Metrics-driven DevSecOps

Wissam Mallouli[1][a], Ana Rosa Cavalli[1,2][b], Alessandra Bagnato[3][c] and Edgardo Montes de Oca[1][d]

[1]*Montimage EURL, Paris, France*

[2]*SAMOVAR, Telecom SudParis, Institut Polytechnique Paris, France*

[3]*SOFTEAM, Paris, France*

Keywords: Continuous Measurements, DevOps, Security, Feedback Loop, Software Quality, Automation.

Abstract: Due to the modern iterative development practices and new automated software engineering tools and methods brought by the DevOps agile method, the traditional metrics and evaluation methods are not enough to ensure software security. Besides, the recent years have seen probably the most continuous and extreme software security attacks ever recorded against organizations in an assortment of enterprises. Security is presently a vast range, critical for business achievement. The existing metrics must be redefined, and new security metrics should be determined based on multiple measures to increase the reliability of the values. Due to the short cycles of iterative processes in DevOps method, the feedback must come quickly, so the measurement should be automated and continuous. Due to the massive amount of information, the results must be visualized at a suitable level of abstraction, which may be different for the various stakeholders. In this paper, we propose a unique Metric-driven approach to help improve the software engineering processes by increasing the quality, adaptability and security of software and decreasing costs and time-to-market.

1 INTRODUCTION

The current cybersecurity landscape has evolved to the point that companies are facing increasingly sophisticated attacks (Tounsi and Rais, 2018), more and more adapted to their specific products, targeting all security measures across the application's software stack. At the same time, companies are faced with the need to deliver products to the market at an increasingly rapid rate and to adapt to new customer needs and requirements. As such, many businesses must find a difficult balance between the speed of software delivery, dictated by the market they are part of, and the need to invest effort and time to ensure that each iteration of their product is hardened from a cybersecurity perspective.

The need to deliver software products in a fast but still controlled and reliable manner has led to the growth of DevOps teams in most large software companies. However, since continuous delivery must be integrated into security, the DevOps teams have evolved to DevSecOps (Myrbakken and Palacios, 2017). As such, the tool chain used by teams to automatically test, prepare for deployment and deliver a software product to the market has been changed to include security controls. Each step of a software deployment pipeline should, to date, include a series of controls that ensure that the delivered product passes a series of security requirements designed to prevent malicious actors from exploiting them, or at the very least, to alert all concerned parties when any security breach occurs.

It is certainly necessary to include security controls as part of the DevOps process, but what is more important is to ensure that they are appropriate and relevant for the system deployed. In most cases, DevSecOps teams must manually define the controls to include in their process. The use of knowledge specific to an expert domain in this decision-making process adds value, since the controls and validations implemented are intended to prevent or alert against the exploitation of known vulnerabilities. However, the metrics measured to guarantee security (Medeiros et al., 2017) are, in most cases, standard measures intended to protect the application against known threats.

[a] https://orcid.org/0000-0003-2548-6628
[b] https://orcid.org/0000-0003-2586-9071
[c] https://orcid.org/0000-0003-2675-0953
[d] https://orcid.org/0000-0001-6771-0689

Mallouli, W., Cavalli, A., Bagnato, A. and Montes de Oca, E.
Metrics-driven DevSecOps.
DOI: 10.5220/0009889602280233
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 228-233
ISBN: 978-989-758-443-5

This leads to software applications not protected against the so-called ”unknown threats” - previously unknown vulnerabilities which, when exploited, can lead to problems ranging from loss of system services to more serious cases such as data leaks. As such, there is a clear gap that needs to be filled - develop a way to discover the metrics that need to be monitored in the DevSecOps pipeline in order to prevent or alert when the system behaves in a non-expected manner, indicating possible security issues.

In this paper, we propose a unique Metric-driven approach to improve software engineering processes by increasing the quality, adaptability and security of software and decreasing costs and time-to-market by:

- Defining innovative metrics and developing methods and tools for measurement of software engineering activities and artefacts, focusing on security, privacy and robustness related metrics, that will be adopted during the complete software lifecycle.
- Developing Machine Learning and Arificial Intelligence (ML/AI) based methods and tools for analysing real-time data produced by the continuous measurement to enable increasing trust, security, and reliability, while at the same time maintaining the necessary system performance and energy savings.
- Supporting decision-makers by visualising the results of continuous measurement at targeted level of abstraction, i.e. providing different visualisations for developers, operators, security experts, managers and other stakeholders. This includes providing clear and actionable recommendations for continuous security assurance.
- Proposing an agile development process (i.e. DevSecOps) that relies on the developed metrics to ensure better reliability and resiliency of artefacts.

The rest of paper contextualises the proposed approach in Section 2, then it details the proposed methodology and presents a concrete architecture of such framework in section 3. Section 4 concludes this paper.

2 CONTEXT AND MOTIVATION

2.1 Metric-driven Software Engineering

Measuring has become a fundamental aspect of software engineering (Bagnato et al., 2017). Accurate measurement is proving to be highly effective in various domains. For example, measurement is vital for the provision of high quality decision-support and prediction systems, in the context of improving software development and maintenance. Furthermore, accurate measurement is required for the evaluation and enforcement of a system's quality (by highlighting problematic areas), and in the determination of better work practices with the aim of assisting practitioners and researchers in their work.

Software Measurement needs tools that assist in the evaluation and industrialisation of Software Process Improvement in the organizations that develop them. Software Measurement is, in fact, a key element in initiatives such as SW-CMM (Capability Maturity Model for Software), ISO/IEC 15504 (SPICE, Software Process Improvement and Capability determination) and CMMI (Capability Maturity Model Integration). The ISO/IEC 90003:2004 standard also highlights the importance of measurements in managing and guaranteeing quality.

There exist various methods and standards concerning how to carry out measurements in a precise and systematic manner, of which the most representative are:

- Goal Question Metric (GQM): the basic principle of GQM is that measurement must always be oriented towards an objective. GQM defines an objective, refines that objective into questions and defines the measures which attempt to answer those questions (Koziolek, 2005).
- Practical Software Measurement (PSM): the PSM methodology is based on the experience obtained from organizations on the manner in which to best implement a software measurement program with guarantees of success (Card, 2003).
- ISO/IEC 15939: this international standard identifies the activities and tasks which are necessary to successfully identify, define, select, apply and improve software measurement within a general project or within a business's measurement structure.

In addition, the availability of a formal description language that allows representing the elements which must be taken into account by the measurement process can be considered as important in the decision making and process improvement.

2.2 Cybersecurity Related Risks

The recent years have seen probably the most continuous and extreme software security attacks ever recorded against organizations targeting different kinds of enterprises. Security has become critical for business achievement and shown the importance of safety and risk management for dealing with the protection of a company from destructive cyber-attacks and implementing more and more strict regulations. Thus, assessing the trustworthiness of a software from its early stages in its development lifecycle becomes a must. Many security metrics and indicators, have been identified and implemented in several security tools from requirements to operations. However, a global view of security practices are not always available for product owners, developers, operators, managers and other stakeholders (Casola et al., 2017).

2.3 DevOps Agile Process

Today, DevOps is a concept that is gaining serious traction within organizations – large to small – trying to bridge the gap between development and operations. DevOps outlines a cyclic approach to various stages of the software development lifecycle such as conception, test and deployment of software artefacts. Specific tools are used in every stage of the DevOps cycle to improve the final outcome. DevOps is centred on two core principles, automation and continuous improvement, and has become a crucial part of software development.

Nowadays, real-time integration, deployment, and monitoring of applications is a necessity where assessment and feedback are the foundations and embedded in every step. It forms a series of feedback loops that provide immediate insight and response to each preceding step in the process and delivers feedback from customers and users into the application's business value. This automatic and continuous feedback process is referred to as Continuous Assessment.

By integrating security in DevOps, the main purpose is to build on the mindset that "everyone is responsible for security". The goal is to distribute security decisions with speed and at a scale required so that those that can act can do so with out sacrificing other factors of the business process.

With an increased business demand for DevOps, agile and cloud services, traditional security processes have become a major roadblock that sometimes needs to be bypassed all together. Traditional security operates from the position that once a system has been designed, its security defects can determined by the security staff and corrected by the business operators before the system is released. This required a limited amount of security skills for producing the outcomes and avoided the need to expand the security context of larger systems. But a process designed this way only work when the business activities are organised as a waterfall with agreement among all involved parties. Unfortunately, the belief that security must operate this way is flawed with the introduction of iteration, creating inherent risks within the system due to the lack of coordination and cooperation.

In this paper, we target to answer these challenges by introducing a new Metrics-driven DevOps methodology (Forsgren and Kersten, 2018) that allows to continuously check the quality of a software from design to operation by gathering different metrics from different software artifacts. A specific focus on trustworthiness related metrics is performed to ensure reliable and resilient products.

3 METRICS-DRIVEN DevSecOps METHODS

The proposed architecture is based on the state-of-the-art architecture derived from two projects, in particular from ITEA3's MEASURE project[1] and H2020's MUSA Project[2].

The solution (represented in Figure 1) is built on a centralized platform dedicated for measuring, analysing, and visualising metrics (mainly quality and trustworthiness metrics) to extract and derive information concerning the software engineering process (Dahab et al., 2019). This platform:

- Implements methodologies and tools to automatically measure software engineering processes during the whole software lifecycle by executing measures defined using the SMM (Structured Metrics Metamodel) standard and extracted from a catalogue of formal and platform-independent measurements;
- Provides methodologies and tools which allow developing a catalogue of formal and platform-independent measures with a specific focus on trustworthiness;
- Integrates a storage solution for storing and processing measurements obtained by capturing metrics in a big data context;
- Relies on the visualization tools to expose the extract results in an easy-to-read fashion, so allowing a quick understanding of the situation to determine the possible actions that can be taken to improve the diverse stages of the software lifecycle.
- Includes an open API facilitating the integration of the measurement platform with external tools and services including other measuring and analysis tools.

[1]https://itea3.org/project/measure.html
[2]https://www.musa-project.eu/

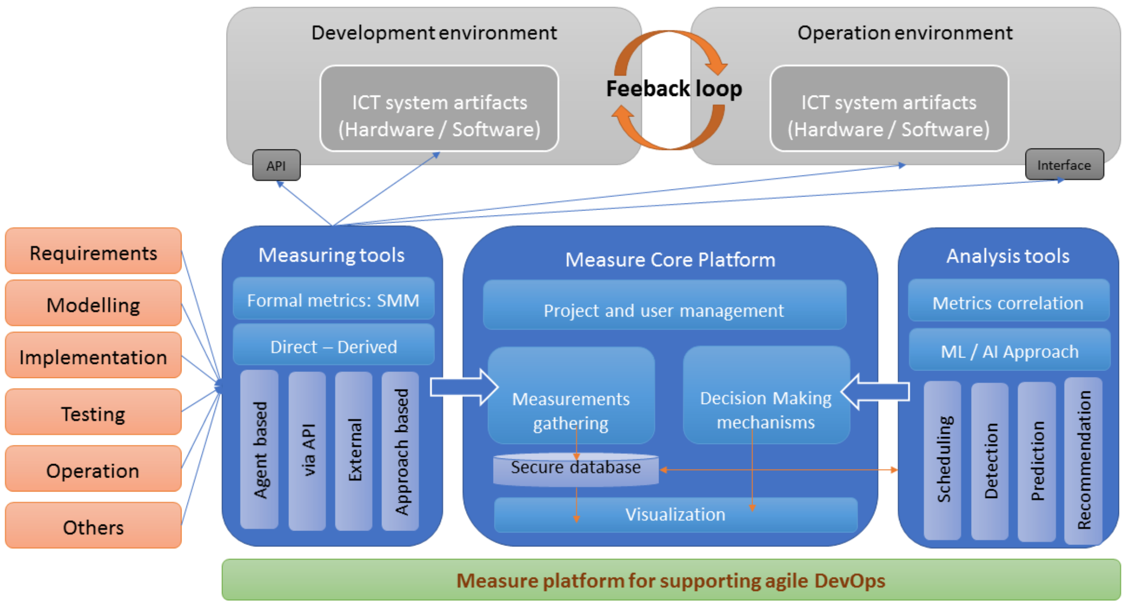

Figure 1: The Metrics-Driven DevSecOps Architecture.

The platform activity is organized around its ability to collect measurement by executing measures defined by the SMM standard. SMM measures are auto-executable components, implemented externally, which can be interrogated by the platform to collect measurements.

The Metrics-Driven DevSecOps presented in this paper consists of defining metrics and its supporting tools for measuring modern software engineering activities with respect to trustworthiness, energy efficiency and quality. The measurement process needs to be automated in order to reduce the development time. This can help the DevOps team during their decision-making. The measuring methods and tools rely on predictive and reactive security enablers that can be applied during the development and operation phases. These enablers include: security requirements specification, secure modelling (i.e. security-by-design), secure coding, vulnerability scanning, static and dynamic code analysis, continuous risk analysis, intrusion and anomaly detection, root cause analysis, and the application of resilience mechanisms at the design (called remediation mechanisms) or at the operation (called countermeasures) phases.

3.1 Measure Catalogue

The measures rely on an extension of SMM (Structured Metrics Meta-model) format, an OMG Standard. A "collect and compute component" provides services to register a "Metric" defined in specific executable format derived from the SMM standard using a communication API. Registered measures are stored in the Measure Catalogue[3]. Once registered, the measure catalogue allows the platform to access: the measure's meta-data that contains the information required to identify the measure (i.e. an unique id); the measure's type (e.g. Direct Measures, Collective Measures, Binary Measures); the information on the measure's implementation, which is the format of data returned by the measure (i.e. Unit) and a list of configuration points provided by the measure (i.e. Scope).

3.2 Measure Configuration and Instances

A Measure is composed of a set of meta-data associated with an implementation and represents a generic data collection algorithm that has to be instantiated and configured to be applied on a specific context. Consequently, in order to be executed by the platform, it's required to define an instance of a registered measure that fills the configuration values described in the measure's meta-data. As an example, a measure instance contains:

- An identifier for the measure instance.
- Information related to the collection cycle of the measurement; in other words, when and in what interval a measurement is to be collected.

[3]Examples of metrics can be found following this link: https://github.com/ITEA3-Measure/Measures/wiki

- Information required for identifying the measured system (i.e. configuration values).
- Information related to the required measure input and output for a Derived Measure.
- Information related to the data structure returned by the measure.

Once instantiated, the data related to the measure instances are stored in a centralized database.

3.3 Collecting Measures using the SMM Engine Measuring Tools

The SMM standard defines two main types of measures: (1) the direct measures, which are measures collected from the physical world, and (2) calculated measures (e.g. Collective Measures and Binary Measures), which are measures derived from the direct measures.

- Direct Measure Collecting. The SMM engine will invoke the implementation of the direct measure after having communicated to it the configuration parameters defined by the measure instantiation.
- Derived Measure Calculation. Using pre-existent measurements stored in the measurement database and from direct measures, the SMM Engine will provide services to calculate the derived measures.

The SMM Engine provides services to execute measures. The execution of a measure can be triggered manually or can be scheduled and repeated over time. In the second case, the SMM Engine uses a specific scheduling component to organise the executions of measures. Once collected or calculated, the SMM engine stores the resulting measurements in the measurement database.

3.4 Measurement Storage

Collected and calculated measurements are to be stored in a database for future exploitation by the analysis tools. In this context, it is necessary to deal with a very large quantity of data. Furthermore, due to the nature of the measures, flexibility related to the nature of the data to store in this database is also needed. In fact, it should not be necessary to make hypothesis on the nature of the measurements returned by the measurement tools. A measurement can be a simple value or a complex and structured data. Elastic Search can be a good candidate to perform this task.

3.5 Analysis Tool and Decision Support

The primary goal of the analysis is to provide a working environment driven by measures that can help managers during their decision-making process. It implements analytic algorithms, to correlate the different phases of software development and to perform the tracking of metrics and their values. The platform also connects and assures the interoperability among the tools and defines actions for improvement and possible countermeasures. The analysis tools provide a way to display graphical representations of the measured information as graphics and produce reports by defining and organizing several dashboards. Each dashboard presenting a selected pool of measures using graphics, tables and/or specific indicators. The main services offered by this component are:

- Provide a web application which allows the DevOps team (including the product owner and the project manager) to organise and configure measures computation.
- Provide a web application which will be able to present measures execution results and measurements as customisable charts.
- Allow organising these charts with respect to projects, phases and dashboards, in order to facilitate modern iterative development practices.
- Integrate consolidated results provided by the analysis tools.

One of the approach's innovation consists in analysing measurement results for identifying what and how to automatically improve the software quality or the software engineering process quality. Such highly automated and easy-to-deploy solution can be considered a breakthrough solution, as current tools only support this type of approach with a very limited scope. The analysis of retrieved measurements can rely on several algorithms for correlating them and learning from previous experiences in similar development projects. Machine learning (ML) and artificial intelligence (AI) can be used to perform clustering, measurement plan scheduling, metrics predictions and recommendations to the DevOps team in order to react to any detected/predicted issues (Dahab et al., 2019). As a concrete example, if we measure that one detected risk was classified as low (e.g., data breaches) and not mitigated at the design phase, but during operation, we detect several attacks exploiting the vulnerability resulting to this risk (data is exfiltrated). A recommendation would be to activate a security control (e.g., protect data by encrypting it or by using a secure communication channel).

4 CONCLUSIONS

The main objectives of Metrics-driven DevOps approch presented in this paper is to design and develop a centralized platform provided as PaaS to be deployed on the users premises or in a public/private cloud for building trustworthy software that rapidly adapts to changing requirements while maintaining key qualities indicators (e.g. reliability, availability, performance, security, privacy). This platform gathers measurements from the different software development lifecycle phases and during the production in order to detect/predict potential issues and prevent them. This verification relies on different software engineering tools (like risk analysis, intrusion and anomaly detection etc.) and allows providing real time recommendations to the DevOps team to improve the security/privacy of their software as well as resiliency. Different ready-to-deploy security mechanisms are needed to support such a platform.

ACKNOWLEDGEMENTS

This work is partially funded by the ongoing European project ITEA3-MEASURE started in Dec. 1st, 2015 (https://itea3.org/project/measure.html), and the H2020 ENACT project started in Jan. 1st, 2018 (https://www.enact-project.eu/).

REFERENCES

Bagnato, A., Sadovykh, A., Dahab, S., Maag, S., Cavalli, A. R., Stefanescu, A., Rocheteau, J., Mallouli, S., and Mallouli, W. (2017). Modeling OMG SMM metrics using the Modelio modeling tool in the MEASURE project. *Génie logiciel*, (120):46 – 52.

Card, D. N. (2003). Practical software measurement. In Clarke, L. A., Dillon, L., and Tichy, W. F., editors, *Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA*, pages 738–739. IEEE Computer Society.

Casola, V., Benedictis, A. D., Rak, M., and Villano, U. (2017). A security metric catalogue for cloud applications. In Barolli, L. and Terzo, O., editors, *Complex, Intelligent, and Software Intensive Systems - Proceedings of the 11th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2017), Torino, Italy, July 10-12, 2017*, volume 611 of *Advances in Intelligent Systems and Computing*, pages 854–863. Springer.

Dahab, S., Maag, S., Mallouli, W., and Cavalli, A. (2019). Smart measurements and analysis for software quality enhancement. In van Sinderen, M. and Maciaszek, L. A., editors, *Software Technologies*, pages 194–219, Cham. Springer International Publishing.

Forsgren, N. and Kersten, M. (2018). Devops metrics. *Commun. ACM*, 61(4):44–48.

Koziolek, H. (2005). Goal, question, metric. In Eusgeld, I., Freiling, F. C., and Reussner, R. H., editors, *Dependability Metrics: Advanced Lectures [result from a Dagstuhl seminar, October 30 - November 1, 2005]*, volume 4909 of *Lecture Notes in Computer Science*, pages 39–42. Springer.

Medeiros, N. P. D. S., Ivaki, N., Costa, P., and Vieira, M. (2017). Software metrics as indicators of security vulnerabilities. In *28th IEEE International Symposium on Software Reliability Engineering, ISSRE 2017, Toulouse, France, October 23-26, 2017*, pages 216–227. IEEE Computer Society.

Myrbakken, H. and Palacios, R. C. (2017). Devsecops: A multivocal literature review. In Mas, A., Mesquida, A. L., O'Connor, R. V., Rout, T., and Dorling, A., editors, *Software Process Improvement and Capability Determination - 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4-5, 2017, Proceedings*, volume 770 of *Communications in Computer and Information Science*, pages 17–29. Springer.

Tounsi, W. and Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Comput. Secur.*, 72:212–233.