Design of Secure Coding Challenges for Cybersecurity Education in the Industry

Tiago Gasiba[1,2][0000−0003−1462−6701], Ulrike Lechner[2][0000−0002−4286−3184], Maria Pinto-Albuquerque[3][0000−0002−2725−7629], and Alae Zouitni[4][0000−0002−8809−7657]

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

Abstract. According to a recent survey with more than 4000 software developers, *"less than half of developers can spot security holes"*. As a result, software products present a low-security quality expressed by vulnerabilities that can be exploited by cyber-criminals. This lack of quality and security is particularly dangerous if the software which contains the vulnerabilities is deployed in critical infrastructures. Serious games, and in particular, Capture-the-Flag(CTF) events, have shown promising results in improving secure coding awareness of software developers in the industry. The challenges in the CTF event, to be useful, must be adequately designed to address the target group. This paper presents novel contributions by investigating which challenge types are adequate to improve software developers' ability to write secure code in an industrial context. We propose 1) six challenge types usable in the industry context, and 2) a structure for the CTF challenges. Our investigation also presents results on 3) how to include hints and penalties into the cyber-security challenges. We evaluated our work through a survey with security experts. While our results show that "traditional" challenge types seem to be adequate, they also reveal a new class of challenges based on code entry and interaction with an automated coach.

Keywords: education · teaching · training · secure coding · industry · cybersecurity · capture-the-flag · game analysis · game design · cybersecurity challenge

1 Introduction

To improve the quality (ISO250xx [16]) of software in terms of security, several standards such as IEC-62443-4-1 [15] and ISO 27001 [17] mandate the implementation of a secure software development lifecycle (S-SDLC). Additionally, in recognition of the importance of secure code and need to develop secure products [23, 26], several companies have joined together and formed the SAFECode [25] alliance to promote security best practices. Automatic tools such as

Static Application Security Testing (SAST) [24] can be used to automate and aid in improving code quality. These tools scan the code basis for existing vulnerabilities, which must be fixed by software developers. However, previous research shows that this is not enough [22]: the reliability of such tools is still not good enough, and they cannot automatically fix the code - this is done by software developers who must also be trained in secure software development.

One of the methods currently being investigated and that is showing promising results are training methods based serious games of the type Capture-the-Flag (CTF). The concept of these kinds of games was originally developed in the pen-testing community. Several such games are continually being deployed around the world [7] nowadays by universities, companies, and even groups of individuals. However, most of the existing CTFs are not geared towards software developers in the industry. Gasiba et al. [10] have recently shown that, in order to raise awareness on secure coding in the industry, the game design must address the specific requirements of its target audience.

Typically CTFs can be categorized as follows: 1) *Attack-Only*, 2) *Attack-and-Defend* and 3) *Defend Only*. The participants of these CTFs are generally split into two categories: Red Team (attackers) and Blue Team (defenders). In *Attack-Only* Red team players try to exploit several systems to gain access and control. In *Attack-and-Defend* competitions, the Red team players attack systems that are being hardened and protected by blue team members. Finally, in *Defend Only* CTFs, the players answer questions on cybersecurity for points or configure and harden systems to be resilient to simulated attacks.

To address the needs of the industry and to better adapt to the players, Gasiba et al. [10] have proposed a defensive CTF approach and also outlined the requirements for the design of the defensive challenges. A proper design of challenge types based on these requirements is especially important in an industrial setting, as shown by an experiment by Barela et al. [3], where the type of the challenge (based on comics) was seen to be inadequate for CTFs in the industry.

Therefore, in this paper, we extend previous work by addressing the question of which types of defensive challenges are suitable for software developers. In particular, we are interested in the 1) structure of the said challenges and also on 2) which types of challenges can be used in a CTF-like competition to raise awareness of software developers in the industry. Our work is based on surveys administered through interviews with expert security trainers from the industry. The main contributions of this work are the following:

- design of defensive challenges for CTFs in the industry which aim at raising awareness on secure coding and secure coding guidelines
- definition of a challenge structure for industrial CTFs,
- definition of six different challenge types for industrial CTFs, and
- insight into different options on how to include hints and penalties in industrial CTFs.

We hope that this work can be used by designers of serious game and quality engineers as a guideline on how to design defensive challenges for CTFs aimed at raising awareness on secure coding on software developers in the industry.

In section 2, we present previous work related to our research. Section 3 discusses our approach to the design of the defensive challenges. The results of our study are presented in Section 4. This section also presents a critical discussion on the obtained results, presents our main contribution to practical scenarios for possible games, and briefly discusses the threats to the validity of our findings. Finally, section 5 summarizes our work and briefly discusses possible next steps.

2 Related Work

In [11], Graziotin et al. have shown that *happy developers are better coders*, i.e., produce higher quality code and software. Davis et al. in [8] show that CTF players experience fun during game play. Furthermore, Woody et al. [32] argue that *software vulnerabilities are quality defects*. Since fun and happiness are inter-related [30], these facts can be seen as a motivator to use Capture-the-Flag (CTF)-base serious games [9] to raise awareness [14] on the topic of secure coding for software developers in the industry, in order to improve code quality.

In [19] Mirkovic et al. introduced classroom CTF exercises as a form of cybersecurity education in academia. Their results show that the students that participated in this kind of event have enjoyed the training and have shown increased interest, attention, and focus towards cybersecurity topics. Additionally, in their study, Gonzalez et al. [13] shown similar results and state that cybersecurity training through serious games improves the students' education and skills, and has a positive impact on attracting students to cybersecurity field. They conclude that this kind of training can reduce the shortage of professionals in the field of cybersecurity. Several additional studies [2, 4, 8] also show the positive benefits of CTF in students' attention and performance.

However, using CTFs as a tool to raise cybersecurity awareness comes with different obstacles. In [10], Gasiba et al. elicit requirements for designing CTF challenges geared towards software developers in the industry and show that these CTF challenges should focus on the defensive perspective. Chung et al. [6] also evaluated different aspects related to CTFs and concluded two important issues related to CTFs: the challenge difficulty level and suitability the target audience.

In our work, we are interested in designing high-quality defensive CTF challenges for software developers in the industry that address the topic of secure coding guidelines [5] (SCG) and secure software development best practices [21] (SDBP). However, most of the currently existing work focuses on academia, where the target group is composed of current or future security experts, or pen-testers. Furthermore, most existing studies also focus on the offensive perspective and do not address the topic of SCG, and SDBP. As such, this study is driven by both the need to raise awareness on secure coding [1,20,33], and by the lack of design of defensive CTF challenge geared towards software developers in the industry. The research method used in this work is based on semi-structured interviews [31] and survey best practices as described by Grooves et al. [12] and Seaman et al. [28]. The design of the serious games is based on [9].

3 Approach to Challenge Design

In order to design defensive challenges for industrial CTFs, the authors have decided to focus on two different aspects: the challenge structure (CS) and the challenge type (CT). The content of the challenge (e.g. questions or example of software vulnerability), which are not the focus of the current work, can be derived from existing SCG [5] and SDBP [21]. The challenge structure reflects the mechanics of the challenge, i.e., how it is supposed to be deployed and how it should work. The challenge type specifies the different ways that the challenge can be presented to a participant. Figure 1 shows the steps that we have followed in our approach to design the defensive CTF challenges for an industrial context.

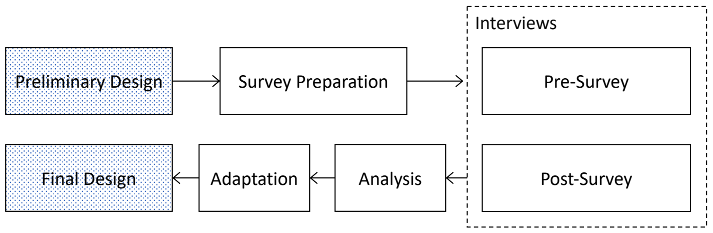


Fig. 1. Study Approach

In the first step, we have created a preliminary design, containing a proposed CS and different CTs. For this, we conducted several informal discussions with one security expert. Additionally, based on our experience with past Capture-the-Flag events, we concluded the preliminary design of challenge structure and challenge types. In the next step, we created a two-phase survey [12,28]. The goal of the survey was to gather feedback and opinions, in a structured way, on the preliminary challenge structure and challenge types. It was used to facilitate the semi-structured interviews with several security experts. The interviews, carried out in the following step, were realized in face-to-face meetings. The meetings consisted of three parts: pre-survey, post-survey, and informal discussions. After the interviews, the collected feedback was transferred to digital form and was analyzed. The analysis step aims at understanding the joint agreement on the different suggested improvements by the security experts. The commonly suggested improvements were then used to adapt and change the preliminary design, which resulted in the final challenge design.

In the following sub-sections, we present details on the different phases of our approach. The results of the analysis, adaptation, and also the final challenge design will be presented in section 4.

3.1 Preliminary Design

In the preliminary design, the authors conducted several informal discussions with a security expert which is also a trainer of secure coding in the industry. The security expert has more than 10 years of experience in the industry and has also knowledge and had previously participated in Capture-the-Flag events. Based on the experience of the security expert and also on the experience of the authors, preliminary design was derived.

3.2 Survey Preparation

In order to prepare for the interviews with security experts, a two-part survey [18] was developed by the authors. The developed survey underwent three reviews by three different cybersecurity experts: one holding a master of science in computer science and two holding a Ph.D. in IT security, whereby one is additionally a university lecturer in cybersecurity. The main goal of the pre-survey was to understand what types of challenges do experienced industry security experts find suitable for CTF-based awareness training. The post-survey' primary goal is to understand the level of agreement with the different preliminary challenges types. The pre-survey was conducted at the beginning of the meeting, before presenting the preliminary design. The post-survey was conducted after presenting the preliminary design. This split allowed the participants to think and reflect on their answers from the pre-survey and be prepared and more open-minded for the discussions on the post-survey. Splitting the survey into two parts was done in order to guarantee unbiased feedback collection from the security experts during the pre-survey. Both the pre-survey and post-survey asked the participants - *if they were to design a CTF challenge about secure coding for software developers in the industry, what kind of challenge structure and type would they use?*. The post-survey additionally asked questions on the preliminary design, in particular on *what would the participant change, add or remove to the presented preliminary design*, *what other challenge types would they additionally consider* and also on the expert opinion on *how to use penalties and hints in the challenges*. In total, the pre-survey consisted of 16 questions, of which 12 were multiple-choice, three were based on a Likert scale, and 1 was an open-ended question [29]. The post-survey consisted of 11 questions, whereby 5 were feedback questions based on a Likert scale, and 6 were open-ended questions.

3.3 Interviews

For the interview, the authors engaged 20 security experts with an average of 4 years of experience in the industry (minimum one year and maximum of 12 years). The experts were selected based on their experience, position, and background in the company - engaged in several consulting projects as a cybersecurity expert. A large part of the participants were also trainers themselves of different topics on cybersecurity. The selected participants were all familiar with CTF competitions. Half of the experts hold a Ph.D. degree in computer science or equivalent, and the remaining half holds a master of science in computer science or equivalent. The face-to-face interviews lasted for one hour and were carried out between the 1st of October 2019 and the 16th. During the interview, the first 20 minutes were dedicated to the pre-survey. Afterward, the preliminary design of CTF challenges was presented to the participants. The remaining 30 minutes were then spent on the post-survey, open-ended discussions and finished with 10 minutes of informal discussions on the results.

3.4 Analysis

In this stage, we gathered all the collected data from the pre-surveys, post-surveys, and informal discussions. The results using a Likert scale were analyzed using standard statistical methods. Due to its nature, the open-ended questions and the informal discussions need to be coded [28]. In order to guarantee the quality of this step, the transcripts were given to three security experts who were asked to perform the coding step manually. We have opted for a manual procedure rather than automated to ensure high quality, as automated coding has been previously shown not to achieve high accuracy [27]. The coding outcome of each expert was then collected and discussed together. Similarities and differences were then systematically addressed, and the final coding was derived by mutual agreement between the three experts.

3.5 Adaptation and Final Design

The last step consisted of using the feedback from the previous step to adapt and change the preliminary design accordingly. Only the proposed changes that were agreed by the majority of the participants (i.e., more than 2/3 after coding or 80% of participants) were considered for the final design. In section 4, the final challenge design, including challenge structure and derived challenge types, will be presented in more detail.

4 Analysis and Results

In this section, we describe the results from the two-part survey interview, as outlined in the previous section. We present the final challenge structure and types, which take into consideration the feedback provided by all the security experts. Finally, we summarize the main contributions and briefly discuss the threat to the validity of our work.

4.1 Preliminary Design

As a result of the informal discussions with the security expert, the challenge structure was defined in two rounds: *round 1*: main challenge and *round 2*: presentation of secure coding guideline related to the challenge. No further details will be given for the initial design, as this was changed after the interview with the security experts, as shown in the next sub-sections. Section 4.4 details the final challenge structure. The derived initial challenge types were the following: Single Choice Question (CSQ), Multiple Choice Question (MCQ), Text Entry Challenge (TEC), Code Snippet Challenge (CSC), and Code Entry Challenge (CEC). Table 3 shows a summary of the challenge types. Further details are given in section 4.5, together with the final design.

4.2 Pre-Survey Results

Pre-survey results showed that the majority (55%) of the participants thought an adequate type of challenge would be of type *question and answer*, without specifying what they mean. Additionally, 85% answered that *some form of challenge involving coding* would be adequate, since the challenges should be based on SCG. However, some participants replied that *friendly hacking* exercises would also be a good exercise - this was discarded as these types of challenges are not defensive. One participant mentioned that an appropriate challenge would involve *fixing a problem on a vulnerable code snippet.*

Question	**Pre-survey Results**
When would you add the hints?	(30%)For all challenges (50%) For difficult challenges (20%) No opinion
How to design the hints?	(50%) Giving details on the answer (75%) Disclosing important concept (70%) Include an external reference
When would you add penalties?	(60%) Agree to introduce penalties (35%) Retrying the same challenge (65%) When using a hint (30%) Disagree to introduce penalties (30%) No opinion

Table 1. Coding Results On Hints and Penalties

Table 1 shows a summary of the agreement level of the participants towards questions asked during the pre-survey related the hints and penalties. The usage of hints was backed by 80% of the survey participants, for difficult challenges (50%) or all questions (30%). The hints should include details on how to solve the questions (50%) and point-out the secure coding concept behind the challenge (75%). The majority of the participants agreed that adding an external reference (e.g. link to an article on the web) is an appropriate way to design hints for challenges. Half of the participants agree that hints should disclose the essential concept behind the challenge, e.g., on which secure coding guideline the challenge is based. Only 75% of the participants agree that giving targeted hints (e.g., disclosing an important concept) is a good idea. During the informal discussions, several participants mentioned that the goal of the hints should be to make sure that the CTF players are learning secure coding concepts during the game. The hints should also be designed in order to lower player frustration and maximize the learn-effect. In particular, the types of hint should be precise and to the point, as industry players have a limited time to play the game.

In terms of penalty-points, 60% agreed to introduce them, 30% disagreed, and 10% had no opinion. The ones that agreed to introduce penalty points, 65% agreed that using hints should be penalized, and the remaining 35% agreed that retrying a challenge should be penalized. During the informal discussions, the survey participants mentioned that the intention to add penalties should be to motivate the player to find solutions by him/herself and not to rely on hints. Furthermore, the penalties should be small to lower the frustration level while maximizing the learning effect of the CTF players.

4.3 Post-Survey Results

In the post-survey, the participants were shown all the derived challenge types and were asked to rate their agreement on the suitability for a CTF-like event with software developers in an industrial setting on a Likert scale. Table 2 shows the results of the post-survey for the five different challenge types. We use the standard mapping of the Likert scale as follows: from 1-*strongly disagree* to 5-*strongly agree.*

	SCQ	**MCQ**	**TEC**	**CSC**	**CEC**
Average	3.95	3.80	3.15	4.30	4.30
Std. Deviation	0.76	1.00	1.04	1.26	0.92

Table 2. Average Agreement Level

The derived ranks of the preferred challenge types are the following (from highest agreement to lowest agreement): 1) Code-Entry-Challenge, 2) Code-Snippet Challenge, 3) Single-Choice Question, 4) Multiple-Choice Question and 5) Text-Entry Challenge. Although CSC and CEC have the same average agreement level, CSC has a higher standard deviation (i.e., higher uncertainty) than CEC; therefore, we have placed CEC in first place in the rank.

When the participants were asked about ideas for additional challenge types, 80% had *no new idea*, 15% answered yes, *they had an additional idea* and 5% had *no opinion.* The additional collected ideas were the following: a) *"something dynamic and fun"*, b) *"associating left and right lists"* (ASL) and c) *"modify code that has one vulnerability"*. The contribution (a) and (c) could not be mapped into an existing challenge type, nor could a new challenge type be discerned. However, (b) resulted in a new challenge type.

The participants were also asked what could be added to the existing challenges. The following additional points were collected with this question:

- Provide explanation at the end of the challenge, together with the flag
- Add explanations on multi-stage challenges
- Ask which coding guideline is not being followed in a code snippet
- Randomize the answers and randomize of the solutions
- Do not forget about the fun aspect when designing the challenge

These additional points were also used to improve the final challenge structure, as shown in the next sub-section.

4.4 Final Challenge Structure

The final challenge structure (CS) contains three phases consisting of four stages: *introduction* (phase 1), *challenge* and *logic* (phase 2) and *conclusion* (phase 3), as shown in Figure 2. In the *introduction* stage (phase 1), a topic related to secure coding is introduced, which is helpful to solve and frame the challenge (e.g., secure coding guideline or previously related cybersecurity incident). This optional stage and can include a single-choice or multiple-choice question before proceeding to the next phase. In the second phase, the *challenge* stage contains the main CT according to a given challenge type, as presented in section 4.5. In this stage, several hints can be given to the player depending on several factors, e.g., time taken by the player to solve the challenge or the previous number of attempts to solve the challenge. The *logic* stage is responsible for evaluating the solution to the challenge provided by the player and determining if it is correct (acceptable) or wrong (not acceptable). According to the analysis of the answer provided by the player, points or penalties might be awarded.

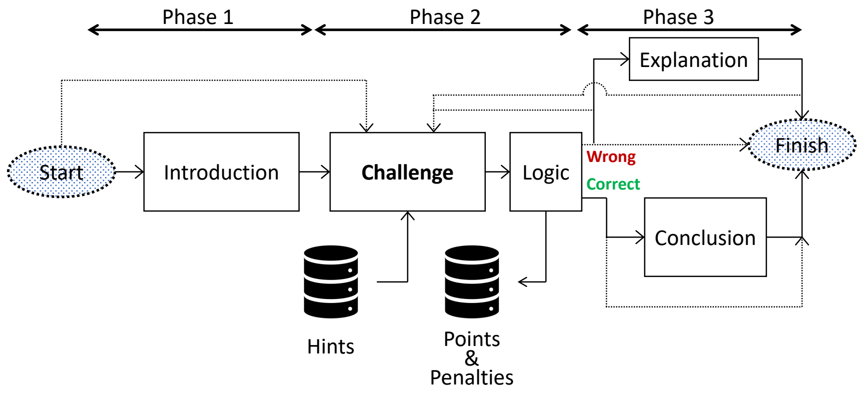


Fig. 2. Challenge Structure

The third phase depends on the result of the logic stage. In case the player's answer was wrong, the following four options can occur: return to the challenge stage, give some explanation why the solution is wrong and return to the challenge stage, proceed to the finish or give an explanation why the solution is wrong and proceed to the finish. In case the player's solution was correct, the following two options can occur: give some concluding remarks (with an optional additional question) and then proceed to the finish or proceed to the finish. If a correct solution is achieved at the finish state, then a flag is presented to the player (according to the CTF rules). In the conclusion stage, additional useful information can be given to the player, e.g., an explanation of secure coding guidelines related to the challenge, the importance of the challenge in the industry context, for example, through lessons learned from past incidents or vulnerabilities.

4.5 Final Challenge Types

Table 3 shows the final six derived challenge types. In single-choice questions (SCQ), the participant is asked a question, and only one of the possible answers is the correct solution. In multiple-choice questions (MCQ), the correct solution must include more than one different answers. In text-entry questions (TEQ), the participant needs to type in the solution as text - this can be achieved, for example, by completing or writing a full sentence as the answer to the challenge. Code-snippet challenge (CSC) presents a piece of code to the participant and lets the participant select lines of code containing vulnerabilities or select changes to the code that would avoid vulnerabilities (i.e., respect SCG and SDBP). In code-entry challenges (CEC), the participant is given vulnerable code that needs to be changed or rewritten to eliminate the vulnerability by complying with SCG and SDBP. In associate left-right challenges, the participant needs to associate items in a list on the left to items in a list on the right.

Figures 3-8 show mock-up sketches of possible implementations on how to create a defensive CTF challenge based on the six challenge types. Each challenge contains a guiding question, an area where the player can interact with the challenge and a *submit* button to submit the results to the backend and trigger the logic stage.

Challenge Type	**Description**
Single Choice Question	Select a single correct answer
Multiple Choice Question	Select multiple correct answers
Text Entry Challenge	Type the answer to the question
Code Snippet Challenge	Identify lines or expressions in a code snippet
Code Entry Challenge	Write or adapt code to eliminate vulnerabilities
Associate Left-Right	Associate elements in left-list to those in right list

Table 3. Description of the derived challenge types

4.6 Observations

In this work, we designed defensive challenges for CTF events, which aim to raise secure coding awareness of software developers in the industry. Code-Entry Challenges were found to be among the most popular choice, while Text-Entry Questions among the least popular. Both the initial CS and the CT were updated as a result of the interviews with security experts. Interestingly, the informal discussions with the security experts did not result in CTs based on comics [3]. Another interesting observation is that all the security experts considered "simple" game types, i.e., no discussions took place on advanced challenge types based e.g., on Virtual Reality or Role-Playing-Games. This fact is likely related to the particular nature of the topic and deployment environment (industry). As such, challenge types that are more simple and traditional have been selected (e.g., Single-Choice Questions and Multiple-Choice Questions). One unexpected challenge type was the Code-Entry Challenge. Due to its complex nature, this type of challenge requires more investigation to understand how to create a challenge based on this type effectively.

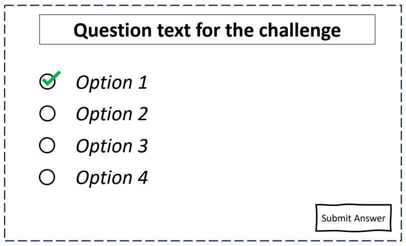


Fig. 3. Single-Choice Question

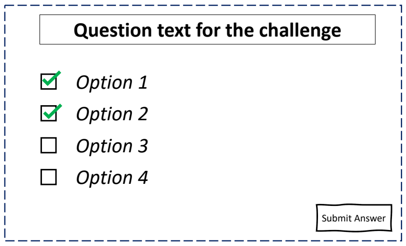
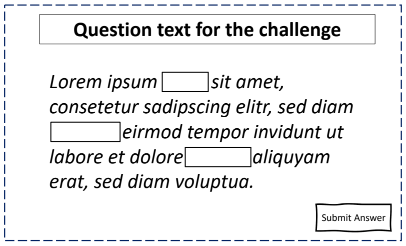


Fig. 4. Multiple-Choice Question



Lorem ipsum ____ sit amet, consetetur sadipscing elitr, sed diam ____ eirmod tempor invidunt ut labore et dolore ____ aliquyam erat, sed diam voluptua.

Fig. 5. Text-Entry Question

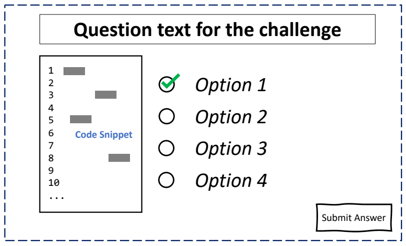
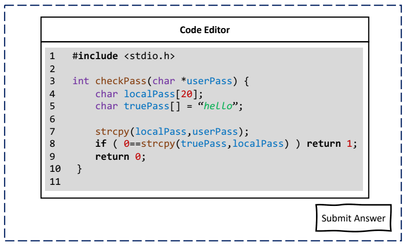


Fig. 6. Code-Snippet Question



```
#include <stdio.h>

int checkPass(char *userPass) {
    char localPass[20];
    char truePass[] = "hello";

    strcpy(localPass,userPass);
    if ( 0==strcpy(truePass,localPass) ) return 1;
    return 0;
 }
```

Fig. 7. Code-Entry Challenge

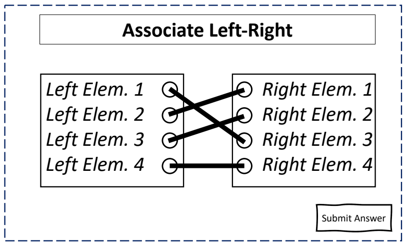


Fig. 8. Associate Left-Right

4.7 Threats to Validity

The work hereby presented is based on the knowledge and know-how obtained through interactive discussions and surveys from a group of 20 security experts from the industry. A possible threat to our conclusions is the limited number of participants and their company background.

Although the authors found previous work on defensive challenges for Capture-the-Flag events, they were not focused on secure coding, software developers, and the industry. Nevertheless, since the authors did not perform a systematic literature review, it might be that some challenge types present in scientific literature might also apply to our situation and constraints.

Another limitation of our work is that it was based only on feedback from security experts and not from players, i.e., real CTF participants. As such, no direct feedback from the target group was used in our evaluation, especially in the preferred challenge type ranking. The authors will address these issues in a subsequent publication.

Although the present work follows survey methodology and semi-structured interviews best practices, it lacks a systematic and academic approach. The reason for this is that the study was conducted in an industrial setting. However, extensive searches were conducted in scientific publication search engines to identify previous relevant work. These findings constituted part of the initial CT and CS design.

5 Conclusions

Nowadays, there is an increasing demand for awareness training of software developers in the industry on secure coding. This demand is motivated by requirements from quality standards and security security standards. One promising new method to raise security awareness is the usage of Capture-the-Flag events. However, these events need to be specially designed in order to address the target audience and its requirements - software developers in the industry.

Recently the requirements that are needed for designing these games in an industrial setting have been investigated [10]. However, the authors of this previous work did not provide details on the challenge types but rather requirements on the overall game. The design of challenge types is not a trivial task, and poor quality challenges may result in inefficiencies (e.g., loss of productivity) that industrial companies are not willing to accept. Barela [3] et al. gives one such example, which has shown that challenge types based on comics, when deployed in CTF, might not be adequate for the event and its goals. Furthermore, the majority of the existing literature not only focuses on defensive challenges but also mostly addresses a target audience of security professionals, e.g., pen-testers.

In this work, we have addressed the design of defensive challenges for CTFs for the industry. We have derived a challenge structure and six different challenge types. Our work is based on semi-structured interviews with security experts and comprises a two-part survey and additional informal discussions. Our results show that security experts prefer "*traditional*" challenge types, based e.g., on Single-Choice and Multiple-Choice Questions. We have seen that the least preferred challenge type by security experts is the Text-Entry Challenges. Three additional challenge types have been discussed: Association Left-Right, Code-Entry Challenge, and Code-Snippet Challenge. The two latter types are well adapted to secure coding challenges since they use software code.

However, the unexpected new challenge type was the Code-Entry Challenge, where the player submits code to the backend, which decides if the challenge is correctly solved. A topic that needs additional investigation is the details on how to create such a challenge type. The results presented in this publication have been derived solely based on feedback from interviews with security experts. Further work is required to validate the derived challenge structure and challenge types in real CTF events in an industrial setting. In particular, the authors intend to give concrete examples of the implementation of the different derived challenge types in an upcoming publication. This further work will allow to refine further the challenge structure and challenge types based on the feedback from the CTF players themselves.

Acknowledgement

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

References

1. Acar, Y., Stransky, C., Wermke, D., Weir, C., Mazurek, M.L., Fahl, S.: Developers need support, too: A survey of security advice for software developers. In: 2017 IEEE Cybersecurity Development (SecDev). pp. 22–26. IEEE (09 2017)
2. Aoyama, T., Nakano, T., Koshijima, I., Hashimoto, Y., Watanabe, K.: On the Complexity of Cybersecurity Exercises Proportional to Preparedness. Journal of Disaster Research **12**(5), 1081–1090 (2017)
3. Barela, J., Espinha Gasiba, T., Suppan, S., Berges, M., Beckers, K.: When interactive graphic storytelling fails. In: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). pp. 164–169. IEEE (Sep 2019)
4. Beuran, R., Chinen, K.i., Tan, Y., Shinoda, Y.: Towards Effective Cybersecurity Education and Training. Research report (School of Information Science, Graduate School of Advanced Science and Technology, Japan Advanced Institute of Science and Technology) **IS-RR-2016**(April), 1–16 (2016)
5. Carnegie Mellon University: SEI-CERT Coding Standards, `https://wiki.sei.cmu.edu/confluence/display/seccode`, [Online]
6. Chung, K., Cohen, J.: Learning Obstacles in the Capture The Flag Model. In: 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14). USENIX Association, San Diego, CA (2014)
7. CTFtime team: CTFTime - All about CTF, `https://ctftime.org`, [Online]
8. Davis, A., Leek, T., Zhivich, M., Gwinnup, K., Leonard, W.: The fun and future of CTF. In: 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14). USENIX Association, San Diego, CA (2014)
9. Dörner, R., Göbel, S., Effelsberg, W., Wiemeyer, J.: Serious Games: Foundations, Concepts and Practice. Springer International Publishing, 1 edn. (2016)
10. 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.) 27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019. IEEE (2019)
11. 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)
12. Groves, R.M., Fowler, F., Couper, M., Lepkowski, J., Singer, E.: Survey Methodology. John Wiley & Sons, 2 edn. (2009)

13. Hugo Gonzalez, Rafael Llamas, F.O.: Cybersecurity Teaching through Gamification: Aligning Training Resources to our Syllabus. Research in Computing Science **146**, 35–43 (12 2017). https://doi.org/10.13053/rcs-146-1-4
14. Hänsch, N., Zinaida, B.: Specifying it security awareness. In: 25th International Workshop on Database and Expert Systems Applications, Munich, Germany. pp. 326–330 (09 2014)
15. IEC 62443-4-1: Security for industrial automation and control systems - part 4-1: Secure product development lifecycle requirements. Standard, International Electrotechnical Commission (Jan 2018)
16. ISO: ISO 250xx Series. Standard, International Organization for Standardization, Geneva, CH (2005), `http://iso25000.com/index.php/en/iso-25000-standards`
17. ISO 27002: Information technology - Security techniques - Code of practice for information security controls. Standard, International Organization for Standardization, Geneva, CH (Oct 2013)
18. Krosnick, J.A.: Questionnaire Design. In: The Palgrave handbook of survey research, pp. 439–455. Springer (2018)
19. Mirkovic, J., Peterson, P.: Class capture-the-flag exercises. In: 2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14) (2014)
20. Nance, K., Hay, B., Bishop, M.: Secure coding education: Are we making progress? In: 16th Colloquium for Information Systems Security Education. pp. 83–88 (06 2012)
21. OWASP Top 10, `https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf`, Online, accessed June 2019
22. Oyetoyan, T.D., Milosheska, B., Grini, M., Cruzes, D.S.: Myths and facts about static application security testing tools: an action research at telenor digital. In: International Conference on Agile Software Development. pp. 86–103. Springer, Cham (2018)
23. Patel, S.: 2019 Global Developer Report: DevSecOps finds security roadblocks divide teams (July 2020), `https://about.gitlab.com/blog/2019/07/15/global-developer-report/`, [Online; posted on July 15, 2019]
24. Rodriguez, M., Piattini, M., Ebert, C.: Software verification and validation technologies and tools. IEEE Software **36**(2), 13–24 (2019)
25. SAFECode Charter Members: SAFECode - Software Assurance Forum for Excellence in Code, `https://safecode.org`, [Online]
26. Schneier, B.: Software Developers and Security (July 2020), `https://www.schneier.com/blog/archives/2019/07/software_develo.html`, [Online; posted on July 25, 2019]
27. Schonlau, M., Couper, M.: Semi-automated categorization of open-ended questions. Survey Research Methods **10**(2), 143–152 (Aug 2016), `https://ojs.ub.uni-konstanz.de/srm/article/view/6213`
28. Seaman, C.: Qualitative methods in empirical studies of software engineering. IEEE Transactions on software engineering **25**(4), 557–572 (07 1999)
29. Smith, C.: Content analysis and narrative analysis. Handbook of Research Methods in Social and Personality Psychology pp. 313–335 (2000)
30. Tews, M.J., Noe, R.A.: Does training have to be fun? A review and conceptual model of the role of fun in workplace training. Human Resource Management Review **29**(2), 226–238 (2019)
31. Whiting, L.: Semi-structured interviews: guidance for novice researchers. Nursing Standard **22**, 35–40 (03 2008)
32. Woody, C., Ellison, R., Nichols, W.: Predicting Cybersecurity Using Quality Data. In: 2015 IEEE International Symposium on Technologies for Homeland Security (HST). pp. 1–5. IEEE (2015)
33. Yang, X.L., Lo, D., Xia, X., Wan, Z.Y., Sun, J.L.: What security questions do developers ask? a large-scale study of stack overflow posts. Journal of Computer Science and Technology **31**(5), 910–924 (09 2016)