

<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *2025 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2025, Honolulu, Oct 2-3, 2025*.

Citation for the original published paper:

Fucci, D. (2025)

Secure Software Engineering Through Sensible AutoMation (SESAM)

In: *International Symposium on Empirical Software Engineering and Measurement* (pp. 502-504). IEEE Computer Society

International Symposium on Empirical Software Engineering and Measurement

<https://doi.org/10.1109/ESEM64174.2025.00027>

N.B. When citing this work, cite the original published paper.

© 20XX IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:bth-29291>

Secure software Engineering through Sensible AutoMation (SESAM)

Davide Fucci

*Department of Software Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
davide.fucci@bth.se*

Abstract—Background: Security is incorporated late in the Software Development Life Cycle (SDLC), whereas early activities supporting developers in understanding and implementing security measures are difficult to integrate. **Aims:** The project focuses on empowering developers with tools and practices to seamlessly integrate security and understand the role automation plays in it. **Method:** During the project, we perform several industrial empirical studies, qualitative and quantitative, under the Design Science Research paradigm. **Results:** Our studies supporting developers to secure their software supply chain show a positive stance despite low adoption of artifacts such as SBOM and VEX. Other efforts, embedding security in GUI-based testing, are showing promising results. **Conclusion:** The project covers a broad spectrum of development activities that can be enhanced from a security perspective. Initial results show that despite developers’ interest, adoption is limited.

Index Terms—software security, technical debt, GUI testing, code reviews, design science research

I. PROJECT INFORMATION

The SESAM project (Secure software Engineering through Sensible AutoMation) is funded by the Knowledge Foundation of Sweden (KKS)¹ for three years, between November 2024 and October 2027.

The project consortium includes four partners:

- Blekinge Tekniska Högskola (BTH), project leader
- Ericsson AB, industry partner
- CodeScene AB, industry partner
- Synteda AB, industry partner

The project website is available at <https://sesam-project.github.io>.

II. PRESENTER AND CONTRIBUTORS

The presenter is the project principal investigator, Davide Fucci (davide.fucci@bth.se). Other contributors are:

- Oleksii Novikov (BTH)
- Oleksandr Adamov (BTH)
- Emil Alégroth (BTH)
- Nikhil Srivastava (Ericsson)
- Jörgen Johansson (Ericsson)
- Markus Borg (CodeScene)
- Carla Johansson (Synteda)

¹<https://kks.se>

III. PROJECT OBJECTIVES AND EXPECTED CONTRIBUTIONS

SESAM develops and evaluates a framework in which key developers’ practices are augmented to perform security activities supported by sensible automation.

In this project, we operate on the fundamental premise that developers should be able to focus on creating new (or maintaining existing) software without becoming security experts. Consequently, we address this challenge by incorporating security-related activities into a developer’s workflow with minimal disruption. In particular, together with the industry partner, we selected key development practices such as automated testing (including unit testing and GUI testing), code reviews, and technical debt management.

The focus of the proposal can be broken down into two key components. The first identifies how developers can integrate information sources to enhance security into their existing workflows, tools, and techniques. The second component aims to minimize developers’ efforts to adopt these tools and techniques through automation. Automation increases the likelihood of broader tools and techniques acceptance, for example, by delegating security decisions to intelligent software companions. However, it is crucial to be aware that automation can have unintended consequences if based on incorrect assumptions [1]. In the project, we aim at determining the appropriate extent to which automate security-focused activities for developers, and what are the trade-offs.

The project is expected to make three main contributions to the state-of-the-art. First, it will enrich the security aspects of key development practices, which, so far, have only been studied in isolation—e.g., the studies reported in [2], [3] regarding code reviews—with information gathered for different types of artifacts. Second, the project will consider the perils of automation as a first-class citizen as opposed to the current state-of-the-art, which assumes one-size-fits-all automation by default—e.g., for vulnerabilities detection [4], [5]. Third, the project will emphasize the industrial contexts, whereas existing research focused on academia, such as the studies reported in [6], [7] or open-source projects, such as in the studies reported in [8], [9].

Moreover, the project will strengthen the research profile of the partners in the area of Software Engineering that focuses

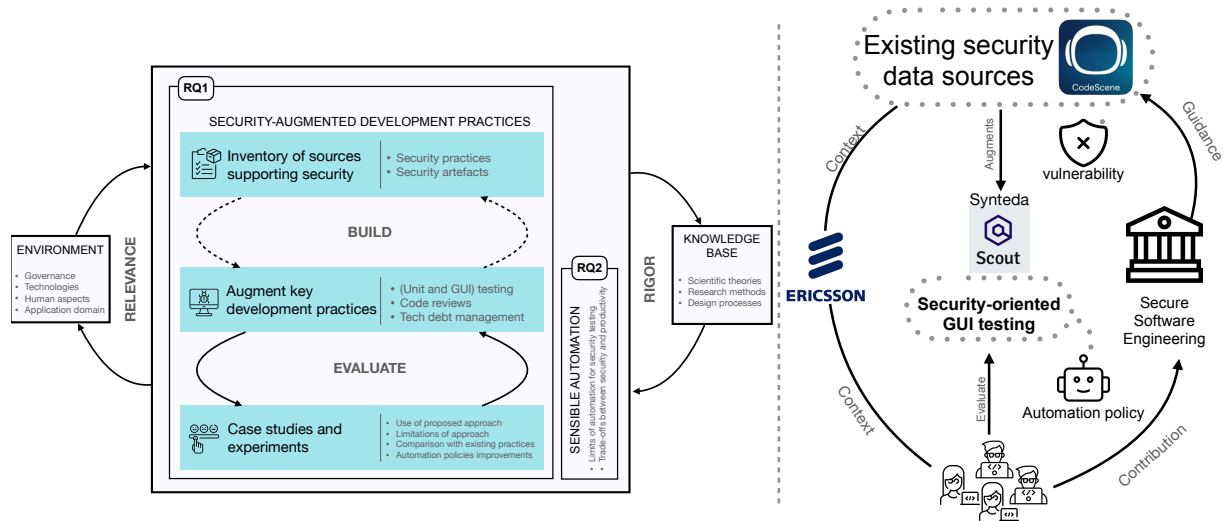


Fig. 1. General DSR cycle (left) and its exemplary instantiation within SESAM (right).

on software security, improve collaboration within the Swedish industry in this research area, and finally translate the results in educational programs.

IV. RELEVANCE TO THE ESEM COMMUNITY

The project is particularly relevant to the ESEM community for the following reasons: i) it contributes to an area of software engineering—i.e., security-by-design—which, despite impacting software professionals (e.g., developers) is currently understudied, ii) under the umbrella of Design Science Research (DSR) it applies several empirical evaluation methods (e.g., experiments, confirmatory case studies), and iii) it will make replication material available for the community. Regarding i), at the moment the SE research community interested in application security focuses, for the large part, on vulnerability management including prediction and automated fixing [10]. A smaller focus is given to topics such as threat modeling [11], security testing [12], secure code reviews [2] among others. With the SESAM project, we will foster interest in other area related to secure software engineering.

Regarding ii), the project is based on the DSR paradigm (Figure 1, left) adapted to Software Engineering by Wieringa [13] in which the design cycle alternates between build and evaluate phases. We use constructive research to co-design security-related improvements to the three selected key development practices used by our industry partners and evaluate them in real-world settings through experiments, case studies, and surveys with developers and security experts. Figure 1(right) presents a concrete scenario in which the partners are involved in design cycles to build and evaluate approaches for augmenting from a security perspective their current practice of software testing. In particular, this scenario focuses on an approach for security-oriented Graphical User Interface (GUI) testing [14], which targets the prevention of XSS vulnerabilities. From the context of one of the partners

(i.e., Ericsson), we access security-relevant artifacts including technical debt data provided by the other partner CodeScene². This data is collected, cleaned, and aggregated to create rules implement in Scout, a GUI testing tool developed by Synteda³. While performing functional testing of the system, developers can now also look for security violations due to XSS vulnerabilities, such as unexpected behaviours caused by malicious inputs provided as suggestions by the Scout tool. For evaluation, we will explicitly compare different variants of the proposed approach based on the automation policies against the current state-of-practice in the business partners' context using controlled experiments and (explanatory) case studies.

Regarding iii), we pay special attention to replicability and reproducibility, endeavours which are particularly difficult in the field of software security due to the sensitivity of such data, particularly when collected from industrial projects. Thus, ESEM represents a venue for the SESAM project to foster collaboration with external partners, both academic and industrial. Such collaborations can explore other practices not included in the project, such as architectural modeling or alternative and competitive solutions to the ones proposed for the development practices under study. Furthermore, we will advertise available artifacts—e.g., to foster replications—within the community.

V. CURRENT STATUS AND INTERMEDIATE RESULTS

The project began in November 2024. The initial weeks were dedicated to establish a reference group as a control mechanism for the project implementation—e.g., ensuring that deliverables are ready on time, initiating dissemination actions, addressing project management and administrative issues. At the same time, the consortium partners created a Data

²<https://codescene.com/product>

³<https://store.synteda.se>

Management Plan (DMP)—i.e., a living document capturing the (meta)data to be collected during the project, how to solve possible ethical and legal issues, as well as plans for data curation, preservation and sharing. Part of the initial efforts was also the creation of a dissemination plan with the objective of making sure that the results of the SESAM project are widely known beyond the project partners’ contexts. One of the activities captured in this plan is the presentation of the project at research conferences specialized tracks.

There are several research studies being developed in parallel. One of the project goal is to support developers performing security code reviews. Specifically, the industry partners are interested in developing techniques to review the software supply chain of the of their products. Accordingly, we are currently exploring how new artifacts, such as Software Bill of Materials (SBOM) and Vulnerability Exploitability eXchange (VEX) can support this task. To that end, in an initial mixed-method study [15], we show that there is little prevalence of these artifacts within open source projects (i.e., GitHub) despite project maintainers positive stance towards their usefulness. We are currently designing an internal replication of this study within one of our industrial partner.

Regarding security testing, we are currently co-developing with Synteda a plugin for Scout, their GUI testing tool, to support the use case presented in Figure 1. Specifically, we have completed a first design-evaluate round, and co-developing proposed improvements based on their feedback. Publication of the results is planned once the second round of assessment is completed.

We are also performing an initial study to understand how different types of technical debt items—measured using CodeScene state-of-practice tooling—and vulnerabilities related to each other. This initial study, currently targeting open source projects, will be replicated internally with the other project partners. Our goal with this part of the project is to provide developers an enriched view of technical debt (e.g., on how it is currently tracked in systems such as Jira) which considers security concerns.

Finally, regarding the second focus of the project—i.e., understanding the trade-offs between automation and effectiveness of tools used to address security concerns, we are currently surveying the rationale for automating Software Application Security Testing. Similarly to the studies presented above, we performed an initial DSR cycle targeting open-source project to understand the feasibility of our approach, and are performing further cycles with the consortium partners (e.g., to address relevance). The results from the preliminary work on this topic are currently under review.

REFERENCES

[1] W. K. Edwards, E. S. Poole, and J. Stoll, “Security automation considered harmful?” in *Proceedings of the 2007 Workshop on New Security Paradigms*, 2008, pp. 33–42.

[2] L. Braz and A. Bacchelli, “Software security during modern code review: the developer’s perspective,” in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 810–821.

[3] A. Edmundson, B. Holtkamp, E. Rivera, M. Finifter, A. Mettler, and D. Wagner, “An empirical study on the effectiveness of security code review,” in *International Symposium on Engineering Secure Software and Systems*. Springer, 2013, pp. 197–212.

[4] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi, “Automation of vulnerability classification from its description using machine learning,” in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[5] Y. Zhou, J. K. Siow, C. Wang, S. Liu, and Y. Liu, “Spi: Automated identification of security patches via commits,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 1, pp. 1–27, 2021.

[6] J. Parker, M. Hicks, A. Ruef, M. L. Mazurek, D. Levin, D. Votipka, P. Mardziel, and K. R. Fulton, “Build it, break it, fix it: Contesting secure development,” *ACM Transactions on Privacy and Security (TOPS)*, vol. 23, no. 2, pp. 1–36, 2020.

[7] D. Van Landuyt and W. Joosen, “A descriptive study of assumptions in stride security threat modeling,” *Software and Systems Modeling*, pp. 1–18, 2022.

[8] Y. Li, S. Wang, and T. N. Nguyen, “Vulnerability detection with fine-grained interpretations,” in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 292–303.

[9] K. Li, S. Chen, L. Fan, R. Feng, H. Liu, C. Liu, Y. Liu, and Y. Chen, “Comparison and evaluation on static application security testing (sast) tools for java,” in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 921–933.

[10] N. Shiri Harzevili, A. Boaye Belle, J. Wang, S. Wang, Z. M. Jiang, and N. Nagappan, “A systematic literature review on automated software vulnerability detection using machine learning,” *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–36, 2024.

[11] K. Tuma, C. Sandberg, U. Thorsson, M. Widman, T. Herpel, and R. Scandariato, “Finding security threats that matter: Two industrial case studies,” *Journal of Systems and Software*, vol. 179, p. 111003, 2021.

[12] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, “Security testing: A survey,” in *Advances in Computers*. Elsevier, 2016, vol. 101, pp. 1–51.

[13] R. Wieringa, “Design science as nested problem solving,” in *Proceedings of the 4th international conference on design science research in information systems and technology*, 2009, pp. 1–12.

[14] E. Alégroth, R. Feldt, and L. Ryrholm, “Visual gui testing in practice: challenges, problems and limitations,” *Empirical Software Engineering*, vol. 20, pp. 694–744, 2015.

[15] D. Fucci, M. Di Penta, S. Romano, and G. Scanniello, “Augmenting software bills of materials with software vulnerability description: A preliminary study on github,” *arXiv preprint arXiv:2503.13998*, 2025.