CyberSecurity Challenges for Software Developer Awareness Training in Industrial Environments

Tiago Espinha Gasiba
Siemens AG, Universität der Bundeswehr München, Germany

Ulrike Lechner
Universität der Bundeswehr München, Germany

Maria Pinto-Albuquerque
Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR-IUL, Lisboa, Portugal

CyberSecurity Challenges for Software Developer Awareness Training in Industrial Environments

Tiago Gasiba[1,2], Ulrike Lechner[2], and Maria Pinto-Albuquerque[3]

[1] Siemens AG, Munich, Germany
tiago.gasiba@siemens.com
[2] Universität der Bundeswehr München, Munich, Germany
tiago.gasiba@unibw.de ulrike.lechner@unibw.de
[3] Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR, Lisboa, Portugal
maria.albuquerque@iscte-iul.pt

Abstract. Awareness of cybersecurity topics facilitates software developers to produce secure code. This awareness is especially important in industrial environments for the products and services in critical infrastructures. In this work, we address how to raise awareness of software developers on the topic of secure coding. We propose the "CyberSecurity Challenges", a serious game designed to be used in an industrial environment and address software developers' needs. Our work distills the experience gained in conducting these CyberSecurity Challenges in an industrial setting. The main contributions are the design of the CyberSecurity Challenges events, the analysis of the perceived benefits, and practical advice for practitioners who wish to design or refine these games.

Keywords: Cybersecurity, Serious Games, Awareness, Industry, Capture-the-Flag · Education

1 Introduction

Over the last years, the number of industrial security-related incidents, e.g., reported by the ICS-CERT [8], has been steadily increasing. When malicious parties exploit security vulnerabilities present in products and services, the outcome of its exploitation has serious negative consequences for society, the customers, and the company that produced the software. Think, e.g., of critical infrastructures as the grid, transportation, or production lines: a security vulnerability in the code may cause interruptions in service quality or cause safety issues for society or individual customers when critical machinery fails. Several efforts can be made to increase the level of security in critical infrastructures. These efforts include, among others: analysis of threat and risks, implementing a secure software development lifecycle process, deployment of static application security testing tools, code reviews, and training. This paper addresses the software vulnerabilities through awareness training of software developers in the industry, based on a serious game: the CyberSecurity Challenges (CSC). Serious

16th International Conference on Wirtschaftsinformatik,
March 2021, Essen, Germany

Games are games that are *designed for a primary purpose other than pure entertainment* [9]. The serious game "CyberSecurity Challenges" (CSC) aims at raising awareness of secure coding topics among industrial software engineers. In this game, software developers are trained to spot security vulnerabilities in software and write secure code. i.e., code that is free from known vulnerabilities and adheres to secure coding policies. Previous work introduced the CyberSecurity Challenges from a theoretical point-of-view [11,16] and focused on particular aspects [15]. The current work extends previous publications by a presentation of a unified view on the design process, tailoring to the industry's needs and the perceived usefulness of the CSC events. Our results are based on data from several CSC events held in the industry from 2017 to 2020. As such, the main contributions of this work are:

- **CSC Artifact**: consolidated view of the design and deployment of CSCs, based on results from thirteen events held in an industrial context, and
- **CSC Evaluation:** analysis of results from industry events covering the following aspects: adequacy of CSC as a means to raise secure coding awareness, impact of CSC on software developers, and success factors for CSC events.

This paper aims to guide practitioners who wish to develop or refine a software developer awareness training in an industrial context, provide a solid reference to the research community who wishes to address serious games for the industry, and close the existing literature gap. This work is organized as follows. In the following section, we give a summary of the game idea and logic of CSC games. Section 3 presents related work. In Section 4, the research method and research questions are introduced. The unified view of the CSC artifact is presented in Section 5. Section 6 presents a summary of the survey results, together with critical discussions. Finally, Section 7 concludes the paper with an outlook of next steps.

2 Cybersecurity Challenges at a glance

The CyberSecurity Challenges (CSCs) are a serious game, designed to raise awareness for cybersecurity topics among industrial software engineers. A CSC game consists of several challenges designed to raise awareness on secure coding guidelines and secure coding on software developers. These challenges are oriented towards improving the defensive skills of the participants. Defensive challenges are challenges that help the players write code that has no (known) vulnerabilities and adheres to secure coding guidelines.

The Capture-the-Flag genre was the original inspiration for the game. Capture the-Flag (CTFs) are associated with offensive skills, e.g., system penetration, and reverse engineering, and they can often last hours or even days [23]. Unlike CTF games, which teach the participants to attack and break into systems, CSC focus on improving skills to write and develop secure code. These games thus have no intention to cause any harm or inspire unlawful actions – they are about "defensive" skills. The challenges are composed of C, C++, Java, and Web exercises. The focus on these programming

languages and genre inspiration is rooted in internal demand for training and internal decisions taken in the company where CSC is developed. Thus, the games are designed to match software developers' interests and organizations' needs for developer training. This interest can be motivated by several factors, e.g., the need to show due diligence and certification purposes.

The CSC event is delivered as a single event (Standalone type) or after a workshop on secure coding (Workshop type). In both cases, the duration of the event is designed to fit a single working day. During the game, the participants solve secure coding challenges related to secure coding guidelines, either individually or as part of a team. Although the challenges can include an offensive part (e.g., on how malicious parties exploit systems), the main focus and emphasis of the challenges is on developing secure software, i.e., on the defensive perspective. For each solved challenge, points are awarded, and the winner of the game is the one with the highest number of points. Participants to the event can have either a background in a single programming language or be mixed, e.g., both C and Web developers.

3 Related Work

Although several methods exist to deal with software vulnerabilities, e.g., requirements engineering and code reviews, we focus on awareness training for software developers. Several previous studies indicate that software developers lack secure programming awareness and skills [2,26,31]. In 2020, Bruce Schneier, a well-known security researcher, and evangelist stated that "more than 50% of software developers cannot spot security vulnerabilities in software" [29]. His comment adds to a discussion on secure coding skills: In 2011, Xie et al. [32] did an interview study with 15 senior professional software developers in the industry with an average of 12 years of experience. Their study has shown a disconnect between "software developers' understanding of security concepts and their attitudes in their jobs". Awareness training on information security is addressed in McIlwraith [22], which looks at employee behavior and provides a systematic methodology and a baseline on implementing awareness training. In their work, Stewart et al. [30] argue that communicators, e.g., trainers, must understand the audiences' constraints and supporting beliefs to provide an effective awareness program.

There is a stream of literature on compliance with security policies, which deals with employees in general and not with software developers specifically. This stream of literature explores many reasons why people do not comply with IT-security policies. The unified framework by Moody et al. [24] summarizes the academic discussion on compliance with IT-security policies. Empirical findings include that neither deterrence nor punishment such as e.g., public blame, works to increase compliance. However, increasing IT-security awareness increases the level of compliance [30]. In their seminal review article, Hänsch et al. [20] define IT-security awareness in the three dimensions: *Perception*, *Protection*, and *Behavior*. The concept of IT-security awareness is typically used in IT security management contexts, and we use this

concept to evaluate our work. While these findings are for the compliance of employees with IT-security policies and awareness of IT security, little empirical research is done on IT-security awareness in software development and what makes software developers comply with security policies in software development.

Graziotin et al. [19] show that *happy developers are better coders*, i.e., produce higher quality code and software. Their work suggests that by keeping developers happy, we can expect that the code they write has a better quality and, by implication, be more secure. Davis et al. [7] show that cybersecurity games have the potential to increase the overall happiness of software developers. Their conclusions support our approach to use a serious game approach to train software developers in secure coding. Awareness games are a well-established instrument in information security and are discussed in defacto standards as the BSI Grundschutz-Katalog [5] (M 3.47, Planspiele) as one means to raise awareness and increase the level of security. Frey et al. [10] show both the potential impact of playing cybersecurity games on the participants and show the importance of playing games as a means of cybersecurity awareness. They conclude that cybersecurity games can be a useful means to build a common understanding of security issues. Rieb et al. [27] provide a review of serious games in cybersecurity and conclude that there are many approaches. However, only a few have an evaluation of their usefulness and are available beyond the immediate context of a consulting or cyber-security company. The games listed mainly address information security rather than secure coding. Documented and evaluated games are [4] and [27].

Capture-the-flag is one particular genre of serious games in the domain of Cybersecurity [7]. Game participants win flags when they manage to solve a task. Forensics, cryptography, and penetration testing are skills necessary for solving tasks and capturing flags. They are considered fun, but there are hardly any empirical results on these games' effects on participants' skill levels. The present work uses serious games to achieve the goal of *raising secure coding awareness of software developers in the industry*. Previous work on selected design aspects and a smaller empirical basis on the CSC includes [11–18].

4 Method

The design science paradigm, according to Hevner [21], Baskerville and Heje [3] guides our research in the industry. Design and evaluation of designs in iterative approaches are an integral part of design research: this article presents our design after 13 CSC events and the evaluation of the design. The events took place from 2017 to 2020, with more than 200 game participants.

Table 1 summarizes the CSC events. CSC games were designed in three design cycles: 1) Initial Design (events 1-5), 2) Refinement (events 6-9) and 3) Sifu/Online (events 10-13).

Table 1. Overview of Cybersecurity Challenges Events

No.	Date	Type	Focus	Where	NP	Data collection
1	Nov. 2017	Standalone	Mixed	Germany	11	SSI
2	May. 2018	Standalone	Web	Germany	12	SSI
3	Jul. 2018	Standalone	Web	Germany	6	SSI
4	Jul. 2018	Standalone	Mixed	Germany	30	SSI
5	Sep. 2018	Standalone	Web	Germany	16	SSI
6	Aug. 2019	Workshop	C/C++	China	14	Survey
7	Aug. 2019	Workshop	Web	China	15	Survey
8	Sep. 2019	Workshop	Web	Germany	7	Survey
9	Oct. 2019	Workshop	C/C++	Turkey	23	Survey
10	Jun. 2020	Standalone	C/C++	Online	15	Survey*
11	Jul. 2020	Standalone	C/C++	Online	21	Survey*
12	Jul. 2020	Standalone	C/C++	Online	20	Survey*
13	Jul. 2020	Standalone	C/C++	Online	15	Survey*

NP: No. of players, **SSI**: semi-structured interview, **(*)** for survey description see [15]

The CSC events participants were all software developers specializing in web technologies and the C/C++ programming language. The events took place mostly in Germany but also in China and Turkey. The players' age ranged from 25 to 60, the background industry of the participants was critical infrastructures, in particular, industry automation (50.85%), energy (37.29%), and healthcare (11.86%), the overall number of years of work experience was as follows: one year (13.7%), two years (11.0%), three years (19.2%), four years (6.8%) and five or more years (49.3%). Regarding the average number of security training over the previous five years, the results are as follows: Germany – 3.57, China 2.10, and Turkey 1.50.

According to the first and second design cycles, the evaluation of these CSC events is structured according to the following research questions. For analysis of the survey results concerning the Sifu platform, we refer the reader to [15].

- **RQ1**: To what extent are CSC adequate to raise awareness about secure coding?
- **RQ2**: What is the impact that CSC workshops have on the participants?
- **RQ3:** Which factors are considered essential for a successful CSC event?

To address these research questions, the authors have conducted semi structured interviews (SSI) [1] and developed a small survey. The semi-structured interview questions were asked to the participants, one after another in a round-the-table. The participants' answers were recorded on paper. The semi-structured interviews were performed during the first design cycle and were part of the feedback round after the CSC event. They were based on the following questions: a) "what went well, and you would you like to keep" and b) "what did not go well and would you like to change". These questions gave a good insight and allowed us to improve later versions of the game. They were also fundamental for requirements elicitation (see [11]).

The survey was administered to the CSC participants, in the refinement cycle, after completion of the event. The survey consisted of an online survey. Participation in the SSI and the survey was opt-in. Furthermore, all participants consented to participate in research, and the collected data was anonymized. We have used a more formal survey

Table 2. Survey 1: Questions

RQ	CT	QID	Question
RQ1	PE		By participating in this awareness training _____
		Q1.1	I learned new techniques and principles of secure software development
		Q2.1	I understand the possible consequences of a security breach
	BE	Q3.1	I feel that I am prepared to handle secure coding related issues at work
		Q4.1	I understand the need to have secure development life-cycle activities
		Q5.1	I feel more prepared to work with static code analysis tools (e.g. SAST)
		Q6.1	I know how to use the information about secure coding guidelines
		Q7.1	Focusing on the challenges improves my practical secure coding skills
	PR	Q8.1	I have learned about new issues that I would like to check in my own code
		Q9.1	I know where I can find more information about secure coding guidelines
		Q10.1	I understand the importance of secure coding guidelines
RQ2	–	Q11.1	The learning goals of the challenges were clearly explained
		Q12.1	CSC games help me to understand the need to develop secure software
		Q13.1	The help from the coaches was adequate
RQ3	–	Q14.1	I want to learn about new tools, even if I do not use them at work
		Q15.1	I prefer to solve challenges sequentially rather, even if it takes too much time
		Q16.1	Working in teams is better than working individually on the challenges
		Q17.1	I like the fact that different kinds of challenges are presented
		Q18.1	I prefer challenges that address the same problem from different point-of-views
		Q19.1	I prefer challenges that are related with my work environment
		Q20.1	I prefer challenges that are based on real-life examples
		Q21.1	I prefer challenges that can be systematically solved with some tool

RQ: Research Question, **CT**: Construct , **QID**: Question Identifier, **PE**: Perception, **BE**: Behavior, **PR**: Protection

methodology to evaluate the game's usefulness concerning the level of awareness and the skills in secure coding. Table 2 shows the questions that were asked in the survey and the related research questions. The survey used a five-point Likert scale of agreement with the following mapping: strongly disagree (1), disagree (2), neutral (3), agree (4), and strongly agree (5). RQ1 addresses the aspect of the usefulness of the CSC artifact, and the corresponding survey questions are based and adapted from the three dimensions of awareness, as defined by Hänsch et al. [20]: Perception (PE – knowing existing software vulnerabilities), Protection (PR – knowing how to write secure software) and Behavior (BE – actual behavior of software developer) (cf. Sec. Related Work). The questions for RQ2 focus on clarity of the description of the challenges, the coaches' role during the game, and the general motivation of training secure coding. These questions address the design of CSC games and events. RQ3 questions address the challenges and their relation to software developers' everyday work practices in the industry. The survey questions for RQ2 and RQ3 are based on the authors' experience in industrial software engineering, feedback from CSC evaluations of events 1 to 5, and various discussions with colleagues. All the collected data were processed using the statistics package RStudio 1.2.5019. Availability of the gathered data is provided in the same authors' included references and on a forthcoming publication.

5 Design of the CyberSecurity Challenges

In this section, we present the design of the CyberSecurity Challenges for industrial software developers. The sub-sections provide a detailed overview of the architecture, the schedule, and the design of challenges. The results presented in this section distill

the experience obtained through the three design cycles of the CSC games, i.e., of the thirteen CSC events.

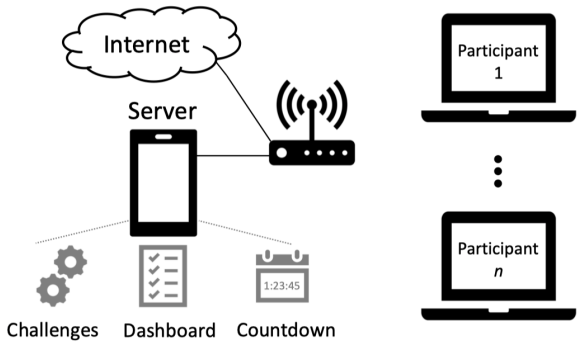

Fig. 1. Architecture of CyberSecurity Challenges infrastructure

5.1 Architecture

Figure 1 shows the architecture of the CSC infrastructure. Each participant, either individually or after forming a team, accesses the challenges through a computer. A server hosts the applications that run the game logic, a "countdown" clock, and a dashboard that records individual players and teams' progress. The dashboard uses the open-source CTFd [6] project. A description of the challenges will be given in the following.

Table 3. Agenda for a one-day cybersecurity challenges game event

Duration	What	Description
10 min	*Welcome*	Welcome to participants and accessing CSC infrastructure
20 min	*Team building*	Participants select partners and build teams that will play against each other
30 min	*Introduction*	Challenge types are presented. One challenge in each category is solved in order to show the participants how the game works
320 min	*Main event*	Game is open and teams are free to play the game. They are responsible for defining their own strategy for time-out (e.g. for lunch break).
10 min	*Winner*	Game is closed and teams can no longer submit points to the dashboard. Winning team is announced. A brief review of the game-play is done together with the participants.
30 min	*Feedback*	Participants are asked to fill out a survey about the game. Additionally, discussions with players is held in short non-systematic interviews. Main points of discussions is recorded for later analysis.
60 min	*Walk-through*	Participants are shown solution to the exercises they considered most difficult. These exercises are solved together in interaction with all the participants. Discussion on how to solve the challenge is highly encouraged.

5.2 CSC Time Schedule

Table 3 shows a typical time-plan for the one-day CSC event consisting of seven blocks: 1) welcome, 2) team building, 3) introduction, 4) main event, 5) winner announcement, 6) feedback and 7) walk-through. The last block, the walk-through, was not initially planned and is the direct result of players feedback — the participants

preferred to dedicate one hour of the main event to provide final explanations and closure on selected exercises. The authors decided to place the feedback and survey before the walk-through to increase the chance of collecting feedback from the participants.

The duration of similar training events ranges from several days [23] (less common) to a single day [28] (more common). Note that the first CTF is done in academia, while a commercial provider does the latter. Additionally, a difference to typical Capture-the-Flag events are the two agenda items *Introduction* and *Walk-through*.

5.3 Defensive Challenges

The primary focus of the CSC game's challenges are Web and C/C++. In contrast to C/C++, for the web challenges, it was decided not to focus on a single programming language or framework since many of these programming languages and frameworks are in everyday use in the company where the CSC game was developed. In this case, we chose a generic approach based on the Open Web Application Security Project – OWASP [25]. The challenges' design took two approaches: 1) based on open-source components and 2) design of own challenges. The first approach was used in the Refinement design cycle, while the second approach in the Sifu/Online design cycle. A common approach to the design of the challenges is given in [16]. Each challenge is presented to the participants according to the following phases: *Phase 1* - introduction, *Phase 2* - challenge, and *Phase 3* - conclusion. The types of challenges are: Single Choice Questions (SCQ), Multiple-Choice Questions (MCQ), Text-Entry Questions (TEQ), Associate-Left-Right (ALR), Code-Snippet Challenge (CsC), and Code-Entry Challenge (CEC). Second, Phase 1 presents an introduction to the challenge and sets up the scenario; the main part of the challenge is phase 2; phase 3 concludes the challenge by adding additional text related to secure coding guidelines or additional questions related to phase 2.

Challenges using Open-Source Components Challenges on secure coding for software developers can be implemented by using and adapting existing open-source components. Since most of the available projects focus on the offensive perspective, the following adaptations are suggested: 1) include an incomplete description on how to solve the challenge, and 2) provide follow-up questions related to secure coding guidelines. Fig. 2-4 shows an example of a challenge for Web developers using OWASP JuiceShop. This challenge's learning goal is to understand what SQL injections are and how to identify an SQL injection quickly. Phase 1 sets the stage for the challenge (Fig. 2). In Phase 2, the player is assisted with how to find the vulnerability, through the textual description, as in Fig 3, or also directed by the game coaches. The last phase consists of an additional question related to the exercise, as shown in Fig. 4, which directs the player to secure coding guidelines. Table 4 shows the open-source projects and components in which have been used to design CSC challenges for Web and for C/C++, along with the expected effort required to modify them. Note that the design of these challenges is based on open-source components that

include an offensive perspective. Therefore, after the components' adaptation, these types of challenges are described as being *defensive/offensive*.

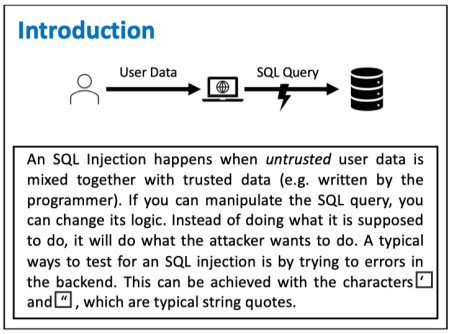

Fig. 2. Web Challenge: Phase 1

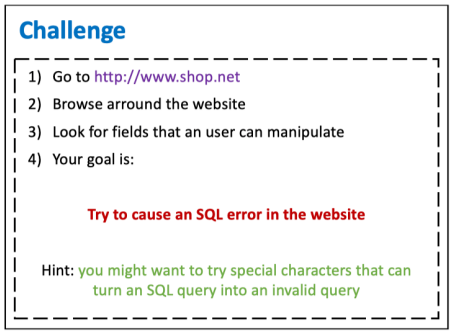

Fig. 3. Web Challenge: Phase 2

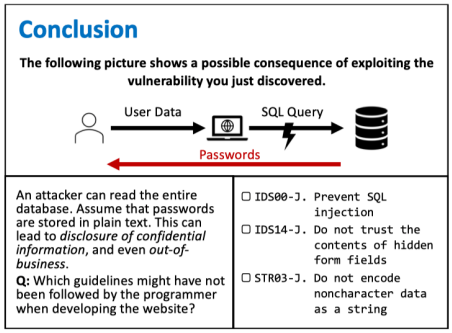

Fig. 4. Web Challenge: Phase 3

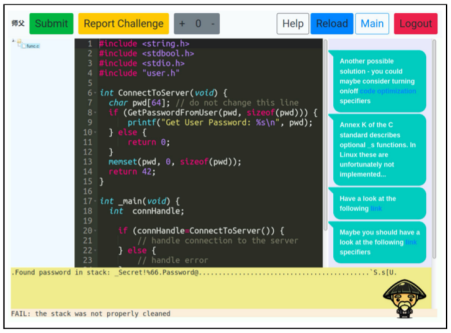

Fig. 5. Sifu Platform

Table 4. Open-Source Tools used for Cybersecurity Challenges

Type	Project	Effort	Description
Web/Java	Juice Shop	Minimal	Insecure web application for training purposes from the OWASP project.
Web/Java	Java SEI-CERT	Medium	Secure coding guidelines dedicated to Java from Carnegie Mellon University
Web	Vulnerable API	Medium	REST API containing several vulnerabilities
C/C++	MBE	Small	Vulnerable code from RPISEC course at Rensselaer Polytechnic Institute
C/C++	C/C++ SEI-CERT	Medium	Secure coding guidelines dedicated to C/C++ from Carnegie Mellon University
C/C++	Vulnerable code snippets	High	Vulnerable C/C++ code from NIST (Juliet Set)

Defensive Challenges using Sifu Platform The Sifu platform hosts code projects containing vulnerabilities in a web application. The reason to choose a web interface is to avoid that the players need to install any software on their machine, which might be difficult in an industrial setting. The players' task is to fix the project's source code to bring it to an acceptable solution (therefore focusing on the defensive perspective). An acceptable solution is a solution where the source code is compliant to secure coding

guidelines and does not have known vulnerabilities. The Sifu platform contains two main components: 1) challenge assessment and 2) an automatic coach. The challenge assessment component analyses the proposed solution submitted by a player and determines if it is acceptable. Analysis is based on several tools, e.g., compiler output, static code analysis, and dynamic code analysis. The automatic coach component is implemented through an artificial intelligence technique that provides hints to the participant when the solution is not acceptable, with the intent to guide the participant to an acceptable solution. Figure 5 shows the Sifu platform. Note that only phase 2 is shown in the figure. The player can browse the different files of the project. All the hints issued by the automatic coach are available on the right-hand side. If the player experiences errors when using the platform, these can be reported for later analysis and improvement. The Sifu platform's main advantage is that the participants do not need to install any software in their machine - a browser with internet or intranet access is sufficient. However, since untrusted and potentially malicious code will be executed in the platform during the analysis stage, several security mechanisms need to be implemented to guarantee that the players cannot hack it. These challenges were developed in the Sifu/Online design cycle, and further and detailed information on the implementation is available in [15]. For more information about the Sifu platform we also refer the reader to [14]. The Sifu platform is available for download as an open-source platform under the MIT license in [18].

6 Results

This section presents a quantitative analysis of the CSC artifact based on the semi-structured interviews and online survey collected during the design cycles Initial Design and Refinement.

6.1 Initial Design Cycle —— CSC 1 to 5

As discussed in section 4, in this design cycle, the participants were asked to provide feedback on what should be kept and what should be changed in the CSC event. The participants were encouraged to discuss openly what they felt was important. These discussions were used to inform the design of future CSC events. In this cycle, requirements were collected on traits that serious games for software developers in the industry should have. A summary of the findings is as follows: 1) *challenges should focus on the defensive perspective*, 2) *challenges should reflect real-world examples*, 3) *challenges should be aligned with the work environment*, 4) *careful planning in terms of duration should be performed*, and 5) *participants should be able to solve challenges without knowledge of extra tools*. A more in-depth analysis of the feedback and resulting requirements is available in [11].

6.2 Artifact Refinement Cycle —– CSC 6 to 9

Figure 6 shows the overall results of the answers to the survey. The research questions are used to group the results. We observe an overall agreement on all the survey questions. In particular, considering negative answers (-), neutral answers (N) and positive answers (+), this table shows the following overall results for each research question: $RQ1^{-} = 7.89\%$, $RQ1^{N} = 16.13\%$, $RQ1^{+} = 75.99\%$, $RQ2^{-} = 4.82\%$, $RQ2^{N} = 12.05\%$, $RQ2^{+} = 83.13\%$, $RQ3^{-} = 4.19\%$, $RQ3^{N} = 12.56\%$, and $RQ3^{+} = 83.26\%$.

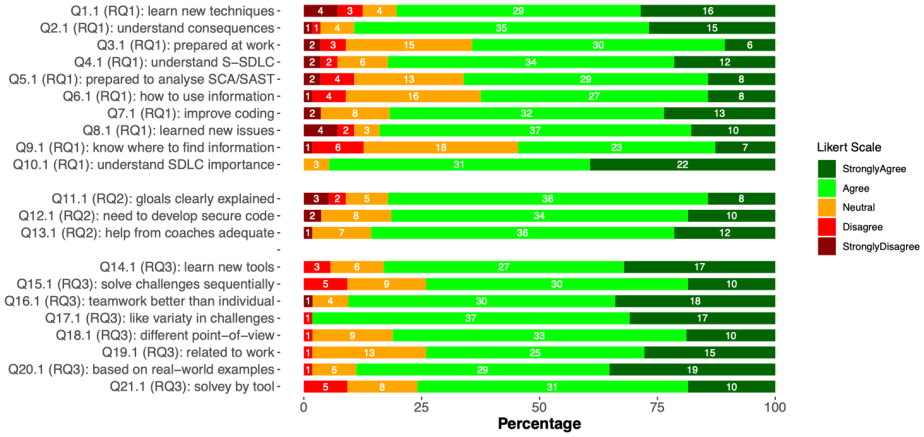

Fig. 6. Evaluation of Usefulness of CSC Events 6-9

These results give a good indication that CSC games are suitable as a means to train software developers in secure coding guidelines, as the factors on awareness (RQ1) and impact on participants (RQ2) have high levels of agreement (i.e., higher than 75%. However, we observe the difficulty in making every participant happy, in particular, due to the residual values on negative and neutral answers. Further analysis is required to understand this. Based on our experience, we believe that this fact might be correlated with the participants' previous experience.

Table 5. Analysis of Research Questions for Survey on CSC Events 6-9

	Rank	1	2	3	4	5	6	7	8	9	10
RQ1	W.Avg.	4.34	4.11	3.98	3.93	3.89	3.84	3.66	3.66	3.63	3.53
	QID	Q10.1	Q2.1	Q7.1	Q4.1	Q1.1	Q8.1	Q5.1	Q6.1	Q3.1	Q9.1
RQ2	W.Avg.	4.04	3.93	3.82	-	-	-	-	-	-	-
	QID	Q13.1	Q12.1	Q11.1	-	-	-	-	-	-	-
RQ3	W.Avg.	4.27	4.22	4.21	4.09	4.00	4.00	3.85	3.83	-	-
	QID	Q17.1	Q20.1	Q16.1	Q14.1	Q18.1	Q19.1	Q21.1	Q15.1	-	-

Table 5 shows a ranking of the different survey questions, grouped by research question. The ranking is performed by sorting the questions based on the average agreement value. In terms of adequacy (RQ1), and impact on the participants (RQ2), the two highest-ranking answers are: to understand the importance of SDLC (Q10) and understand consequences of a breach (Q2) for RQ1, and help from coaches (Q13), and

understand the need to develop secure software (Q12) respectively. The lowest-ranked factors for RQ1 are "find more information" (Q9) and "prepared to handle secure coding issues at work" (Q3). Although the rank is low, the average agreement is positive. The surprising result obtained for Q3 is likely related to the large number of neutral answers. Further investigations are required to determine the root cause of this observation.

The collected results for RQ3 serve to inform practitioners who wish to design such games for an industrial context. It provides a ranked list of factors that participants consider having a positive impact on CSC games. The three top factors that contribute to the success of a CSC game that should be considered by practitioners who wish refine the CSC game are the following: different kinds of challenges (Q17), based on real-life examples (Q20), and participants should work in teams rather and individually (Q16).

In terms of awareness, taking into consideration negative answers (-), neutral answers (N), and positive answers (+), the perception (PE), behavior (BE), and protection (PR) show the following results: $PE^{-} = 8.04\%$, $PE^{N} = 7.14\%$, $PE^{+} = 84.82\%$, $BE^{-} = 7.89\%$, $BE^{N} = 20.79\%$, $BE^{+} = 71.33\%$, $PR^{-} = 7.78\%$, $PR^{N} = 14.37\%$, $PR^{+} = 77.84\%$. These results show similar values for the negative answers (around 8%), which might be related to the players' background. The highest result is related to perception, which also has the least number of neutral answers. While we observe strong agreement on the behavior and protection constructs (more than 70%), there are still many neutral answers. We believe that the large number of neutral answers is also related to player background and the fact that the challenge type is not purely defensive, i.e., it is defensive/offensive, as discussed in section 5. The reasoning for this is based on the better results obtained in the study of the Sifu platform (see [15]).

6.3 Sifu/Online Cycle —— CSC 10 to 13

In this design cycle, the CSC challenges were further developed as the Sifu platform [15]. The participants were asked to evaluate the platform through 5point Likert scale questions. Survey questions were based on the Awareness [20], and Happiness [19] dimensions. The following is a summary of the results, in terms of the three awareness dimensions: perception (PE), behavior (BE), and protection (PR); and in terms of happiness (HP). $PE^{-} = 2.22\%$, $PE^{N} = 8.89\%$, $PE^{+} = 88.89\%$, $BE^{-} = 0.0\%$, $BE^{N} = 8.06\%$, $BE^{+} = 91.94\%$, $PR^{-} = 6.67\%$, $PR^{N} = 11.11\%$, $PR^{+} = 82.22\%$, $HP^{-} = 8.22\%$, $HP^{N} = 10.27\%$, $HP^{+} = 81.51\%$. The negative results (-) correspond to strongly disagree and disagree, neutral (N) to neutral answers, and positive results (+) correspond to agree and strongly agree. A more in-depth analysis of these results, along with the Sifu platform's design, and the survey questions, can be found in [14]. The collected answers again indicate an agreement with the awareness theory, in the following sequence: behavior, perception, and finally, protection. Also, the participants report having fun and being happy while playing challenges in the Sifu platform.

The Hellinger distance is used to measure the distance between two probability mass functions (PMF). The distance between the PMF of the three awareness constructs was computed to compare the results obtained in the second (refinement) and third

cycle (Sifu/Online). The obtained results are as follows (from higher distance value to smaller distance value): behavior (d = 0.25), perception (d = 0.10), and protection (d = 0.04). These results show that using the Sifu platform results in the most significant improvement in agreement on the behavior construct. Although both cycles indicate positive results, the participants have a more substantial agreement that solving the Sifu platform's challenges helps in actual behavior (i.e., using defensive challenges), than using defensive/offensive challenges. In terms of protection, the distance between the PMF is low (0.04), indicating that the agreement level is similar for the protection construct for both the defensive/offensive and the defensive challenges. These results were as expected since the improvements to the challenges and the corresponding design cycles performed in the Sifu platform increase the adequacy to improve software developer awareness in terms of behavior.

6.4 Discussions

In this work, we have presented and evaluated an awareness training program for software developers in the industry, which was designed through three design cycles [21]. The types of CSC challenges for each design cycle were as follows: offensive, defensive/offensive, and defensive. The initial design cycle was mostly used for requirements elicitation to further develop and refine the CyberSecurity Challenges for software developers in the industry. In the second design cycle, defensive/offensive challenges were introduced. These challenges adapt existing open-source projects to adopt a defensive perspective. Finally, in the third design cycle, defensive challenges are introduced using the Sifu platform. Our experience has shown that software developers highly appreciate playing CSC games based on direct feedback from participants. It was also observed that playing CSC games can be done as either a standalone event or after a secure coding training. Furthermore, the participants have claimed that the challenges have helped solidify, understand, and practice secure coding in real scenarios, the concepts discussed during training. While the challenges, as described for the second and third design cycle, seem to address software developers and management's needs adequately, the third design cycle was shown to result in a higher agreement in terms of the behavior (BE) awareness construct.

Participants report on the happiness and fun in participating in these events. However, a long-term study on the impact of CSC events on software quality is not possible. The reason for this is related to the large number of factors that hinder this study, which include, among others: job rotation, changing and evolving IT security technologies, discovery of new attack vectors, and evolving programming languages and programming language standards. Therefore, we need to suffice with the fact that these events are both welcome by software developers and, with the fact that CSC has had continuous management approval throughout the years, and also the fact that it has been introduced in the standard teaching curriculum in the company where it was developed.

While previous work such as McIlwraith [22] provides a generic approach for awareness training, we show a method that explicitly addresses software developers in

the industry and is based on a serious game inspired by the Capture-the-Flag format. Nevertheless, some of the traits introduced by McIlwraith are also common with our artifact, e.g., the usage of web-based media and web-based text. While the CSC artifact was designed for Web and C/C++ challenges, we think our approach can be generalized to other programming languages. Other possible usages of our artifact include a refresher on previously acquired knowledge, a self-evaluation tool for individuals, and a recruiting tool used by human resources. However, further work might be required either for non-industrial environments or participants with different backgrounds, e.g., management or human resources.

6.5 Threats to Validity

There are threats to the validity of or findings - threats as they are typical or inherent to design research. Both the evaluation of the game in a survey and in the mixed workshop might lead to socially desired bias. Moreover, participants might evaluate the game positively in order to be able to get the awareness training done as it is a task that is mandated to them by management. The authors cannot control the types of workshops and the participants; however, we think that the conclusions are valid as they contributed to improving the serious game over time.

The authors claim that the game has a positive impact on IT security awareness. The path from awareness training to secure products and services is long and potentially tedious, and other kind of research would be needed to evaluate whether such a game has an impact on the quality of code. Due to the large number of factors that affect code quality, this is, in practice, not possible. Nevertheless, awareness is a well-established endpoint in IT security research.

As in any design research, we cannot argue that our solution is the best, and we need to suffice with the argument that our artifact and outcome of research is useful, both in terms of developers' happiness and management approval. There are several external variables that we cannot control in an industrial setting that can limit our evaluations' validity. Although we have explicitly mentioned to the participants that the survey questions refer to the CSC event, we cannot exclude questions' misinterpretation due to the participants' different cultural and language backgrounds.

Also, we cannot exclude a bias for socially desired answers and positive bias with the game setting. However, for the validity of our findings, we refer to the fact that all game participants were industrial software engineers, and participation in the survey was not mandatory. Our results demonstrate that these are a viable method for awareness training on secure coding in the industry in terms of the CSC game's usefulness. We base this observation on the fact that it is approved by management, has high internal demand, and is liked and enjoyed by most participants.

7 Conclusions and Further Work

In this work, we provide an overview of the design and implementation of CyberSecurity Challenges - a serious game to raise awareness on secure coding for

software developers in the industry. The CyberSecurity Challenges have been developed following a design science research design structured in three design cycles: Initial Design, Refinement, and Sifu/Online. The design cycles extended from 2017 until 2020 and consisted of thirteen events where more than 200 software developers participated. Our contribution addresses practitioners who wish to develop or refine a software developer awareness training for the industry and the research community by understanding the usage of serious games targeting software developers in the industry.

This paper consists of two main parts: 1) an overview of the design of the CyberSecurity Challenges and 2) an evaluation of the CyberSecurity Challenge game and events, including the usefulness of CyberSecurity Challenges. In the first part, we presented a consolidated view of CyberSecurity Challenges. This consolidated view is the result of all the lessons learned throughout the three design cycles. We provide an analysis and report of the main results that practitioners can use to design a similar awareness training program. We also discuss the differences and similarities to other existing awareness training programs. In the second part, we analyze results from semi-structured interviews from the first design cycle and a survey collected during the second design cycle. Overall, software developers enjoy playing CyberSecurity challenges, either as a standalone event or together with a training workshop on secure programming. Furthermore, we present results on the impact that the game has on the participants and discuss essential factors for successful awareness training. Our positive results, continuous management endorsement, and the fact that these games have been introduced as a standard part of the company's teaching curricula validate our design approach. Additionally, our results show that CyberSecurity challenges are a viable approach for awareness training on secure coding.

As further steps, the authors would like to design a systematic approach to identify topics for challenges and assessing these challenges for relevance. Towards this, more empirical analyses are required. Thus, parallel and next steps include an empirical study on the awareness of various secure coding topics to tailor the challenges to different software developer groups' needs. Also, as the COVID-19 crises limits travel and physical presence, we will continue to enhance the online version of the game. We also plan to enrich the scope of defensive challenges.

Acknowledgements

The authors would like to thank the participants of the CyberSecurity Challenges for their time and their valuable answers and comments. Also, the authors would also like to thank Kristian Beckers and Thomas Diefenbach for their helpful, insightful, and constructive comments and discussions.

This work is financed by national funds through FCT - Fundação para a Ciência e Tecnologia, I.P., under the projects FCT UIDB/04466/2020 and UIDP/04466/2020. Furthermore, the third author thanks the Instituto Universitário de Lisboa and ISTAR, for their support.

References

1. Adams, W.: Conducting Semi-Structured Interviews. In: Newcomer, K., Hatry, H., Wholey, J. (eds.) Handbook of Practical Program Evaluation, chap. 19, pp. 492–505. Wiley Online Library (2017)
2. Assal, H., Chiasson, S.: 'Think secure from the beginning' A Survey with Software Developers. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. pp. 1–13. CHI'19, Association for Computing Machinery, New York, NY, USA (2019)
3. Baskerville, R., Pries-Heje, J.: Explanatory design theory. Business & Information Systems Engineering 2(5), 271–282 (2010)
4. Beckers, K., Pape, S.: A Serious Game for Eliciting Social Engineering Security Requirements. In: 2016 IEEE 24th International Requirements Engineering Conference (RE). IEEE (08 2016)
5. Bundesamt für Sicherheit in der Informationstechnik: BSI IT-Grundschutz-Katalog, 2016, 15. ed. (2016), https://tinyurl.com/zkbmfb6
6. Chung, K.: CTFd : The Easiest Capture The Flag Framework, https://ctfd.io/
7. Davis, A., Leek, T., Zhivich, M., Gwinnup, K., Leonard, W.: The fun and future of CTF. 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14) pp. 1–9 (2014), https://tinyurl.com/y97enbtr
8. Department of Homeland Security: Industrial Control Systems - Computer Emergency Response Team. https://us-cert.cisa.gov/ics, accessed: 2020-08-26
9. Dörner, R., Göbel, S., Effelsberg, W., Wiemeyer, J.: Serious Games: Foundations, Concepts and Practice. Springer International Publishing, 1. Ed, Switzerland (2016). https://doi.org/10.1007/978-3-319-40612-1
10. Frey, S., Rashid, A., Anthonysamy, P., Pinto-Albuquerque, M., Naqvi, S.A.: The Good, the Bad and the Ugly: A Study of Security Decisions in a Cyber-Physical Systems Game. IEEE Transactions on Software Engineering 45(5), 521–536 (2019)
11. Gasiba, T., Beckers, K., Suppan, S., Rezabek, F.: On the Requirements for Serious Games Geared Towards Software Developers in the Industry. In: Damian, D.E., Perini, A., Lee, S. (eds.) Conference on Requirements Engineering Conference. pp. 286–296. IEEE, Jeju, South Korea (09 2019). https://doi.org/10.1109/re.2019.00038
12. Gasiba, T., Lechner, U., Cuellar, J., Zouitni, A.: Ranking Secure Coding Guidelines for Software Developer Awareness Training in the Industry. In: Queirós, R., Portela, F., Pinto, M., Simões, A. (eds.) First International Computer Programming Education Conference (ICPEC 2020). OpenAccess Series in Informatics (OASIcs), vol. 81, pp. 11:1–11:11. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020)
13. Gasiba, T., Lechner, U., Pinto-Albuquerque, M.: Awareness of Secure Coding Guidelines in the Industry - A first data analysis. In: The 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, Online (12 2020)
14. Gasiba, T., Lechner, U., Pinto-Albuquerque, M.: Sifu - A CyberSecurity Awareness Platform with Challenge Assessment and Intelligent Coach. In: Special Issue on Cyber-Physical System Security of the Cybersecurity Journal. SpringerOpen (12 2020)
15. Gasiba, T., Lechner, U., Pinto-Albuquerque, M., Porwal, A.: Cybersecurity Awareness Platform with Virtual Coach and Automated Challenge Assessment. In: 6th Workshop On The Security Of Industrial Control Systems & Of Cyber-Physical Systems. Springer, Online (09 2020)

16. Gasiba, T., Lechner, U., Pinto-Albuquerque, M., Zouitni, A.: Design of Secure Coding Challenges for Cybersecurity Education in the Industry. In: 13th International Conference on the Quality of Information and Communications Technology. Springer, Online (09 2020)
17. Gasiba, T., Lechner, U., Rezabek, F., Pinto-Albuquerque, M.: Cybersecurity Games for Secure Programming Education in the Industry: Gameplay Analysis. In: Queirós, R., Portela, F., Pinto, M., Simões, A. (eds.) First International Computer Programming Education Conference (ICPEC 2020). OpenAccess Series in Informatics (OASIcs), vol. 81, pp. 10:1–10:11. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020)
18. Gasiba, T. Sifu Platform, https://github.com/saucec0de/sifu (2020)
19. Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P.: What happens when software developers are (un)happy. Journal of Systems and Software 140, 32–47 (2018)
20. Hänsch, N., Benenson, Z.: Specifying IT security awareness. In: 25th International Workshop on Database and Expert Systems Applications, Munich, Germany. pp. 326–330. IEEE, Munich, Germany (Sep 2014). https://doi.org/10.1109/DEXA.2014.71
21. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Q. 28(1) (03 2004)
22. McIlwraith, A.: Information Security and Employee Behavior: How to Reduce Risk Through Employee Education, Training and Awareness. Gower Publishing, Ltd. (2006)
23. Mirkovic, J., Peterson, P.A.: Class Capture-the-Flag exercises. In: 2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14) (2014)
24. Moody, G.D., Siponen, M., Pahnila, S.: Toward a Unified Model of Information Security Policy Compliance. MIS quarterly 42(1), 1–50 (2018)
25. OWASP Foundation: Open Web Application Security Project, https://owasp.org/
26. Patel, S.: 2019 Global Developer Report: DevSecOps finds security roadblocks divide teams (July 2020), https://about.gitlab.com/blog/2019/07/15/globaldeveloper-report/, [Online; posted on July 15, 2019]
27. Rieb, A.: IT-Security Awareness mit Operation Digitales Chamäleon. Ph.D. thesis, Universität der Bundeswehr München, Neubiberg (2018)
28. SANS Institute: SEC642: Advanced Web App Penetration Testing, Ethical Hacking, and Exploitation Techniques, https://tinyurl.com/yytoawyn, online, Visited Nov 2020
29. Schneier, B.: Software Developers and Security. Online (July 2020), https://www.schneier.com/blog/archives/2019/07/software develo.html
30. Stewart, G., Lacey, D.: Death by a Thousand Facts: Criticizing the Technocratic Approach to Information Security Awareness. Information Management & Computer Security 20(1), 29–38 (2012)
31. Tahaei, M., Vaniea, K.: A Survey on Developer-Centered Security. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). pp. 129–138. IEEE (2019)
32. Xie, J., Lipford, H.R., Chu, B.: Why do Programmers Make Security Errors? 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) pp. 161–164 (09 2011). https://doi.org/10.1109/VLHCC.2011.6070393