about privacy risks, enabling them to interact with the system and request privacy improvements based on their preferences. The quantification result can be mentioned explicitly at a service contract level, such as in the privacy policy. Moreover, this result can be updated regularly during the *Privacy Setup* in the *Privacy Indicators* stage. At some point, services can be composed of different services, for example, microservices architectures. This composition implies a distributed evaluation considered in the *Distributed Privacy Tests* stage. We can individually evaluate the different services, and the quantification result of each of them can

be described to users and developers at the service contract level. For this reason, this stage considers privacy tests about the developed service and the resulting interactions, dependencies, and communications with other services. Thus, it is possible to update the privacy level regularly and continuously update users with the privacy level offered by a system. Finally, ML algorithms will also be useful in creating patterns of users' privacy preferences, which will simplify the decision-making process for users in the future.

Maximizing 6G Potential With DevPrivOps

6G is the next-generation cellular communication technology currently being developed by researchers and engineers worldwide. 6G has already gained significant attention from the research community, with some experts predicting that it could become commercially available by the early 2030s. One of the crucial features of 6G is its ability to support extremely high data rates, which potentially enables new applications in areas such as augmented and virtual reality, high-resolution video streaming, and real-time remote control of vehicles and other machines.

The development of 6G promises to offer new opportunities for integrated localization and sensing applications that can transform how we interact with our environment, live, and conduct business. It will continue to operate at higher frequency ranges, wider bandwidths, and massive antenna arrays. However, 6G will enable sensing solutions with very fine range, angular resolutions, and location accuracy down to the centimeter level. As a result, network operators will be able to reshape and control the electromagnetic response of the environment [15].

The role of AI systems will also be significant in the 6G development, particularly with the proliferation of smart devices, autonomous vehicles, and IoT architectures. More intelligent and connected devices usage will produce unprecedented amounts of data. Furthermore, it is expected that more computational resources will be used to tackle the most challenging problems in wireless communication systems. These advancements suggest that 6G will be a genuinely intelligent wireless system, offering ubiquitous communication, high-accuracy localization, and high-resolution sensing services [15].

However, as with any new technology, there are also potential risks and concerns associated with 6G. However, this are aggravated as 6G proposes to extensively make of use of user location, and to integrate higher layer activities at the network level, in a way to enable richer, and low latency services (e.g, related with IoT and Virtual Reality, through edge computation). Potential privacy issues could arise with the development and deployment of 6G, such as: Increased data collection; Location tracking; Data security; Facial recognition; Social profiling; and Lack of transparency.

Since software development will play a crucial role in the realization of 6G's potential, the complexity of the software that will be used in 6G networks is expected to be higher than that of 5G networks.

By emphasizing continuous integration, delivery, and deployment, DevPrivOps enables teams to develop, test, and deploy software more efficiently while taking privacy considerations into account from the earliest stages of development. This methodology can help ensure that privacy is a core aspect of 6G software development, enabling network operators to build systems that are both innovative and private.

The proposed DevPrivOps methodology offers several advantages for 6G software development. By incorporating privacy quantification, this approach enables 6G operators (developers and SP) and users to quickly and easily understand the privacy risks associated with the software. Users can manage their privacy rights more effectively using a simple privacy scale, while operators can use the output of the privacy quantification process to update the software as needed.

The proactive approach to privacy quantification represents an improvement over the current state of the art in DevPrivOps since it allows the identification and addressing of potential privacy risks before they become problematic.

Additionally, by incorporating privacy indicators resulting from laws and standards, software developers can ensure that their systems are designed to comply with existing privacy regulations from the outset. Thus, the proposed DevPrivOps methodology represents an important step forward in ensuring that 6G software development is conducted with privacy considerations on top of the priorities.

Adopting privacy engineering strategies, specifically incorporating DevPrivOps, can significantly benefit network management in 6G. As the deployment of network infrastructure becomes more reliant on software, it is essential to ensure the software developed is reliable, functional, secure, and private. Privacy engineering principles can be integrated into the design and development of network management tools and systems using DevPrivOps, thereby ensuring that privacy considerations are taken into account throughout the development process. Additionally, privacy engineering and DevPrivOps can aid in identifying potential privacy risks associated with network management practices and provide recommendations for mitigating these risks. This approach allows privacy engineers to develop effective strategies for addressing privacy issues in network management.

Furthermore, privacy engineering and DevPrivOps can help to ensure network management tools and systems comply with current regulations, minimizing the risk of privacy violations. Finally, privacy engineering can enable network managers to be more transparent about data collection and processing practices, promoting user trust and satisfaction.

Conclusion

The proposed DevPrivOps methodology plays a crucial role in increasing the privacy level of software development, benefiting developers and users. The model encompasses local and distributed privacy quantification approaches, considering the analysis of the software's base code and infrastructure, user behavior and actions, and the inference of personal data through the combination of multiple data sources. Additionally, the proposed DevPrivOps incorporates an intelligent approach to predict potential privacy breaches and determine when re-quantify the privacy level.

The proposed DevPrivOps methodology offers several advantages for 6G software development. By incorporating privacy quantification, this approach enables 6G operators (developers and SP) and users to quickly and easily understand the privacy risks associated with the software. Users can manage their privacy rights more effectively using a simple privacy scale, while operators can use the output of the privacy quantification process to update the software as needed.

Privacy quantification within DevPrivOps offers several advantages, including prioritizing privacy considerations, ensuring compliance with data privacy regulations, promoting transparency, optimizing performance, and informing decision-making processes.

Privacy quantification within DevPrivOps offers several advantages, including prioritizing privacy considerations, ensuring compliance with data privacy regulations, promoting transparency, optimizing performance, and informing decision-making processes. Furthermore, the adoption of privacy engineering strategies and DevPrivOps in network management can significantly improve the privacy, compliance, and transparency of network management practices in 6G. Given the potential privacy risks and concerns, as well as the complexity and importance of software in 6G networks, it is vital to integrate a continuous development and integration of privacy-based technologies. Thus, we can ensure that privacy considerations are consistently addressed and included throughout the software development life cycle in 6G networks.

Acknowledgment

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101095933 (RIGOUROUS project).

References

[1] A. Mishra and Z. Otaiwi, "DevOps and Software Quality: A Systematic Mapping," *Computer Science Review*, vol. 38, Nov. 2020, p. 100308.
[2] A. Senarath, M. Grobler, and N. A. G. Arachchilage, "Will They Use It or Not? Investigating Software Developers' Intention to Follow Privacy Engineering Methodologies," *ACM Trans. Privacy and Security*, vol. 22, no. 4, Nov. 2019, pp. 1–30.
[3] F. H. Semantha *et al.*, "A Systematic Literature Review on Privacy by Design in the Healthcare Sector," *Electronics*, vol. 9, no. 3, Mar. 2020, p. 452.
[4] C. Silva *et al.*, "Privacy Preservation in Temporary use of IoT Environments," *CSNDSP*, IEEE, July 2022.
[5] L. Leite *et al.*, "A Survey of DevOps Concepts and Challenges," *ACM Computing Surveys*, vol. 52, no. 6, Nov. 2019, pp. 1–35.
[6] P. Leloudas, Challenges and Solutions in Software Testing, *Apress*, 2023, pp. 189–204.
[7] J. A. Shah and D. Dubaria, "NetDevOps: A New Era toward Networking & DevOps," *UEMCON*, IEEE, Oct. 2019.
[8] W. John *et al.*, "Service Provider DevOps," *IEEE Commun. Mag.*, vol. 55, no. 1, Jan. 2017, pp. 204–11.
[9] R. N. Rajapakse *et al.*, "Challenges and Solutions When Adopting DevSecOps: A Systematic Review," *Information and Software Technology*, vol. 141, Jan. 2022, p. 106700.
[10] M. Sanchez-Gordon and R. Colomo-Palacios, "Security as Culture: A Systematic Literature Review of DevSecOps," *Proc. Int'l. Conf. Software Engineering Workshops*, ser. ICSE '20, ACM, June 2020.
[11] L. Sion, D. V. Landuyt, and W. Joosen, "The Never-Ending Story: On the Need for Continuous Privacy Impact Assessment," *EuroS&PW*, IEEE, Sept. 2020.
[12] E. Grunewald, *Cloud Native Privacy Engineering through DevPrivOps*, Springer, 2022, pp. 122–41.
[13] P. Aqueveque *et al.*, "A Novel Privacy Preservation and Quantification Methodology for Implementing Home-Care-Oriented Movement Analysis Systems," *Sensors*, vol. 22, no. 13, June 2022, p. 4677.
[14] T. Hughes-Roberts, "Privacy and Social Networks: Is Concern a Valid Indicator of Intention and Behaviour?" *Social Computing*, IEEE, Sept. 2013.
[15] C. De Lima *et al.*, "Convergent Communication, Sensing and Localization in 6G Systems: An Overview of Technologies, Opportunities and Challenges," *IEEE Access*, vol. 9, 2021, pp. 26,902–25.

Biographies

Catarina Silva is a Ph.D. Candidate and Researcher at the University of Aveiro, and she is a Researcher at Instituto de Telecomunicaes in Aveiro. Her main research area focuses on Digital Privacy and Privacy Engineering approaches. She works on privacy-enhancing technologies development and proposes techniques to evaluate privacy in dynamic scenarios.

Vitor Cunha received his Ph.D. in computer engineering from the University of Aveiro, in 2022. He is a Researcher at Instituto de Telecomunicaes. He is currently working on dynamic security mechanisms for softwarized and virtualized networks. His interests include network security, SDN, NFV, and pervasive computing.

João Paulo Barraca is an Associate Professor at the University of Aveiro and Vice-Director of the Masters in Cybersecurity program. He is also a Researcher at the Institute of Telecommunications and a member of the Scientific Committee of the DETI Doctoral Program, as well as the Executive Board of UNAVE. He has published over 100 articles in peer-reviewed journals and conferences on networking, software engineering, and cybersecurity.

Paulo Salvador is an Associate Professor at the University of Aveiro, and a Researcher at the Institute of Electronics and Informatics Engineering of Aveiro. P. Salvador teaching and research interests focus mainly on network infrastructure resilience, user and service behavioral characterization, and security solutions based on AI. Has published over 120 scientific articles.