information

MDPI

Article

Cybersecurity Challenges in Industry: Measuring the Challenge Solve Time to Inform Future Challenges

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

[1] Siemens AG, 81379 München, Germany
[2] Institut für Angewandte Informatik, Universität der Bundeswehr München, 85579 Neubiberg, Germany; ulrike.lechner@unibw.de
[3] Departamento de Ciências e Tecnologias da Informação, Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR-IUL, 1649-026 Lisboa, Portugal; maria.albuquerque@iscte-iul.pt
* Correspondence: tiago.gasiba@siemens.com

Received: 17 October 2020; Accepted: 11 November 2020; Published: 16 November 2020

Abstract: Cybersecurity vulnerabilities in industrial control systems have been steadily increasing over the last few years. One possible way to address this issue is through raising the awareness (through education) of software developers, with the intent to increase software quality and reduce the number of vulnerabilities. CyberSecurity Challenges (CSCs) are a novel serious game genre that aims to raise industrial software developers' awareness of secure coding, secure coding guidelines, and secure coding best practices. An important industry-specific requirement to consider in designing these kinds of games is related to the whole event's duration and how much time it takes to solve each challenge individually—the challenge solve time. In this work, we present two different methods to compute the challenge solve time: one method based on data collected from the CSC dashboard and another method based on a challenge heartbeat. The results obtained by both methods are presented; both methods are compared to each other, and the advantages and limitations of each method are discussed. Furthermore, we introduce the notion of a player profile, which is derived from dashboard data. Our results and contributions aim to establish a method to measure the challenge solve time, inform the design of future challenges, and improve coaching during CSC gameplay.

Keywords: education; training; secure coding; industry; cybersecurity; capture-the-flag; game analysis; cybersecurity challenge; challenge solve time

1. Introduction

Over the last decade, the number of alerts issued for Industrial Control Systems by the United States Department of Homeland Security (DHS) [1] has been steadily increasing. This increase is especially significant and relevant for companies that provide products and services for critical infrastructures. According to an estimation by the DHS, about 90% of the reported security incidents result from exploits against defects in the design or code of software [2]. Adding to this, a recent large-scale study by Patel et al. [3] showed that more than 50% of software developers cannot spot vulnerabilities in source code [4]. The combination of these facts is a serious problem.

IT security standards such as ISO 27k [5], IEC 62443 [6], PCI/DSS [7], and the German BSI Grundschutz Kompendium [8] mandate the implementation of secure software development practices in the software development life-cycle. Compliance with these standards is often the aim of audits, especially in the critical infrastructure industry. To address customers' concerns and prove commitment to cybersecurity, the Charter of Trust (CoT) initiative was created to establish industry-wide principles in software and product development. The signing parties to the CoT commit to fulfilling these established principles. One of the CoT principles is the commitment to cybersecurity awareness and

education of industrial software developers. In addition to the CoT, several companies together have formed the SAFECode [9] alliance, which specializes in promoting secure coding best practices in the industry. This initiative is a rich source of information on the topic of industrial secure coding best practices.

One possible method to address secure coding and software quality issues and reduce the number of software vulnerabilities is through Static Application Security Testing tools (SAST). Several vendors have presented their proposed solutions to reduce software vulnerabilities. However, these have actually been shown not to perform well in detecting security vulnerabilities in source code [10]. One factor that makes the usage of such automated tools complicated is the excessive amount of false positives and false negatives [11,12] that these tools produce. The additional work, which must inevitably be done by a human, can be considerably high to understand SAST tools' reports and filter the false positives from the true positives. Furthermore, the presence of false negatives is a clear indicator that these tools are not perfect and that additional measures are required to ensure a good quality software [13].

In our work, we concentrate on awareness training of software developers, i.e., on the human factor. The motivation to focus on the software developer is that he or she is ultimately responsible for writing the software and interpreting the output of SAST tools. Based on the first author's industrial experience, we conclude that one possible way to improve software quality starts with raising awareness of software developers' secure coding through education. Kapp [14] provided a meta-analysis of game-based learning and concluded that using games as an educational method is an effective way of learning.

1.1. Cybersecurity Awareness through Cybersecurity Challenges

Capture-The-Flag (CTF) events are games that were introduced in the pentesting community as a means to exercise and improve system penetration skills and to simulate cybersecurity breach scenarios [15]. The participants in this kind of event are typically pentesters, cybersecurity experts, hobbyists, and academic students. For an overview of existing CTF games, see Kucek et al. [16]. CTF games pose complicated security puzzles to the participants, and often, these games take days [17]. It is not unusual that only a few of the participants manage to solve all the challenges, which have no fixed solution or no a priori solution (i.e., might even be impossible). This inability to solve challenges often leads to high frustration levels and eventually to players giving up on the CTF event itself. The reason for this is that challenges in CTF events generally have no clear solution, can go beyond typical (realistic) IT security topics, and are not coached events—i.e., each participant is left alone to learn IT security by himself or herself. In practice, the participants in CTF games increase their knowledge on attacking and defending IT systems by working together in groups, by searching for information on the Internet, or by asking friends.

CyberSecurity Challenges (CSCs) are serious games that refine the popular CTF format and adapt it to the industry. CSCs were introduced in [18] by Gasiba et al. to raise the secure coding awareness of software developers in the industry. Similar to CTF, a CSC event consists of a collection of challenges. However, these challenges are designed to consider secure coding guidelines and coverage of business-relevant topics. The challenges also contain at least one pre-defined solution. Furthermore, the CSC event is coached. This means that one or more persons supervise the event and help participants in solving the exercises. The goal of the coaches is twofold: to ensure that participants do not get stuck when solving exercises and to lower the frustration of the participants while increasing the overall happiness and learning effect. By solving CSCs, the participants focus on secure coding issues and enter a flow [19] state, which is characterized by increased concentration, attention, involvement, and enjoyment. Previous studies [20,21] indicated that these events result in the increased awareness [22] of the participants (industrial software developers) on the topic of secure coding. Furthermore, the CSC coaches ensure that participants are not frustrated during the event and help the participants remain in the flow state.

In contrast to CTF, an essential requirement for designing CSC games in an industrial context is to correctly plan the duration of a CSC event. This requirement exists because the time a software developer spends playing an awareness game is counted as unproductive hours and, if not planned properly, can introduce severe delays in ongoing projects [18]. Furthermore, not only can one software developer participate in such events, but also a whole team of software developers (e.g., up to 30 participants per event); this makes it even more urgent to perform a proper planning of the duration of the CSC event. Another critical factor to consider in the design of these games is the desired learning effect. This goal can be achieved by carefully planning the amount of exercises allocated for each different topic [23]. To achieve both these requirements, the whole event's duration needs to be planned carefully before the event. To do this, the duration of each challenge individually and its timing characteristics need to be known or estimated a priori to plan the event.

Good measurements of the time that players require to solve a given challenge must be based on data acquired from past events, i.e., from real-world data. Several methods can be used to measure this time, including simply looking at the clock and registering the data on paper. The precision of the measurement depends heavily on the technique used to measure. Note that different players with different backgrounds will solve the challenges in different amounts of time. In the present work, however, all the participants in the event have an industry background, and the challenges are related to their work environment. Therefore, the variance of the measurement is expected to not be as critical as if the background of the participants were much more diverse. In this work, we use a method to measure the challenge solve time based on collected timestamps, which are generated by standard tools (e.g., AJAX scripts). This is possible because the players in the CSC game always need to interact with a server component. Using the recorded data from previous events helps CSC designers estimate the challenge solve time and to further improve, refine, and inform future CSVevents. Based on this scenario, this work intends to address the following research questions:

RQ1: How doe we measure the time it takes to solve a cybersecurity challenge?
RQ2: What are the limitations of to measuring the time it takes to solve a cybersecurity challenge?
RQ3: Which factors can be used to inform CSC coaches during gameplay?

While we address the first two research questions quantitatively, the latter research question is addressed qualitatively, based on the experience gathered in the field. The present work introduces two methods to measure the challenge solve time and provides a first analysis of the data from cybersecurity challenges to aid CSC game designers in planning CSC events. The objective is to understand the interaction in cybersecurity challenges, identify implications for the game design, and optimize the gaming experience of participants. Furthermore, the present work aims at improving scientific knowledge on methods and limitations that can be used to measure the challenge solve time for CSC games. This study bases its conclusions on the data collected from 12 CSC events, with a total of 190 participants from the industry, which took place between 2017 and 2020. Furthermore, we identify player profiles based on player interactions with the infrastructure, which enable CSC coaches to direct their help towards individual players or teams while lowering their frustration, maintaining the flow state, and increasing overall happiness.

The rest of the paper is structured as follows. The next sub-section introduces the related work. Section 1.3 gives an overview of cybersecurity challenges in the industrial context. Section 1.4 introduces the two main proposals to measure the challenge solve time to address RQ1 and also briefly presents the data collection and research method. In Section 3, the results of the analysis of the collected data are presented. This section gives practitioners practical indications of challenge solve times and CSV event planning. Furthermore, this section addresses RQ2 and RQ3 by comparing the obtained results through the two different computation methods, and several factors are identified that aid CSC coaches in their activities during a CSC event. The last section concludes with an outlook on the work presented and briefly discusses future work.

1.2. Related Work

Awareness training on information security was addressed in McIlwraith [24], who looked at employee behavior and provided a systematic methodology and a baseline for implementing awareness training. In their work, Stewart et al. [25] argued that communicators, e.g., trainers, must understand the audiences' constraints and supporting beliefs in order to provide an effective awareness program. A serious game was defined as "a game designed with a primary goal other than pure entertainment", by Dörner et al. [26,27]. Kapp [14] defined serious games as an experience designed using game mechanics and game thinking to educate individuals in a specific content domain. Furthermore, he defined gamification as a careful and considered application of game thinking to solving problems and encouraging learning using all the elements of games that are appropriate. Cybersecurity challenges are serious games, according to the definition by Dörner et al. [18,20,28]. Using games as a means to raise IT security awareness has recently gained much attention in the research community to raise information security awareness [18,29,30]. The design of serious games' design, in a general form, along with the discussion on several aspects can be found in Alexis et al. [31]. In [32], Pesantez et al. performed a systematic literature review to understand serious game design methodologies, frameworks, and models. Lameras et al. [33] summarized in their study how design features can be planned, developed, and implemented. Their work focused on the learning outcomes, teacher roles, pedagogic value, and game attributes.

Capture-The-Flag (CTF) is a serious game genre in the domain of cybersecurity. Such serious games' common goals include skill improvement and training purposes, identifying the "best students", and assessing new employee skills, e.g., potential pentesters. In a CTF, a series of challenges needs to be solved by single players or teams of players. Various authors have identified that CTF games are fun activities with a high potential educational value [15,34]. Game activities are known to lead to increased experience in concentration, interest, and enjoyment resulting from increased levels of affective, behavioral, and cognitive involvement with the task at hand [19,35–37].

In [38], Mirkovic and Peterson investigated an adapted CTF method that they proposed to enhance students' cybersecurity education. While their work included both attack and defend exercises, they focused on fostering adversarial thinking. After the CTF takes place, the tutor explains possible solutions to the exercises. Additionally, an in-class post-mortem analysis of the event helps students identify their mistakes and further improve their skills.

While many of these publications mostly take for granted the suitability of CTF games as a tool to enhance cybersecurity awareness, this has been put into question [39–42]. In [43], Chung and Cohen evaluated several possible obstacles to effective learning through CTF. The major conclusions that they arrived at were that the challenges need to be adapted to the participants; the difficulty level should be adequate for the participants; and that the challenges should undergo a well-defined design process. These conclusions further emphasize the need to properly determine the time it takes for challenges to be solved, in order to plan the whole event. The duration of similar training events ranges from several days [38] (less common) to a single day [44] (more common). The first CTF is done in academia, while the latter is done by a commercial provider. Events that last several weeks have a severe financial impact and possible delay ongoing projects [18] and therefore typically do not find high levels of approval by management. Therefore, such a long duration is not adequate for the industry.

In [45], Rademacher gave some hints about the differences between academia and industry. Although their work did not focus on secure coding, we believe that their conclusions also extend to this area. The consequences of their work corroborate the need to address industry-specific requirements in the design of cybersecurity challenges. Gasiba et al. [18] discussed the requirements needed to address software developers in the industry as the primary target group. One requirement pertains to the design and planning of the challenges themselves, especially in terms of the time it takes to solve them individually (challenge solve time). The rationale is that the challenges presented to the participants should be solved in the amount of time that the event is designed to last, and the design should consider the desired and designed learning effect.

The evaluation of participant performance in a serious game is a vital part of assessing the game itself [46]. It is critical to understand how to refine and improve the game and know how effective a particular game is to raise secure coding awareness among its participants. Mäses et al. [47] proposed additional metrics to measure and track participants' progress. In their work, they mostly looked at weighted scoring and the time required to solve challenges. These metrics can be used by game designers to increase the effectiveness and efficiency of the learning experience. Recent work by Andreolini [48] proposed a scoring algorithm based on modeling of trainee activities during cyber range games. A comparison of both trainee scores and the desired activities path was the basis for the scoring algorithm. Path activities modeled as directed graphs were analyzed to identify player profiles.

Furthermore, in their work, Graziotin et al. [49] argued that happy developers are better coders. They showed that developers that are happy at work tend to be more focused, adhering to software development processes and following software development best practices. This result leads to the conclusion that happy developers can produce higher quality [13] and more secure code than unhappy developers. One important task for CSC coaches during gameplay is to ensure that the participants are happy.

In [22], Hänsch et al. introduced the concept of IT security awareness. They concluded that there are three components to awareness: (1) perception, or knowing threats, (2) protection, or knowing existing mechanisms, and (3) behavior, or correctly acting according to the circumstance. Cybersecurity challenges aim to increase software developers' awareness of these three dimensions as introduced by Hänsch et al. Both the CSC games and the CSC coaches address these three dimensions, ensuring that the participants at CSC events are exposed to and exercise these three awareness components.

The open-source CSC challenges are based on existing projects, which are left unmodified, and do not provide a mechanism that can be used to measure the challenge solve time. Since a decision was made not to modify these challenges, in order to reduce implementation and maintenance effort and costs, an alternative source of information was used. This alternative source was the logging data, which were stored in a SQLite [50] database, as generated by the CTFd [51]. As will be briefly introduced, later refinements of the CSC game used an in-house developed Sifu platform. Since this platform was developed by the authors, a mechanism using standard AJAX [52] requests was implemented. The AJAX requests were stored in a log file in the back-end and served as the second source of information. Note that both methods use logging information. In [53], Westera et al. also looked at log files to derive learning analytics for their serious game and concluded that log files contain a "hidden treasure" of information for further analysis.

CSC events are also coached events. The participants in the game are assisted during gameplay by one or more coaches, who act as supervisors of the event and serve as a guarantee that the frustration level of the participants is lowered and that the learning goals are achieved. Bird et al. [54] provided a good framework that can be adopted by CSC coaches to reach these objectives.

1.3. Cybersecurity Challenges and the Sifu Platform

A cybersecurity challenge event [18] is a one day serious game [26] event in which 10 to 30 software developers from the industry participate. During the event, the participants form teams (a maximum of four players each) and compete against each other by solving secure coding challenges. Although teams consisting of singular participants are allowed in this game, we have observed that in practice, the participants have always created or joined an existing team. The game's goal is to raise the awareness on secure coding, secure coding guidelines, and secure coding best practices of software developers working in the industry. For a thorough description and review of CSC events, we refer the reader to a forthcoming publication [21].

The CSC game was designed, based on internal demand for training, for the following software developers' backgrounds: web application, C/C++, and mixed. The choice of programming languages (C/C++) for the challenges was also driven by company internal demand for training. Note that these programming languages are widely used in the industry [55]. Additionally, software

implemented in C/C++ is considered the most vulnerable, according to a study by WhiteSource [56]. Due to its generic nature, the web application challenges are not based on a particular technology, but follow the OWASP [57] approach of generalizing the different existing web vulnerabilities. Both challenge types (web and C/C++) address many different cybersecurity topics, all of which are related to secure coding and focus on secure coding guidelines (company internal guidelines, OWASP [57], SEI-CERT [58], MISRA [59], and AUTOSAR [60]).

The main components of the CSC game are challenges, dashboard, and countdown. The challenges are the exercises that need to be solved by the individual teams. The dashboard is based on the CTFd [51] and contains the list of the challenges to be solved by the participants, along with their categories and points. It is used by teams and players to choose challenges to work on, ask for hints, collect points, and track the game (in the form of a dashboard). A countdown timer is shown to the participants during the event. This countdown serves as an incentive for the participants to continue solving the challenges, and it also serves as a guide for coaches to steer their efforts in helping individual teams. Figure 1 shows the overall architecture of a CSC game, along with its main components. Note that the elements present in this figure are general and not unique to CSC games. While this architecture is not unique to serious games, its usage is in the context of a serious game. The development of the CSC architecture and platform was done to facilitate integration into internal in-house components and to follow internal training methodologies.

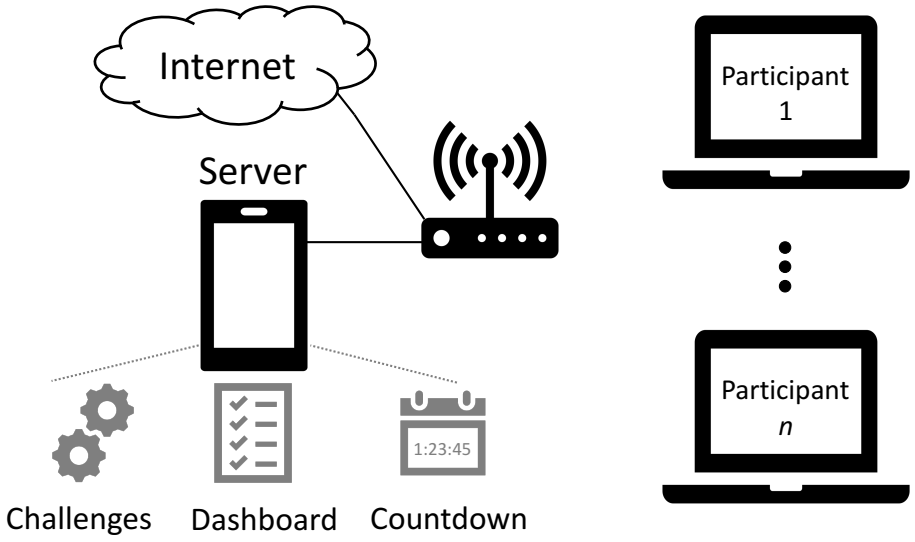

Figure 1. Architecture of cybersecurity challenges.

The CSC event starts with a one hour welcome to the participants, a briefing on how to play the game, team formation, and live examples on how to solve one challenge in each category. The main event lasts for six hours, including a lunch break. During this time, the individual players in each team collaborate together in solving the challenges. Two coaches (IT security experts) monitor the game while being played and assist with players' and teams' individual problems. The coaches' important task is to ensure that participants do not lose interest in the game and are kept motivated throughout the day—i.e., that the players achieve a flow [19] state. At the end of the main event, the winning team is announced, the participants are asked to fill out a small survey about the game (the participation of which is not mandatory), and finally, the coaches, in an open discussion with all the players, show the solutions of some challenges live. This last part of the CSC event (from team announcement until challenge solution) lasts one hour.

Each CSC challenge is developed to target one secure software weakness, which corresponds to one or more secure coding guidelines. The challenge consists of three phases [61]: Phase 1—introduction, Phase 2—challenge, and Phase 3—conclusion. In Phase 1, an optional phase,

the challenge, environment, and scenario are introduced. Furthermore, references to the secure coding guideline(s) which the challenge is about are discussed. In Phase 2, the player is presented with the challenge itself, described in Phase 1. In Phase 3, the player is given a debriefing on the challenge; this includes background information on previous real-world exploits and a short discussion of the possible consequences of exploiting the vulnerability.

Upon solving one challenge, a flag is awarded—i.e., a random-like code is given to the participants. This code can be submitted to the dashboard and be redeemed for points. During the workshop, the players accumulate points by solving challenges. At the end of the event, the team with the most points wins the CSC game. Further challenges can be unlocked upon solving a given challenge. The unlocked challenges can consist of follow-up questions (e.g., to test the understanding of the information given in Phase 3), or a more difficult challenge can be unlocked. Player interactions with the dashboard are kept in the dashboard's internal database. Although the team with the most amount of points is declared the winner of the CSC event, the game intends that every participant profits from the game and that by concentrating on solving the challenges [19], the awareness about secure coding, secure coding guidelines, and secure coding best practices is exercised.

1.3.1. Cybersecurity Challenges—Base Challenges

To address industry requirements [18], the CSC challenges are tailored to the participants' job description, level of experience, and the allocated time for the event. The tailoring of the challenge to the job description and the participants' experience is done to ensure the relevancy of the game for the day-to-day work of those participating as software professionals in the industry. The CSC games focus on the following domains [21]:

- Web: secure coding problems in web development with challenges related to web back-end and web front-end,
- C/C++: secure coding problems in the C and C++ programming language,
- Mixed: for mixed groups with background on web development and C/C++ development.

For each CSC event, challenges from the following categories are chosen to be incorporated into the particular CSC event for a given group of developers. The choice of which challenge category is offered in which event is discussed and agreed upon during the project setup, together with management.

- C/C++: challenges related to C/C++ secure coding guidelines,
- Comics: challenges related to general user behavior presented in a comic style (cf. also [62]),
- Forensics: challenges with analytic methods, e.g., the analysis of PCAP files and the traffic captured in these files with tools such as, e.g., Wireshark,
- Python: secure coding topics specific to the Python programming language, i.e., secure coding problems in data analysis,
- Questions: topics related to company IT security processes, software life-cycle, or specific to secure coding guidelines,
- Web: questions related to secure coding of web applications (both front-end and back-end).

The questions in the CSC game are multiple-choice questions [61]. These include questions related to company internal policies and address specific secure coding topics; they are also related to internal supporting organizations, as well as software code analysis. Possible answers to the questions are multiple-choice, and the number of tries is limited to avoid players answering by brute-force (see [61]).

The base CSC challenges are developed based on existing open-source components and projects. These challenges are continuously internally developed and integrated into the CSC platform to address the internal demand for different secure coding topics and ensure a wide variety of challenges. At the end of 2020, the pool of base CSC challenge comprised 223 different challenges on the different categories detailed above. A defensive perspective is given to the CSC challenges constructed by

adapting existing open-source components [21]. Therefore, the type of these challenges is more correctly identified as being defensive/offensive.

1.3.2. Cybersecurity Challenges—Sifu Platform

To address the shortcomings of the base CSC challenges, which are built on open-source components, a new platform was introduced—the Sifu platform. The design of these challenges is based on [18,61] and was presented in [20]. This platform allows CSC challenge designers to write interactive defensive challenges on secure coding, which are based on software vulnerabilities [23,63] and related to secure coding guidelines. The main differentiating factor of the Sifu platform compared to other existing challenges is that the challenges present in the Sifu platform are defensive, but the player interaction with the back-end is fully automated, while hints are also generated automatically. Individual players submit possible solutions to challenges to the back-end, which automatically assesses the challenge and generates hints if the submitted solution is not acceptable. These hints are intended to help the game participants in solving the challenge. The automatic challenge assessment evaluates to what extent the submitted solution follows secure coding guidelines, does not contain vulnerabilities, and works as intended (through functional tests).

At the end of 2020, the pool of Sifu challenges comprised 19 different secure coding challenges, all based on Common Weakness Enumeration (CWE), as shown in Table 1. Henceforth, we will refer to each challenge individually by its CWE identifier. For each challenge implemented in the Sifu platform, Table 1 also shows the SEI-CERT secure coding guidelines [58], which are related to the challenge.

Table 1. Common weakness enumeration used in Sifu challenges.

CWE	Ref.	Related SCG	Description
CWE-14	[64]	MSC06-C	Compiler Removal of Code to Clear Buffers
CWE-77	[65]	ENV33-C	Improper Neutralization of Special Elements used in a Command
CWE-121	[66]	ARR38-CSTR31-C	Stack-based Buffer Overflow
CWE-127	[67]	ARR30-CARR38-C EXP39-CSTR31-CSTR32-C	Buffer Under-read
CWE-134	[68]	FIO30-CFIO47-C	Use of Externally-Controlled Format String
CWE-190	[69]	INT18-CINT30-C INT32-C INT35-C MEM07-CMEM35-C	Integer Overflow or Wraparound
CWE-208	[70]		Observable Timing Discrepancy
CWE-242	[71]	POS33-C	Use of Inherently Dangerous Function
CWE-320	[72]		Key Management Errors
CWE-330	[73]	CON33-CMSC30-CMSC32-C	Use of Insufficiently Random Values
CWE-331	[74]	MSC32-C	Insufficient Entropy
CWE-338	[75]	MSC30-C	Use of a Cryptographically Weak Pseudo-Random Number Generator (PRNG)
CWE-562	[76]	DCL30-CPOS34-C	Return of Stack Variable Address
CWE-676	[77]	CON33-C ENV33-C ERR07-CERR34-C FIO01-C MSC30-C STR31-C	Use of Potentially Dangerous Function
CWE-682	[78]	FLP32-CINT07-C INT13-C INT33-C INT34-C	Incorrect Calculation

Table 1. *Cont.*

CWE	Ref.	Related SCG	Description
CWE-758	[79]	ARR32-C ERR34-C EXP30-C EXP33-C FIO46-C INT34-C INT36-C MEM30-C MSC14-C MSC15-C MSC37-C	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior
CWE-768	[80]		Incorrect Short Circuit Evaluation
CWE-778	[81]		Insufficient Logging
CWE-783	[82]	EXP00-C	Operator Precedence Logic Error

CWE: Common Weakness Enumeration, SCG: SEI-CERT Secure Coding Guideline.

1.4. Measuring Challenge Solve Time

Cybersecurity challenges can contain challenges based on two different backgrounds: (1) challenges based on existing open-source components and (2) own in-house developed challenges (Sifu platform). As discussed in [18], an essential requirement for CSC events in the industry is the proper planning of the event in terms of overall duration and learning goals. These requirements lead to the critical question of how to compute the time it takes to solve a CSC challenge, which we call the challenge solve time. The definition of the challenge solve time includes all three challenge phases, as described in the Introduction: introduction, challenge, and conclusion; this means that the challenge solve time counts the time from which a player or team starts working on the challenge until a correct solution is found, i.e., the challenge is solved. Knowing the challenge solve time is required to inform the design and refinement of future CSC events.

Challenges based on open-source components, since they are not necessarily designed to be integrated into a CSC event, typically have no implemented mechanism that easily allows us to measure the challenge solve time. According to the first author's experience, adding this functionality might either not be possible or be very expensive (i.e., high effort). The reasons that motivated the decision to not modify the open-source projects are based on development and maintenance costs, since for every new release of the open-source component, the functionality might need to be implemented again. Therefore, a different (lightweight) method is required to achieve this goal. The alternative solution is to re-use the data available in the CSC dashboard [51]. One main advantage of this solution is that these data are readily available, and no particular setup needs to be performed (the data only need to be processed and analyzed). For the Sifu platform, which is developed in-house, this functionality was implemented by the authors directly in the platform using standard existing methods (details will be given in the following).

The main limitation of using this proposed strategy (i.e., using dashboard data) is that it is not possible to determine the exact start and end time-points required to compute the challenge solve time. It might also be difficult to distinguish when more than one player on a team is working on different challenges, i.e., different team members work in parallel in solving challenges. However, a more in-depth analysis of these limitations will be addressed in the Results and Discussion Section.

Next, we will present the two different methods used to compute the challenge solve time: (1) based on dashboard data and (2) based on Sifu platform data. We intend to make use of the proposed computational methods, which will also be described in the following, to address our research questions. Additionally, we will use the collected data to determine the timing characteristics of the challenge solve time and use the data to analyze the player profiles with the goal to inform game coaches during gameplay.

1.4.1. Challenge Solve Time from the Dashboard

Figure 2 shows a CSC dashboard based on the open-source CTFd [51] project. The challenges present in the figure are web application challenges, as described previously. Note that, although the figure shows web challenges, this platform hosts all the CSC challenges. In particular, it hosts web, C/C++, comics, forensics, Python, and questions. Note also that this trait of hosting all the challenges will be used in later sections to compare differences in the estimation of the challenge solve time using dashboard data and separately using the Sifu platform.

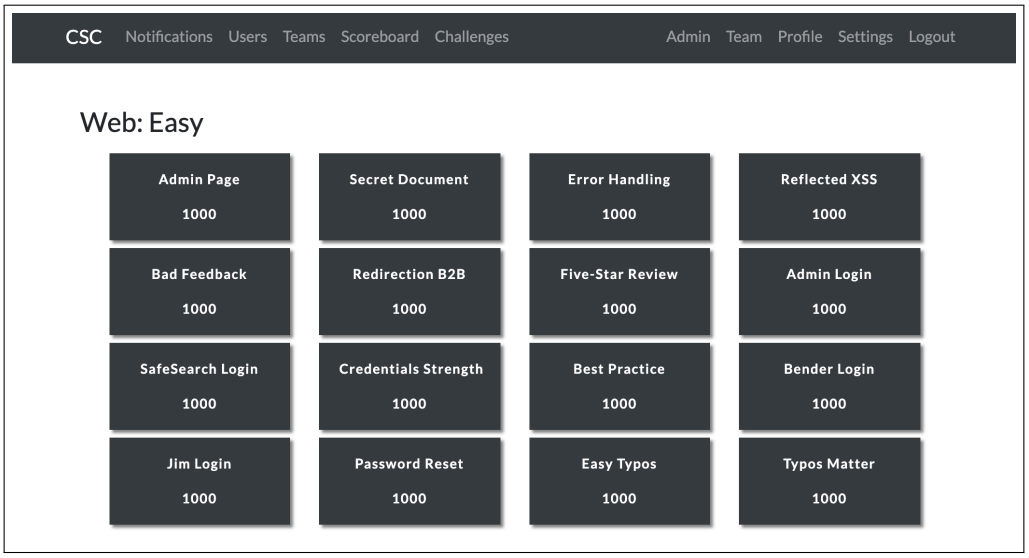

Figure 2. Exemplary dashboard of the cybersecurity challenge using CTFd.

Then, a player interacts with the dashboard, and an event is recorded in the internal SQLi database. This mechanism is an integral part of the CTFd [51] and is implemented using the standard Flask Python framework. There are three types of events that are stored in this database by the CTFd platform, and they are relevant for our analysis: C—Correct solve, W—Wrong solution, and H—Hint. The event C occurs when a player submits a correct flag to the dashboard; the event W occurs when a player submits a false flag or solution to a challenge to the dashboard; the event H occurs when a player requests a challenge hint from the dashboard. For each of these events (λ), the dashboard captures the following data in a database: event timestamp (t_λ), event type (e_λ), CSC challenge (c_λ), and team (τ_λ), in a 4fourtuple as follows: λ: $(t_\lambda, e_\lambda, c_\lambda, \tau_\lambda)$.

Figure 3 shows an example of events recorded in a dashboard database for a given team. The x-axis corresponds to time, and the y-axis marks the events that occurred. The first three events are as follows: the first event, which occurs at time t_1, is a correct flag submission C_1 for Challenge Number 1; the next event is a request for a hint H_2 for Challenge Number 2, which occurs at time t_2; the third event occurs at time t_3 and corresponds to a correct flag submission C_2 to Challenge Number 2.

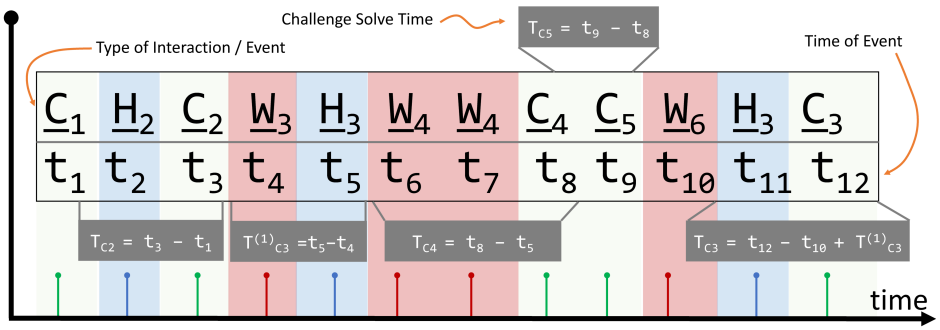

Figure 3. Computing the challenge solve time from dashboard interactions.

Using the definition of the challenge solve time and the dashboard events stored in the database, the challenge solve time for challenge n, i.e., T_n^S, is computed as follows:

$$T_n^S = \sum (t_i - t_j), \quad (1)$$

where i and j represent non-intersecting consecutive timestamps such that $i < j$ and Event i is related to a different challenge than n, and all events from $(i+1)$ to (j) are only related to challenge n. Figure 3 illustrates the computation of the challenge solve times for C_2, C_3, C_4, and C_5. Note that it is not possible to compute $T_{C_1}^S$ since only one timestamp (t_1) is available. For $T_{C_2}^S$, the related timestamps used for the computation are t_1 and t_3, since Event 1 is not related to Challenge 2 (it is related to Challenge 1), and all events between 2 and 3 are only related to Challenge 2 (i.e., H_2 and C_2). Therefore, $T_{C_2}^S = t_3 - t_1$. Note also that the time between Event 3 and Event 11 does not count toward the challenge solve time computation because it intersects with other computations, and between the two events, there are events related to other challenges other than Challenge Number 3.

The main idea to exclude time differences that include events related to other challenges is because it cannot be excluded that between these times, someone on the team is working on a different challenge (as recorded by the event). Therefore, in order to reduce computational errors, these time intervals are removed from the computation. The overall result of this restriction is that less data from the dashboard might be considered valid, i.e., it will discard some timestamp computations, which we argue should be the case.

1.4.2. Challenge Solve Time from the Challenge Heartbeat

Figure 4 shows the web interface to the Sifu platform containing a C/C++ challenge. When a player opens this page in the browser, the JavaScript code starts a routine that we call the challenge heartbeat. The challenge heartbeat consists of the web interface contacting the back-end server with a message stating that the web page is open. This functionality is implemented using standard AJAX. To reduce load on the client-side and on the server-side, the heartbeat message is sent only every 20 s. Note that the choice of the heartbeat time also corresponds to a trade-off between precision and load on the server side, to process several parallel heartbeat connections from all the concurrent players accessing the Sifu platform.

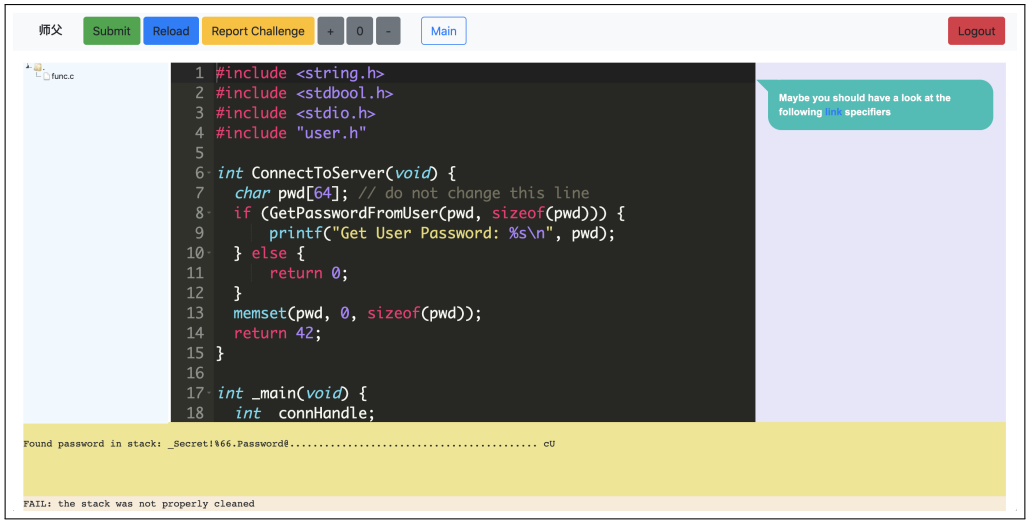

Figure 4. Web interface of the Sifu platform.

Every time a challenge heartbeat is received on the server-side, it is stored in an internal database. The values that are stored in this database are a three-tuple as follows: λ: $(t_\lambda, c_\lambda, \tau_\lambda)$. Similar to the

data stored in the database of the dashboard, t_λ represents the timestamp when the heartbeat message was received, c_λ the challenge, and τ_λ the team.

Like the dashboard, computing the challenge solve time of a given challenge consists of counting the total number of received heartbeats (only for solved challenges) that correspond to the challenge and multiplying this number by the heartbeat time (i.e., 20 s). This can be computed as follows:

$$T_n^S = |\mathcal{H}(n)| .\Delta^S, \tag{2}$$

where $|\cdot|$ represents the number of elements in a set, $\Delta^S = 20$ s is the heartbeat time, and:

$$\mathcal{H}(n) = \{\forall j \mid c_j = n\}, \tag{3}$$

is the set of all indexes such that the event is related to challenge n.

Figure 5 shows an example of heartbeat messages for a given team and for four challenges C_1 to C_4. The x-axis represents time, and the y-axis shows heartbeat messages. The star in the graph represents a solved challenge. Note that, since C_3 was never solved, i.e., the flag was not obtained, the challenge solve time cannot be computed. The challenge solve time for challenge C_1 consists of counting the number of heartbeats, which, as per the example in the figure, are split over two time intervals: Δ_1 and Δ_7.

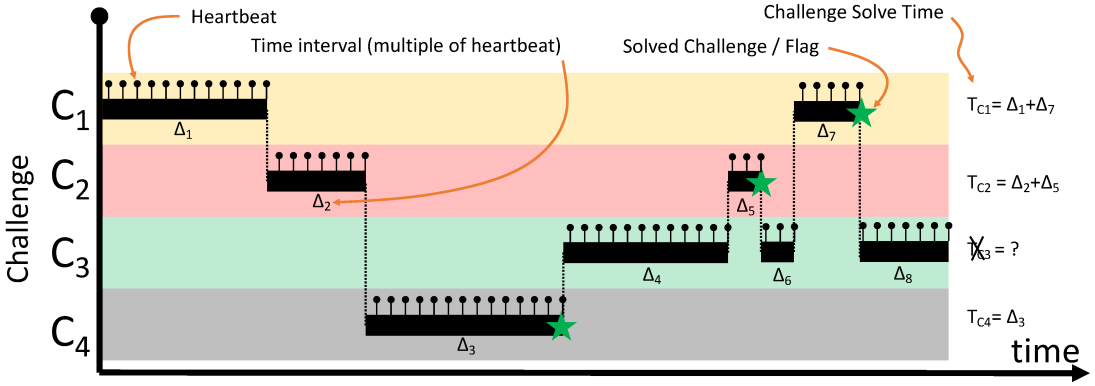

Figure 5. Computing challenge solve time from the Sifu platform's heartbeat.

1.4.3. Timing Characteristics: Probability of Solving the Challenge

To understand the dynamics of CSC exercises, we define a new function: the challenge failure probability, $P_f(t)$. This function is dependent on time and captures the timing characteristics of the challenge solve time, in the form of a probability. The main idea is to compute the probability of a challenge not being solved until time t has passed since the beginning to solve the challenge. This function will have a value of one at time $t = 0$, since, at the beginning of working on a challenge, no player has, by definition, yet solved the challenge, i.e., all players have failed to solve the challenge; therefore, $P_f(0) = 1.0$. Assuming that, given enough time, all challenges will eventually be solved, as time passes, the probability will monotonically decrease towards zero, i.e., $P_f(\infty) = 0$. To compute this function, we make use of the recorded events. Note that these events can be both collected from the CTFd database and also from the log files of the Sifu platform. Therefore, this method works for both methods of data collection. For each challenge and for each team, we can define the following function:

$$s(t) = \sum_{t_\lambda \in \mathcal{S}} \delta(t - t_\lambda), \tag{4}$$

where $\delta(t)$ is the Dirac-Delta function and $\mathcal{S}$ is the set of all times when a challenge was solved. The challenge fail probability at time t, $P_f(t)$ is then given as follows:

$$P_f(t) = \frac{\int_t^{\infty} s(u)du}{\int_0^{\infty} s(u)du} = 1 - \frac{1}{|\mathcal{S}|}\int_0^t s(u)du \tag{5}$$

where:

$$|\mathcal{S}| = \int_0^{\infty} s(u)du \tag{6}$$

represents the total number of challenges solved. Knowing the shape of this function and piloting its graph can give more information to a practitioner other than more straightforward statistic measurements such as Quartiles. An advantage of defining and computing this curve is comparing the results obtained using data from the dashboard against the results obtained from using data from the Sifu platform.

1.4.4. Team Profiles

Based on the information available in the CTFd database, we define the team profile based on the cumulative team interactions over time [28]. Furthermore, we define this function as being normalized in time (according to the entire duration of the workshop) and also being normalized in terms of the total number of interactions. A more formal definition of the team interactivity function is given as:

$$I(\tau) = \frac{\int_0^{\tau.T} i(u)du}{\int_0^{\infty} i(u)du} = \frac{1}{|\mathcal{I}|}\int_0^{\tau.T} i(u)du, \tag{7}$$

where:

$$|\mathcal{I}| = \int_0^{\tau.T} i(u)du \tag{8}$$

and

$$i(t) = \sum_{t_\lambda \in \mathcal{I}} \delta(t - t_\lambda), \tag{9}$$

and $\mathcal{I}$ represents the set of all times when an interaction with the dashboard occurred, T is the total duration of the workshop, and $0 \leq \tau \leq 1$. The definition of the function $I(\tau)$ suggests that its graph, for a team that has regular and constant interactions with the dashboard, is a simple straight line from the origin to the point (1, 1).

1.5. Challenge Playing Graphs

Another insight into the collected data, which is applicable to the Sifu platform, is described as follows. Since the Sifu platform contains a heartbeat mechanism, we draw a graph of time versus challenges. What is drawn in the graph are filled rectangles, which are placed between the last received heartbeat and the current heartbeat for every challenge. The idea of this graph is to know which challenges are being developed simultaneously be every member on the team. Furthermore, this graph can be used to determine the sequence of challenges that teams choose on average. The resulting black horizontal bars indicate that at least one player is currently browsing the corresponding challenge in the Sifu platform. The additional information that this graph can add to the knowledge of the dynamics of the game is knowing and understanding when players are playing a certain challenge and how many challenges are being solved in parallel. According to the first author's experience, teams work better when all team members are involved in a discussion and try to solve a challenge together.

2. Results

This section presents the results of the twelve CSC games played in an industrial setting. From 2017 to 2020, the authors collected data from 12 different CSC events (which took place online and in four different countries), whereby 190 software developers, pentesters, and test engineers with ages ranging from 25 to 60 and with an industrial background participated. Table 2 summarizes the 12 game events in chronological order with the number of participants and the focus domain of the CSC

event. Furthermore, it shows how data were collected: either using CTFd or CTFd together with the Sifu platform. The results were computed from the dashboard data contained in the SQLi database and from the Sifu platform logs. Data were pre-processed using Python scripts and then analyzed using R-Studio Version 1.2.5001 [83].

Table 2. Overview of CSC events.

Event No.	1	2	3	4	5	6	7	8	9	10	11	12
When	2017 November	2018 May	2018 July	2018 July	2018 September	2019 August	2019 September	2019 September	2019 October	2020 July	2020 July	2020 July
No. Players	11	12	6	30	16	14	15	7	23	21	20	15
Where	DE	DE	DE	DE	DE	CH	DE	DE	TK	OL	OL	OL
Focus	Mixed	Web	Web	Mixed	Web	Mixed	Mixed	Web	C/C++	C/C++	C/C++	C/C++
Data Collection	DB	DB	DB	DB	DB	DB	DB	DB	DB	DB Sifu	DB Sifu	DB Sifu

DB: Dashboard, DE: Germany, CH: China, TK: Turkey, OL: Online.

Event 1 validated the core CSC design and tested the gaming infrastructure. The participants to this event were cybersecurity experts. The participants in Events 2 and 3 were professional software developers. Participants in Event 4 were software developers, but it also included pentesters and quality engineers. The participants' distribution was as follows: 16% were non-software developers, and 84% were professional software developers. The comic challenges were used in Events 2 and 3 and were not part of any further CSC event (see [62]). Except for Event 10, participants of the remaining events were exclusively professional software developers. Participants in Event 10 included 7 computer science students, 7 professional software developers, and 1 assistant professor.

During a CSC event, participants gave their consent on the data collection in anonymized form. The participants also consented to be used for scientific research purposes. Since all the collected data were anonymized, tracing back to each person was not possible. This restriction applies in an industrial setting, where personal data are not allowed to be collected.

The data collected, as shown in Table 2, were comprised of dashboard data (for Events 1 through 9) and dashboard data and Sifu platform data (for Events 10 through 12). In the last three events, the Sifu platform was used in conjunction with the dashboard, as described in the Introduction. Note that the data collected through both means were related to the same CSC challenges and the same teams. The data collected in these events thus allowed us to compare the results obtained using both challenge solve time computational methods, as discussed previously.

In the following, we structure the presentation of the results in the following way. First, we explore the results obtained for challenge solve time using the dashboard (Events 1–9). Next, we present the results for challenge solve time based on the data collected from the Sifu platform, during Events 10 to 12. In the following section, we compare the challenge solve time as computed using both dashboard data and Sifu platform data for Events 10 to 12. Following this, we discuss the team profiles, as obtained through player interaction with the dashboard. Next, we compare the player profiles with the final ranking in terms of points, obtained by the individual teams. Finally, we present the results related to the challenge playing graphs.

2.1. Time to Solve Challenges Using Dashboard Data

Table 3 gives a summary of the challenge solve time, based on collected dashboard data for Events 1 to 9. The challenge solve time is shown for each category of challenges, and the following results are summarized in the table: average (avg.), minimum (min.), maximum (max.), standard error (stde.), 25% quartile (q25), 50% quartile (q50), 75% quartile (q75), 99% quartile (q99), and kurtosis (k) values.

Table 3. Detailed challenge solve time results for dashboard. stde., standard error; q, quartile; k, kurtosis.

	avg. (s)	min. (s)	max. (s)	stde.	q25	q50	q75	q99	k
C/C++	1973	69	6852	201.5	666	1172	2810	6702	3.24
Comics	245	14	1494	41.2	42	105	275	1444	7.22
Forensics	555	10	6772	54.4	81	227	545	4988	19.30
Python	1269	63	6893	176.3	375	743	1844	5553	7.07
Questions	246	3	6904	14.3	23	52	153	3865	40.20
Web	1025	7	6973	65.9	197	492	1173	5876	7.07

The analysis illustrates that different topics and kinds of questions yield different times to respond. The results obtained in this work show that it takes on average about 30 min to solve C/C++ challenges, 20 min for Python challenges, 15 min for web challenges, 4 min for multiple-choice questions, and 4 min for the comics challenges. These results indicate that C/C++ challenges are more difficult to solve than web challenges. Generic multiple-choice questions and questions based on comics are less challenging to solve, as expected.

Surprisingly, in Table 3, the forensic challenges, although not the core competency of the players, are solved on average in only slightly more time than multiple-choice questions. The authors attribute this to the fact that, in order to solve these challenges, specialized tools (in our case, Wireshark) helped the participants, and it cannot be excluded that the participants that chose to solve these challenges had experience with these tools. Even if participants did not previously know this tool, they could navigate it and find the appropriate option(s) to solve the problem.

Concerning the comics challenges, the first author's previous work determined that the participants did not find the comics to be useful during the CSC events. These challenges were rather experienced as distracting; see [62]). For this reason, these types of challenges were only present in the second and third events.

Note that the two categories forensics and questions have high kurtosis values. These high values are an indicator that there is no given, well-known path to the result. The participants might know how to use an appropriate tool, know a simple method to solve the challenge quickly, or use their skills to solve it. Potentially, players' background and experience lead to the differences in time that it takes to solve a challenge. For the Python challenges, however, the average time is higher, and the kurtosis lower, i.e., the distribution is sharper. We interpret this as an indication that it takes more effort to solve a Python challenge, but players may follow a defined strategy to solve the challenge in a given time. A similar effect is observed for the C/C++ and web categories.

2.2. Time to Solve Challenges Using the Challenge Heartbeat

Table 4 gives a summary of the challenge solve time, based on collected Sifu platform data for Events 10 to 12. Note that all the challenges in the Sifu platform are C/C++ challenges. The challenge solve time is shown for each category of challenges, and the following results are summarized in the table: average (avg.), minimum (min.), maximum (max.), standard error (stde.), 25% quartile (q25), 50% quartile (q50), 75% quartile (q75), 99% quartile (q99), and kurtosis (k) values.

The minimum average challenge solve time is about 14 min, and the maximum average challenge solve time is about 31 min. All the kurtosis values are small (single digit), which indicates that the data distribution is a sharp curve.

Surprisingly, some challenges (CWE-190, CWE-242, CWE-330, and CWE-338) show low minimum values for the challenge solve time (40 to 98 s). This observation indicates that, for some players, these challenges were easy to solve.

Table 4. Detailed challenge solve time results for the Sifu platform.

	avg. (s)	min. (s)	max. (s)	stde.	q25	q50	q75	q99	k
CWE-14	1425	303	3969	431	625	996	1807	3854	3.33
CWE-77	1866	479	4691	560	1068	1259	2249	4592	2.85
CWE-121	1545	325	3381	657	501	619	2899	3362	1.26
CWE-127	1348	120	5972	776	550	740	750	5659	5.05
CWE-134	1460	220	5121	925	300	700	961	4954	3.14
CWE-190	1280	40	5821	219	290	800	1385	5526	5.20
CWE-242	983	60	4380	352	340	459	1049	4118	6.09
CWE-330	839	98	3036	552	280	340	440	2933	3.20
CWE-338	1858	40	6512	789	380	729	2938	6284	3.29
CWE-676	1681	500	3138	318	1146	1429	2106	3116	2.09
CWE-758	1248	140	5567	651	190	547	1155	5311	5.01

2.3. Comparison of Time to Solve Challenge Computation Methods

Events 10, 11, and 12 were used to compare the two different challenge solve time computation methods previously introduced. For this, the challenge solve time was computed using dashboard data and, independently, computed using data from the Sifu platform. Figure 6 shows the computed challenge solve time, which is based on the dashboard data, and Figure 7 shows the computed challenge solve time, which is based on data collected from the Sifu platform.

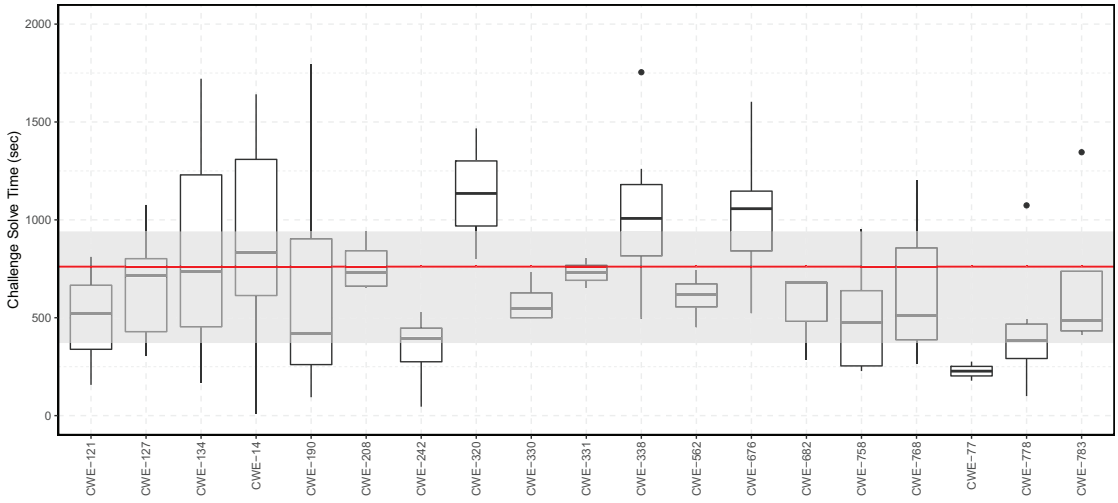

Figure 6. Challenge solve time from the dashboard based on flag submission of Sifu challenges.

In these graphs, the upper limit of the gray bar represents the 75% quartile, while the lower limit of the gray bar represents the 25% quartile, both based on combining all the results for all the challenges. The red line represents the average challenge solve time of the combined results for all the challenges.

This analysis shows that for challenges CWE-14, CWE-121, CWE-127, CWE-242, CWE-758, and CWE-783, both computation methods show approximately the same result. For the remaining challenges, the differences in computation are that either the average time is higher for results collected from the dashboard or that the average time is higher for results collected from the Sifu platform's challenge heartbeat.

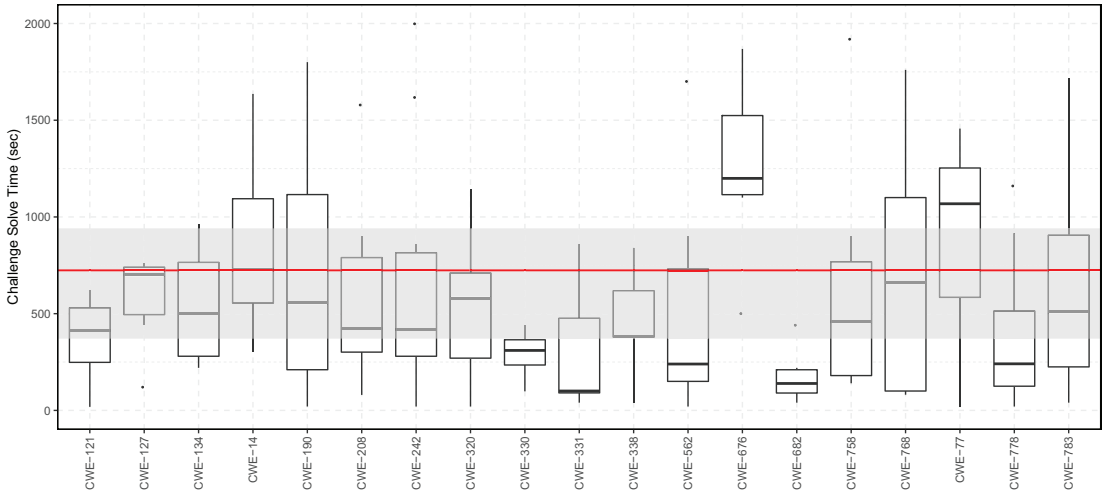

Figure 7. Challenge solve time from the challenge heartbeat in the Sifu platform.
Figure 7. Challenge solve time from the challenge heartbeat in the Sifu platform.

The discrepancies between these two results were observed in practice by the first author during the three events. This difference occurs because of the two observed phenomena: (1) some teams did not immediately submit the flag to the dashboard upon solving it in the Sifu platform, and (2) some teams did not close the web browser window containing the Sifu challenge after submitting the flag to the dashboard. These two phenomena can either result in an overestimation or an underestimation of the challenge solve time using data from the dashboard or Sifu platform.
The discrepancies between these two results were observed in practice by the first author during the three events. This difference occurs because of the two observed phenomena: (1) some teams did not immediately submit the flag to the dashboard upon solving it in the Sifu platform, and (2) some teams did not close the web browser window containing the Sifu challenge after submitting the flag to the dashboard. These two phenomena can either result in an overestimation or an underestimation of the challenge solve time using data from the dashboard or Sifu platform.

To better understand the overall estimation between the two methods, the challenge fail probability, as previously defined, was computed for the consolidated data of all the challenges both for the dashboard and for the Sifu platform. This analysis is possible since all challenges are of the same category. The analysis is, therefore, valid for a typical C/C++ challenge in the Sifu platform.
To better understand the overall estimation between the two methods, the challenge fail probability, as previously defined, was computed for the consolidated data of all the challenges both for the dashboard and for the Sifu platform. This analysis is possible since all challenges are of the same category. The analysis is, therefore, valid for a typical C/C++ challenge in the Sifu platform.

Figure 8 shows the computed probability of challenge solve versus time. The results show that this probability is high (close to 1.0) for small time values and becomes very low (close to 0.0) for large values of time. This result is expected as the participants need time to solve the challenges, i.e., initially, the failure rate is high; however, with time, the different challenges start being solved, and the probability that challenges are not solved starts decreasing with increasing time.
Figure 8 shows the computed probability of challenge solve versus time. The results show that this probability is high (close to 1.0) for small time values and becomes very low (close to 0.0) for large values of time. This result is expected as the participants need time to solve the challenges, i.e., initially, the failure rate is high; however, with time, the different challenges start being solved, and the probability that challenges are not solved starts decreasing with increasing time.

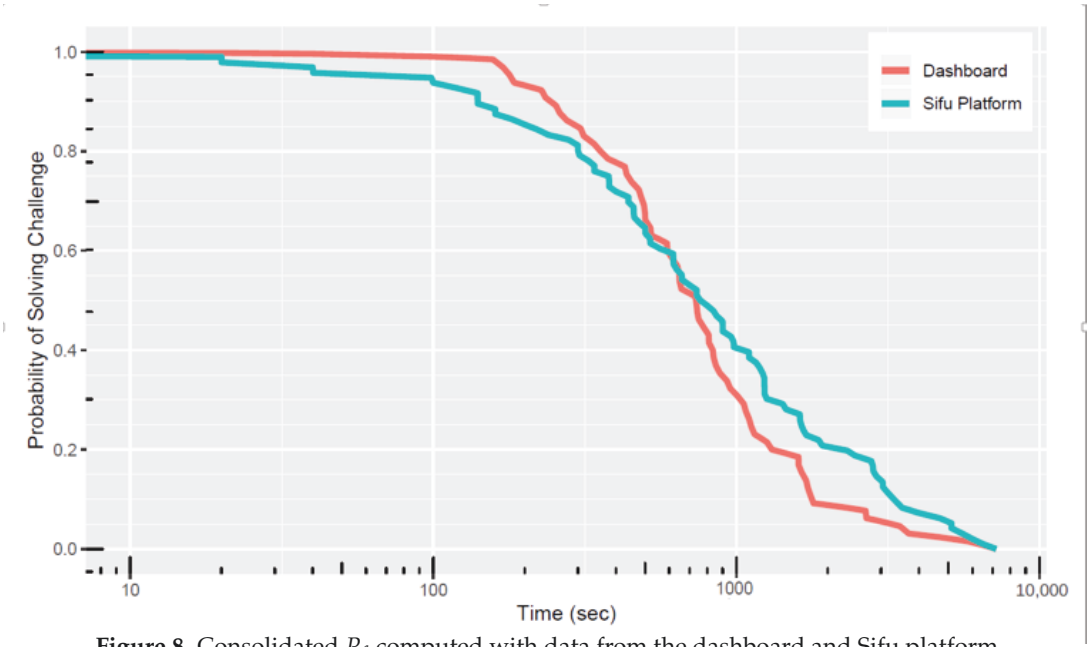

Figure 8. Consolidated P_f computed with data from the dashboard and Sifu platform.
Figure 8. Consolidated P_f computed with data from the dashboard and Sifu platform.

As expected, according to results shown in previous subsections, there are differences between the computed challenge solve time using the dashboard data and the challenge solve time computed using the challenge heartbeat from the Sifu platform. We can observe from the figure that both curves have a similar behavior and are close together, showing similar results. A practitioner can use these curves to understand the margin of error incurred by using either measurement technique. It can also be used for the planning of CSC events.

One insightful aspect of the graph is that the dashboard curve has a higher slope and that both curves cross each other at about 50% probability. The reason for this is currently unknown and is a topic that requires further investigation. The authors suspect that this difference might be related to the two different observed phenomena as detailed above.

Finally, Figure 9 shows the same curve, but for the eleven individual challenges. The plots show a general agreement between both measurements and exemplify the individual margin of errors for both measurement techniques.

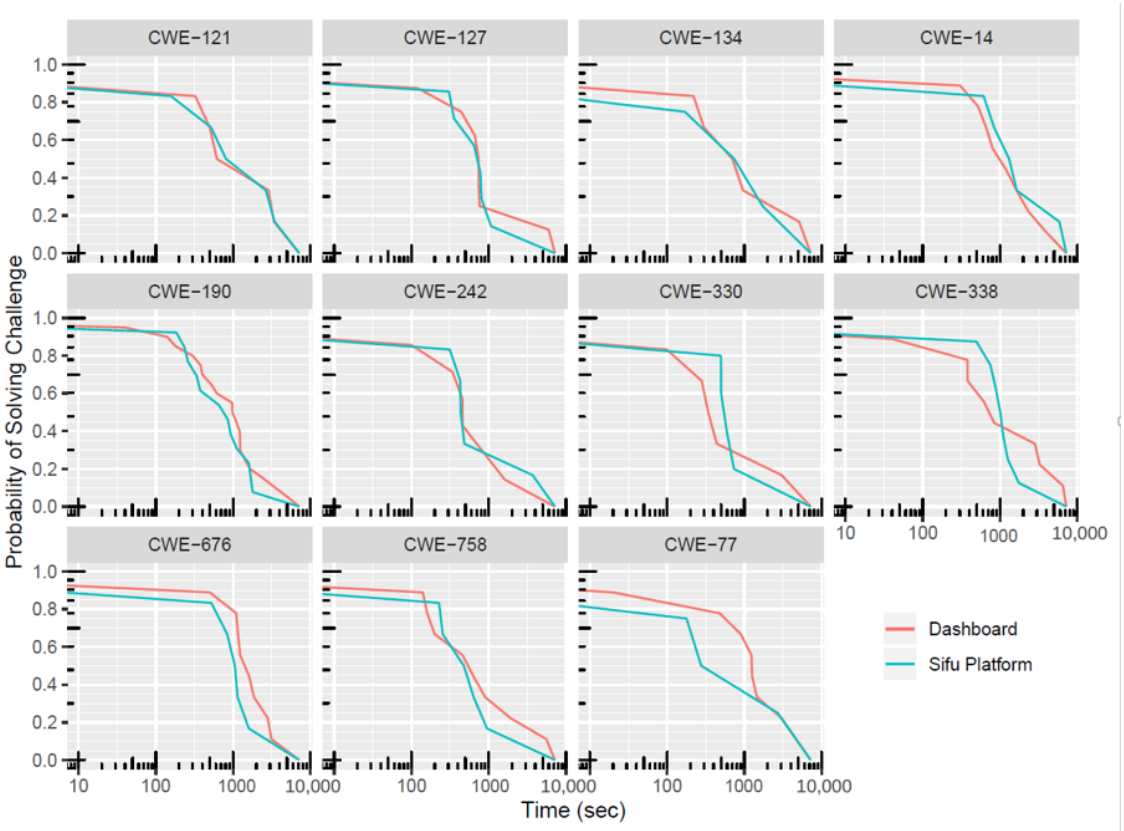

Figure 9. P_{ff} for different challenges computed with data from the dashboard and Sifu platform.

Another interesting conclusion for a practitioner is the fact that, by computing these probability curves, not only can a practitioner find better estimates for individual challenges, but also combining the curves into a consolidated view (as in Figure 8) leads to a better overall estimation of the challenge solve time for a complete CSC event. The more data a practitioner collects, the better the computed curves and estimates are, and therefore, the better the overall estimation for a complete CSC event is.

Concerning RQ3, obtaining these curves is also essential for a good coaching strategy. The individual curves are plotted in a graph that has a logarithmic x-scale. This fact means that the time increases exponentially when moving towards the right-hand side of the graph. We observe that, after about 20% of the challenge failure rate (i.e., at approximately Minute 15), moving more to the right increases the time considerably; however, both curves' slope is more flattened than in the transition zone at about 50%. This observation means that it takes a considerably higher amount of time (exponentially increasing) to solve a challenge after a team has spent about 15 min on its solution.

Note that the 15 min mark is well in agreement with the average challenge solve time presented in the previous section for this type of challenge.

At this time (15 min), the authors recommend that the CSC coaches ask if the teams need assistance in solving the current challenge. With this, the highest effort (80%) was already performed by the team members, and the coach can intervene to help finish the challenge. This help from the CSC coaches can be done by providing additional hints and indications towards the solution. According to experience in the field, the participants report that the coaches have helped reduce frustration during gameplay and increase overall happiness with playing the game. One possible way to keep track of this is to implement a panel in the Sifu platform for CSC coaches; when the challenge solve time of a given team exceeds a certain threshold, the coaches can intervene and help the team.

2.4. Team Profiles

Team profiles represent the overall gaming playing mode that a team has experienced when solving the individual challenges. Here, the authors looked at the curves resulting from the normalized cumulative interactions of teams with the dashboard versus the normalized CSC event time (typically 6 h). Analysis of the team interactions with the dashboard resulted in three team profiles: fast, slow, and automated. The automated profile was rejected in our study since it resulted from pentesters (during Event No. 2) attacking the dashboard using automated scripts. This profile was later confirmed by asking the team members directly about the phenomenon observed in the data.

Although rejected in further analysis, this profile is a clear indicator of non-human interaction with the dashboard; this can be used by coaches to determine possible foul play during a CSC game. However, for all the remaining CSC events, the first author has not seen this scenario occur in practice anymore. This behavior further illustrates the difference between the typical CTF mentality and target group and the target group of players to CSC events.

After discarding the automated profile, the two remaining profiles from human interaction and gameplay are considered:

- Fast: the interaction takes place mostly at the beginning, but wears out as gameplay advances,
- Slow: most of the interactions happen towards the end of the gameplay, with fewer interactions at the beginning.

By looking at the resulting curves, the authors determined that the shape of the curves resembles the following formula:

$$I(\tau) = \frac{a^{\tau} - 1}{a - 1}, \quad a > 0, \quad 0 \leq \tau \leq 1. \tag{10}$$

Note that τ represents the normalized time (i.e., 0.0 represents the start of the CSC event and 1.0 the end of the CSC event). The parameter a changes the shape of the curve: for $a > 0$, this curve results in a slow profile, and for $a < 0$, the formula results in a fast profile.

Analysis of the CSC Events 1 through 9 shows that the resulting minimum average error has a value of 0.05 and is similar for both fast and slow profiles. Although a theoretical explanation for the curve shape formula is not available, it has been shown to produce relatively good results by curve-fitting using a minimum-squared-error algorithm from all the collected data from the first nine events. Furthermore, this value indicates a sound fit between the model given by Equation (10) and the collected data from the real CSC events.

Figure 10 shows two examples of best-fitting curves, for fast and slow profiles, using minimum-squared-error criteria for Events 4 and 3. Note that, for completeness, we included the automated profile curve, which was observed in CSC Event No. 2.

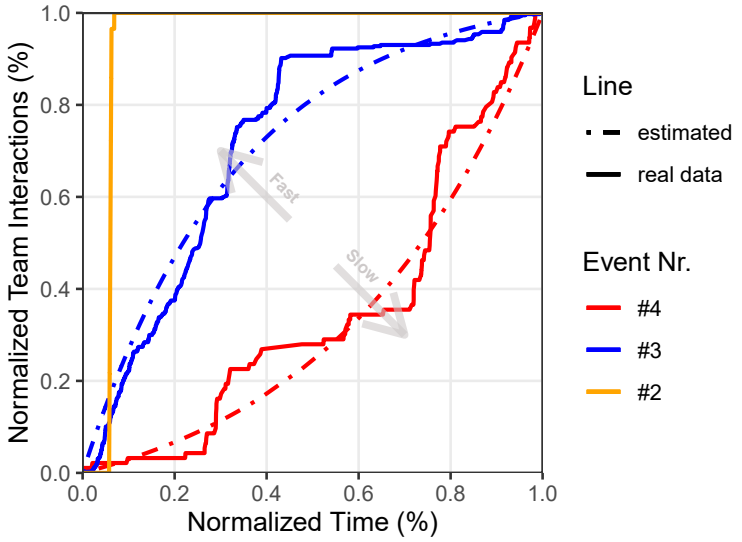

Figure 10. Examples of normalized time vs. normalized total interactions.

Plotting the normalized team interactions with the dashboard over normalized time (see Figure 10) depicts the two expected team profiles: slow and fast.

Implications for practitioners and CSC coaches include facilitating the management of the gaming experience of the participants. Both curves indicate that time management needs attention for both profile types: a coach should understand which profile a team is showing and advise and guide participants accordingly. Field experience has shown that teams that start with a fast profile (i.e., solving too many challenges in the beginning) tend to become more frustrated towards the second half of the game, while teams with a slow profile tend to be frustrated at the beginning. The authors perspective is that a coach should try to help the teams follow a more linear profile—i.e., solving challenges regularly over time. In this way, a coach manages the team expectation during gameplay and helps the team members experience the flow [19] state.

Management of expectations during gameplay and strategies to ensure that players experience the flow state are topics that require further investigation. However, practitioners can use the simple model presented as a first approach to expectation management. An additional consequence is that, with careful game planning, the total number of challenges can be pre-determined before the event starts. Therefore, it is possible to compute the player profile's approximation during the CSC event by normalizing the current interactions with the dashboard by the possible interactions through dashboard pre-configuration.

2.5. Profile and Team Performance

The gathered data during Events 1 through 9 were analyzed in terms of the relation between team profile and team performance in the game. The team performance is given in terms of final team rank, as computed using the final score for each team in the dashboard. The authors identified all the corresponding curve types (fast and slow) utilizing curve-fitting, for all the teams, and compared them with the final game ranking (i.e., first place, second place, and so on). Table 5 summarizes these results; for the nine games played, four teams ended in first place with a fast profile, while five teams ended in first place with a slow profile, and so on for the remaining places.

Table 5. Ranking of profiles and scores.

Place	1st	2nd	3rd	4th	5th	6th	7th	8th
Fast	4	4	5	4	3	3	2	3
Slow	5	5	4	5	2	2	2	0

The data in Table 5 also show that, if looking at teams that finish in first place, fifty-six percent belong to the slow profile and 44% to the fast profile. However, looking at the first, second, and third place, the distribution is 48–52% for the fast and slow profile, respectively. The expected value for the place of fast and slow profiles is $E(Fast) = 4.0$ and $E(Slow) = 3.3$, corroborating the previous observations.

This observation means that a team having a slow or fast profile is not guaranteed to win the game. Nonetheless, the slow profiles do show slightly better results than the fast profiles. Note that as the collection of challenges is individual for each CSC event, the teams' ranking is used for this comparison of performance and not the number of flags acquired in the challenge.

Further research is needed to establish a relation between the number of flags won in a game, the number of challenges mastered, the team's profile, and learning outcomes in terms of awareness, knowledge, and skills. Further empirical evidence and analysis are needed to understand the differences between slow and fast profiles. Nevertheless, the results presented can be used as a first approximation by practitioners.

2.6. Playing Sifu Challenges

Figure 11 shows real-world examples of the Sifu challenge heartbeat for six different teams that participated in Events 10, 11, and 12. Figure 5 shows an overview of the theoretical model corresponding to this figure. While in the theoretical model, only the heartbeat of a single challenge is shown at any point in time, the results in Figure 11 show a different picture. Here, we can observe that all the teams had at least two Sifu challenges heartbeats running simultaneously. Observation during these events, by the first author, showed the following factors that contributed to this result.

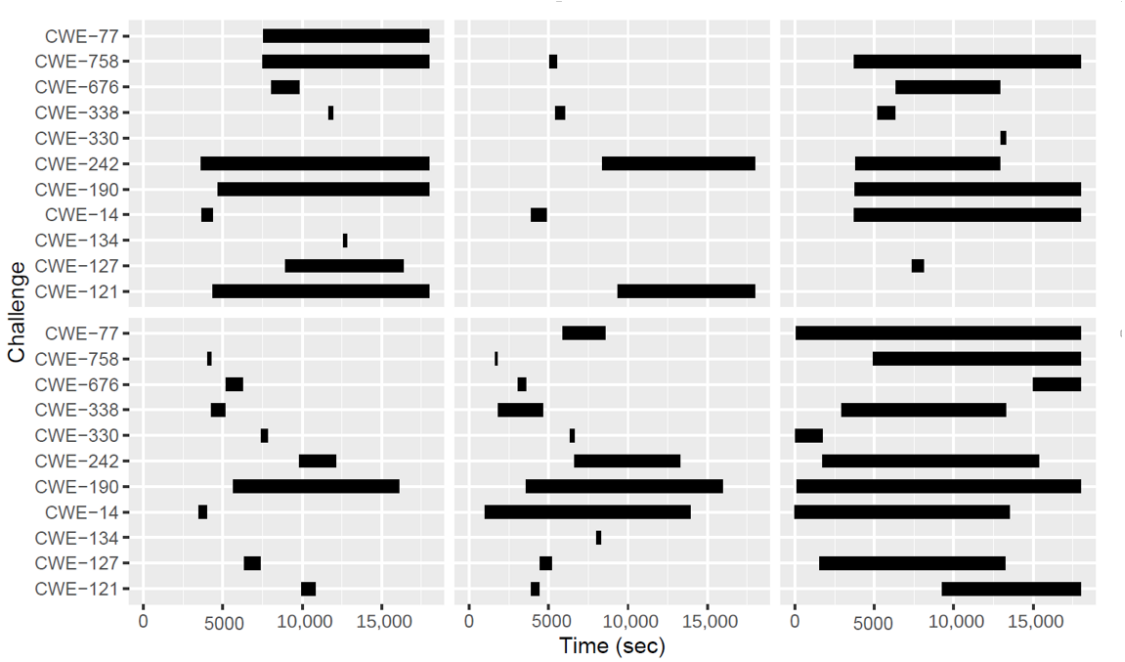

Figure 11. Six real-world examples of the challenge heartbeat.

- Teams were composed of several players. It was observed that more than one player was working separately and in parallel towards solving the Sifu challenges. The team members decided to use this strategy in order to increase the chances of collecting a higher amount of points and, therefore, winning the CSC event,
- Some team members have decided to keep web pages open while working on other challenges in the Sifu platform. During this time, two or more challenge heartbeats were running in parallel for the same player.

Note that these figures can also be used by practitioners to determine the average path that CSC teams and players take in solving the challenges. For example, the graphs shown in Figure 11 indicate that teams tend to start early in solving challenge CWE-14, while the tendency to start solving challenge CWE-121 is after more than two hours, i.e., half-way through the entire time of the workshop. This can be used to determine "popular" challenges, i.e., challenges where players on average try an early solve. Knowing which challenges are more popular than others can serve to inform CSC coaches on, e.g., the distribution and focus of advice among teams.

3. Discussion

In this work, we presented two methods to compute the time it takes to solve a challenge in the cybersecurity challenge platform. The challenges in this platform are either based on open-source components or an in-house developed platform called the Sifu platform. Computation of the challenge solve time is based on post-processing the timestamps that are either recorded in the CSC dashboard or through a heartbeat mechanism that is implemented in the Sifu platform. The recording of the timestamps uses standard and widely established methods, such as AJAX. We also discuss how to use the player interactions with the dashboard to derive an interaction profile, which we call the team profile. Furthermore, we use the timestamps to compute a curve that represents the probability of not solving a challenge as a function of time. Finally, we use the heartbeat events to plot a challenge playing graph. This graph represents when players are trying to solve CSC challenges. We conclude that, from all the rich set of information that we are able to derive from the back-end logs, these logs constitute a treasure trove containing information ready to be analyzed [53]. The following provides a brief list of the results we presented. This list serves both as a short reference to the previous sections that deal with each specific topic, but also as a source of ideas for practitioners designing a similar game and who wish to better understand the mechanics of their own artifact.

1. Method to measure the challenge solve time, based on dashboard data;
2. Method to measure the challenge solve time, based on the implementation of a heartbeat function;
3. Analysis of the team profile based on player interaction with the dashboard;
4. Timing characteristics of the challenge solve time in the form of the probability of solving a challenge as a function of time;
5. Challenge playing graphs to determine player paths.

Knowing good estimates of the challenge solve time is a pre-requisite for practitioners who wish to design or refine the cybersecurity game. The results hereby presented can also be extended to other serious games, in particular those that use a dashboard, implemented in software, and that captures interactions and timestamps, to track the progress of individual teams. Our results show, as expected, that both methods to compute the challenge solve time produce similar results. Therefore, we can conclude that, for the cybersecurity challenges using open-source components, changing these components to add additional mechanisms to determine the challenge solve time is not really necessary. This can lead to lowering the costs for the maintenance and update of these components.

Our results show the different challenge solve times for the different categories of challenges existing in the CSC platform. Table 3 shows a summary of our results for the base CSC challenges, while Table 4 shows the results for the Sifu platform. These tables also indicate that the average challenge solve time is lower for the comics, questions, and forensic challenges (i.e., lower than 560 s), while being higher for the C/C++, Python, and web challenges (more than 1000 s). Furthermore, the C/C++ challenges take on average about double the time it takes to solve web challenges. The kurtosis for the challenges forensics and questions is much larger than 3.0, indicating a leptokurtic curve, i.e., a very sharp curve, while the kurtosis for C/C++ is about three for both the dashboard results and for the Sifu platform results. This indicates that the distribution of the challenge solve time for the C/C++ challenges follows a normal distribution.

Both Tables 3 and 4 show the values for the 25%, 50%, 75%, and 99% quartiles. These values can be used by a designer who wishes to control the level of difficulty of solving a game. The advantage of these tables is that they can be easily consulted. A more detailed analysis of the characteristics of the challenge solve time is shown in Figure 8. This figure shows the average probability that a team has to fail the solution of a challenge as a function of time. We expect that practitioners that use this method will in time be able to gather more and more statistics, thus reducing computational errors. While this curve allows a more granular tuning and design of a CSC event, practical results might vary due to inherent human factors.

Based on the experience of the first author, considering the player profile results with the challenge playing graph, the following advice can be drawn for designers of CSC events. Since players in teams with a fast profile tend to become more frustrated towards later stages of the game, a better balance can be found; to keep the team profile more constant (i.e., linear), we suggest to make challenges available to participants in "waves". This means that, instead of releasing all the challenges immediately at the beginning, a CSC designer can program the dashboard to initially hide some challenges, which will be slowly released during gameplay. Although further research is needed at this point, our experience shows that this further helps the players lower their frustration during gameplay and also helps them to remain in a continuous state of flow [19].

Coaches who have a good understanding of the characteristics of the challenge solve time can also use this to decide on the best time when to approach teams to give advice. This point is especially relevant when the game is played in different countries where the participants have different backgrounds. Our experience has shown that in certain countries, players have a more active role when participating in the CSC event, e.g., by actively asking questions of coaches, while in other countries and cultures, the opposite is true. Therefore, knowledge of this curve can potentially improve the situation (lower frustration and increase learning effect) for the latter case. Additionally, the same methodology can be extended to the Sifu platform, where automatic hints can be given to players dependent on the characteristics of this timing curve. Furthermore, an automatic hint system based on artificial intelligence, such as the one in the Sifu platform, can adjust the hint level based on the timing profile, i.e., more vague hints in the beginning and more directed and solution-oriented towards the end.

Based on the experience of the first author, field experience gathered over the twelve CSC events, together with the presented results, we further propose that the following factors be considered by coaches to decide which team to aid:

- Consider long times to submit solutions, which can also indicate a team with a slow profile. In order to lower frustration, a CSC coach should understand to which teams to provide additional hints,
- Teams that have several challenges started at the same time (information available through the challenge heartbeat) should be considered candidates for coaching, together with
- coaching should also be considered for teams that have consumed all or most of the available hints for a given challenge.

The following shows a small list of key factors to CSC's success that coaches should foster, which are based on field experience, feedback from participants, and the results presented in this work.

- Open discussions: the teams that worked together in solving challenges reported on the effectiveness of the discussions with their colleagues as a positive experience during the CSC event;
- Do not give up on the exercise: teams that did not immediately switch to a different exercise and were persistent towards a solution reported increased understanding of the challenge and takeaways for their own self-improvement; CSC coaches can also foster this behavior;
- Ask for help from coaches: teams that actively asked for help from coaches when needed reported to have increased fun and understanding of the challenges; although they were also able to finish

the challenges faster, they also reported on the positive learning effect as expected; it is therefore recommended to announce and encourage the players at the beginning of the CSC event to ask for help from coaches when necessary proactively;

- Mixed experienced players: teams that mixed junior and senior players also reported on the benefits of the information exchange; we also observed that more experienced players took an active role as coaches themselves towards other players, increasing the learning effect.

Together, we conclude on the importance of properly designing the duration of the CSC event and the duration of individual challenges. We also conclude on the importance of the presented methods to improve the quality of coaching during a CSC event. The authors think that the information collected can be automatically processed and displayed to a panel that coaches can use during a CSC event to improve its effectiveness further.

Furthermore, the authors believe that the proposed methods and results can also be used as an indicator of the competence and skill level of CSC participants: e.g., if the player takes less time than the computed challenge solve time average, this indicates a highly skilled participant, and a higher challenge solve time than average indicates the opportunity for coaching.

3.1. Designing CSC Events—Practical Example

Based on the results presented in this work, a practitioner and designer of the CSC event that want to design an event that lasts 8 h (with 6 h CSC games) can use the following guidelines for the agenda: 1 h introduction, 7 C/C++ challenges, 21 questions, and 1 h for conclusions.

This example above is computed as follows (assuming a target of 75% solve rate, i.e., Quantile 75):

- $T^{75\%}_{C/C++} \approx 45$ min
- $T^{75\%}_{Questions} \approx 3$ min
- Total duration = $\sum n_j T^p_j$
- $Total^{75\%} = n_{C/C++} T^{75\%}_{C/C++} + n_{Questions} T^{75\%}_{Questions} +$
- $Total^{75\%} = 360 \text{ min} = 6 \text{ h} = 7 \times T^{75\%}_{C/C++} + 3 \times T^{75\%}_{Questions}$

The values presented were used in practical scenarios with success (see the forthcoming publication by Gasiba et al.). According to the first author's experience, longer duration CSC events, i.e., that last more than one working day, have been met with resistance from software developers and management. The main reason for this, as detailed in [18], is that it can be very costly for a company to allocate 30 software developers to "play games." As such, the recommended maximum duration is one working day (e.g., 8 h) with the number of challenges presented above.

Another vital aspect of computing the individual challenge solve times is that these values allow for a proper planning of CSCs, especially in terms of desired learning effects. Taking the example above, with 7 C/C++ challenges and 21 questions, the ratio of learning topics in questions to C/C++ is one to seven, i.e., 45 min spent on questions and 315 min spent on C/C++ challenges.

3.2. Limitations and Threats to Validity

This study presents an analysis of data from 12 CSC events between 2017 and 2020, with 190 players. These games have been played both on-site and remotely in different organizations in industrial software engineering. The number of events and the number of total players are reasonable for research work in the industry.

The CSC events were a part of different projects (including remuneration) for software development training. Although the participants took part in the mandatory training, participation in the current study, including data collection, was previously agreed upon. According to this scenario, it can be assumed that the players did their best at solving the challenges, and it can also be assumed

that data collection and the analysis results have a high validity. Limitations on the focus of games, the number of participants per event, and variations in the participants' background and experience are inherent to this kind of industrial setting and research. These variations are challenging to control in industrial settings. However, these variations represent the existing reality in the industry. In a different scenario, e.g., with students or voluntary participants acquired through social media, the results would have probably been different. However, we argue that these results' validity would be lower, especially for an industrial setting.

Further research is needed to clarify the measurements due to the high standard errors in the data. The presence of high errors is likely related to the previously discussed limitations on data collection (e.g., the inability to precisely determine the beginning and end times for solving challenges). However, the authors argue that the guiding principles presented are more critical than the absolute values obtained. However, the methods presented are useful in practical examples in the industry. Furthermore, given the limitations in how CSC challenges are mostly created (i.e., based on open-source projects), a considerable effort would need to be made to improve the data measurement. The hereby proposed data collection method and analysis are lightweight, respect the privacy of participants, monitor the games in real and deferred time, and aid in determining the learning outcomes.

The formula for the shape of the interaction curve (Figure 10), i.e., player profile, also needs a more in-depth analysis and an understanding of the theoretical foundation. Although the empirical model presented does fit the observed results well, further analysis is necessary to understand the theoretical basis.

In the present analysis, questions have been considered as one challenge category. However, further investigation is needed to separate generic questions and questions specific to previous challenges (i.e., that are unlocked by solving some challenge).

Finally, since in every game, individual players were part of a team, the authors analyzed the influence on ranking in terms of team performance and not individual performance. More research is needed to validate the data on individual performance.

4. Materials and Methods

The following table (Table 6) lists the open-source projects on which the current work is based.

Table 6. Open-source components used in this research.

Component	Ref.	Description
Dashboard	[51]	Dashboard that hosts the CSC challenges
JuiceShop	[84]	Open-source project on which the web application challenges are based
MBE	[84]	Open-source project on which the C/C++ challenges (non-Sifu) are based
Netresec	[85]	Public PCAP files on which the forensic challenges are based
Sifu	[20]	Description of the implementation reference of the Sifu platform with all the open-source components

We intend to release the Sifu platform as an open source project. At the time of writing, an internal software clearing process is being performed at the company at which the Sifu platform was originally developed. Raw collected results can be made available through contacting the first author. All data were pre-processed with Python [86] and post-processed with R-Studio [83].

5. Conclusions

In this work, cybersecurity challenges are presented as serious games that raises awareness about secure coding and are used to train industrial software developers in secure coding techniques. Due to industry requirements, the design of these events needs to take into consideration the

total event duration and the time it takes to solve individual challenges—the challenge solve time. Furthermore, the careful design of the desired learning goals and effects needs to be taken into consideration.

The current work proposes two different methods to measure the challenge solve time, which is based on timestamps generated and collected using standard existing mechanisms. We also introduce a new tool that can be used both by CSC designers and also by CSC coaches: the challenge failure probability. This tool is a graph that gives an indication of the probability that a given CSC challenge has been solved by a player, depending on the amount of time that has passed since the player has started to solve the challenge. This tool is not only useful to understand the challenge solve time duration, but it can also be used as a tool by CSC coaches to plan their intervention points for different team members. This is especially important to guarantee that the participants are kept actively interested in the game and do not lose concentration due to frustration. Furthermore, we introduce team profiles based on player interaction with the CSC dashboard and discuss practical implications for CSC coaches. Finally, we discuss the usefulness of challenge playing graphs in informing future CSC events.

To address practitioners, the challenge solve times are measured for different challenge types. The presented results provide a good guideline for CSC designers to plan a CSC event. These are based on the analysis of twelve events from CSC games that were played from 2017 to 2020 by more than 190 professional software developers from the industry. Our proposed measurement methods use only data from the game dashboard and from the Sifu platform and allow for an automated computation and estimation of the challenge solve time. The presented results from the analysis of the data are shown to have both implications for game design and for the individualization of cybersecurity challenges.

In the last three events, we collect data from two different sources, using two different challenge solve time computation methods. Using these data, we compare both results against each other and discus the advantages and limitations of each method. Our results show a good agreement between both measurement methods, and our field experience validates our results.

Furthermore, the authors identify, based on the analysis of the challenge solve time, the following profiles: automated, slow, and fast profiles. Our results indicate that the slow profile, with few interactions in the beginning, has advantages over the fast strategy.

The measurement methods hereby presented to analyze the games are pragmatic in nature: it takes only data from the dashboard, no personal data and no linkage to individual persons and to the learning outcome, as per the requirement in the industry. Using the collected data, the authors provide a method that allows for the planning of a CSC event together with the ratio of learning materials. In a provided computed example, which is comprised of an 8 h CSC event (where 6 h belong to active challenge solving), we show that 7 C/C++ challenges and 21 questions can be accommodated. This value has also been verified in practice and matches our experience in the field.

Furthermore, our analysis is useful for the game coaches: monitoring the dashboard allows coaches to provide targeted guidance with the goal to optimize the gaming experience and the learning outcome. Furthermore, we devise factors and recommendations that CSC coaches can take into consideration, so as to determine when to intervene and perform individual coaching of team members.

In further research, the authors plan to further improve the measurement time using the Sifu platform, addressing some of the limitations that have been observed in practice (e.g., that players have not closed their browser window or that flags have not been submitted in a timely manner to the dashboard). The authors would also like to explore the failure probability graph as a means to improve the hint system in the Sifu platform.

Author Contributions: T.E.G. did this work under the supervision of U.L. and M.P.-A. The first author carried out the main idea, implementation, and evaluation in an industrial context. All the authors actively contributed to steering the work and, in particular, to discussing adequate forms of evaluation of the platform, in terms of

surveys and their questions. All the authors made a significant contribution in the preparation, review, and writing of this manuscript. All authors read and agreed to the published version of the manuscript.

Funding: This work is co-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.

Acknowledgments: The authors would like to thank the survey participants for their useful and insightful discussions and their participation in the survey. The authors would like to thank Kristian Beckers and Thomas Diefenbach for helpful, insightful, and constructive comments, discussions, and suggestions. The authors would also like to thank Filip Rezabek for his help setting up the CTFd and for his work on the web application challenges during his working student position at Siemens AG.

Conflicts of Interest: The authors declare that they have no competing interests.

Abbreviations

The following abbreviations are used in this manuscript:

AUTOSAR	AUTomotive Open System ARchitecture
AVG	Average
BSI	Bundesamt für Sicherheit in der Informationstechnik
CERT	Computer Emergency Response Team
CH	China
CSC	Cybersecurity Challenges
CoT	Charter of Trust
CTF	Capture-The-Flag
CTFd	CTF dashboard
CWE	Common Weakness Enumeration
DB	Dashboard
DE	Germany
DHS	Department of Homeland Security
ICS	Industrial Control Systems
IEC	International Electrotechnical Commission
ISO	International Standard Organization
MBE	Modern Binary Exploitation
MISRA	Motor Industry Software Reliability Association
OL	Online
PCAP	Packet Capture
PCI/DSS	Payment Card Industry Data Security Standard
RQ	Research Question
SAFECode	Software Assurance Forum for Excellence in Code
SAST	Static Application Security Testing
SCG	Secure Coding Guideline
SEI-CERT	Software Engineering Institute-Computer Emergency Response Team
TK	Turkey

References

1. Department of Homeland Security. ICS-CERT: Industrial Control Systems—Computer Emergency Response Team. 2020. Available online: https://us-cert.cisa.gov/ics (accessed on 15 November 2020).
2. Department of Homeland Security, US-CERT. Software Assurance. Available online: https://tinyurl.com/y6pr9v42 (accessed on 30 September 2020).
3. Patel, S. 2019 Global Developer Report: DevSecOps Finds Security Roadblocks Divide Teams. Available online: https://about.gitlab.com/blog/2019/07/15/global-developer-report/ (accessed on 15 July 2019).
4. Schneier, B. Software Developers and Security. Available online: https://www.schneier.com/blog/archives/2019/07/software_develo.html (accessed on 21 July 2020).
5. ISO 27001. *Information Technology—Security Techniques-Information Security Management Systems-Requirements*; International Standard Organization: Geneva, Switzerland, 2013.

6. IEC. *Security for Industrial Automation and Control Systems—Part 4-1: Secure Product Development Lifecycle Requirements*; International Electrotechnical Commission: London, UK, 2018.
7. PCI DSS. Requirements and Security Assessment Procedures. 2008. Available online: https://www.pcisecuritystandards.org/ (accessed on 13 November 2020).
8. Bundesamt für Sicherheit in der Informationstechnik. BSI IT-Grundschutz-Kompendium. 2020. Available online: https://tinyurl.com/BSI-Grundschutz-Kompendium (accessed on 13 November 2020).
9. SAFECode Charter Members. SAFECode—Software Assurance Forum for Excellence in Code. Available online: https://safecode.org (accessed on 3 July 2020).
10. Goseva-Popstojanova, K.; Perhinschi, A. On the capability of static code analysis to detect security vulnerabilities. *Inf. Softw. Technol.* **2015**, *68*, 18–33.
11. Aloraini, B.; Nagappan, M.; German, D.M.; Hayashi, S.; Higo, Y. An empirical study of security warnings from static application security testing tools. *J. Syst. Softw.* **2019**, *158*, 110427.
12. Li, J. Vulnerabilities Mapping based on OWASP-SANS: A Survey for Static Application Security Testing (SAST). *Ann. Emerg. Technol. Comput.* **2020**, *4*, 1–8. [CrossRef]
13. ISO. *ISO 250xx Series*; International Organization for Standardization: Geneva, Switzerland, 2005.
14. Kapp, K.M. *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education*; John Wiley & Sons: San Francisco, CA, USA, 2012.
15. Leune, K.; Petrilli, S., Jr. Using Capture-the-Flag to Enhance the Effectiveness of Cybersecurity Education. In Proceedings of the 18th Annual Conference on Information Technology Education, New York, NY, USA, 4–7 October 2017; pp. 47–52.
16. Kucek, S.; Leitner, M. An Empirical Survey of Functions and Configurations of Open-Source Capture the Flag (CTF) Environments. *J. Netw. Comput. Appl.* **2020**, *151*, 102470.
17. Švábenskỳ, V.; Vykopal, J.; Cermak, M.; Laštovička, M. Enhancing cybersecurity skills by creating serious games. In Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, Larnaca, Cyprus, 2–4 July 2018; pp. 194–199.
18. Gasiba, T.; Beckers, K.; Suppan, S.; Rezabek, F. On the Requirements for Serious Games geared towards Software Developers in the Industry. In Proceedings of the 27th IEEE International Requirements Engineering Conference (RE 2019), Jeju Island, Korea, 23–27 September 2019; Damian, D.E., Perini, A., Lee, S., Eds.; IEEE: Piscataway, NJ, USA, 2019.
19. Nakamura, J.; Csikszentmihalyi, M. The Concept of Flow. In *Flow and the Foundations of Positive Psychology*; Springer: Dordrecht, The Netherlands, 2014; pp. 239–263.
20. Gasiba, T.; Lechner, U.; Pinto-Albuquerque, M.; Porwal, A. Cybersecurity Awareness Platform with Virtual Coach and Automated Challenge Assessment. In Proceedings of the 6th Workshop on The Security of Industrial Control Systems & of Cyber-Physical Systems, CyberICPS, Guildford, UK, 14–18 September 2020.
21. Gasiba, T.; Lechner, U.; Pinto-Albuquerque, M. Cybersecurity Challenges for Software Developer Awareness Training in Industrial Environments. 2021, under review.
22. Hänsch, N.; Benenson, Z. Specifying IT Security Awareness. In Proceedings of the 25th International Workshop on Database and Expert Systems Applications, Munich, Germany, 1–4 September 2014; pp. 326–330. [CrossRef]
23. Gasiba, T.; Lechner, U.; Cuellar, J.; Zouitni, A. Ranking Secure Coding Guidelines for Software Developer Awareness Training in the Industry. In Proceedings of the International Computer Programming Education Conference, ICPEC, Porto, Portugal, 23–24 April 2020.
24. McIlwraith, A. *Information Security and Employee Behaviour: How to Reduce Risk Through Employee Education, Training and Awareness*; Gower Publishing, Ltd.: Newcastle, UK, 2006.
25. Stewart, G.; Lacey, D. Death by a Thousand Facts: Criticising the Technocratic Approach to Information Security Awareness. *Inf. Manag. Comput. Secur.* **2012**, *20*, 29–38.
26. Dörner, R.; Göbel, S.; Effelsberg, W.; Wiemeyer, J. *Serious Games: Foundations, Concepts and Practice*, 1st ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016. [CrossRef]
27. Dörner, R.; Göbel, S.; Kickmeier-Rust, M.; Masuch, M.; Zweig, K. *Entertainment Computing and Serious Games: International GI-Dagstuhl Seminar 15283, Dagstuhl Castle, Germany, July 5–10, 2015, Revised Selected Papers*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9970.

28. Gasiba, T.; Lechner, U.; Pinto-Albuquerque, M.; Rezabek, F. Cybersecurity Games for Secure Programming Education in the Industry: Gameplay Analysis. In Proceedings of the International Computer Programming Education Conference, ICPEC, Porto, Portugal, 23–24 April 2020.
29. Rieb, A. IT-Sicherheit: Cyberabwehr mit hohem Spaßfaktor. *Kma Gesundheitswirtschaftsmagazin* **2018**, *23*, 66–69.
30. Rieb, A.; Gurschler, T.; Lechner, U. A Gamified Approach to Explore Techniques of Neutralization of Threat Actors in Cybercrime. In *GDPR & ePrivacy: APF 2017—Proceedings of the 5th ENISA Annual Privacy Forum*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2017; pp. 87–103.
31. Le Compte, A.; Elizondo, D.; Watson, T. A renewed approach to serious games for cyber security. In Proceedings of the 2015 7th International Conference on Cyber Conflict: Architectures in Cyberspace, Tallinn, Estonia, 25–29 May 2015; pp. 203–216.
32. Ávila-Pesántez, D.; Rivera, L.A.; Alban, M.S. Approaches for serious game design: A systematic literature review. *ASEE Comput. Educ. CoED J.* **2017**, *8*. Available online: https://www.asee.org/documents/papers-and-publications/papers/CoEd_Journal-2017/Jul-Sep/AVILA_PES%C3%81NTEZ.pdf (accessed on 13 November 2020).
33. Lameras, P.; Arnab, S.; Dunwell, I.; Stewart, C.; Clarke, S.; Petridis, P. Essential features of serious games design in higher education: Linking learning attributes to game mechanics. *Br. J. Educ. Technol.* **2017**, *48*, 972–994.
34. Cullinane, I.; Huang, C.; Sharkey, T.; Moussavi, S. Cyber Security Education Through Gaming Cybersecurity Games Can Be Interactive, Fun, Educational and Engaging. *J. Comput. Sci. Coll.* **2015**, *30*, 75–81.
35. Cairns, P. Engagement in Digital Games. In *Why Engagement Matters*; O'Brien, H., Cairns, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 81–104. [CrossRef]
36. Henrik, S.F. The Player Engagement Process—An Exploration of Continuation Desire in Digital Games. In Proceedings of the 2011 DiGRA International Conference: Think Design Play, Hilversum, The Netherlands, 14–17 September 2011.
37. Kim, S.; Song, K.; Lockee, B.; Burton, J. Engagement and Fun. In *Gamification in Learning and Education*, 1st ed.; Advances in Game-Based Learning; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 7–14. [CrossRef]
38. Mirkovic, J.; Peterson, P.A. Class capture-the-flag exercises. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14), Utrecht, The Netherlands, 14–17 September 2014.
39. Hendrix, M.; Al-Sherbaz, A.; Victoria, B. Game based cyber security training: Are serious games suitable for cyber security training? *Int. J. Serious Games* **2016**, *3*, 53–61.
40. Davis, A.; Leek, T.; Zhivich, M.; Gwinnup, K.; Leonard, W. The Fun and Future of CTF. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14), Utrecht, The Netherlands, 14–17 September 2014.
41. Alotaibi, F.; Furnell, S.; Stengel, I.; Papadaki, M. A review of using gaming technology for cyber-security awareness. *Int. J. Inf. Secur. Res. IJISR* **2016**, *6*, 660–666.
42. Cheung, R.S.; Cohen, J.P.; Lo, H.Z.; Elia, F.; Carrillo-Marquez, V. Effectiveness of cybersecurity competitions. In Proceedings of the International Conference on Security and Management (SAM), The Steering Committee of The World Congress in Computer Science, Computer, Las Vegas, NV, USA, 27–30 July 2012.
43. Chung, K.; Cohen, J. Learning obstacles in the capture the flag model. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14), Utrecht, The Netherlands, 14–17 September 2014.
44. SANS Institute. SEC642: Advanced Web App Penetration Testing, Ethical Hacking, and Exploitation Techniques. Available online: www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Nzundiz_2016_3_8.pdf (accessed on 13 November 2020).
45. Radermacher, A.; Walia, G. Gaps between industry expectations and the abilities of graduates. In Proceeding of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, USA, 6–9 March 2013; pp. 525–530.
46. Mäses, S. Evaluating Cybersecurity-Related Competences through Serious Games. In Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Joensuu, Finland, 21–24 November 2019.

47. Mäses, S.; Hallaq, B.; Maennel, O. Obtaining Better Metrics for Complex Serious Games Within Virtualised Simulation Environments. In Proceedings of the 11th European Conference on Game-Based Learning (ECGBL), Graz, Austria, 5–6 October 2017; pp. 428–434.
48. Andreolini, M.; Colacino, V.G.; Colajanni, M.; Marchetti, M. A Framework for the Evaluation of Trainee Performance in Cyber Range Exercises. *Mob. Netw. Appl.* **2019**, *25*, 236–247. [CrossRef]
49. Graziotin, D.; Fagerholm, F.; Wang, X.; Abrahamsson, P. What happens when software developers are (un)happy. *J. Syst. Softw.* **2018**, *140*, 32–47.
50. Owens, M. *The Definitive Guide to SQLite*; Apress: New York, NY, USA, 2006.
51. Chung, K. CTFd: The Easiest Capture The Flag Framework. Available online: https://ctfd.io/ (accessed on 13 November 2020).
52. Garrett, J.J. Ajax: A New Approach to Web Applications. 2005. Available online: https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf (accessed on 13 November 2020).
53. Westera, W.; Nadolski, R.; Hummel, H. Learning analytics in serious gaming: Uncovering the hidden treasury of game log files. In Proceedings of the International Conference on Games and Learning Alliance, Paris, France, 23–25 October 2013; pp. 41–52.
54. Bird, J.; Gornall, S. *The Art of Coaching: A Handbook of Tips and Tools*, 1st ed.; Routledge: New York, NY, USA, 2015.
55. IEEE Spectrum. The Top Programming Languages 2018. 2019. Available online: https://tinyurl.com/y75qj2ea (accessed on 13 November 2020).
56. WhiteSource. What Are the Most Secure Programming Languages? 2019. Available online: https://www.whitesourcesoftware.com/most-secure-programming-languages/ (accessed on 13 November 2020).
57. OWASP. OWASP Top 10 Vulnerabilities. Available online: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf (accessed on 17 June 2019).
58. Carnegie Mellon University. Secure Coding Standards. 2019. Available online: https://tinyurl.com/y29mwsyj (accessed on 13 November 2020).
59. Motor Industry Software Reliability Association. *Additional Security Guidelines for MISRA C: 2012*; MISRA: Nuneaton, UK, 2016.
60. AUtomotive Open System ARchitecture. Guidelines for the Use of the C++14 Language in Critical and Safety-Related Systems. 2017. Available online: https://www.autosar.org/ (accessed on 15 November 2020).
61. Gasiba, T.; Lechner, U.; Pinto-Albuquerque, M.; Zouitni, A. Design of Secure Coding Challenges for Cybersecurity Education in the Industry. In Proceedings of the 13th International Conference on the Quality of Information and Communications Technology, QUATIC, 8–11 September 2020; (online conference).
62. Barela, J.; Espinha Gasiba, T.; Reinhard Suppan, S.; Berges, M.; Beckers, K. When Interactive Graphic Storytelling Fails. In Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), Jeju Island, Korea, 23–27 September 2019; pp. 164–169.
63. MITRE. Common Weakness Enumeration. Available online: https://cwe.mitre.org/ (accessed on 4 February 2020).
64. MITRE. CWE 14: Compiler Removal of Code to Clear Buffers. Available online: https://cwe.mitre.org/data/definitions/14.html (accessed on 13 November 2020).
65. MITRE. CWE 77: Improper Neutralization of Special Elements Used in a Command ('Command Injection'). Available online: https://cwe.mitre.org/data/definitions/77.html (accessed on 13 November 2020).
66. MITRE. CWE-121: Stack-Based Buffer Overflow. Available online: https://cwe.mitre.org/data/definitions/121.html (accessed on 13 November 2020).
67. MITRE. CWE-127: Buffer Under-Read. Available online: https://cwe.mitre.org/data/definitions/127.html (accessed on 13 November 2020).
68. MITRE. CWE-134: Use of Externally-Controlled Format String. Available online: https://cwe.mitre.org/data/definitions/134.html (accessed on 13 November 2020).
69. MITRE. CWE-190: Integer Overflow or Wraparound. Available online: https://cwe.mitre.org/data/definitions/190.html (accessed on 13 November 2020).
70. MITRE. CWE-208: Observable Timing Discrepancy. Available online: https://cwe.mitre.org/data/definitions/208.html (accessed on 13 November 2020).
71. MITRE. CWE-242: Use of Inherently Dangerous Function. Available online: https://cwe.mitre.org/data/definitions/242.html (accessed on 13 November 2020).

72. MITRE. CWE 320: Key Management Errors. Available online: https://cwe.mitre.org/data/definitions/320.html (accessed on 13 November 2020).
73. MITRE. CWE 330: Use of Insufficiently Random Values. Available online: https://cwe.mitre.org/data/definitions/330.html (accessed on 13 November 2020).
74. MITRE. CWE 331: Insufficient Entropy. Available online: https://cwe.mitre.org/data/definitions/331.html (accessed on 13 November 2020).
75. MITRE. CWE 338: Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG). Available online: https://cwe.mitre.org/data/definitions/338.html (accessed on 13 November 2020).
76. MITRE. CWE 562: Return of Stack Variable Address. Available online: https://cwe.mitre.org/data/definitions/562.html (accessed on 13 November 2020).
77. MITRE. CWE 676: Use of Potentially Dangerous Function. Available online: https://cwe.mitre.org/data/definitions/676.html (accessed on 13 November 2020).
78. MITRE. CWE 682: Incorrect Calculation. Available online: https://cwe.mitre.org/data/definitions/682.html (accessed on 13 November 2020).
79. MITRE. CWE 758: Reliance on Undefined, Unspecified, or Implementation-Defined Behavior). Available online: https://cwe.mitre.org/data/definitions/758.html (accessed on 13 November 2020).
80. MITRE. CWE 768: Incorrect Short Circuit Evaluation). Available online: https://cwe.mitre.org/data/definitions/768.html (accessed on 13 November 2020).
81. MITRE. CWE 778: Insufficient Logging. Available online: https://cwe.mitre.org/data/definitions/778.html (accessed on 13 November 2020).
82. MITRE. CWE 783: Operator Precedence Logic Error. Available online: https://cwe.mitre.org/data/definitions/783.html (accessed on 13 November 2020).
83. RStudio PBC. RStudio | Open Source & Professional Software for Data Science Teams. Version 1.2.5001. Available online: https://rstudio.com/ (accessed on 16 October 2019).
84. OWASP. OWASP JuiceShop. Available online: https://www.owasp.org/index.php/OWASP_Juice_Shop_Project (accessed on 1 September 2017).
85. Netresec. Public PCAP Files for Download. Available online: https://www.netresec.com/?page=pcapfiles (accessed on 17 June 2019).
86. Van Rossum, G. Python Programming Language. Version 2.7.14. Available online: https://www.python.org/ (accessed on 13 November 2020).