Raising Awareness in the Industry on Secure Code Review Practices

Andrei-Cristian Iosif
Siemens AG
Munich, Germany
email: andrei-cristian.iosif@siemens.com

Tiago Espinha Gasiba
Siemens AG
Munich, Germany
email: tiago.gasiba@siemens.com

Ulrike Lechner
Universität der Bundeswehr München
Munich, Germany
email: ulrike.lechner@unibw.de

Maria Pinto-Albuquerque
Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR
Lisboa, Portugal
email: maria.albuquerque@iscte-iul.pt

***Abstract*—As products and services become increasingly digital and software increasingly complex, all aspects of an industrial software development lifecycle must contribute to quality. Code review serves as a means to address software quality and fosters knowledge exchange across teams. Nonetheless, code review practices require resources and often require more resources than planned, while the benefit of a code review to code quality is less tangible. In our work, we address the effectiveness and efficiency of code review practices and develop an understanding of what is a good and valuable code review practice as part of a software development lifecycle. Our focus is code reviews meant to identify and address security weaknesses in an industrial context. This work presents a design study on how to design a workshop on code review. We conducted and evaluated three workshops with 37 industrial software developers. The findings of our work reveal that presenting constructive code review practices can contribute to raising awareness of secure coding and software lifecycle practices among software development professionals. This contributes to the quality and, in particular, security of software.**

***Index Terms*—code review, cybersecurity, compliance, development lifecycle, quality, standards**

I. Introduction

As more of the modern world relies on digital infrastructure, ensuring software quality is paramount. To address this issue in a tangible and standardized way, the ISO/IEC 25000 series [1] has been created as an international standard that provides guidelines and frameworks for assessing quality in software, encompassing various characteristics, such as functionality, reliability, usability, efficiency, maintainability, and portability. Among these quality aspects, security is a critical domain, as it ensures the protection of sensitive data and prevention and protection against potential cyber threats. The ISO 27000 series explicitly addresses information security management systems, with ISO/IEC 27001 [2] playing an essential role in establishing and maintaining a comprehensive security framework within organizations.

Cybersecurity threats are constantly evolving, and the security of critical infrastructures is at risk. In developing software for Operational Technology, security is an essential requirement. Code reviews are one of the measures to detect weaknesses in the code and address other quality aspects, such as adhering to standards or policies. In contemporary software development practices, industry practitioners are tasked with creating functioning code, adhering to established standards, and integrating their work within the company's adopted software development lifecycle pipeline without considerable overhead. There is a growing push for standardized processes within companies to ensure consistent and efficient practices in the software industry. Compliance with established standards is crucial for maintaining code quality and security. One way of easing this goal is by employing code review focusing on software security.

To achieve this, exposing practitioners to the concepts and principles behind the code review process is imperative. By providing practitioners with a clear understanding of methodologies, organizations can foster a culture of compliance, enhance code quality, and promote a cohesive approach to software development that adheres to industry best practices. This endeavor requires disseminating knowledge across all organizational levels, ensuring a shared understanding of best practices and methodologies. Interactive workshops serve as effective communication channels for achieving this goal, as participants can engage with theoretical concepts and solidify their understanding through hands-on exercises. By fostering a collaborative learning environment, such workshops empower industry practitioners to enhance their technical skills, align with organizational standards, and contribute to the overall betterment of the software development process.

On the other hand, automation technologies and especially DevSecOps, are promising methods in reducing the human load in auxiliary programming tasks, such as software testing, according to Sánchez-Gordón et al. [3]. Nonetheless, according to Mao et al., [4], these emerging fields come with limitations – Static Code Analysis Tools (SAST), for example, produce reports that often include many false positives that need human attention for filtering. The perceived gain in testing coverage and less human involvement may therefore be lost through manual filtration and, more concerning, distract developers from the false negatives, which are not included in the output of SAST tools, and often require deep insight into the code. Manual code review should, therefore, not be disregarded in security-critical applications, as a professional's scrutiny can catch architecture-level bugs and vulnerabilities, whereas tools often fail, as shown by Kupsch et al. [5].

According to our experience, decision-supporting and pro-

 ISBN: 978-1-68558-113-8

cess tools must be enriched with human insight to ensure that the software development lifecycle benefits from the team's expertise behind the software products being developed and delivered. For this purpose, our current work aims toward developing and evaluating an educational medium suitable for the industry in which knowledge about code review, including current standards and practical takeaways, can be disseminated effectively.

This paper presents the initial design of a workshop to train industrial software developers in code review. We also present empirical results on the workshop's evaluation and the participants' awareness of software development. The present work poses and systematically addresses the following research questions:

RQ1 What elements of the workshop contribute to raising awareness on security when performing code review, and are important and helpful for the participants when conducting a workshop on code review?

RQ2 How can code review in practice be improved?

RQ3 How do practitioners receiving training on security awareness compare against SAST tools?

Our work follows Action Design Research (ADR) methodology principles, and the present results shall serve as the first step of ADR, with further refinement to be carried out in the following, future design iterations.

The article is organized as follows: Section II provides an overview of related work on code review and the standards that may govern it. Section III follows by presenting the employed methodology. The intervention is then presented in Section IV. Based on the collected results, a discussion is presented in Section V. Finally, Section VI reiterates our work and presents further potential research directions based on the conclusions reached in the present work.

II. Related Work

The foundations of this work rely on established code review standards and workflows, as well as the international standards that govern code review and software security. To elaborate on the workshop's contents, a literature survey was conducted on these two topics, and each of the two shall be discussed in the remainder of this section.

A. Code Review

Code review is a practice that is well established within the software development lifecycle, with one of the first works on formalizing this process being formulated by Fagan [6], [7] – where the author highlights code inspections in mitigating errors during program development. The research emphasizes how early identification and rectification of defects through inspection processes leads to improved software quality and increased productivity.

More recent research in the industry indicates that practitioners consider a review beneficial primarily if the review comments lead to improved code quality, as per Bosu et al. [8]. In another study, MacLeod et al. [9] looked at the defining characteristics of a code review that is perceived as useful by the individuals that changed the code. Their findings highlight the challenges of code review, e.g., the improper focus of the review, uncertainty about the process, and lack of formal training. The authors also provide recommendations for best practices for all stakeholders – reviewers, change authors, and the organization itself.

In his research, W. Charoenwet [10] conducted work on integrating automated program security analysis into the code review workflow. The work acknowledges the limitations of automated findings and plans to design a review-assisting framework based on tool findings. While similar in focus, the research does not account for security standards and is not explicitly targeted at the industrial work practices.

As previously mentioned, the studies serve as a consistent foundation of insights into the benefits and pitfalls of code review, pointing towards the potential trade-offs that need to be considered when opting to integrate code review in an industrial software development context. Though comprehensive, the studies point to a gap in providing the necessary knowledge for developers to perform a code review that is considered beneficial within their teams.

Our work complements the existing body of knowledge on code review by specifically focusing on reviews that aim predominantly at reducing security flaws in code and which follow the current standards. To the best of our knowledge, there is no other ongoing research on raising industrial practitioners' awareness of cybersecurity-focused and standard-compliant code reviews. We aim to heighten practitioners' security awareness without having them rely on decision-supporting tools.

B. Standards

The three standards we broadly covered as part of the workshop presented in this work will be introduced in the remainder of this section.

The ISO/IEC 62443 is an international series of standards that address cybersecurity for Industrial Automation Control Systems (IACS). The standard is divided into sections and describes technical and process-related aspects of automation and control systems cybersecurity. Furthermore, the standard divides the cybersecurity topics by stakeholder category and roles (e.g., operator, service providers, component and system manufacturers). The different roles each follow a risk-based approach to prevent and manage security risks in their activities. This standard includes code review as part of its Secure Development Lifecycle (SDL) requirements for products intended for use in the industrial automation and control systems environment. Specifically, the IEC 62443-4-1 [11] outlines process requirements for the secure development of products used in IACS. One important aspect is that the standard mentions that the people performing the review should have specialized knowledge. This requirement leads to the need for training and awareness, not only for software developers but also for the parties involved in code review. The IEC 62443-4-2 [12] details low-level technical requirements that need to be fulfilled when implementing software for IACS. These

 ISBN: 978-1-68558-113-8

requirements must also be taken into consideration during code review.

The ISO/IEC 20246 [13] standard establishes a generic framework for work product reviews. It can be referenced and used by all organizations managing, developing, testing, and maintaining systems and software. The standard contains a generic process, activities, tasks, review techniques, and documentation templates applied during the review of a work product, i.e., any artifact produced by a process. This document defines work product reviews that can be used during any phase of the life cycle of any work product. It is intended for parties involved across all levels – i.e., project managers, development managers, quality managers, test managers, business analysts, developers, testers, customers, and others involved in developing, testing, and maintaining systems and software. Following a code review standard is vital during the SDL process, audits, and accreditation.

From a programming-language perspective, the ISO/IEC TR 24772-1 [14] standard provides low-level guidance to avoiding vulnerabilities in programming. This standard is programming-language agnostic. It specifies vulnerabilities to be avoided in developing systems where assured behavior is required for security, safety, mission-critical, and business-critical software. Another broad secure coding standard is provided by MITRE [15] through the Common Weakness Enumeration (CWE). Our work is based on the 2023 release of this standard. The Open Web Application Security Project (OWASP) releases the OWASP Top 10 [16] standard on a regular basis that is specific to software developed for the web. Our work is based on the 2021 release of this standard.

III. Methodology

The research design is guided by the Action Design Research (ADR) method. As stated by Sein et al. [17], ADR is a research method for generating prescriptive design knowledge through building and evaluating ensemble IT artifacts in an organizational setting. It deals with (1) addressing a problem encountered in a specific organizational setting by intervening and evaluating and (2) constructing and evaluating an IT artifact that addresses the class of problems typified by the encountered situation. ADR is done in close collaboration between researchers and practitioners in an iterative way in which problem understanding, action-taking, and evaluation are closely intertwined.

This research is situated in an industrial software development context in which cybersecurity is paramount. We aim to generate knowledge on code review practices to increase code review effectiveness and to raise awareness for security and code review. Designing and evaluating instruments for training industrial software developers are critical activities in our research design. Two researchers are embedded in an industrial software development context and dispose of several years of experience in industrial software development and secure coding. This industrial context is open to collaborating with academia to bring in new ideas for individual and organizational learning and rigorous evaluation of current practices. In this article, we describe the results from activities undertaken in industrial practice to design a workshop format and content to raise awareness for code review as a practice in the software lifecycle. We report on three workshops and the evaluation of data from the workshops.

The first step in our research process was identifying relevant scholarly literature and findings, de-facto standards and norms on code reviews, software quality, and the software lifecycle. Then, relevant content was selected and tailored to the organization's needs, and a workshop was designed. We developed original code review exercises to be part of this workshop to activate workshop participants and foster transfer from the workshop practice to the everyday job situation.

We conducted three workshops as three separate events with 37 participants between May and June 2023. These workshops included semi-structured interviews for evaluation of the code review training. We analyze data from the interviews, together with the participatory observations. The intervention was conducted in the context of industrial training aimed at professionals in software development for operational technology that are not specialized in cybersecurity.

The three workshops had between 7 and 16 participants. Workshop participants aged between 21 and 63, each with a technical background. The professional functions varied – each of the three workshops included representatives for software developers, project managers, and product owners.

Following the workshop, we conducted semi-structured interviews. All the participants were informed clearly about the purpose of the study. Moreover, it was noted that participation in the review was optional, and the collected results shall be collected anonymously. In total, 20 individual feedbacks were collected. The survey was conducted online via Microsoft Forms, with the participants being granted indefinite access to the questionnaire at the end of the workshop. The interviews conducted in our work follow a semi-structured methodology, as per Wilson et al. [18] – the questions delivered to the participants were aimed towards assessing the workshop's content in terms of balance between the topics it spanned across. The questions are not directly addressing security but rather the workshop design itself. Following the principles of action design research, the initial phase of workshop planning is meant mainly to collect information for further design steps.

IV. Intervention – The Code Review Workshops

A. Design of the Code-Review Training Workshop

The workshop was designed to last a day with a target group of software developers and people holding managerial positions. The delivery method was designed to be suitable for both on-site and remote settings. A vital element of the workshop was an exercise consisting of Python code which contained several security vulnerabilities. The goal of the exercise was for the participants to spot these vulnerabilities through code review. The exercise allowed the participants to put to practice the theoretical concepts that were learned during the workshop.

 ISBN: 978-1-68558-113-8

The covered topics included: a taxonomy of common software vulnerabilities, code review standards, and available security and review tooling, as well as practical examples of exploitation, mitigation, and reviewing methodology. The designed workshop was conducted three times in an industrial setting. Table I summarizes the three interventions. A questionnaire followed each intervention, the contents of which can be observed in Table II.

TABLE I
INDUSTRY INTERVENTIONS

IN	Date	NP	No. CWEs	Delivery	Participants from
1	3.05.2023	14	21	Online	Germany, India
2	25.05.2023	7	21	Online	Germany
3	12.06.2023	16	21	On-site	United Kingdom

IN - Intervention Number, **NP** - Number of Participants

The first two interventions took place online through Microsoft Teams with participants mainly from Germany. During the first workshop, four participants joined the online meeting from India. The third workshop took place on-site in the United Kingdom. We conducted a small survey at the end to evaluate the workshop design. Additional details on the survey are provided in Section IV-C.

B. Design of the Exercise

The exercise was developed to engage participants, as these were tasked to work as a group, with a time limit of 15 minutes, and spot as many potential flaws in the code as possible. Participants were also asked to use the SAST tool *Bandit* [19] to enhance their code review. A discussion based on their findings accompanied the exercise, with opinions being exchanged about what lessons could be drawn.

As this workshop is taking part in an industrial context, a requirement for the delivery was to traverse the contents of the training in a manner with adequate pacing for the given time constraints given for disseminating information to participants.

We, therefore, opted for a compact code snippet with a high density of defects per Line of Code (LoC) - see Appendix. As the participant's programming background was polyglot, we opted for a Python Flask Web Application to serve as a snippet. This choice is due to the popularity and readability of the Python language [20], as well as the rising prevalence of web applications overall – see Collins [21]. Nonetheless, a codebase's security is a quality metric that is considered non-negotiable, according to our experience in the industry as security researchers, irrespective of the type or scope of application – no application meant for an end-customer can be insecure and standards-compliant at the same time.

The specially crafted code, although relatively small, contains a high number of vulnerabilities (22) - this is meant to cover as much of OWASP Top 10 [16] and CWE [15] through the breadth of exposure. Excluding blank lines, `include` statements, and rewrapping a multi-line statements, the snippet comprises thirty-six lines of code. Of the thirty-six LoC, expert security professionals managed to uncover 20 vulnerabilities. This observation translates into an average of 0.55 vulnerabilities per LoC. The annotated snippet is provided as an Appendix to the paper. The exercise targets code defects rather than architectural defects, as the architecture design and planning should be carried out earlier in the lifecycle of an application, and the majority of industry reviews mostly concern themselves with code rather than the underlying architecture.

The exercise under scrutiny serves as a way of quantifying the *behavior* aspect from Hänsch et al. [22] in the broader context of the workshop. In the context of the present study, *Perception* refers to disseminating knowledge of possible issues, *Protection* addresses knowledge of possible countermeasures, and *behavior* describes the observed performance of the participants during practical exercises. Gaining insight into the proposed **RQ**s is intended to lead to a well-designed workshop, such that *Perception*, *Protection*, and *Behavior* of the practitioners can be improved.

C. Empirical Evaluation

The evaluation is done in a semi-structured interview. In the evaluation of the workshop content and perceived usefulness of the workshops to the participants, we refer to the definition of awareness as given by Hänsch et al. [22], who structure awareness into three constituent components: perception (knowledge of issues), protection (available countermeasures) and behavior (individuals actively employing countermeasures).

We group the questions into four distinct categories, as per Table II: keep factors (**K**), reject (discard) factors (**R**), delivery of the content (**D**), and perceived value (**V**). All questions were delivered via Microsoft Forms, with the answering options consisting of freeform text, except for questions **Q11** and **Q12**, which employed a 5-point Likert [23] scale that spanned between "Very unlikely" to "Very likely" and "Very dissatisfying" to "Very satisfying" respectively.

Refering to our driving research questions, answering **RQ1** can be done by observing the answers to the questions pertaining to the keep factors (**K**), as well as the discard factors (**R**) serving as counter-examples.

Observing what participants rank as valuable from the training can provide insight into **RQ2**, with their answers serving as tangible measures of what can be done to improve code review in practice.

V. EVALUATION RESULTS

In total, 20 individual feedback results were collected from the structured interviews across the three interventions.

Regarding keep factors, practical examples were the most mentioned among the participants' answers, with seven mentions across the 20 collected answers.

In terms of discard factors, participants consistently mentioned the medium of delivery for the exercise, as usage of the platform sometimes hindered the intended goal of the exercises. The employed medium for conducting the exercise was Microsoft Conceptboard, a generic collaboration and

 ISBN: 978-1-68558-113-8

TABLE II
SURVEY QUESTIONS

QN	Category	Question
Q1	K	What would you keep from this training?
Q2	R	What would you discard from this training?
Q3	K	What did you find most useful/valuable in this presentation/training?
Q4	D	Was the content presented clearly and effectively?
Q5	R	Were there any parts of the presentation/training that you found confusing or difficult to understand?
Q6	K & R	Did the presentation/training meet your expectations? If not, what could have been improved?
Q7	K	Were there any specific examples or case studies that resonated with you?
Q8	D	Did you feel the presentation/training was engaging and interactive enough?
Q9	V	Did the presentation/training meet your learning objectives?
Q10	K	Were there any additional topics or areas you would have liked to see covered in this presentation/training?
Q11	V	How likely are you to recommend this presentation/training to others?
Q12	V	Overall, how would you rate the quality of the presentation/training?

QN - Question Number, **K** - Keep, **R** - Reject, **D** - Delivery, **V** - Value

brainstorming platform. Approximately half of the participants also expressed a lack of familiarity with the platform, which might influence the perception in this regard. Other negatively received aspects of the training were advanced concepts that were mentioned, but not sufficiently addressed, specifically formal verification and advanced pentesting techniques as supplementary means of code testing. Furthermore, through answers on **Q10**, participants expressed interest in the following additional topics/aspects: receiving supplementary practical guidance, such as a sample code review report and strategies in finding issues.

Focusing on the answers to question **Q3**, which inquires about the most useful part of the presentation, the participants' answers can be clustered into two main categories: practical examples and standards. This clustering is in line with the expectations considering their demographic, as the audience consisted of a balanced crowd of developers and managerial and organizational professionals.

The positive results collected from **Q11** and **Q12** encourage the pursuit of further design cycles – Recommending the training to others was responded to with 50% *Very likely*, 41% *Likely* and 9% *Neutral*; The quality of the training was considered by 58% of the participants to be *Very Satisfying*, 25% reported it as *Satisfying*, and 17% *Neutral*.

In terms of delivery, the participants considered the overall format adequate, sans the delivery method for the exercises. The workshop also contained a dedicated section of *Do's* and *Don't* regarding code review comments, where best practices were suggested. As coding style is often highly a matter of personal preference, some debate was expected. This discussion occurred organically across all three interventions, initiated from the participants' side, and denotes a high degree of implication and opinion, which proved valuable in opening up the workshop for fruitful open discussion concerning what affects and does not affect code security. Through open dialogue, a consensus was reached that coding style does not impact security as long as readability is not negatively impacted.

In addition to the questions from Table II, participants were encouraged to provide additional thoughts they considered worth sharing. This free-form feedback was observed to be more brief and sparse than its semi-structured counterpart.

Some examples of replies from the participants include:

- *"Thank you, I really liked it and am happy to get the slides as a reference for later."*
- *"Consider adapting the exercises to the Conceptboard."*
- *"Keep up the great work. Thanks.*"

In order to do a preliminary exploration of **RQ3**, we introduce the metrics collected from the practical exercise in Table III.

TABLE III
NUMBER OF IDENTIFIED VULNERABILITIES

IN	PF	EF	TF
1	11	22	5
2	12	22	5
3	14	22	5

IN - Intervention Number, **PF** - Participant Findings
EF - Expert Findings, **TF** - Tool Findings (SAST tool Bandit)

None of the participants had cybersecurity knowledge or familiarity with Python Flask applications for all three intervention runs. Referring to the discussion following the exercise, participants were positively impressed to see how they outperformed the SAST tool provided to them, considering the group's cumulative experience and the exercise's time limit. Observing the limitation of automated security testing was beneficial to the participants' general awareness of vulnerabilities and the threat of false negatives from tools.

It is interesting to point out that the participants' findings constantly ranked, across all interventions, at approximately the halfway point between the number of findings from the tools and the number of findings from security experts. This indicates that any party involved in the software development lifecycle may positively impact the security of the work product, even after just minimal training.

Furthermore, based on the discussion following the exercise, the participants reached an intended conclusion: deep familiarity with the technology under review (programming language and libraries) is necessary to uncover all underlying security issues fully.

A. Threats to validity

The results of this work rely on data interpreted from the collected from surveying a total of 37 working professionals. The limited number of participants and variance in their backgrounds may introduce bias in the conclusions. Preliminary results are positive for the most part, which could partly be influenced by the voluntary nature of the survey

 ISBN: 978-1-68558-113-8

and participant number – a negative reviewer may opt out of feedback submission, and it is typical for participants that are aware of the purpose of a study's purpose to display positive bias.

Nevertheless, the results of this work are in line with previous and related work in the field of security awareness conducted in the industry. This fact, together with the inherent limitation of design science studies carried out in an industrial context, leads us to regard that the formulated conclusions of the present study would not be subject to significant variation if the participant number had been higher.

According to the design science paradigm by Hevner et al., [24], we are dealing with a *wicked problem* – the requirements dealt with in the industry and in practice are unstable and changing, as conducting the interventions cannot be done with respect to a control group, and the demographics of the participants and their capabilities are changing from one run to the other. In this case, conducting a quantitative measurement would constitute a tedious endeavor, and we, therefore, employ the active design methodology laid out by Hevner et al.

As ADR employs iteration across multiple design cycles, the authors would like to expand the participant pool through industrial interventions to gain more refined insight and validate the current results.

B. Lessons Learned

Through investigating **RQ3**, we conclude that SAST tools do not cover all the aspects of code review. Coding guidelines contain both decidable and non-decidable problems – this translates into automated assessment being a helpful decision-supporting mechanism but not a final solution for the adherence to standards.

The false sense of security provided by the tools also translates into a more relaxed attitude during manual code review – practitioners have been observed to sometimes overlook the *banal* security malpractices that are reported by SAST tools while focusing on the more subtle defects introduced in the snippet presented in the exercises.

The series of industrial interventions was conducted together with audiences that were heterogeneous in terms of knowledge and professional role. We have found that our generalist approach to the content of the workshop was suitable for this context, as focus and time could be reallocated dynamically throughout the workshop to suit better the gaps in each of the audience's awareness.

VI. Conclusions and Further Work

As experience in the industry includes adhering to rigorous standards, practitioners at all layers of the software development lifecycle must be made aware of the aspects governing processes that are part of adhering to current standards. In our study, we have observed that, though code review has been established, it is not yet widespread. Although the ISO 20246 standard on code review found its roots in 2008, through IEEE 1028, general awareness of its practice was lacking in our study group. With the inclusion and actualization of formalized code reviews within ISO 62443, companies are sure to include more code reviews in their internal processes. Raising awareness concerning this standard is therefore needed in the industry.

To address this issue, we propose raising awareness of security issues through code review by means of an interactive workshop. In our training, the participants are given a snippet of vulnerable code and are tasked with finding issues within it, similar to a real-life review.

Our work contributes to understanding how to structure a workshop in the industry to address the participants' needs. We evaluated the workshop through semi-structured interviews spanning three separate interventions, throughout which 37 practitioners participated, and feedback from 20 individuals was collected. Preliminary results indicate that the workshop's content, with emphasis on the practical side, was well received. Having the participants formulate some of the workshop's intended takeaway points during follow-up discussions of the hands-on parts of the training serves as substantial validation that the exercises contained in the workshop are fulfilling their purpose of building and raising awareness on code review in the context of software security.

In further work, the authors would like to refine the implementation of the practical exercises toward a serious game. Based on the feedback, the new iteration shall cover elements that would steer the participant's information toward the content while keeping interface and interaction elements at a minimum necessary. Accounting for the current studies' participants, future content may be added in the direction of an appendix of tailored code review checklists based on the intended objective of the review (e.g., audit, release). Furthermore, the authors would like to assess practitioners' current awareness of the interplay between code review and software security by employing a representative survey.

Acknowledgments

This work is partially financed by Portuguese national funds through FCT–Fundação para a Ciência e Tecnologia, I.P., under the projects FCT UIDB/04466/2020 and FCT UIDP/04466/2020. Furthermore, Maria Pinto-Albuquerque thanks the Instituto Universitário de Lisboa and ISTAR, for their support. We acknowledge funding for project CONTAIN by the Federal Ministry of Education and Research under project numbers 13N16581 and 13N16585.

The authors would like to thank Anton Bartl for the useful discussions and provision of the original starting idea for the vulnerable code snippet used in the study.

References

[1] ISO/IEC 25000:2014, "Systems and software engineering – systems and software quality requirements and evaluation (square) – guide to square," International Organization for Standardization, Geneva, CH, Standard, 2014.

[2] ISO/IEC 27001:2013, "Information technology – Security techniques – Information security management systems – Requirements," International Organization for Standardization, Geneva, CH, Standard, 2013.

 ISBN: 978-1-68558-113-8

[3] M. Sánchez-Gordón and R. Colomo-Palacios, "Security as culture: A systematic literature review of DevSecOps," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: ACM, 2020, p. 266–269. [Online]. Available: https://doi.org/10.1145/3387940.3392233

[4] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, H. Shen, L. Chen, and K. Lu, "Preliminary Findings about DevSecOps from Grey Literature," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, 2020, pp. 450–457.

[5] J. A. Kupsch and B. P. Miller, "Manual vs. Automated Vulnerability Assessment: A Case Study," in *First International Workshop on Managing Insider Security Threats (MIST)*, 06 2009, pp. 83–97. [Online]. Available: pages.cs.wisc.edu/∼kupsch/va/ManVsAutoVulnAssessment.pdf

[6] M. Fagan, "Design and code inspections to reduce errors in program development," *IBM Systems Journal*, vol. 38, no. 2.3, pp. 258–287, 1999.

[7] M. Fagan, "A History of Software Inspections," in *Software Pioneers*. Springer Berlin Heidelberg, 2002, pp. 562–573.

[8] A. Bosu, M. Greiler, and C. Bird, "Characteristics of Useful Code Reviews: An Empirical Study at Microsoft," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 146–156.

[9] L. MacLeod, M. Greiler, M.-A. Storey, C. Bird, and J. Czerwonka, "Code reviewing in the trenches: Challenges and best practices," *IEEE Software*, vol. 35, no. 4, pp. 34–42, 2017.

[10] W. Charoenwet, "Complementing Secure Code Review with Automated Program Analysis," in *Proceedings of the 45th International Conference on Software Engineering: Companion Proceedings*, ser. ICSE '23. IEEE Press, 2023, p. 189–191.

[11] ISO/IEC 64223-4-1:2018-1, "ISO/IEC 62443-4-1:2018 Security for industrial automation and control systems - Part 4-1: Secure product development lifecycle requirements," International Organization for Standardization, Geneva, CH, Standard, 1 2018.

[12] ISO/IEC 64223-4-2:2019-12, "Security for Industrial Automation and Control Systems - Part 4-2: Technical Security Requirements for IACS Components," International Electrical Commission, Geneva, CH, Standard, 1 2019, ISBN 978-2-8322-6597-0.

[13] ISO/IEC 20246:2017, "Software and systems engineering – Work product reviews," International Organization for Standardization, Geneva, CH, Standard, 2017.

[14] ISO/IEC TR 24772-1:2019, "Programming languages – Guidance to avoiding vulnerabilities in programming languages – Part 1: Language-independent guidance," International Organization for Standardization, Geneva, CH, Standard, 2019.

[15] "CWE Top 25 Most Dangerous Software Weaknesses," https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html, MITRE Corporation, 2023, online, accessed 2023.07.24.

[16] "OWASP Top 10 - 2021," https://owasp.org/Top10/, OWASP Foundation, 2021, online, accessed 2023.07.24.

[17] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, "Action Design Research," *MIS Quarterly*, vol. 35, pp. 37–56, 2011.

[18] C. Wilson, "Semi-Structured Interviews," in *Interview Techniques for UX Practitioners*. Burlington, Massachusetts, USA: Elsevier, 2014, pp. 23–41. [Online]. Available: https://doi.org/10.1016/b978-0-12-410393-1.00002-8

[19] "Bandit," https://pypi.org/project/bandit, PyCQA, 2023, online, accessed 2023.07.24.

[20] GitHub, "Why Python Keeps Growing, Explained," https://github.blog/2023-03-02-why-python-keeps-growing-explained/, March 2023, online, accessed 2023.07.24.

[21] V. Collins, "Why You Don't Need to Make an App: A Guide for Startups Who Want to Make an App," https://www.forbes.com/sites/victoriacollins/2019/04/05/why-you-dont-need-to-make-an-app-a-guide-for-startups-who-want-to-make-an-app/, April 2019, online, accessed 2023.07.24.

[22] N. Hänsch and Z. Benenson, "Specifying IT Security Awareness," in *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*. Munich, Germany: IEEE, 12 2014, pp. 326–330.

[23] R. Likert, "A Technique for the Measurement of Attitudes." *Archives of psychology*, vol. 22, no. 140, pp. 1–55, 6 1932, . [Online]. Available: https://legacy.voteview.com/pdf/Likert_1932.pdf

[24] A. Hevner, S. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *Management Information Systems Quarterly*, vol. 28, pp. 75–, 03 2004.

APPENDIX - VULNERABLE FLASK APP

Listing 1. Code snippet used for manual review

```
import sqlite3, random
from flask import Flask, abort, request, jsonify
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
database = './login.db'

def create_response(message):
    response = jsonify({'message': message})
    response.headers.add('Access-Control-Allow-
        Origin', '*')
    # TODO Ticket: id91263
    return response

@app.route('/setup', methods=['POST'])
def setup():
    connection = sqlite3.connect(database)
    SECRET_PASSWORD = "letMeIn!";
    THIS_IS_A_VARIABLE = "WBneKJw1fHch8Qd3XFUS";
    print("Super Secret Password SSH Server Password
         to 10.10.10.1:22: " + SECRET_PASSWORD)
    connection.executescript('CREATE TABLE IF NOT
        EXISTS login(username TEXT NOT NULL UNIQUE,
        password TEXT NOT NULL);INSERT OR IGNORE
        INTO login VALUES("user_1","123456");')
    return create_response('Setup done!')

@app.route('/login', methods=['POST'])
def login():
    username = request.json['username']
    password = request.json['password']
    connection = sqlite3.connect(database)
    cursor = connection.cursor()
    cursor.execute('SELECT * FROM login WHERE
        username = "%s" AND password = "%s"' % (
        username, password))
    user = cursor.fetchone()
    if user:
        response = create_response('Login successful
            !')
        response.set_cookie('SESSIONID', str(random.
            randint(1,99999999999999999999)),
            httponly=False,secure=False)
        response.set_cookie('TESTID1', str("
            TESTSTRING1"), httponly=True,secure=True
            )
        response.set_cookie('TESTID2', str("
            TESTSTRING2"))
        return response
    else:
        response = create_response('Login failed!')
        response.delete_cookie('username')
        return response, 401

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080, debug=True)
```