INSTITUTO
UNIVERSITÁRIO
DE LISBOA

A Serious Game for teaching Java Cybersecurity in the Industry with an Intelligent Coach

Luís Afonso Maia Rosa Casqueiro

Master Degree in Computer Science and Business Management

Supervisor:
PhD Maria Cabral Diogo Pinto-Albuquerque, Assistant Professor,
ISCTE - Instituto Universitário de Lisboa, Portugal

Co-Supervisor:
Tiago José Espinha de Mendonça Gasiba, Cybersecurity Researcher,
Siemens AG, Munich, Germany

November 2021

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

A Serious Game for teaching Java Cybersecurity in the Industry with an Intelligent Coach

Luís Afonso Maia Rosa Casqueiro

Master Degree in Computer Science and Business Management

Supervisor:
PhD Maria Cabral Diogo Pinto-Albuquerque, Assistant Professor,
ISCTE - Instituto Universitário de Lisboa, Portugal

Co-Supervisor:
Tiago José Espinha de Mendonça Gasiba, Cybersecurity Researcher,
Siemens AG, Munich, Germany

November 2021

Acknowledgments

It is hard to put into words all the experiences I went through over the past months. The events and development detailed in this master thesis were much more than the information retained inside of it. As a normal project, the writing of this thesis had its ups and downs along the process. I am grateful for the people that helped me travel along this bumpy road.

In February of 2021, I went to Munich to work in Siemens and develop the research detailed in this thesis. I am extremely grateful to my supervisor Maria Pinto-Albuquerque for her constant help and detailed scrutiny that helped smooth the development and writing of this thesis. I would also like to thank my co-supervisor Tiago Gasiba. My time at Siemens was highly productive and working with him was an extraordinary experience. Not only was he extremely helpful but also pushed me into achieving new milestones. He was always ready to teach and offer assistance any time I encountered a bump in the road.

Without their assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. I would like to thank you very much for your support and understanding over these past months. Without you both, my experience would not have been the same and for that I am genuinely thankful.

Writing this dissertation required more than academic support, and I have many people to thank.

To my family for the constant support and kind words that helped me during these past months.

To my friends for the fun, actions and laughs that helped me find motivation to write this thesis.

Lastly, I take this opportunity to express gratitude to professor Ruben Pereira who was always available to solve and elucidate any potential problems that I faced.

Resumo

A cibersegurança tem vindo a ganhar mais importância nos últimos anos. Hoje em dia, continuamos a ver um aumento no número de vulnerabilidades conhecidas e ataques cibernéticos bem-sucedidos. Vários estudos mostram que uma das causas desses problemas é a falta de consciência dos programadores de software em termos de segurança. Ao não estarem cientes de como escrever código seguro, os programadores podem adicionar vulnerabilidades ao software sem saber. Este estudo foca-se em aumentar a conciencia dos programadores de software de Java, no que toca à segurança cibernética, através de uma abordagem baseada em jogos sérios. O nosso artefacto *Java Cybersecurity Challenges*, consiste em exercícios de programação que pretendem providenciar aos programadores de software com uma experiência prática sobre vulnerabilidades relacionadas à segurança da linguagem de programação Java. A solução desenvolvida inclui um treinador inteligente que visa ajudar os jogadores a compreender as vulnerabilidades e a resolver os exercicios. Esta pesquisa foi desenvolvida com base na metodologia Action Design Research. Esta metodologia permitiu-nos chegar a uma solução útil, para o problema encontrado, aplicando uma abordagem de desenvolvimento iterativa. Os nossos resultados mostram que o artefacto desenvolvido é um bom método para responder ao problema definido e foi aceite e incorporado num programa de treino da indústria. Este trabalho contribui para investigadores e praticantes através de uma descrição detalhada sobre a implementação de um processo de análise automática de código, bem como de feedback, para avaliar o nível de segurança dos Java Cybersecurity Challenges.

Keywords: Java, Consciencialização de Cibersegurança, Industria, Educação, Treino, Código Seguro, Treinador Inteligente

Abstract

Cybersecurity as been gaining more and more attention over the past years. Nowadays we continue to see a rise in the number of known vulnerabilities and successful cyber-attacks. Several studies show that one of the causes of these problems is the lack of awareness of software developers. If software developers are not aware of how to write secure code they can unknowingly add vulnerabilities to software. This research focuses on raising Java developers cybersecurity awareness by employing a serious game type of approach. Our artifact, the *Java Cybersecurity Challenges*, consist of programming exercises that intend to give software developers hands-on experience with security related vulnerabilities in the Java programming language. Our designed solution includes an intelligent coach that aims at helping players understand the vulnerabilities and solve the challenges. The present research was conducted using the Action Design Research methodology. This methodology allowed us to reach a useful solution, to the encountered problem, by applying an iterative development approach. Our results show that the developed final artifact is a good method to answer the defined problem and has been accepted and incorporated in an industry training program. This work contributes to researchers and practitioners through a detailed description on the implementation of an automatic code analysis and feedback process to evaluate the security level of the Java Cybersecurity Challenges.

Palavras-Chave: Java, Cybersecurity Awareness, Industry, Education, Training, Secure Coding, Intelligent Coach

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The last couple of decades have seen a huge growth in the usage of software-related services. Nowadays this technology component is embedded in almost everything we interact with in our daily routine. However, with the exponential increase of digitization and overall user acceptance of these technologies, there has also been a drastic increase in cyber-attacks aimed at both the public and private sectors. Now, more than ever, the software industry is suffering from these cybersecurity issues that can compromise entire critical infrastructures [1].

A 2011 study conducted by Dimensional Research with 853 information technology (IT) experts from all around the world, showed that about 48% of big companies and 32% of small ones suffered from 25 or more attacks between 2009 and 2011, with very costly consequences [2]. According to PurpleSec [3], the reported malware-related incidents have been steadily increasing for the past decade, from an already high 12.4 million successful attacks in 2009 to an astonishing 812.67 million in 2018. Furthermore, the Department of Homeland Security [4, 5], depicts 2020 has one of the most critical years in cybersecurity related issues.

These reports highlight a drastic increase in confirmed cyber-attacks to companies and other critical infrastructures. This can be explained by the current unprecedented worldwide situation on account of the pandemic crisis caused by the COVID-19 virus [6]. Since people are forced to work from home, critical data that once was only accessed in secure environments is now being transmitted through a multitude of networks that do not possess the same level of security provided by an office network. This gives malicious users a potential easier access to critical information.

There are multiple recent examples of successful cyber-attacks. In 2020 multiple U.S agencies were targeted and compromised due to the successful cyber-attack on the software provider Solarwinds, that, when hacked, gave the malicious users access to their entire network [7]. Although this particular situation was due to a supply chain attack, a significant number of other cyber-attacks occurs due to one main cause: poor code quality. In fact, the U.S Department of Homeland Security [8], states that the root cause of about 90% of cyber related incidents is poor software quality. This issue is related to vulnerabilities in the source code of applications that a malicious user can exploit to his advantage.

Throughout the digital history of the world, we have watched the damages that situations like those can have. For example, malicous software like the WannaCry ransomware, released in 2017, had a estimated cost of four billion US. dollars and affected several industry control systems by exploiting a vulnerability known since the 70's (buffer overflow) [9]. However, the damages done by these breaches go beyond a monetary value. Successful attacks often create mistrust among the users and can lead to the destruction of a company's reputation [10]. But why do software developers keep creating bad quality software that leads to these kinds of problems? The root of this issue extends to the cybersecurity awareness side of software development and the knowledge of how to develop secure software. If a developer does not know how to produce secure code, these problems are overlooked and the software remains ex-

ploitable. Furthermore, by not being aware of how to produce secure code, software developers can unknowingly add vulnerabilities when developing software.

According to Patel et al. [11], less than 50% of software developers can identify a code vulnerability in the source code. This leads to one obvious yet frightening conclusion, software developers lack the awareness necessary to develop secure source code when developing software [12].

Nowadays, companies tend to have higher standards when it comes to software quality and, by default, security. Furthermore, the relationship between software quality and software security is many times mixed. A good quality software has the security aspect already embedded within. However, this inherent correlation is not applicable the other way around. A software can be secure and of poor quality. To deal with this situation, several standards like the *ISO 25000 Standards* [13] provide frameworks to evaluate the quality (including security) of any given software.

To meet these quality criteria, in particular cybersecurity, companies are now engaging in several initiatives that aim at mitigating security problems. Initiatives like the Charter of Trust [14], intend to make companies address major issues linked to cybersecurity. Within companies themselves, a number of tools and methods have been developed and employed to try and mitigate potential situations related to code security problems. These tools and methods range from static code analysis to code reviews and threat and risk analysis, among others. One of the methods to mitigate potential security issues is by following secure coding guidelines. Guidelines such as the SEI-CERT Coding standards, defined by the Carnegie Mellon University Software Engineering Institute [15], provide software developers with in-depth explanations and examples on how to produce more secure code. These guidelines, when correctly applied, give source code another level of robustness when dealing with cyber-threats. Although this type of guidelines and standards are available for public use, it has been found that most software developers do not follow these recommendations [16]. One of the existing ways to deal with this situation, which is the focus of the present work, is the awareness training of software developers when it comes to these issues. When in contact with these type of security issues in a safe environment, software developers learn to mitigate them, ultimately increasing their security awareness and safe code development.

Towards this goal, Siemens, developed a platform, the Sifu-Platform, that combines several fields of study to create a way to raise cybersecurity awareness in software developers [16]. This platform is inspired on the Capture-the-Flag game genre, where participants compete against each other, and for each correct answer they receive points in the form of a flag. The platform also employs analysis tools and contains an intelligent coach that aids the participants to solve the challenges. Ultimately, the Sifu-Platform intends to educate software developers on how to create safer code by giving them a game type of experience where they can safely deal with security issues. Therefore, this project is based on a serious game type of approach.

Unlike mainstream games, where the sole purpose is to provide the player with some type of entertainment, serious games intend to transmit knowledge and teach the player about a certain topic ([17, 18]). The goal of the Sifu-Platform is to raise awareness of software developers by extending their knowledge on secure coding guidelines and how to address them.

The proposed work extends previous work by Gasiba et al. [19] through a study on how to develop Java challenges for the Sifu-Platform. Java [20] is not only widely used in the industry and taught at universities, but it is also in the top-3 of the programming languages that have the most known vulnerabilities [21]. We tackle this problem by developing programmable challenges to give participants hands-on experience on dealing with security vulnerabilities.

These new exercises are focused on the Java programming language and offer the player an interactable way of learning secure coding. This type of exercises, or challenges, present the player a piece of vulnerable code with one or more security issues. The player must find the security vulnerabilities and re-write the code in order to secure it against possible attacks.

We decided to base our challenges on the SEI-CERT rules for secure Java coding [15]. Each one of these rules is linked to at least one security vulnerability. In addition, the solutions to these challenges are also in line with the good practices and solutions provided by these coding standards. Once the player solves the challenge, by modifying the code and eliminating the vulnerability, a flag is given as a reward that can be converted into points.

To achieve this goal of developing Java programmable challenges, we defined one major research question which constitutes the focus of the present work:

RQ How to adapt CyberSecurity Challenges to create a useful serious game to raise Java cybersecurity awareness of industrial software developers?

We used an Action Design Research (ADR) [22] approach, guided by our research question, to reach a suitable solution to this problem. In each conducted ADR cycle, we defined more specific, cycle intrinsic research questions that tackle a specific angle of the though-out artifact. By providing a suitable solution to these more concrete problems we can develop a good solution to the initially defined research question. In short, each development cycle of the research intends to improve a specific aspect of the artifact. The issues we want to solve through this iterative ADR research are defined in the early stages of each corresponding cycle of development.

We started our research by conducting a literature review to help us answer the general research question. This was fundamental to both understand the theoretical background and the previous work preformed in the field of study. The developed artifact was initially based on this literature review, and improved through the experience gathered in the life-cycle of this research. We evaluated the proposed solution through the feedback we gathered from the interventions held in an industry setting.

The developed artifact implemented a multi-step framework on which the Java challenges are based on. This framework allows for the backend analysis of the players submitted code. This analysis scans the players code for security issues, and employs unit-tests and other testing methods to guarantee the intended functionality of the code. The tests performed by the backend derive from the final subset of tools selected. The platform also employs a hint-system to aid the players to solve the challenges. In fact, we extended previous work by Gasiba et al. [19] through the development and adaptation of the hint-system (intelligent coach) algorithm to the Java programming language. Like previous work, the intelligent coach algorithm presented in this work is also based on the laddering technique. Although this technique was not originally developed to answer code related problems, we believe that an approach based on this concept is a good and straightforward way to provide meaningful feedback to the participants. More details about this technique can be found in [16, 23].

Our research was conducted following the Action Design Research (ADR) methodology as introduced by Sein et al. [22]. This methodology is based on a cyclic approach and gives opportunity for the end-users to express their opinion on the artifact whilst still in development. This approach allows for continued testing and improvement of the artifact leading to a more complete and user-oriented final product. The Action Design Research methodology will be presented in detail in Chapter 2.

Figure 1.1 shows an overview of the timeline of the present research. Between September 2020 and January 2021 we pinpointed the initial scope of the project and performed a lightweight literature review on the topic at hand. During the development of the research more scientific work on the subject was published. Due to this, the related work review linked to this research, accompanied the development of the artifact. After the initial scope definition we started the development of the artifact itself.

Following the ADR methodology the artifacts' development underwent four distinct development and testing cycles. These cycles can be categorized by the development performed in

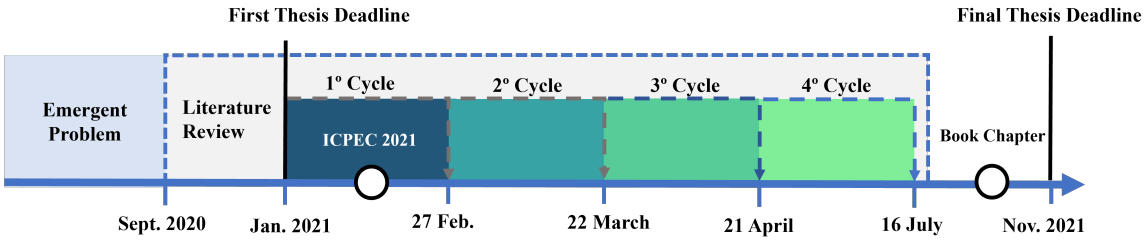


Figure 1.1: Research Timeline

each one. The first cycle encompassed an initial design of the artifact and serves as a proof-of-concept of the work as a whole. The development and testing performed in this cycle was presented in the form of a paper [24] in the second International Computer Programming Education Conference - ICPEC 2021 [25], which took place on the 28th of May through an online environment. The second cycle, built on the results of the previous one, had the main goal of developing more challenges and refining the backend analysis process. The third cycle of development, focused on improving the Hints and Hint-System given to the players and further validate the previously obtained results. The forth and final cycle of development focused on improving the hints and hints-system by addressing the feedback provided by players in the previous cycle.

Each version of the artifact suffered an evaluation phase. Several industry-based workshops were used to evaluate each version of the developed artifact. These workshops were conducted thought virtual meetings from late February to mid July of 2021. These workshops were mainly focused on an industry setting. However, a couple of evaluations with participants from the academia were also conducted. From each workshop the participants had the opportunity to give feedback that was latter used to improve the artifact in an iterative way. These results are presented in further sections of this work and provide an insight on how participants perceived the platform.

The artifact underwent an evaluation stage in each cycle of development. A total of 117 participants tested the artifact during the research life-cycle. Of this total amount, 99 where from an industry setting and the remainder 18 from an academia context. Furthermore, of the 117 participants that tested the artifact, 44 chose to answer the given questionnaire, and give us feedback about the artifact. The participants were informed and chose to participate in the present study of free will. The total amount of answers gathered were obtained in the four conducted development cycles of this research.

The main contribution of this work is to provide a working approach for practitioners and researchers who intend to develop Java related programming exercises with automatic hint generation with the end goal of raising awareness about Java secure coding. This was accomplished by adapting a previously conducted approach, used in C++, and applying it to the Java programming language. Furthermore, this work also presents a subset of tools that can be used to help determining the level of security of Java code and a complete framework on how to implement the challenges. Our approach is validated through the analysis of the results and feedback gathered by the questionnaires and overall interventions in the industry.

The present work was performed in a master's degree setting in Iscte - Instituto Universitário de Lisboa, Portugal in collaboration with Siemens AG and the Universität der Bundeswehr München (UniBW), the latter two based in Munich, Germany. The development and data presented in this research was submitted for a chapter publication in a book entitled "Handbook of Research on Gamification Dynamics and User Experience Design".

This thesis is organized as follows: Chapter 2 provides an analysis on the Action Design Research methodology used in the present work. Chapter 3 gives a theoretical background on

some key aspects and discusses relevant related work. Chapter 4 provides a description on the final version of the developed artifact. Chapter 5 describes the several development cycles inherent to this research. Chapter 6 provides an overview of the research and a comparison of the results between the different cycles, and Chapter 7 summarizes and concludes this work.

Chapter 2

Methodology

This section describes the methodology on which the development of this work was based. A methodology can be defined as the application of practices and procedures to a specific branch of knowledge [26]. It serves has a guide for researchers to base their work on. By using an already defined methodology the researchers can focus on applying its framework instead of creating the guidelines of the work from scratch. Furthermore, an already established and validated methodology comes with an intrinsic property of allowing for the creation of knowledge when correctly applied. The main goal of this work was to create a working artifact that would both answer the research question previously defined and be useful for future use in an industry setting. For this purpose we chose a development approach based on the Action Design Research methodology (hereafter ADR), as defined by Sein et al. [22]. This methodology aims at addressing a problem or situation encountered in an organizational context by creating and evaluating an artifact capable of solving the issue. The ADR methodology intends to decrease the gap between end-users and the researchers solution. This is achieved by employing a cyclic approach of testing and improvement where end-users can give feedback on what aspects need to be improved. This link between development and testing leads to a more complete and robust final artifact that meets the end-user's and company demands. The present work arises from an industry-based context and requires a development process capable of providing a suitable solution. Due to this we believe the ADR method to be an adequate approach to our research. The ADR methodology allows for a continuous improvement of the initial artifact throughout its effective development life cycle. The result is a solution that is both based on the first intent of the researchers and molded by the organizational setting.

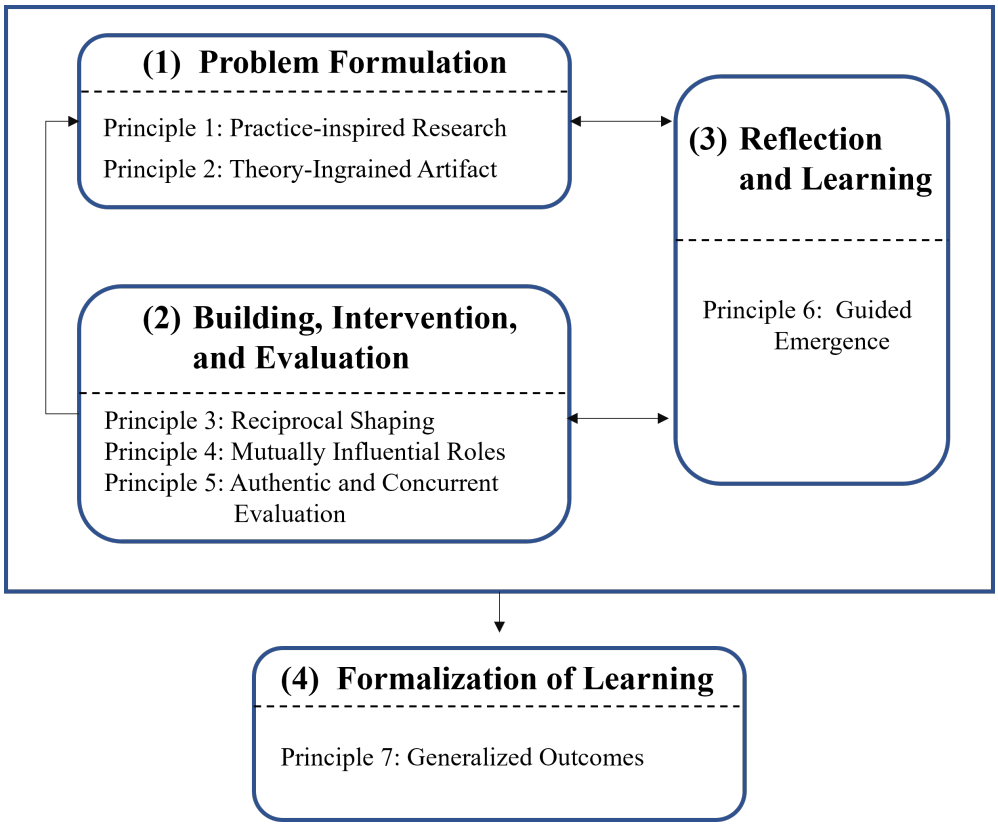


Figure 2.1: Action Design Research Methodology

The ADR methodology has four main stages, (1) Problem Formulation, (2) Building, Intervention, and Evaluation, (3) Reflection and Learning and (4) Formalization of Learning. Each of the four stages has a set of principles linked to itself. Each stage also has an inherent relationship with each of the other stages. Figure 2.1 shows the relationships between the different stages of the ADR methodology and the principles inherent to each one.

In the following subsections the Action Design Research methodology stages will be presented.

2.1 Problem Formulation

This stage serves as the initial motivation for the development cycle. It is in this stage that a problem encountered or anticipated by the researchers is defined. The identification and formulation of the problem serves as foundation to a research opportunity. By pin-pointing a research opportunity the researchers can start to define the initial scope of the work. After determining a first concept of the work, a set of initial research questions are formulated. Finding the answer to these research questions will lead to the creation of knowledge. The design artifact, guidelines and lessons learned will be the product of the designing stages undergone later in the process. After defining the research questions to be addressed the researchers must identify the work related to the topic. This step includes understanding the theoretical bases on which the research is based upon and what other scientific methods have already been carried out to deal with similar problems. The problem formulation stage is not immutable and its contents can suffer modifications throughout the research life cycle. The initially defined research questions and the related work can suffer adjustments over time. The adjustments made to these components serve as a means to improve the quality of the research and the usefulness of the artifact.

2.2 Building, Intervention and Evaluation

This stage is where the development, test and improvement of the artifact occurs. Firstly, by using the work performed in the previous stage, the researchers can now start to determine the initial scope of development. Based on the theoretical background and related work described in stage one, the development of the initial artifact starts. The artifact will undergo changes throughout the development life cycle. This is due to the cyclic nature of the ADR methodology that brings the end-user feedback and concerns into the development of the artifact itself. This is possible by gathering the development of each version of the artifact, its testing and evaluation into one cycle. In the development phase of each cycle the artifact is improved based upon both the goals of the researcher as well as the feedback and input given by the participants of the testing phase of the previous cycle (if exists). In each development cycle, an intervention occurs in order to give end-users an opportunity to test the current version of the artifact. After the intervention the participants are free to give feedback and evaluate the artifact, and it's this input that is later used as a basis of improvement for the artifact in the next cycle. Each cycle builds upon the prior based on an iterative process in order to obtain a more complete final artifact. Once the artifact shows itself useful in answering the emergent problem defined, then the research has reached its final cycle and the development of the artifact is done.

2.3 Reflection and Learning

This stage focuses on the broader aspect of the project. Here a reflection about the design, and all the redesigns the artifact suffered, is made. However, this stage goes beyond the simple solution of the problem. In this stage the researchers must perform a conscious reflection about the theoretical and problem framing side of the research. This is necessary in order to correctly identify the contributions to knowledge.

2.4 Formalization of Learning

This final stage is where the main conclusions of the undergone project are stated. Firstly the focused knowledge of the particular problem suffers an abstraction step to create concepts that can be applied to a broader class of problems. Secondly, the general outcomes from both the design of the artifact as well as its assessment and employment are shared. Finally, a formalization of the results found occurs in order to facilitate the dissemination of the created knowledge. In the present research, this stage relates to the final version of the master's thesis.

2.5 Action Design Research Principles

The ADR methodology has a set of principles that are applied to any research that is conducted following this framework. In total, there are seven distinct principles inherent to any ADR methodology-based work. These principles are ingrained in the stages of the ADR methodology and intend to justify the artifact design options, and to express the underlying values, assumptions and beliefs. From the seven principles that can be applied to an ADR research, we choose to highlight the three that contain more relevancy to the present work.

The first is the *Practice-Inspired Research*. This principle addresses the basis on which the research is conducted, and the artifact is improved. The data used to continuously improve the

artifact in the iterative process of the ADR is collected through practical interventions. This guarantees the relevancy of the gathered data and that the research conducted is inspired in the results from practical application of the concepts and testing of the artifact in an industry setting.

The second principle we want to emphasize is the *Theory-Ingrained Artifact*. This principle states that the development of the artifact is based on existing theories and previously conducted research. Moreover, the final artifact will apply and reflect these theories, leading to a more complete and legitimate research. In the present context, the developed artifact emerges from an already extensively researched and scrutinized platform. Additionally, a literature review was performed in order to understand important concepts about the topic at hand, and understand previous similar work.

The last ADR principle applied to our research is the *Guided Emergence*. This principle draws upon one of the most critical aspects of the current methodology. It states that the final artifact is not only a product of the initial design done by the researchers, which was heavily based on theory, but also all the changes applied due to the continuous testing and refinement along the projects' life cycle. Likewise, the final artifact developed in the present research is not only based on the initial design, used to start the first cycle of development and testing, but also all the changes that were introduced after the iterative process of testing the artifact in an industry context. This principle allows for researchers to introduce and adjust design concepts of the artifact along the research development cycle.

2.6 Action Design Research Overview & Application

Applying the ADR methodology to a research gives the researchers a clear, straightforward method to solve an organizational problem. Furthermore, this methodology allows researchers to create a solution to an emerging problem through iterative development stages. In the end, the research will have both an artifact that represents the work performed in the research, as well as lessons learned that arose throughout the development of said artifact. This simple approach on how to view this methodology is represented in figure 2.2.

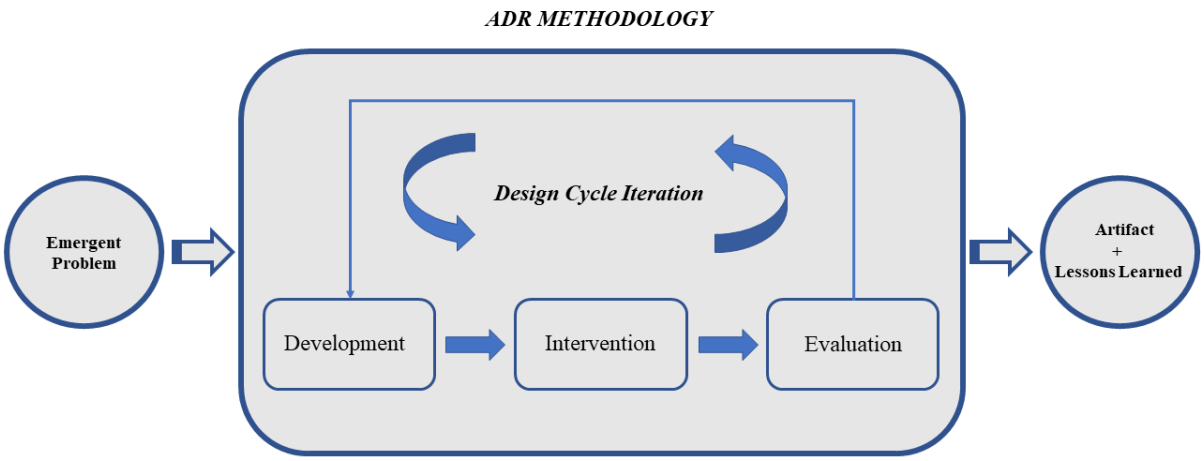


Figure 2.2: Applying the ADR

The present work had a total of four distinct ADR cycles. Cycle one, developed and tested an initial designed artifact to obtain a proof-of-concept on which the research could be based upon. After this initial cycle of development was finished, the researchers defined new goals and improvements that could be applied to the artifact.

Cycle two focused on refining the initial solution encompassing the feedback given by the first testing stage. Due to the inherent iterative process of the ADR methodology, the artifact tested in this design cycle was already impacted by the feedback of the users of the previous cycle.

The third cycle of development tackled the feedback gathered from the participants of cycle two, and hint generation process by the intelligent coach.

Reaching the final cycle of development and testing (cycle four), we focused on performing some final improvements to the artifact and testing it to guarantee the artifacts' usefulness. Due to the overwhelmingly positive nature of the gathered feedback in this cycle, we conclude that the artifact has reached a good maturity level, and is useful for the company in question. Along this four cycle development approach the artifact was improved to not only answer the problems perceived by researchers but also reflect and solve issues raised by industry participants.

Implementing the ADR methodology comes with its own set of challenges. This approach is inherently difficult to fully apply in practice as the research must be done very rigorously by combining several cycles of development and improvement of the artifact. Furthermore, this methodology is mostly suited for a dissertation that can take some years to conclude. However, the present work benefits from previous research, which was carried out using the ADR methodology in a successful manner. In short, this ADR based research is embedded in a broader ADR based work and helps build upon the results gathered. Furthermore, the present work extends previous research by covering a new aspect of the Sifu-Platform through the development of Java Challenges, and through the refinement of the hint generation process.

The results of the present research complement the already validated parts of the previous work and help build upon the overall acceptance of using serious games to raise awareness on cybersecurity issues.

In the present work, we show that our artifact has reached maturity and acceptance by both the software developers and the company. As such, the ADR iterative cyclic approach used in this research was concluded with success.

Chapter 3

Related Work

This section is focused on understanding the state-of-the-art related to the topic at hand. In order to understand the full scope of the work we divided this section in three distinct stages. We begin by identifying theories that serve as the basis and guiding principles to the development of the artifact. We then proceed to perform a lightweight literature review to understand previous work that has been carried out in the present field of study. This approach is a simplified version of the Systematic Literature Review approach, as defined by Kitchenham et al. [27]. After performing these two stages, we gathered information and case studies that served has the initial starting point to the development of the artifact. In addition to this initially collected data, we also performed a continuous search during the different ADR cycles, as defined by its methodology.

This three stage process provides us with a full scope of the theories and other applications in order to create a useful solution to the problem at hand.

In the following sections we will present the theoretical background, describe the literature review protocol defined, and the results gathered trough it's application.

3.1 Theoretical Background

Due to the complex nature of the topic of this research, we can classify the underlying issue as a *Wicked Problem*. Catrien et al. [28] defines this type of problems as *"having no clear solution, and perhaps not even a set of possible solutions"* and *"solutions to wicked problems are not true or false, but good or bad"*. In short, a wicked problem relates to a unique problem that does not have one clear, good solution but a set of possible approaches to dealing with the encountered issue. This research focuses on creating a useful artifact for raising cybersecurity awareness of Java software developers. This problem falls in the category of wicked problems, as no clear single best-solution exists that addresses this particular situation. Only a set of good approaches can be defined to try and mitigate the issue.

To fully understand the dimensions of the work a mind map was created to help focus the main ideas and topics of interest. This allowed us to determine the different relevant areas to the present work. Figure 3.1 represents the main areas of interest. In the center is the core idea of the work. This main objective arises from five distinct concepts: Awareness, Code Analysis, Serious Games, Artificial Intelligence and Secure Coding Guidelines.

These five topics are the main concepts that guided the design of our proposed solution. For each of these five concepts, we have selected representative theories and ideas based on previous work. The following sub-sections present an explanation as well as some examples on each of these chosen topics.

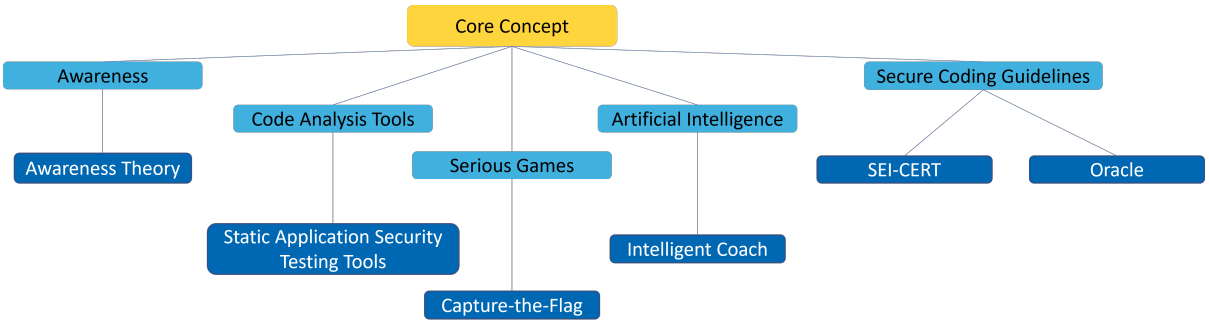


Figure 3.1: Research's Topics of Interest

3.1.1 Awareness

The present research has the main goal of developing a good way to raise cybersecurity awareness. On the awareness topic, we can highlight one major theory. The *IT Security Awareness Theory*. While originally, this theory refers to the IT security awareness as defined by Benenson et al. [29], it was later extended by Gasiba et al. [30], to the field of software development. This theory states that the awareness is composed by three aspects: *Perception*, *Protection* and *Behavior*. The perception aspect in the extension to IT security awareness relates to developers being able to perceive potential threats in software. The second aspect refers to the protection aspect of software coding: developers should have the knowledge, measures and experience needed to mitigate potential threats and protect themselves and their organizations. The last aspect relates to the behavioral aspect of writing software: theoretical knowledge is not enough to reduce security problems, and a change of bad coding habits is needed to produce safe software. This means that the developers should understand what type of programs or strategies can be used to mitigate the issues and how to correctly employ them. These three aspects of security awareness have been shown to serve as the basis to create awareness campaigns targeted at software developers.

3.1.2 Code Analysis Tools

One method to mitigate security vulnerabilities that arose in popularity is the use of code analysis tools. One type in particular is the Static Application Security Testing tools, SAST tools. This type of tools give software developers a fast and automated way to analyse and report potential cybersecurity flaws that might be present in the source code [31, 32]. Not only do SAST tools report the encountered coding issues, but can also provide software developers solutions to the vulnerabilities found through the analysis of source code [33]. When applied correctly to the production of software, SAST tools present a critical ally, providing a powerful insight to developers. However, although they represent a fast and low-cost way to analyse the source code, SAST tools aren't fully reliable [32]. In fact, there are no SAST tools with full detection capabilities (can't find all the possible errors). Furthermore, these tools are not always correct and so can generate a considerable amount of false positives and false negatives when analysing code. This situation creates more work for developers who need to understand if the reports actually represent a vulnerability that needs to be addressed. In fact, trying to discern whether a report is correct or not implies a certain degree of cognitive effort by the developers, that can lead to fatigue and exhaustion. These two unwanted consequences can influence the overall practical experience and execution within companies themselves. Therefore, these tools alone are not sufficient and must be combined with other mitigation techniques (e.g. code reviews) in order to effectively increase software security [32, 33].

3.1.3 Serious Games

Serious games have brought a new approach to teaching all kinds of subjects. Whereas typical games are mainly focused in the entertainment side of the experience, serious games tend to have a more educational and training nature, other than the entertainment feature [17]. These coaching instruments represent a powerful tool in knowledge acquisition since, not only do participants get to be in contact with new information, but also experiment and learn from their own mistakes, all this in a protected and secure environment [34]. As shown by Ros et al. [35] overall learning theories suggest that the amount of information retained, when learning a subject, is directly linked to the learning process itself. Furthermore, the authors claim that more dynamic and active learning processes leads to an overall better knowledge retention. Using a serious game type of approach seems an appropriate method to meet these goals to create a useful learning tool. Being highly stimulating and encouraging to the players, serious games meets all the criteria to solve this issue [17]. However, according to several papers [10, 17, 35, 36], the game itself must be adapted to the audience in question. Both the content of the game and the difficulty must be targeted to the intended level of the participants. If a serious game takes all of this into consideration it can lead to very good results across several fields of study. This can possibly explain the recent growth in interest that both the academic and industrial communities are showing towards these types of learning tools [37]. There are several types of serious games. In the cybersecurity field a well known game is the Capture the Flag, or CTF, events where the user (or team) must exploit security vulnerabilities, in order to get flags that translate into points. These events can be of the following styles:

- *Jeopardy-Style:* In these CTF events the team of hackers exploit the security vulnerabilities of the given system, with the intention of gaining access to any potential information required to get the flags. These events don't usually require between-teams attacks, since they are standalone challenges. In short, all the teams competing are against the "same" system [37].

- *Attack-and-Defense:* In these CTF events, a team protect a vulnerable system containing important data, while attacking other teams systems [37]. Successful attacks to other teams system results in collecting flags.

- *Defense-Only:* This type of CTF events consist of giving players an insecure system that they must defend. Each time the team mitigates one of the vulnerabilities they receive a flag [36].

3.1.4 Artificial Intelligence

Artificial intelligence (AI) is currently a very active field of study. Being capable of automating and solving the most complex processes in a way beyond human reach, this technology is considered by many has a general solution technology, and it's unlikely that it will ever go away [38]. AI can be applied in many different ways. One in particular is gaining traction in the health and education fields: intelligent coaches. This kind of AI aims at improving certain activities, giving the users a helping hand whenever necessary. This helping interaction between the player and the coach seeks to ultimately decrease the fatigue, boost the interest and motivation towards a specific goal [39]. In [40], Tironi et al. presents a case study linked

to the health industry. In this work, a comparison between a competitive virtual coach and a cooperative one is performed. The results show that users have a preference when the coach is actively helping them reach their goal (cooperative scenario), rather than otherwise (competitive scenario). This implies that when developing an intelligent coach is preferable to create one with the capability to understand the users errors and give feedback to what can be improved. In [41], Rodrigues et al. present the challenges of distance learning and possible tools to improve the motivation and interest of students. One of the explored methods was the use of an experimental coach that aided the students in the context of online learning, which attained great results. In [16], Gasiba et al. discuss the use of an AI chatbot-like system that functions like an intelligent coach in helping players solve problems. This latter approach also gathered positive results and serve as the basis for our own development and improvement of the intelligent coach used in this work.

3.1.5 Secure Coding Guidelines

There seems to be an agreement amongst software developers and designers that achieving a fully secure software is impossible. From the complexity of overall systems to the different design of their components, reaching the status 100% security is out of reach [42]. However, best coding practices and guidelines are available in order to promote good code quality amongst software developers in the industry [43]. If not identified and solved in early stages of implementation, security flaws can reach the final product, leaving the company and the client at the mercy of hackers. Since it's better to implement secure code from start than to leave its improvement to the threat analysis team, awareness regarding secure code implementation must be raised. By increasing the security awareness of software developers, companies can save precious time otherwise used to fix implementation and security errors. Furthermore, by possessing a better understanding and knowledge of the security aspect of coding, developers can create better and safer software, ultimately protecting the people that work with it [2]. Accompanying the rise in cyber-attacks the amount of laws and legislations regarding software security guidelines and best practices has also seen a rapid increase. Some organizations in charge of the creation and management of this guidelines have made an effort to increase security regulatory efforts in a global manner [44]. Guidelines such as the OWASP Top 10 [45], and the SEI-CERT secure coding guidelines [15], developed by the CERT Coordination Center, give a thorough list of vulnerability examples and what software developers can do in order to fix vulnerable code. Initiatives like the Charter of Trust drive the industry towards greater digitalization through the establishment of rules and standards created to increase cybersecurity trust [14].

3.2 Literature Review Protocol

Understanding the state-of-the-art around a research topic is essential to develop a more complete and legitimate work. In this research we used a simplistic approach to the Systematic Literature Review method defined by Kitchenham et al. [27], which we designated by lightweight literature review. This approach differs from the original method in terms of the amount of time and articles used to understand the state-of-the-art on the topic. Due to the scope of this research, we believe it to be an appropriate method to understand the topic at hand. To tackle this lightweight literature review, a review protocol was created. A total of two databases were used in our search. To determine the best keywords we created a quadrant map, figure 3.2. This element was based on the previously presented mind-map, to provide a visual aid for keyword determination. From the six keywords present in figure 3.2, only four provided relevant results

to the state-of-the-art for this work. After obtaining the first list of papers, a set of filters were applied in order to narrow down the important content. The following subsections present this information in detail.

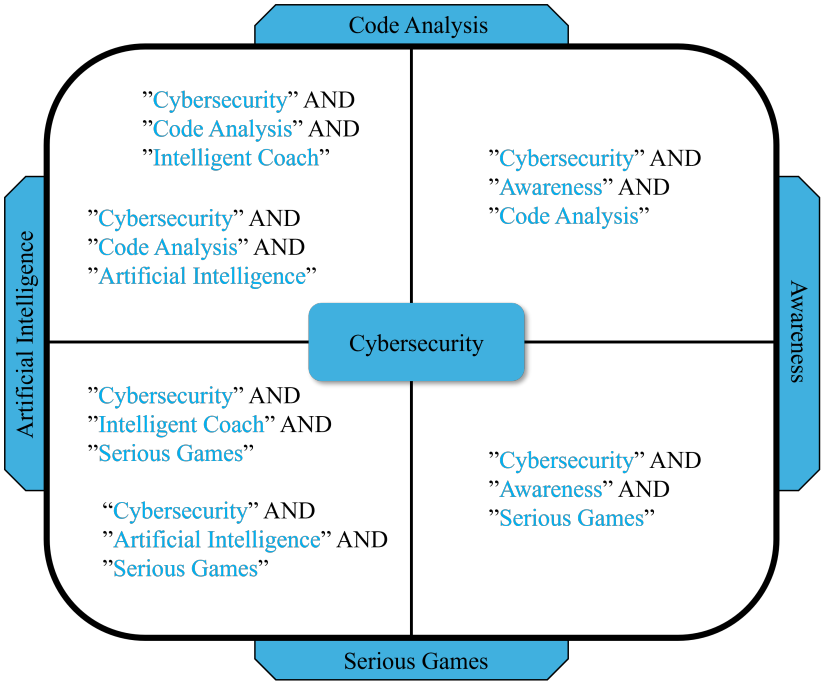


Figure 3.2: Keyword Quadrant

3.2.1 Data-Bases

In our lightweight literature review we used a total of two databases. It was from these databases that we gathered the relevant papers used in this section of the present research. Initially we applied the protocol to other data-bases, however, the results from these other sources comprised only of duplicates or empty results. Table 3.1 shows the two used databases and the designation attributed to each one.

Table 3.1: Used Databases

Designation	Databases
DB1	Web of Science
DB2	IEEE

3.2.2 Keywords

Based on the quadrant map of figure 3.2, we defined six different keywords. Table 3.2 shows the mapping between the defined keywords and the used designation.

Table 3.2: Defined Keywords

Designation	Keywords
KW1	”Cybersecurity” AND ”Code Analysis” AND ”Intelligent Coach”
KW2	”Cybersecurity” AND ”Intelligent Coach” AND ”Serious Games”
KW3	”Cybersecurity” AND ”Awareness” AND ”Serious Games”
KW4	”Cybersecurity” AND ”Awareness” AND ”Code Analysis”
KW5	”Cybersecurity” AND ”Code Analysis” AND ”Artificial Intelligence”
KW6	”Cybersecurity” AND ”Artificial Intelligence” AND ”Serious Games”

The keywords (KW5 - ”Cybersecurity” AND ”Code Analysis” AND ”Artificial Intelligence”), (KW6 - ”Cybersecurity” AND ”Artificial Intelligence” AND ”Serious Games”) didn't return any relevant content and so were not included.

3.2.3 Filters Used

A total of four distinct filters were applied to the set of papers found by our protocol. These filters intend to retrieve the relevant materials from the results provided by the databases. Table 3.3 shows the filters applied.

Table 3.3: Applied Filters

Designation	Filters
F1	In the Text
F2	In the Abstract
F3	Inclusion/Exclusion Criteria
F4	Duplicates

F3 includes the following criteria:

- Papers from 2010 to present;
- Papers in English;
- Papers with Relevant Content.

3.3 Applying the Protocol

After applying the review protocol previously defined, we obtained the results present in table 3.4.

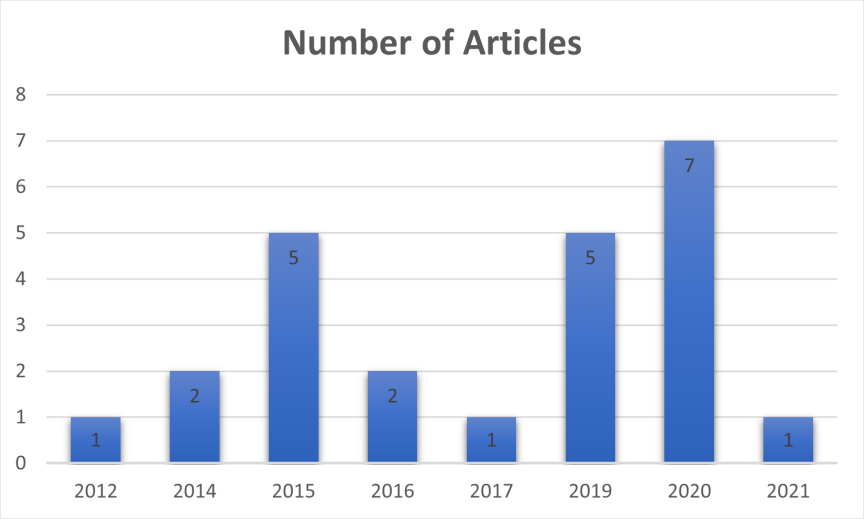
Table 3.4: Final Table of Papers

Databases	Keywords	F1	F2	F3	F4
DB1	KW1	0	0	0	0
	KW2	0	0	0	0
	KW3	8	8	5	4
	KW4	10	3	3	3
DB2	KW1	38	13	5	3
	KW2	17	5	2	2
	KW3	237	1	1	1
	KW4	1.667	10	1	1
TOTAL		**1.977**	**40**	**17**	**14**

After applying the protocol we obtained a pool of 14 articles with relevant information. The total of analysed results in this section amounts to 24 distinct reports. These results combine the ones obtained trough the application of the protocol (14), the ones provided by the thesis supervisors (6) and the ones gathered by applying the snowball method [46] to the two previous sources (4).

The totality of the papers were analysed in order to view the distribution by year of publication, Figure 3.3 and by the source, Figure 3.4.

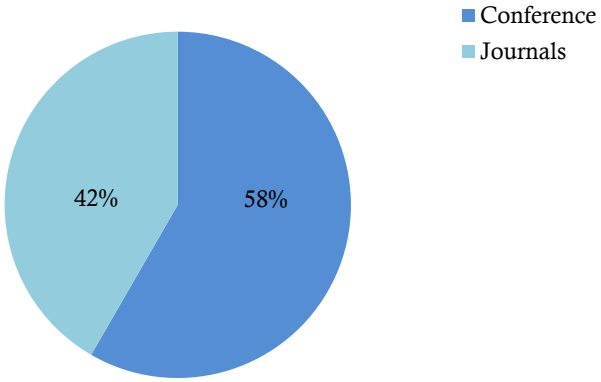
Figure 3.3: Year of Publication



From Figure 3.3 we can identify that 2020 was the year with the most contribution of articles, followed by both 2015 and 2019. These three years represent around 71% of the papers gathered in this work.

From Figure 3.4 we can identify a majority of conference papers. However, this only represents 58% of the overall data used.

Figure 3.4: Source of Articles



3.4 Reporting the Review

Due to its critical nature, cybersecurity, is considered one of the most serious aspects of technology as a whole [47]. In order to prevent malicious attacks there has been a recent new focus and emphasis to preserve the security of information systems [35]. Nevertheless, this effort has been met with a continuous increase in successful cyber attacks all across the globe, both in public and government controlled organizations [48]. This research focuses on raising the awareness of software developers with a serious game to try and mitigate these encountered problems. There are already several examples of serious games that not only aim to raise the cybersecurity awareness but also reinforce the knowledge of its players.

In the academia, there are already some success stories. Probably the most known CTF is the International Capture-the-Flag (ICTF) annual event that as been steadily increasing the number of attending universities and, therefore, the number of participants [49]. The ICTF consists on the Attack-and-Defend type of CTF where the teams must protect their system whilst attacking the other teams. The more successful a team is the more flags they gain, ultimately leading to victory [50]. Although this event has the gamification approach embedded in the experience, it does not incorporate any type of intelligent helper or coach.

In the University of Birmingham a serious game based on Jeopardy-Style attack-only CTF was introduced has one of its evaluation of the academic curriculum. The students had a lab-like evaluation where they had to take the security teachings from theoretical classes and apply them in this CTF game along the school year. Following other CTF-based games, the practical evaluation given to the students in this class was linked with the amount of flags obtained. More flags, higher the grade. Results showed that this approach is an effective measure of the students skills and knowledge in the cybersecurity field [37]. Although this approach does not employ an intelligent helper or coach, teachers stated that throughout the year some hints were given to the classes regarding harder types of challenges.

In the Industry, the serious games approach is also gaining a good amount of recognition. Several CTF based games have been reported, at the industry level, with very promising results. In [51] the game Control-Alt-Hack is introduced. With a main goal of raising the awareness for cybersecurity problems this tabletop game gives the players the ability to make a fictitious attack on a network. The player starts by selecting a card containing a determined network description and creating an attack scenario. Following this, the player must give insight on how the attack works, why it works on the particular network and what software developers can do to mitigate this risk on their code. The success of the attack is determined by a dice. Each

successful attack gives the player a point. At the end of the game the player with most points wins. In [52] is introduced Riskio. This card-based game takes into account the defensive side of the exploitable vulnerability. The players are able to pick a card to defend against the given attack. This results in players leaving the game with an enhanced cybersecurity awareness and an overall increased interest and enthusiasm in this topic [51]. These CTF-like approaches focus more on developers discussing the topic amongst one another, instead of actually programming safe code. Although they employ a gamification approach, no intelligent helper or coach is present in the developed game. In fact, results regarding an intelligent coach linked to helping learn cybersecurity were very scarce. At the time of the application of this lightweight literature review, the only usage of an intelligent coach in the current field of study was applied to the Sifu-Platform [53].

The Sifu-Platform, a serious game developed for the industry, contains some Jeopardy-Style, defense-only type of cybersecurity challenges. This code-entry programmable type of challenge provides the players with insecure code from a software project. In order to obtain the flags (points) the player must modify the code in order to mitigate the security issues present. Once playing, the player interacts with the web platform, and their code is run through several layers of analysis tools in order to identify potential vulnerabilities. As of the beginning of the present research, the Sifu-Platform only employed C/C++ challenges. Furthermore, also implemented in the platform, is an intelligent coach that provides hints that aid the player in the solving of the challenges. This behaviour intends to reduce the fatigue and frustration, ultimately improving the motivation and overall experience of the player [16].

It is based on this initial approach to C/C++ challenges that this research intends to build upon. In this work, we develop a similar approach to raise cybersecurity awareness of Java developers, as well as an improvement of the intelligent coach's algorithm.

Chapter 4

Java Cybersecurity Challenges

Being the building blocks of software, every programming language comes with its own set of challenges. One of the most critical aspects of these challenges is the security issues inherent to any programming language and developed software. When not correctly handled, these issues can cause major problems to both software companies and its users. Being one of the most known programming languages, Java is no exception. A study by Rayome [21] states that Java is the third programming language with the most known security vulnerabilities.

There are several ways to deal with security related issues. Some approaches to raising software quality range from code reviews to static code analysis. Another approach to this issue is to target the human factor (software developers) by educating them in terms of secure coding. This latter approach is the focus of the present research. To raise the awareness of software developers in the industry, we developed a serious game that aims at giving developers hands-on experience on cybersecurity issues and how to solve them. Our serious game, Java Cybersecurity Challenges (JCSC), is comprised of several security challenges that aim to raise awareness to distinct security related issues. The JCSC combines a serious game type of approach with a capture-the-flag type of event. Figure 4.1 shows the distinct stages that comprise the JCSC events.

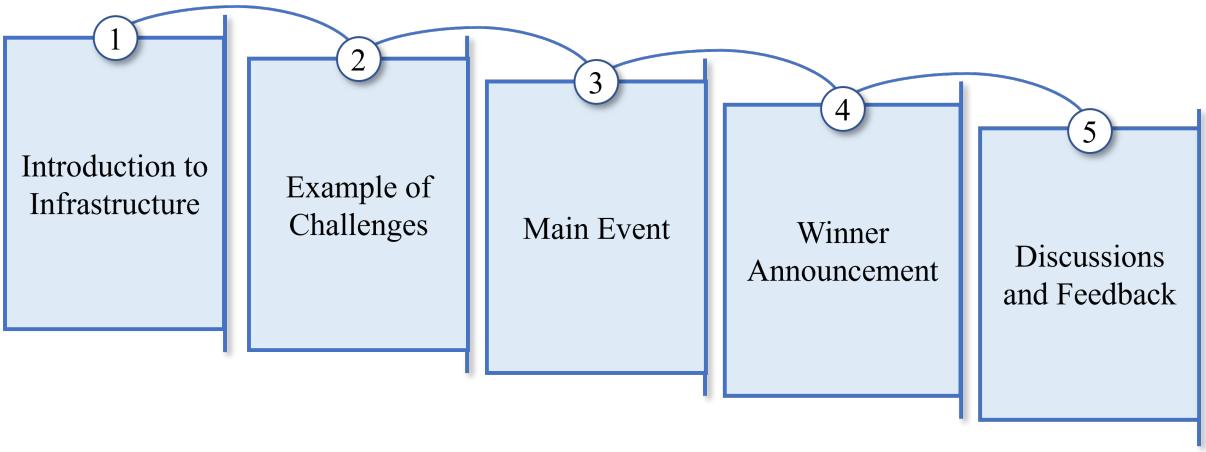


Figure 4.1: Java Cybersecurity Challenge Event

The event itself is comprised of five distinct stages: (1) Introduction to Infrastructure, (2) Example of Challenges, (3) Main Event, (4) Winner Announcement, (5) Discussions and Feedback . The event starts with a brief introduction of the coaches followed by an overview of the platform and the dashboard. The goal of this stage is to teach players how to correctly handle the platform and the dashboard in order to decrease potential confusion during the event. Moreover,

the players are also divided into teams that will remain the same across the workshop. After this stage is completed, the coaches introduce the challenges and provide a brief explanation on how teams should approach the exercises. Furthermore, the coaches solve one, or more, challenges to further explain the exercises and answer any potential questions the teams might have. The main event represents the majority of the workshop. It's in this stage that teams must solve the challenges to gain the largest amount of points they can. Furthermore, these points serve as an incentive to make players want to win the event, learning secure coding in the process. During this stage, the coaches are also monitoring the teams and can intervene if a team has a particular question or if they are stuck in an exercise.

Once this main event is completed, the coaches end the workshop and announce the winning team. This team is the one that was able to gather more points during the main event. The workshop finalizes with a brief feedback round concerning the good, the bad and what can still be improved regarding the challenges and concept as a whole. If requested by any of the teams, the coaches can provide the solutions of exercises the teams could not solve or show implementations performed by other teams.

During the workshop event, each player accesses a server hosted in the cloud that serves the JCSC Challenges. Through this virtual machine the players have access to all the necessary material. Furthermore, the workshop has a dashboard where players can obtain the links to the challenges. It's in this dashboard that players can redeem the flags given when a challenge is solved. Once redeemed, each flag gives the player a pre-determined number of points.

Figure 4.2 depicts the three step sequence of a Java coding challenge.

Figure 4.2: Java Challenge's Sequence

Firstly, when a team enters a challenge, they are faced with a text file (Readme.txt) that explains in detail what the purpose of the code is. This description ensures that the players understand what the code should do and sets the stage for the coding challenge itself. Once the desired functionality of the code is understood, they can advance to the interactive game itself. In this step, the team is faced with a block of code that contains at least one security vulnerability. In order to correctly solve the challenge and win the flag, the team must solve the security issue present, without compromising the functionality of the code. It's in this stage that the teams interact directly with both the programmable challenges and the Sifu-platform. Once the team submits the code to the backend, one of two outcomes can be encountered. If the security vulnerability was correctly solved, then the team is rewarded the corresponding flag and the challenge is solved. However, if the code still has security issues, the Intelligent Coach will provide meaningful hints and feedback about the encountered problem present in the submitted code, and the team must try again. Once the challenge is solved and the flag is given, the team can either be provided with an additional simple multiple choice question or a brief explanation on what the issues was. This last step aims at cementing the knowledge that the players gain from each challenge.

4.1 Platform Defenses

When a team submits the code, the backend analysis process employed by the platform is triggered to analyse the code. Since the teams are free to change every aspect of the code, the platform employs security measures that aim at mitigating potential threats to the analysis process and the platform itself.

Table 4.1: Platform Security Defenses

Requirement	Attack Vector	Counteraction
Infinite Loop in Code	Denial-of-Service	Container has Limited Lifetime
Malicious Players	Remote Code Execution	Sanboxing Environment
Malicious Players	Brute Force Attack	Hint-Timer

Table 4.1 pinpoints the three major significant problems that the platform can encounter when analyzing the submitted code. The backend applies protective measures to shield itself against these possible issues. Against infinite loops, whether intentional or not, the platform is equipped with several timers that break attempts to consume too many system resources. Dynamic tests run in a sandboxed environment, with no internet access, to protect against malicious players and minimize the threat of a player trying to hack into the backend. Finally, in order to mitigate the chance of a player speed-running the challenges by re-submitting the code and getting all the hints at once, the platform employs a hint timer. This mechanism prevents the hint generation algorithm to provide hints in short amounts of time, only generating them after the timer is reached.

These measures assure that the backend analysis process is secure from these possible attack vectors.

4.2 Backend Analysis Process

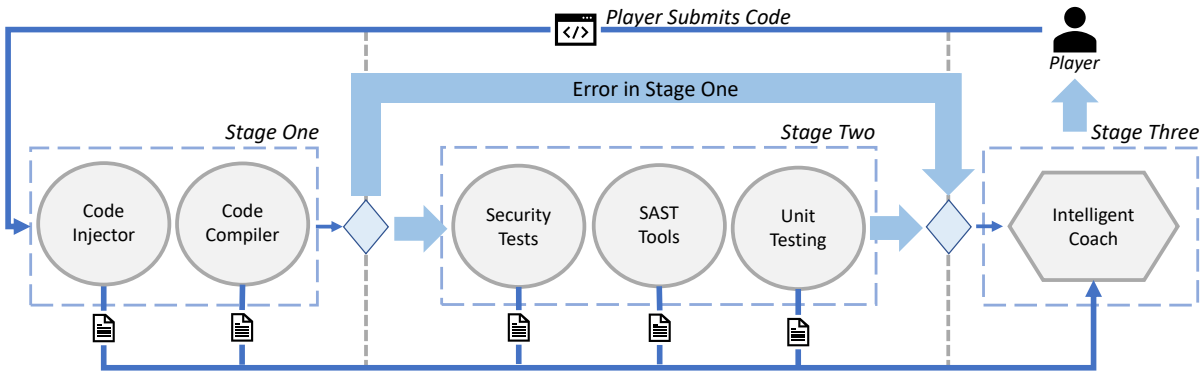


Figure 4.3: Automatic Assessment Process

Figure 4.3 shows the distinct stages that make up the designed framework employed for backend analysis of submitted code. Each of these stages is comprised by several steps that test the code in different manners. The first stage (S1) is composed by two critical steps, code injection and code compilation. Code injection occurs when the players' code needs to be modified. These changes comprise of the insertion of additional functionalities, such as new classes, methods, import statements, etc., that need to be added to the submitted code in order to provide extra context to the testing tools down-stream. This step is done without the player's knowledge. Since the players don't know what type of tests run in the backend, giving any

insight about this could lead to players wanting to deceive and overcome our testing methods to win the challenge.

Next, the compilation of the resulting code is then performed. If any errors are found in this stage, the process skips stage two and moves directly into stage three where the player receives feedback about the critical compiler error encountered. This means that if any compiler issues arise from stage one, no analysis of the submitted code is performed by the backend. This behaviour is due to the nature of the tests employed in the analysis. The backend makes use of dynamic testing and tools, that cannot run in un-compiled code. On the contrary, if no compilation issues are detected then the process continues normally into stage two. Stage two (S2) can be seen as the main analysis stage of the backend process. Here the previously compiled code of S1 is analysed and tested using three distinct testing methods: security tests, static code analysis and unit tests. For each testing need the researchers selected a subset of potential tools that could aid in the analysis of the submitted code. The selected tools were chosen due to their capability to flag and report security issues related to the Java programming language. Table 4.2 shows the tools used in this stage as well as their mapping to the different testing methods.

Table 4.2: Analysis Tools

Tools	Unit Tests	Security Tests	SAST Tools	Reference
JUnit	•	•		[54]
Self-Developed Tools		•		[55] & [56]
SonarQube			•	[57]
FB-Infer			•	[58]
SpotBugs			•	[55]
PMD			•	[56]

Unit testing is performed to ensure that the player's code works according to the challenge' specifications. Each developed challenge has a text base description file. This file contains an detailed explanation on what the purpose of the exercise is. Based on this description, the player has to secure the code without compromising the desired functionality. Due to this, we employ unit-tests to guarantee that the submitted code behaves like intended. For unit testing, we employ a framework wildly used in an industry setting called JUnit tool. This tool guarantees that the submitted code respects the exercises' desired functionality, flagging issues like empty classes (where no security vulnerabilities are found).

Security tests are dynamic unit tests that mimic malicious user actions and input to try and exploit the exercise's vulnerability. When applying these tests we try to break the players' submitted code by using distinct methods (random inputs or method calls) that will either crash the system or detect any underlying issues present within insecure code. For this purpose we also employed the JUnit framework as well as some custom made dynamic tests developed by the researchers. These tools provide, among others, a series of tests that test the given code by creating random inputs in order to try and trigger any potential issues with the given code. This testing approach is necessary to complement the analysis performed by the Static-application security testing tools (SAST). These tools, like most of the developed software, are not perfect and might provide results that are not completely correct. Furthermore, the dynamic tests performed should be randomized to mitigate the probability that players might try to develop solutions against the employed testing in the backend.

SAST tools analyse the submitted code and provide a report based on the findings. These tools give reports concerning a set of problems including code vulnerability issues. For the SAST tools we gathered industry recognized free and open-source software.

Table 4.3: SAST Tools Outcomes

Type	Description
True Positive	Code has a vulnerability and the tool flags it
True Negative	Code doesn't have a vulnerability, and the tool does not flag it
False Positive	Code doesn't have a vulnerability, but the tool flags it
False Negative	Code has a vulnerability, and the tool doesn't flag it

These tools however can produce wrong evaluations of the given code. Since the tools were developed to flag issues in a certain situation, sometimes they can fail and provide us with unsatisfactory results. Table 4.3 shows the four different outcomes that can occur when dealing with SAST tools. Of the four possible results only two represent the desired functionality, True Positive and True Negative. The other two represent situations where the tool can either flag issues that are not present in the code or fail to do so when vulnerabilities indeed exist. In order to mitigate the potential false results in our security assessment all the challenges were extensively tested before being available to the participants of the security workshops. In the development of the artifact, only one challenge (resource leakage) encountered a false positive from a SAST tool. In this particular situation, the coaches were aware of this issue and so, if any participants were to submit code that was correct but resulted in this false positive, the coaches would provide the flag manually to the team in question.

In S2, the analysis steps create reports about the findings related to the submitted code. These reports can be of several types, e.g. *Bad Pratices*, *Code Performance*, *Security Issues*. In this research, since the main goal is to educate developers on security related topics, we focus on the security related findings produced by the tools.

Once the analysis of the code is concluded, the findings from the tools suffer a normalization process to a common, simple format, type of file (JSON) [59].

This new normalized reports contain all the findings related to security issues found in the submitted code. Each finding is assigned a set of parameters such as a PASSFAIL boolean type of value (that indicated whether a test passed or failed), a Common Weakness Enumeration, or CWE [60], a Priority, among others. These parameters are used by the intelligent coach to determine the nature of the given hint.


```
"bug_type": "RESOURCE_LEAK",
"qualifier": "resource of type 'java.io.FileOutputStream' acquired by cal to 'FileOutputStream(…)' at line 9 is not released after line 13.",
"severity": "ERROR",
"line": 13,
"column": -1,
"procedure": "resource_leak.createsFile(java.io.File.java.lang.String):void",
"procedure_start_line": 8,
"file": "resource_leak.java",
```


Figure 4.4: Reports Generated from Tools

Figure 4.4 represents the findings that the SAST tool FB-Infer generates when analysing a code that contains a resource leakage issue. This finding will then go through the normalization process originating the finding contained in Figure 4.5. This normalized version of the results contains some indicators that will be later used to generate meaningful feedback and hints. The first value is the *originTool*. This value represents the name of the tool that performed the analysis linked to the presented report. In this case, this test was conducted by the SAST tool *FB-Infer*. Secondly, each report comes with a *Priority* associated with it. This value is one of the most important values in the analysis and each type of testing has its own range of possible

priorities. The priority of compiler errors range from 0 to 1000, while the unit/dynamic tests one range from 1000 to 10.000. The priority range linked with SAST tools analysis is from 10.000 to 100.000. The ranges of the priority value were mapped by the researches. This value is used by the intelligent coach's' algorithm to determine which issues are chosen to serve as basis to generate the hints. The third value in these files is the *Description*. This indicator contains the error message generated by the tool that flagged the issue. This value is mostly used as the basis of the static feedback employed by the platform. The platform creates most of the static feedback by joining the information of the *Description*, *LineNumber* and *FileName* values. The last three values represent critical indicators. The *ErrorID* corresponds to the mapped CWE of the found issue. In this case, we mapped the resource leakage issue to CWE 772 [61]. The *Tag* value is the name of the ladder corresponding to the particular issue. This name was created by us and each ladder has it's own unique name. The last indicator, *PASSFAIL* value, is a Boolean type of value that indicates whether the particular test has either passed or failed. The mapping to individual CWE's is based on our own experience in the cybersecurity field.


```
[
  {
    "originTool": "FB-Infer" ,
    "priority": 10000,
    "description": "resource of type  'java.io.FileOutputStream' acquired by cal to 'FileOutputStream(…)' at line 9 is not released after line 13.",
    "lineNumber": 13,
    "fileName": "resource_leak.java",
    "errorID": "772",
    "tag": "INCREMENTAL_3_Leakage_",
    "PassFail": "FAIL",
  }
]
```


Figure 4.5: Normalized Findings

Normalized reports like the one presented in Figure 4.5 are used as input for the last stage of the backend analysis process, Stage Three (S3).

The three stage analysis process of Figure 4.3 is applied through a *Yaml* file. This language is mostly used in configuration and data interchange files and represent a human-readable type of programming language [62]. In our research, this file defines all the analysis and testing steps, inherent to each stage, that the backend must run each time a player submits the code. This file also defines the behavior of the process flow i.e. the skip of S2 that occurs if an error is encountered in S1. The final stage of the backend analysis process refers to the intelligent coach present in S3.

4.3 Intelligent Coach

This third and final stage, refers to the artificial intelligence engine that generates relevant hints to aid the player when the code still contains vulnerabilities. If no tests from the previous stages have failed (shown by the PASSFAIL value), then the challenge is complete and the player has won. Otherwise, the intelligent coach proceeds to generate the hints to aid the player. Hints are generated based on the both the CWE selected from the list of normalized findings and the prior history of the exercise, using a laddering technique [16, 23]. Figure 4.6 depicts the process of the CWE selection performed by the intelligent coach algorithm. This algorithm represents a more detailed description of the process performed in the last stage (S3) of the pipeline shown in figure 4.3.

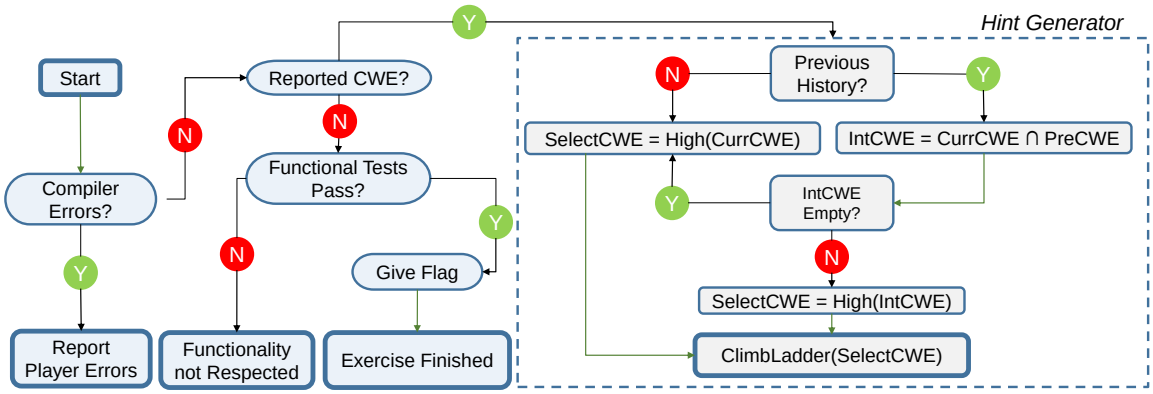


Figure 4.6: Intelligent Coach's Algorithm

When the backend process reaches S3, the intelligent coach will first check if the analysis performed flagged any compilation errors. This is done by checking the reports with priorities ranging from 0 to 1000 for the FAIL value. If these reports include any compilation errors, an error message is sent back to the player reporting the critical errors that prevent the code from correctly compiling. The feedback given to the player contains both the name of the file that prevented the project from correctly compiling, as well as the adjusted line number of the compilation error. If errors in the injected code are the reason the project cannot correctly compile, the player receives a report that the backend is broken.

If the code has compiled with no errors then the algorithm will check for the existence of FAIL values regarding the analysis reports with priority value higher than 1000. If no reports are found the code has no CWEs'. In this case, the code no longer contains a security issue, and so, the process advances to check whether the functional tests passed.

If all these tests pass then the player has corrected the security issues of the code while respecting the desired functionality depicted in the challenge description. This means that the player has mitigated the challenge's vulnerability and is rewarded a flag.

In case of failed functional tests found, the player is given feedback about the corresponding functionality not respected by the submitted code.

When security issues in the submitted code are found, i.e. reported CWE's exist among the reports, the process advances to the hint generator algorithm. This simple straightforward algorithm will choose one of the CWEs found in the analysis reports and generate a hint to provide to the player. Once the process reaches this step, the hint generator checks whether the player has received previous help from the intelligent coach based on the exercise's previous history of hints. By having a prior history of the challenge, the intelligent coach can check what problems the player has faced in earlier submissions and avoid giving hints related to other problems that might have arisen in the meantime. If no hints have been previously given then, the chosen hint, will be based on the highest ranking CWE reported out off all current CWE's. However, if the previous history exists, the intersection between the currently detected CWE's and the past CWE's is executed. After this interception is performed, the intelligent coach will check whether this list is empty (no common CWE found from current and past submissions). If this is the case, then the previously detected security vulnerabilities have been solved and so the hints must be purely based on the currently found CWE. Therefore, the chosen CWE is the highest-ranked out of all current CWE's found.

Otherwise, if the interception list is not empty, the selected CWE will be computed based on the highest-ranked CWE found on the intersection list. In this case, the player has faced the same problem before and so the hint must be more precise and helpful than the previously given one. After the CWE is selected, the intelligent coach will apply the laddering technique to generate the appropriate hint. The ranking of CWEs was internally done based on the supervisors

past experience in the field and was adapted to each exercise goal.

The final step of the backend analysis process is to generate the hint to be given to the player. The intelligent coach makes use of the laddering technique previously described [16]. The system itself is based on different ladders, each containing a sequence of hints. Each ladder is linked to a particular CWE and is defined by several distinct levels. Each level contains a hint to be given to the player about the corresponding CWE. Each ladder can be defined by a sequence of hints that range from generic nature (lower levels) to more specific ones (higher levels). The lower level hints give a generic explanation on what the encountered problem is. The higher level ones tend to provide the player with a more concrete and detailed explanation about the problem and how to solve it. This system of different types of hints functions as a spectrum that increases the specificity of the given hints when increasing the level of the ladder. This incremental approach to the level of hint specificity is to encourage the players to think for themselves and try to solve the exercise without being given the answer directly from the start. Furthermore, this model of hint generation can potentially mitigate possible players' frustration experienced when solving the challenges. Each given hint aims at taking the player one step closer to one of the possible solutions.

Table 4.4 contains an example of a ladder mapped to CWE 772 [61].

Table 4.4: Resource Leakage Hints

Hint Level	Description
Level 1	CWE 772 - The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.
Level 2	You must guarantee that the stream is being closed in every possible scenario...
Level 3	To assure maximum security you need to close the stream after using it. You must also check if you are closing the stream in all possible scenarios.
Level 4	Don't forget that even when the code throws the exception the stream must be closed. Do you know about the finally operator of the try/catch/finally block?
Level 5	The finally block assures that every time the program exits it passes through it. By closing the stream inside the finally block you are guaranteed to always close the stream no matter the situation. You can learn more about it here Finally Operator

We mapped this CWE to the FIO04-J rule of the SEI-CERT Java coding guidelines [63]. This particular vulnerability occurs when a program fails to release resources such as file descriptors or database connections. This can lead to resource exhaustion attacks.

The hints provided by the Intelligent Coach aim at teaching software developers on how to deal with these situations. The first hint provided refers to the definition of the encountered CWE. From here, lower level hints, e.g. level 2, tend to have a generic, and playful feedback. Going up through the levels we can depict an increase in the specificity of each hint, until the last level is reached. This last level, e.g. level 5, gives a concrete explanation of the found vulnerability and provides a solution to the problem.

Once the intelligent coach selects a CWE, the laddering technique is applied. The intelligent

coach chooses a hint from the ladder mapped to the corresponding CWE. The chosen hint is based on how many times the player has submitted the code containing the same vulnerability. If a player has performed two distinct submissions with the same issue, then the hint provided is the one stored in the second level of the corresponding ladder.

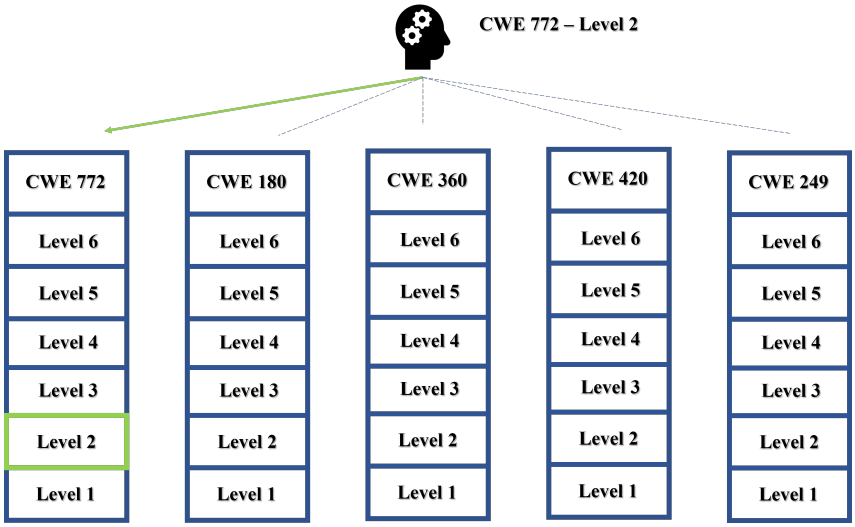


Figure 4.7: Hint Selection Process

Figure 4.7 represents the intended behaviour between the intelligent coach and the laddering technique. Every time a player submits the code the corresponding ladder is selected and the level is determined. If a player re-submits the code containing the same vulnerability, the ladder selected is the same, however the hint chosen will be the one present on the previous level + 1. This process of hint generation is repeated until the player either solves the challenge or solves the vulnerability in question. In the later case, if the code has more vulnerabilities, this process will be repeated, this time with a new ladder.

All this analysis and feedback generation steps performed by the backend intend to give the player the a good environment on which they can learn secure coding. Figure 4.3 shows the perspective of a player after submitting unsecure code to the backend. The feedback within the red rectangle represents the static feedback employed by the platform. The blue rectangle contains the hints provided by the intelligent coach.

Figure 4.8: Players Perspective of the Game

Chapter 5

Solution Development

In this section, all the stages related to the development of the artifact and its evaluation in an industry setting will be presented according to the ADR methodology [22]. This methodology is based on a cyclic approach of development and evaluation. In the present research a total of four distinct cycles were held. Each cycle is defined by a goal determined by the researchers and by the results gathered from previous cycles of development. Following the artifacts development of each cycle, several interventions were held to test and evaluate the current version of the artifact. The results gathered from each cycle originate in these interventions and were gathered through surveys. These surveys were developed by the researchers and contains a total set of 16 questions related to the artifact and the research in question. Furthermore, of the 16 questions, 13 were based on a 5-point Likert Scale [64], with 1 being the lowest (Strongly Disagree) and 5 being the highest (Strongly Agree). The remainder three were open-answer type of questions and were of non-mandatory nature.

The awareness theory, presented in chapter 3, served as a guiding tool when developing the questionnaire and its questions. In the present work, we employ the three defined aspects by means of the Sifu-platform and the Java challenges. The questions present on the provided questionnaire were mapped to represent one of these three aspects. This approach allowed us to correlate the awareness level perceived by participants with the development of the artifact.

Table 5.1: Questionnaire

Identifier	Question	Theoretical Construct	Sub-set
Q.1	How do you rate the overall experience with the platform?	G	S1
Q.2	The error messages and hints issued by the platform are relevant to the exercise.	HI	S2
Q.3	I can learn secure coding by solving the challenges in the platform.	PRO	S1
Q.4	I can relate the hints to the code I have written.	HI	S2
Q.5	The hints provided by the platform make sense to me.	HI	S2
Q.6	It is fun to play the coding exercises in this platform.	HP	S1
Q.7	I like to play the exercises in this platform.	HP	S1
Q.8	If I am given the opportunity, I would like to play more exercises.	HP	S1
Q.9	The hints I have received, helped me to understand the problem with the code I have written.	HI	S2
Q.10	Playing the exercises makes me more aware of security vulnerabilities in Java code.	PER	S1
Q.11	I have learned new threats that can result from vulnerabilities in Java code through playing the exercises.	PER	S1
Q.12	I feel compelled to improve the security of my code in the future.	BE	S1
Q.13	I have learned new mechanisms to avoid vulnerabilities in Java code.	PRO	S1
Q.14	Would you change the name of the challenge, to hide the issue with the code?	G	N/A
Q.15	What would you suggest to change or improve in the platform?	G	N/A
Q.16	What do you think the platform does good?	G	N/A

Legend - BE: Behavior, G: General, HI: Hints, HP: Happiness, PER: Perception, PRO: Protection

Table 5.1 shows the questions present in the questionnaire given to participants in the interventions performed in each cycle of development. All the answers gathered were collected anonymously and there is no indicators that would allow for the identification of a participant. Furthermore, the participants agreed to take part in the study and were instructed about the

objective of the research. The questions from the survey were divided into two sub-sets. The first one (S1) contains the questions related to the challenges, platform and concept as a whole. Sub-set two (S2) contains the questions related to the hints and feedback given to the players. This distinction was made to better identify and analyse the improvements that could still be made regarding each topic.

5.1 Cycle One - Proof of Concept

This section describes the initial concept and design of the proposed work. The early artifact was evaluated in both an industry and academic setting. The interventions served to validate the proposed approach to raising awareness to cybersecurity issues and understand how Java developers perceive the platform and concept as a whole.

The work developed during this initial cycle was published and presented in ICPEC 2021 – 2nd International Computer Programming Education Conference on the 28th of May 2021 [24].

5.1.1 Problem Formulation

This initial cycle aimed at developing two major primary aspects of the Java programmable challenges. Both these features represent a fundamental part of the proposed research. In order to raise cybersecurity awareness by developing Java programmable challenges, we firstly need to develop a framework on which the analysis of the code can be based on. Furthermore, we also require a set of tools that can be easily applied to this though-of framework. Due to this, this initial cycle of development aims at providing a solution to the following problems. Firstly, we need to understand how to automatically assess the level of security of a player's Java code. Secondly, we need to select a sub-set of open-source tools that can be used to assist the Java security code assessment.

We can tackle these issues by creating and deploying a solution that extends the Sifu-Platform for Java challenges based on previous work [16]. This approach provides us with a initial basis on which we can build upon our final code analysis framework and tool subset.

5.1.2 Building

In order to produce a working prototype, a series of search, development and test phases where held. Firstly, we started by determining an appropriate framework that could be applied to analyse the players submitted code. This framework has the main goal of establishing the different testing needs and scenarios that can happen when analysing untrusted submitted code. Furthermore, after reaching a suitable solution to this problem, we researched and selected a sub-set of free open-source tools that could be applied to the distinct testing stages of the defined framework. This selection was based on our previous experience combined with a research of the available analysis tools. Lastly, we developed one Java challenge to test the proposed analysis process and tools. By the end of this cycle of development, the artifact contained an initial version of the analysis process (Figure 4.3), a sub-set of tools to analyse the submitted code (Table 4.2), and an initial Java challenge where both the designed framework and analysis tools employed could be tested. The following subsections detail each one of the developed aspects of the artifact performed in this cycle.

5.1.2.1 Initial Java Challenge

Our initial Java challenge is based on the FIO04-J rule [61, 63] of the SEI-CERT coding standards, and contains a resource leakage vulnerability. This vulnerability was selected to be the

basis of our initial Java challenge due to its simple nature. Furthermore, the selected subset of analysis tools could be easily applied to this example, making it a perfect initial challenge to develop.

Figure 5.1 shows the code present in this Java challenge, which was given to the participants.

```
import java.io.*;

public class ResourceLeak {
    public void writeToFile(File file, String msg) throws IOException {
        FileOutputStream fos = new FileOutputStream(file);
        fos.write(msg.getBytes());
      }
  }
```

Figure 5.1: Initial Java Challenge

This code contains a resource leakage vulnerability. The presence of the vulnerability is due to the fact that the FileOutputStream (fos) used to write the message to the file is never closed. This challenge aims at making players realise that although Java is a garbage-collected language, the garbage collection mechanism does not work for file handles and database connections. Once these resources are no longer needed they should be immediately closed otherwise a resource leakage can occur. A failure to do so can have several consequences. One of those, resource starvation, occurs when an attacker triggers the code too many times making the system open too many file handles. This problem can ultimately lead to a denial of service attack or even a potential crash of the system. This problem is many times overlooked mainly because the code compiles and runs ”correctly”. This malpractice can create serious problems to the user of the program.

To solve this problem a developer can use one of two solutions. Either employing a *try-catch-finally* block, as seen in figure 5.2, or by using a *try with resources* approach, as seen in figure 5.3, the resource leakage problem will be mitigated.

```
import java.io.*;

public class ResourceLeak {

    public void writeToFile(File file, String msg) throws IOException {

        FileOutputStream fos = new FileOutputStream(file)
        try{
            fos.write(msg.getBytes());
        }catch(Exception e){
            //Handle Exception
        }finally{
            fos.close();
        }
    }
}
```

Figure 5.2: *try-catch-finally* Block solution

In this design cycle, the platform employed an initial version of the analysis framework, were the security related aspects of the analysis were more worked on than the other tests. Furthermore, the feedback and hint generation was not considered in this cycle of development.

```java
import java.io.*;

public class ResourceLeak {

    public void writeToFile(File file, String msg) throws IOException {

        try(FileOutputStream fos = new FileOutputStream(file)){
            fos.write(msg.getBytes());
        }catch(Exception e){}

    }
}
```

Figure 5.3: *try* with Resources solution

Table 5.2: Cycle 1 - Interventions

Date	Context	Interview Type	Participants
22 to 27 February 2021	Mixed	Individual	11

Instead, the coaches (researchers) served as the intelligent helper that aid the players in solving this challenge.

5.1.3 Intervention

Once the Initial Java Challenge was working according to the researchers expectations, an intervention was held. Table 5.2 contains the information regarding the intervention performed in this cycle. This intervention occurred from 22 to 27 February 2021 through individual online meetings that lasted 30 minutes. Each meeting was divided in two events. Firstly the participants had 20 minutes to try and solve the challenge. During this time, they were free to interact and test the platform and challenge in any way they desired. Following this, the final 10 minutes were used to perform a semi-structured, individual interview. In this latter event, participants were free to voice their opinions about the concept, the platform and the challenge design and functionality as well as describe their experience with the game. A total of 11 participants took part on our trial run. The age varied from 20 to 35 years and, of the 11 participants, nine were working in the industry and the remaining two were from academia. At the time of the trial run, the industry participants were developers for critical infrastructures with several years of experience from both Germany and Portugal. The two academia participants were senior students in a computer science related course in Portugal. The participants tested and solved the challenge and were free to experiment with the platform without any restrictions. After each individual meeting the participants were asked to answer the questionnaire presented in table 5.1.

5.1.4 Evaluation

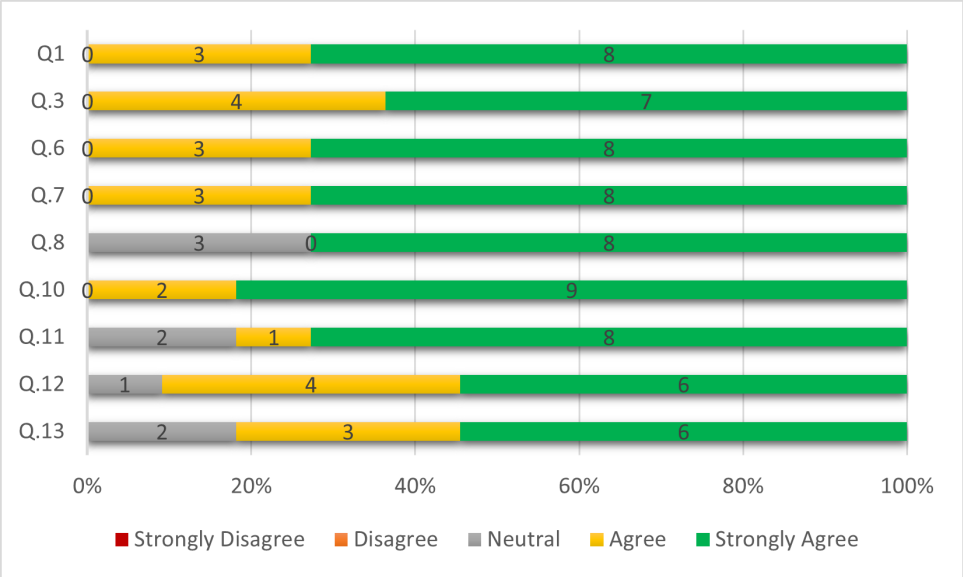
This section presents the results collected from our trial run of the initial Java challenge. A critical discussion of the results is also present.

5.1.4.1 Results

After the first intervention, the survey had collected a total of 11 results. Our initial artifact lacked the hint generation algorithm and gave very simple and primitive feedback to the players. Due to this, we selected a subset of questions to analyse from the given survey. Only the

questions related to the concept and challenge itself were selected. The ones regarding the hints and feedback were not analysed due to the fact that this system was not yet fully implemented. The most meaningful results of this cycle are directly linked to questions that can indicate whether the participants perceive our implementation as a useful approach to learning cybersecurity. The selected subset of questions is composed by seven questions: Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11. Figure 5.4 shows the results of this subset gathered from the participants of our trial run.

Figure 5.4: Proof-of-Concept Results



Analysing these results we can highlight that all the participants had a good experience when playing in the platform. The answers to Q.1 show that every participant had an overall positive experience with 27.3% of participants agreeing with the statement (4/5) and the remainder 72.7% strongly agreeing (5/5). Moreover, Q.3, Q.10 and Q.11 show that the participants see the coding exercises as a good means to learn secure coding and avoid security vulnerabilities when developing code. This can be extrapolated by the positive results we gathered on the three questions. On Q.3 36.4% of participants agreed (4/5) and 63.6% strongly agreed (5/5) with the corresponding statement. Furthermore, on Q.10, 18.2% of participants agreed (4/5), and the remainder 81.8% strongly agreed (5/5). On Q.11, 18.2% of participants provided neutral answers (3/5), 9.1% agreed with the statement (4/5), and the final 72.7% strongly agreed with the given statement (5/5). Q.6 and Q.7 generated the same results. 27.3% of participants agreed with the statement (4/5) and the rest 72.7% strongly agreed (5/5). Q.8 shows that about 73% of the participants would like to play more exercise by strongly agreeing with the statement (5/5). However, 27% provided neutral answers (3/5), and so are not sure. The final two questions gathered similar results. On Q.12, 36.3% agreed with the statement (4/5), 54.6% strongly agreed (5/5), and the remainder 9% felt neutral (3/5). Q.13 gathered 54.6% of the results with strongly agreed (5/5), 27% of participants agreed (4/5) and the remainder 18.18% answered neutral (3/5).

Valuable input was also received during the semi-structured interviews. Some comments from participants helped defining and structuring changes that could be implemented in further work. Amongst the comments received we can highlight the following:

"Very practical tool, the instant feedback helps a lot but is a bit underdeveloped"
"Better than mainstream environments because it also tests the code for security issues"
"Had no idea that this was a security issue and now I know how to solve it."

5.1.4.2 Discussions

The overall feedback gathered from both the questionnaire and the individual meetings suggest that the participants enjoy and welcome this type of exercises. Furthermore, although the hints given to the players were of rudimentary nature the players also perceived this aspect as useful, providing positive ratings. However, the participants also stated that this aspect should be improved in further developments. This is due to the fact that by the time of this intervention, the platform employed static feedback that remained the same each time the same issue was detected. Furthermore, the hints only gave the generic description of the CWE in question. These conclusions align with previous similar studies and serve as encouragement to carry out the further development of the platform.

From the results gathered in this cycle, no possible distinction could be made between the participants from the industry or the ones from academia. Furthermore, there where no indicators in the collected data that would allow us to identify any of the participants.

The results gathered from this cycle of development suffer from a positive bias, due to the nature of the conducted interventions. Since the intelligent coach was yet to be developed, the coaches had to take a more active and helpful approach. Furthermore, these interventions were of individual nature, one participant by intervention. Both these aspects lead to the extremely positive results gathered in this cycle of development. However, as discussed in the following sections, these mostly positive results do not impact the conclusion of this work, but rather validate our initial approach.

5.1.5 Reflection

After analysing the general results of this first cycle we can observe that the participants perceive the Sifu-Platform as a fun and useful way to learn about Java security awareness. The positive results gathered also validate our proposed framework and overall though-of approach to analysing the players submitted code.

We believe that with the creation of more Java challenges the overall experience will be enhanced. With the existence of more challenges the players will have more contact with security issues, improving their overall security awareness. Furthermore, the development of more Java programmable challenges is required in order to fully validate the hints and hint-system that will be developed in the upcoming cycles. Therefore, the creation of new Java Challenges will be the main focus of the upcoming ADR - cycle.

5.2 Cycle Two - Challenge Development

The development of the artifact in this cycle was guided by the findings and conclusions that arose from the previous cycle of development. The main goal of this cycle was to improve the backend analysis process as well as develop more Java challenges.

Since the present research was conducted as part of an awareness campaign embedded in an industry context, additional challenges had to be developed to encompass a wider variety of possible situations. The development of additional Java challenges, aligned with the organization' teaching curriculum, allowed our artifact to be embedded in internal software developer trainings. A total of three new challenges were developed using the same methodology as in the first design cycle. The backend analysis process was improved in the present cycle. The improvements in the backend were made in regards to the tools used in the submitted code testing stage.

The remaining of the present section elaborates into both these developments and reports the findings collected in the interventions performed in the second design cycle.

5.2.1 Problem Formulation

This cycle deals with two main problems: 1) the need for additional challenges, and 2) improving the backend analysis framework.

Firstly, we selected other security issues from the SEI-CERT coding standards [65], that could be used as reference for new challenges. We selected the challenges with the intent to create a balanced teaching curriculum. Our decision to chose the guidelines on which to base our challenges on was influenced by previous work conducted by Gasiba et al. [66].

Secondly, we needed to understand if the analysis framework developed in the previous cycle can be applied to these newly selected challenges. The undergone changes to the analysis process were mainly focused on the subset of SAST tools employed.

5.2.2 Building

The security issues present in the Java Challenges, were based on the SEI-CERT Coding Standards for Java. We started this development cycle by creating a web scrapping application that would allow us to export the full list of vulnerabilities to an excel document. This document allowed us to quickly sort the vulnerabilities listed in the coding standards. This process was fundamental to select the issues that would serve as basis for the new challenges. Each SEI-CERT vulnerability represents a cybersecurity issue, and is mapped according to three attributed values - Severity (S), Likelihood (L) and Remediation Cost (RC). Each of these values have a three point scale that matches the cybersecurity issue, e.g. *SEI-CERT IDS01-J* : S - HIGH, L - Probable, RC - Medium.

Each issue also has a priority that a result of the multiplication of the three previous values, higher the priority, worst the issue, e.g. *SEI-CERT IDS01-J* P - 12.

Issues with the highest priorities tend to be the hardest to solve. The created challenges were based on a set of issues selected due to the distinct priority levels in order to give the player a wider experience of difficulties. By the end of this development cycle we had a total of four different Java coding challenges with priorities that ranged from 4 to 27.

5.2.2.1 Challenge Development

Table 5.3 contains the name of each chosen challenge, the corresponding SEI-CERT issue and priority as well as the difficulty in terms of fixing the issue. The final column of the table represents the internally mapped CWE chosen for each issue.

Table 5.3: Developed Java Challenges

Code Name	SEI-CERT	Priority	Difficulty	Mapped CWE
Resource Leakage	FIO04-J	4	Easy	404
File Creation	FIO50-J	8	Medium	367
String Normalization	IDS01-J	12	Medium	180
Privileged Blocks	SEC01-J/FIO00-J	27	Hard	266

Apart from the already specified *Resource Leakage* challenge three more challenges were developed in this cycle: *File Creation*, *String Normalization* and *Privileged Blocks*. Each challenge focuses on different security vulnerabilities that developers can encounter when creating software. The chosen challenges were mapped to a difficulty level. At the end of this cycle the

platform had one Easy (Resource Leakage), two Medium (File Creation & String Normalization) and one Hard (Privilege Block). Each new challenge was developed to raise awareness to at least one security issue.

In the *File Creation* challenge, players are faced with a potential Time-of-check, Time-of-Use (TOCTOU) vulnerability when creating files. The players must correct the code to safely create a file without the possibility of a TOCTOU based attack. This issue can be exploited by attackers that will force the system to open a different file than the one created. This could lead to severe potential damages in the users system such as privilege escalation.

The *String Normalization* challenge focuses on the treatment of user controlled strings. Every system that accepts untrusted user input should normalize it before validating it for security flags. In this challenge, players are given a system that does not perform this step and so it is vulnerable to attacks. If the input strings are not normalized, or normalized in the wrong way, tags such as <script>avoid detection and run. This is very problematic and can once again damage a system and results in information disclosure.

Finally, the *Privileged Blocks* challenge refers to the possibility of using untrusted variables or data in privileged blocks of code. This issue relates do platform security and occurs when unsanitized data is able to run with system privileges beyond the ones intended. This situation can potentially lead to path traversal attacks and cause major damages and breaches in the system. To solve this issue, players must ensure the variable, in this case a filename, is within the users directory.

5.2.2.2 Backend Improvements

After choosing these three issues, we followed the developed method for the initial Java Challenge and applied it to these three exercises. When applying the defined subset of tools, described in table 4.2, to the analysis of the new challenges we noticed a surprising result. When applying these tools to the analysis of the first challenge no issues regarding the time of analysis was detected. However, by increasing the complexity (amount) of code analysed, some tools could take an undesirable amount of time. Some analysis tools (PMD and SonarQube) were experiencing analysis times of over 10 seconds. The goal of the platform is to provide the player an interactive IDE type of coding environment with instant feedback. Therefore, this type of analysis times were not acceptable to be employed. To mitigate this timing issue we were able to disable unnecessary analysis performed by most tools. Tools like PMD had several types of analysis besides the security aspect. These analysis, ranging from code best practices and naming conventions, among others, were deemed unnecessary. This conclusion is derived from the present research main goal, to raise cybersecurity awareness. Since the main goal of this research is aimed at cybersecurity, those types of analysis were suppressed. By disabling these secondary analysis and leaving the fundamental one, we lowered the running time of most analysis tools to acceptable intervals, i.e, $<$2 seconds. Although most of the tools responded well to this change, one previously employed tool did not make sufficient improvements. In particular, the SonarQube analysis tool remained with analysis times well over the acceptable range ($>$30 seconds of analysis time), and so, its usage was terminated.

To ensure the maximum coverage of security issues we developed some custom rules to be applied to the challenges. These rules were created due to the fact that some vulnerabilities were not flagged by the current versions of the SAST tools. Furthermore, these tools were developed using the PMD and Spotbugs developer tools. These tools provide developers with a good documentation on how to implement new and custom rule-sets based on the tools own methods. The newly developed rule-sets, in collaboration with the SAST tools own analysis, serve to further strengthen the analysis results gathered by the backend. Furthermore, following the initial Java challenge approach, a series of security and unit tests were developed for each challenge in order to cover other potential issues with the submitted code. For each challenge

the backend implementation of the tests followed the framework defined in figure 4.3 (page 25).

5.2.3 Intervention

Table 5.4: Cycle 2 - Intervention

Date	Context	Interview Type	Participants
22 March 2021	Industry	Group	15

Once all four challenges were working according to specifications and the changes to the backend analysis process were applied, an industry focused workshop was held. The goal of this intervention was to further test the challenges in a real-live training scenario in the industry, the platform, and also the concept as a teaching method.

Table 5.4 shows the aspects of the intervention performed in this cycle. The developed challenges were included into an industry based internal security training program as part of the practical exercises section. The training was based on the *Java Cybersecurity Challenges*. Furthermore, the training was held on the 22 of March of 2021 from 9 a.m to 16 p.m. The workshop was done through an online meeting with a total of 15 people. In this test all the participants were Java developers from the industry. Lastly, before concluding the training, there was a discussion between the participants and the coaches on which aspects were positive and negative about the overall experience. In the end the participants were given the developed questionnaire, table 5.1, to give some anonymous input about the test. Of the total 15 participants, 10 chose to answer the survey.

5.2.4 Evaluation

This section presents the results collected from our second cycle where we tested the artifact containing four challenges. A critical discussion of the results is also presented in this section.

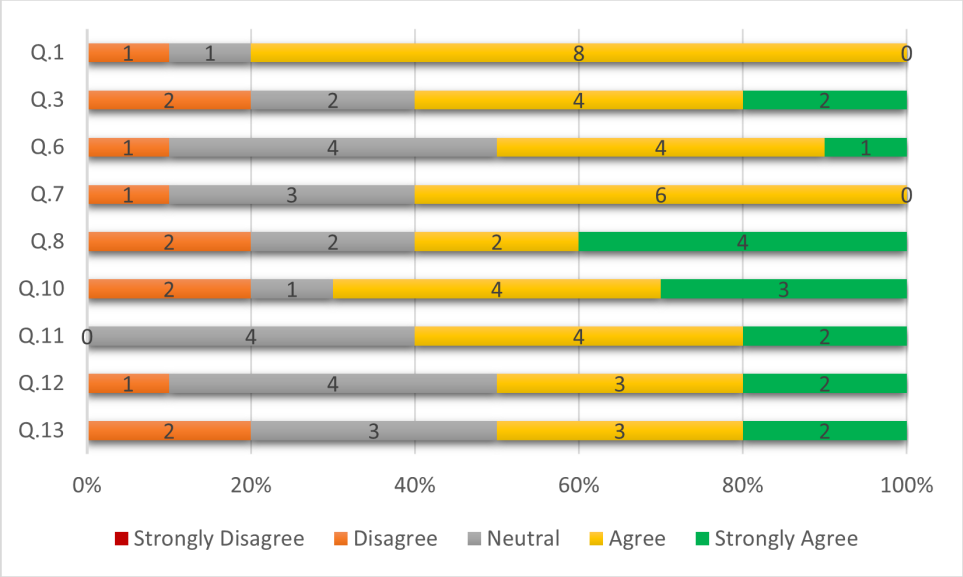
5.2.4.1 Results

The results gathered in this cycle can be analysed in two different sub-sets. The first is the sub-set of questions examined in the previous cycle Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11. These questions relate to the challenges and platform as an appropriate way to raising cybersecurity awareness. The second sub-set of questions to be analyse are the ones concerning the hints & hint-system employed in the platform. This sub-set: Q.2, Q.4, Q.5 and Q.9, give the researchers an inside on how the players perceive the hints and overall feedback as useful.

Regarding the first sub-set of questions, we can highlight that the results show a majority of positive ratings. We can see from figure 5.5 that all the questions have an agree and strongly agree majority of results. We can also pinpoint that of the non-positive results of each questions the majority was of neutral nature and very few negative results were gathered.

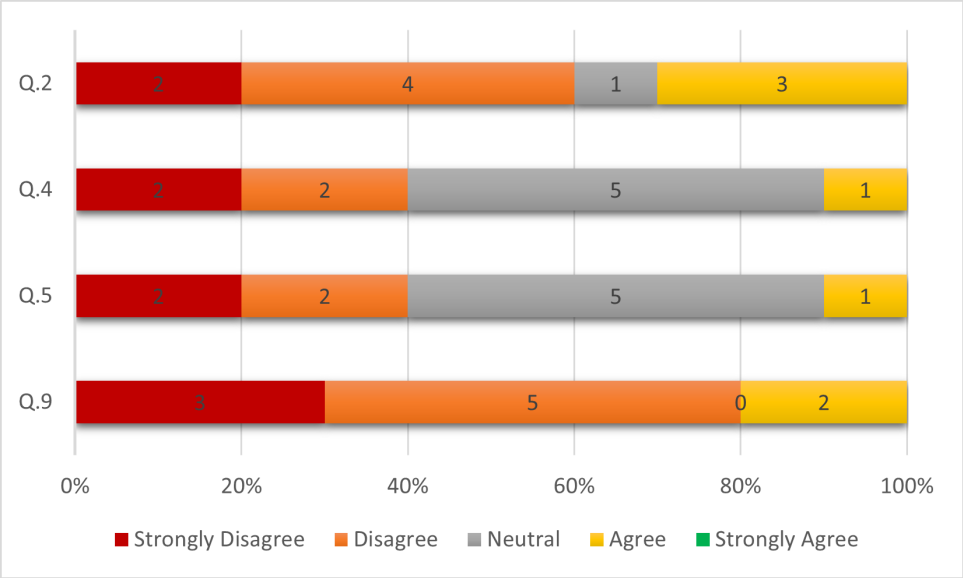
Regarding the sub-set of questions related to the hints & hints-system, Q.2, Q.4, Q.5 and Q.9, we can see that the results are of a less positive nature. Analysing figure 5.6 we can see that these questions related to the hints produced negative results. In Q.2 we see that 30% of participants agreed with the statement (4/5). Of the remaining 70% of participants, 10% gave a neutral answer (3/5), 40% disagreed (2/5), and 20% strongly disagree with the statement (1/5) . Q.4 and Q.5 generated the same results and showed that 10% of participant agreed with

Figure 5.5: Cycle Two - Results from Initial Subset of Questions



the given statement (4/5). Of the remainder 90% of participants, 50% gave a neutral answer (3/5), 20% disagreed (2/5), and the remaining 20% strongly disagreed with the statement (1/5). Lastly, Q.9 shows that 20% of participants agreed with the given statement (4/5). In this case no neutral answers were received and, of the remaining 80% of participants, 50% disagreed (2/5), and 30% strongly disagree with the given statement (1/5).

Figure 5.6: Cycle Two - Hint Related Results



On the last part of the training a feedback round was held and participants could manifest their opinion about the experience. In general, the feedback given by the participants was unanimous and some recorded statements are as follows:

"It is an interesting approach to cybersecurity learning. The hints should be a bit more worked on and specific."
"I had fun while playing, but I kept receiving generic hints that didn't really help that much"

5.2.4.2 Discussions

After analysing the results from the intervention of this cycle, we can pinpoint two major findings. First, we can extrapolate that the participants like the challenges and concept as a way to learn secure coding. The results from the first sub-set of questions, Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11, show that the participants of this cycle agree with the participants of the previous cycle in terms of the usefulness of the platform to raise cybersecurity awareness. However, comparing the results from both cycles shows a clear difference. Although the results of the present cycle remain majoritively positive they are, in fact, less positive than the ones of the previous cycle. At the time, we hypothesize that a possible positive bias could have occurred in the first cycle. When comparing the results from both cycles we can see that a positive bias has indeed occurred in the proof-of-concept cycle.

We believe that this positive bias occurred due to the nature of the interventions. The interventions performed in the first cycle were based on a more personal experience. Each one of the 11 interviews of the undergone interventions was conducted individually which could lead to more positive results. Furthermore, in the first cycle, the coaches were more involved in helping the players solving the challenges, were in interventions of this cycle, the coaches only helped in some specific cases. Both these situations could have impacted the participants of the first cycle leading to the positive bias encountered.

However, since most of the results of this cycle are still majority positive in nature, we conclude that they further validate our approach. Although we pay attention to this sub-set of questions in the upcoming cycles, we can conclude that the concept is viewed as a useful means to raise cybersecurity awareness through learning secure coding. These results are in line with the conclusions of previous researches.

Regarding the sub-set of questions related with the hints and hint-system we can highlight that the results are not so positive. Most of the results gathered are of negative nature and only very few are positive. These results were to be expected due to the fact that by the time the intervention was held, the hint-system was yet to be fully developed. The hints and hint-system that the participants of this cycle got to experience was a very initial and primitive version of the final algorithm. Since the main purpose of this cycle was to implement new java challenges the hints and hint-system were a bit overlooked. The only hints provided to the player at this moment were of a basic nature. When encountered security issues in the code the backend would firstly present the player with an overall feedback. This message would be a non-changing description, meaning that if the same error is encountered then the same feedback would be presented. The hints provided by the intelligent coach to the player were very generic in nature and only based on the security vulnerability found in the code (CWE). Ultimately, the results from both the questionnaire and the final interview of this intervention gave critical insight to the artifacts' development. Firstly it corroborated the findings of the proof-of-concept cycle. Secondly that the hints and hint-system must be further developed to aid the players in the solving of the challenges

In this cycle, all of the participants were industry software developers. However, another concern that the coaches had was the constant rushing on the programmable challenges. Each challenge contains a ReadMe.txt file that explains the goal of the challenge and a brief overview of the security issue. We noticed a constant disregard of these files by the participants across the test. This lead to more participants getting stuck and needing help from the coaches. This issue is addressed in further interventions.

5.2.5 Reflection

After analysing the answers given from the participants and the results of the development of the exercises we can extrapolate two main conclusions. Firstly is that the proposed framework of testing is applicable to other Java coding exercises with only minor changes needed to be done in particular cases. Secondly that most of the participants found the hint system lacking and needing improvement. Since the players hints given did not completely aid the players in solving the exercises this was the major focus on the next cycle of development.

5.3 Cycle Three - Hint Development

This third cycle of development aimed at addressing the concerns raised by the participants of the previous cycle. The main goal of this cycle was to perform an improvement to the hints and hint system. In the last cycle, the intelligent coach only gave the players some feedback and two distinct generic hints. In other words, each mapped CWE had a ladder of hints containing only two levels. Furthermore, the hints were of generic nature and so they were insufficient in aiding the player solve the challenges. The version of the intelligent coach and the hint-system developed in this cycle, contained more levels per ladder and so the participants had more help during the challenge. In addition, after developing the hints, we followed the path of previous cycles and performed two interventions, one in an industry context and the other in an academia setting. These interventions were held to understand how the new hints were perceived by participants. The following subsections give a detailed description on the development of the hints and the ladders.

5.3.1 Problem Formulation

This cycle of development focused on addressing one major issue, the quality of the hints provided to the participants. To be a good addition to the artifact, the intelligent coach and the hint-system must be viewed as a useful way to aid the player. The results from the previous cycle showed that the hints needed improvements in order to to be deemed useful to the participants. Therefore, the development of more hints regarding the vulnerabilities found in the challenges is the main problem this cycle addresses. Our approach to this issue is based on an approach defined in [53] by Gasiba et al. with an improvement in the overall algorithm of the intelligent coach.

5.3.2 Building

We began by developing a set of ladders containing hints to be provided to the players. Each ladder was mapped to a certain vulnerability (CWE) and initially contained between seven and nine levels. This means that for every CWE encountered by participants, the system could provide between seven to nine hints per issue. Furthermore, the developed ladders contained more generic and playful hints than specific ones. The idea was that this approach would better suit the dynamic of the platform and would be the best method to encourage participants to think for themselves. Table 5.5 shows an example the ladder created for CWE 180 [67], its levels and the nature of each hint.

Table 5.5: Initial Normalization Hints

Hint Level	Description	Nature
Level 1	CWE 180 - The software validates input before it is canonicalized, which prevents the software from detecting data that becomes invalid after the canonicalization step.	G
Level 2	Hmm, what could be wrong here, are you really going to accept that input like that?	G
Level 3	That input string sure seems treatable to me, maybe give it a try....	G
Level 4	Should you match the pattern directly with the input string, or should you perform operations on the string beforehand?	G
Level 5	I think there is a method that transforms Unicode text into the standard normalization forms, you should try and use it!	S
Level 6	Here's a hint, you should Normalize the string! This step is very important!	S
Level 7	The untrusted input is not being Normalized before the validation. Input strings given by an untrusted user must always be Normalized. The following link shows the correct Normalization of a string - How to Normalize	S

Legend - G: Generic, S: Specific

The CWE 180 is encountered when untrusted input is used without the proper normalization and sanitation. After each CWE had a ladder with hints, we performed industry based interventions to understand how participants feel about the hints provided in the challenges during the workshop.

5.3.3 Intervention

Table 5.6: Cycle Three - Interventions

Date	Context	Interview Type	Participants
21 April 2021	Industry	Group	12
3 May 2021	Academia	N/A	16

Once the developing and testing of the hints and hint-system was completed we conducted two separate interventions. Table 5.6 summarizes the relevant information regarding the performed interventions. The first one was with an industry focused group with a total of 12 participants and was inserted into internal industry based security training programs. The training was held through and online meeting and lasted from 9 a.m to 16 p.m on the 21 of April of 2021. The session had a feedback round where participants could voice their opinions about the platform, challenges and overall experience while playing. At the end of the training the participants were given the questionnaire, Table 5.1, of which eight participants chose to answer. The second test, with a total of 16 participants, was held in an academia setting, and was inserted into the practical part of an university lecture on software security awareness. The lecture was conducted through an online meeting that lasted two hours of which the last 15 minutes were reserved to the testing of the challenges. In the end the participants were asked to fill out the questionnaireand, of the 16 participants, six chose to answer. In this final intervention, a feedback round was not included.

5.3.3.1 Results

Figure 5.7: Cycle Three - Results from Hint Related Questions

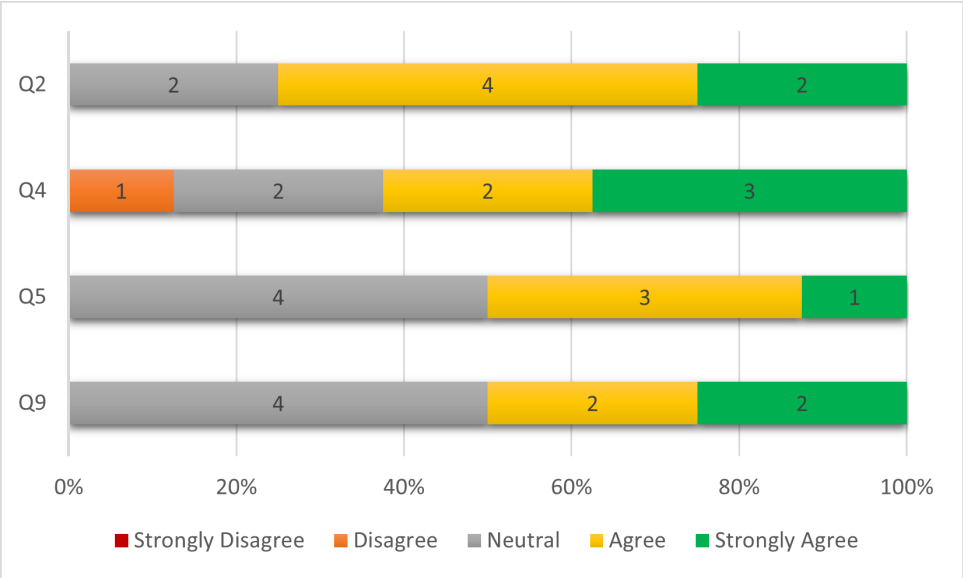
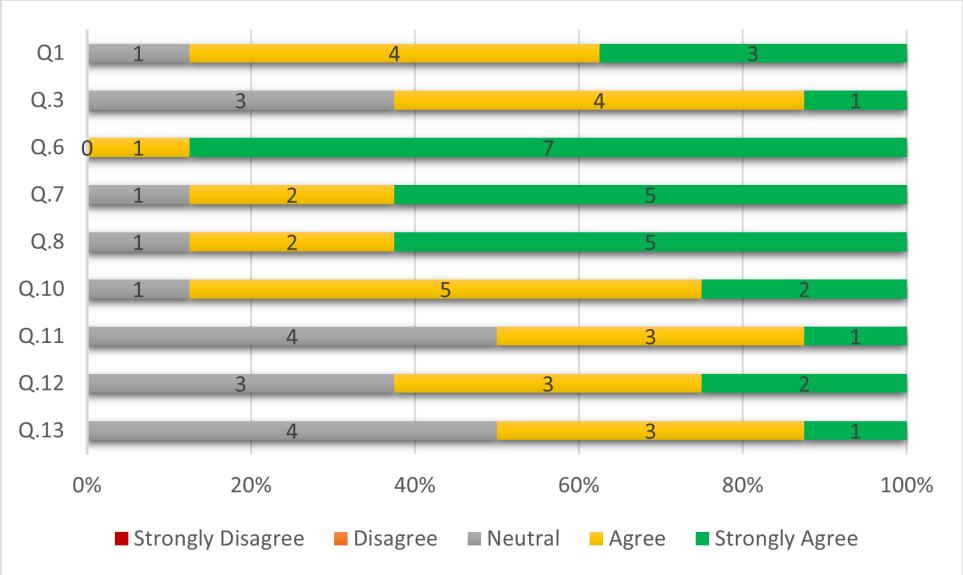


Figure 5.7 shows the results gathered referring the sub-set of questions related with the hints and hint-system, Q.2, Q.4, Q.5, Q.9. From the results of the first test we can identify some aspects that will further help the development of the artifact. We see that all these four questions had a positive majority of results. In Q.2, we see that 50% of participants agreed with the statement (4/5), 25% strongly agreed (5/5), and the remainder 25% gave neutral answers (3/5). Q.4 shows that 25% of participants agreed with the statement (4/5), 37,5% strongly agreed (5/5), 25% gave neutral answers (3/5), and the remainder 12,5% disagreed with the statement (2/5). On Q.5 we see that 37,5% of participants agreed with the statement (4/5), 12,5% strongly agreed (5/5), and the remainder, 50% of participants gave neutral answers (3/5). Lastly Q.9 shows that 25% of participants agreed with the statement (4/5), 25% strongly agreed (5/5), and the remainder 50% gave neutral answers (3/5).

Additionally, we can also analyse the results gathered from the first sub-set of questions regarding the participants view of the challenges, platform and concept as a whole. Figure 5.8 shows the data gathered from this sub-set in the present cycle.

Overall, this first sub-set of questions remains positive. In Q.1, we can see that 50% of participants agrees with the statement (4/5), 37.5% strongly agrees (5/5), and the remainder 12.5% feels neutral about the statement (3/5). In Q.3, 50% agrees with the statement (4/5), 12.5% strongly agrees (5/5), and the remainder 37.5% gave neutral answers (3/5). Q.6 showed that 12.5% agrees with the statement (4/5), and the remainder 87.5% strongly agrees (5/5). Q.7 and Q.8 gathered the same results. In both statements, 25% of participants agreed (4/5), 62.5% strongly agrees with the statement (5/5), and the remainder 12.5% provided neutral answers about the statement (3/5). In Q.10, 62.5% of participants agreed with the statement (4/5), 25% strongly agreed (5/5), and the remainder 12.5% provided neutral answers regarding the corresponding statement (3/5). In Q.11, 37.5% of participants agreed with the statement (4/5), 12.5% strongly agreed (5/5), and the remainder 50% gave neutral answers (3/5). The results form Q.12 showed that 37.5% of participants agreed with the statement (4/5), 25% strongly agreed (5/5) and the remainder 37.5% felt neutral (3/5). Finally on Q.13 37.5% agreed with the statement (4/5), 12.5% strongly agreed (5/5), and 50% of participants gave neutral answers (3/5).

Figure 5.8: Cycle Three - Results First Sub-set



On the feedback part of the test the players also gave valuable input on what the platform does good and the aspects that can be improved. The following statements are representative of the group's opinion on the whole experience.

> *"It's cool to see the platform respond and analyze the code for security issues. The feedback is also helpful. The hints could be more succinct and straight to the point."*
> *"The competitive aspect of it gives you a different motivation to understand what's wrong. The hints should be more direct, I was trying to be not lose much time and the first hints didn't really help that much."*

5.3.3.2 Discussions

One of the main goals of this cycle was to further develop the hint-system so that it could provide meaningful hints associated with the problems found in the analysis stage. Since the developed system, based on the laddering technique, was able to be implemented and employed to each of the challenges without any major difficulty, we conclude that the described method can be used as a suitable solution to answer this particular goal of the cycle. We looked at the gathered data to understand if the participants found the provided hints helpful in solving the challenge. Our results show that the hints given are both welcomed by the players, as well as useful in aiding to the completion of the challenge. However, we can pinpoint two major concerns related to these results. The first is that the results gathered are, in some cases, lower than intended. The second is that on the interview part of the test there was an almost unanimous critique: the hints need to be more straight to the point. We devised the hints to fill a spectrum ranging from more generic to more specific in nature. This was intended to stimulate the players to try and reach the solution without being given the answer directly. Although some people agreed that this was a good approach, the majority of participants pinpointed this as an improving aspect. Our next design cycle aims at addressing this encountered situation.

After gathering the results of both the interventions we noticed one concerning aspect about the workshop held in the academia setting. Although the results gathered were majoritively

positive, we chose not to include them in the present research. The decision arose from the circumstances of the intervention.

The academia participants were of the second year of a bachelor's degree on computer science. They had no previous experience with cybersecurity issues or secure coding. Furthermore, these participants only had a limited amount of time (15 minutes) to interact and evaluate the artifact. Lastly, they were not originating from an industry setting. Due to these three aspects, the gathered results from this academia intervention were not used in this research. However, these results indicate that our approach could also be applicable to an academia-based context.

5.3.4 Reflection

The results of this cycle gave us two main conclusion. Firstly, that the developed hint-system is a good approach to provide simple and meaningful hints to the players. Secondly, that the participants found the developed hints as a useful way to aid in the completion of the challenges. However, they also pinpointed that these hints, even though they are helpful, could be more succinct and straight to the point. To address this latter issue, we decided to perform a new design cycle.

5.4 Cycle Four - Final Version

This final cycle of development had one major goal: to address the issue raised in the previous design cycle. Participants from the previous cycle stated that the hints could be more precise and straight to the point. We tackled this problem by lowering the amount of generic hints given in each challenge. Participants stated that if the hints were more straight to the point from the beginning, the overall experience would be improved. We changed the ladders of each challenge accordingly with the amount of generic hints given in each one. Once the adjustments to the hints were performed, we conducted four additional interventions in the industry. The following subsections detail the development, intervention and evaluation of the final version of our artifact.

5.4.1 Problem Formulation

Based on the results gathered from the previous cycle we can underline one major problem that still needs solving. The participants stated that if the hints were more of a succinct and specific in nature from the start, the overall experience would be be improved. Therefore, the main goal of this cycle is to conduct an upgrade in the developed hints regarding the aspects raised by the participants of the intervention of the previous design cycle.

5.4.2 Building

The hints applied to each challenge have two major natures, generic (G) and specific (S). In the previously presented case of CWE 180, Table 5.5 we see that a total of seven hints (reference with levels) were given to the player in order to help solve the challenge. Based on the participants feedback and results, the platform should start providing specific hints from earlier on. There are two possible ways to correct the existing ladders.

The first approach is based on changing the generic levels to contain more specific hints, maintaining the overall number of levels per ladder. The second approach focuses on deleting the generic levels and leaving the specific ones, increasing the overall specificity of the ladder. Although we improved the content of some of the hints our approach was more focused on the second option, discard the initial generic hints and leave the more specific ones. Based on our

Table 5.7: Normalization Improved Hints

Hint Level	Description	Nature
Level 1	CWE 180 - The software validates input before it is canonicalized, which prevents the software from detecting data that becomes invalid after the canonicalization step.	G
Level 2	Should you match the pattern directly with the input string, or should you perform operations on the string beforehand?	G
Level 3	I think there is a method that transforms Unicode text into the standard normalization forms, you should try and use it!	S
Level 4	Here's a hint, you should Normalize the string! This step is very important!	S
Level 5	The untrusted input is not being Normalized before the validation. Input strings given by an untrusted user must always be Normalized. The following link shows the correct Normalization of a string - How to Normalize	S

Legend - G: Generic, S: Specific

experience, and by perceiving that the last levels of the ladder were rarely reached, we decided to focus on the second approach. Even with the initial generic hints the players would almost never be given the last hints in the ladder. This situation occurred because players reached the solution of the challenge before reaching these higher levels of the hint ladders. If we changed the initial generic hints to specific and maintained the overall number of hints, the players would solve the challenge even with less tries and the last hints would once again be rarely reached. In the case of the ladder of CWE 180 in particular, we discarded the second and third level, leaving the other five hints. Figure 5.7 shows the adapted hints for this challenge.

After the performed changes, the ladders had suffered an overall shrinking regarding the number of levels. Where the earlier version consisted of ladders containing between seven and nine levels, the new version had a maximum of six levels per ladder. In this particular example, the final ladder of CWE 180 consisted of five levels, of which two were more generic in nature and three more specific and solution oriented.

In addition, due to additional requirements from the hosting organization, we developed one more Java programmable challenge. This new challenge was based on the SEI-CERT EXP01-J rule, that references a null Dereference type of vulnerability. This challenge was mapped to CWE 476 and provides the player with a program that tries to count the occurrences of an object in a predetermined collection. However, the code can suffer from a null pointer dereference whenever the upcoming value to be checked is null. This problem can cause a crash and exit of the system. In this final version of the artifact, a total of five distinct challenges had been developed. Table 5.8 shows the full list of developed Java challenges that were available to participants in the fourth cycle of development.

Once these new improvements to the platform met our required quality we conducted more tests to understand if the new approach to designing the hints is considered good by the participants so that they perceive them as helpful.

Table 5.8: Final Java Challenges Developed

Code Name	SEI-CERT	Priority	Difficulty	Mapped CWE
Null Dereference	EXP01-J	3	Easy	476
Resource Leakage	FIO04-J	4	Easy	772
File Creation	FIO50-J	8	Medium	367
String Normalization	IDS01-J	12	Medium	180
Privileged blocks	SEC01-J/FIO00-J	27	Hard	266

5.4.3 Intervention

Table 5.9: Cycle Four - Interventions

Date	Context	Interview Type	Participants
19 May 2021	Industry	Group	13
16 June 2021	Industry	Group	17
23 June 2021	Industry	Group	18
16 July 2021	Industry	Group	15

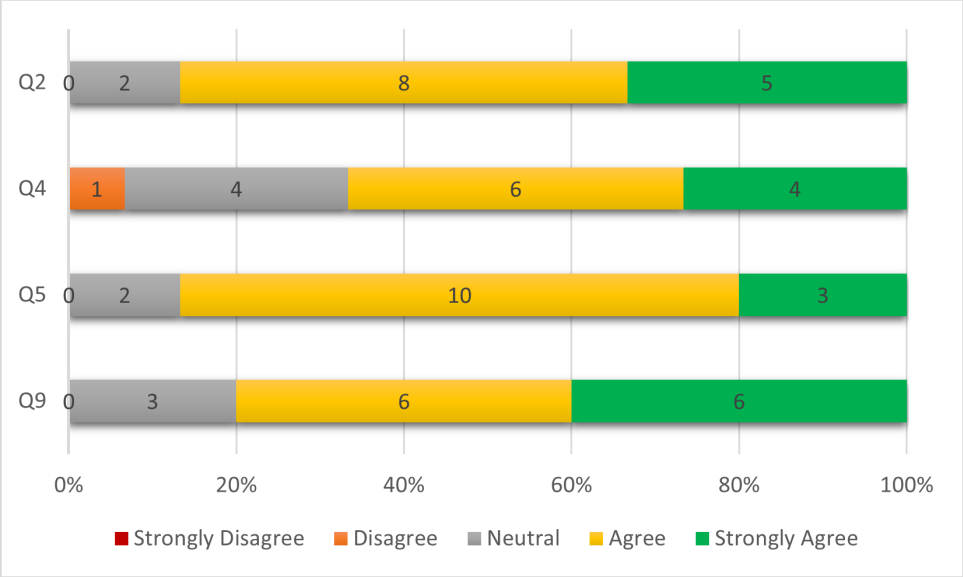
The fourth cycle of development had several distinct interventions to test the final version of the artifact. These interventions were held in an industry context. In addition, these interventions were included into an internal security training program. The interventions were conducted through a teams-based online meetings. Furthermore, at the end of each workshop, the participants could voice their opinion, in the feedback round, as well as fill out the questionnaire of 5.1. Table 5.9 summarizes the information regarding each of the conducted interventions. A total of 13 participants were present in the first intervention that was held on the 19th of May 2021. Of the 13 participants, five chose to answer the given survey. The second workshop was held on the 16th of June 2021 and had a total of 17 participants of which two chose to answer the survey. Conducted on the 23 of June 2021, this intervention counted with 18 participants. Of the 18, four chose to answer the survey. Finally, in the last intervention held on the 16th of July 2021, a total of 15 participants were present of which four chose to fill out the given survey. In the end, from the four distinct interventions we gathered a total of 15 answers.

5.4.4 Results

Figure 5.9 shows the final results regarding the sub-set of questions related to the hints and hint-system,Q.2, Q.4, Q.5, Q.9. Overall, the data gathered shows majoritively positive results. In Q.2, 53.3% of participants agreed with the statement (4/5), 33.3% strongly agreed (5/5), and the remainder 13.3% provided neutral answers (3/5). In Q.4, 40% of participants agreed with the statement (4/5), (26.7%) strongly agreed (5/5), 26.7% gave neutral answers (3/5), and the remainder 6% disagreed with the statement (2/5). In Q.5, 66.7% of participants agreed with the statement (4/5), 20% strongly agreed (5/5), and the remainder 13.3% of participants gave neutral answers regarding the statement (3/5). Lastly, in Q.9, 40% of participants agreed with the statement (4/5), 40% strongly agreed (5/5), and the remainder 20% felt neutral about the statement (3/5).

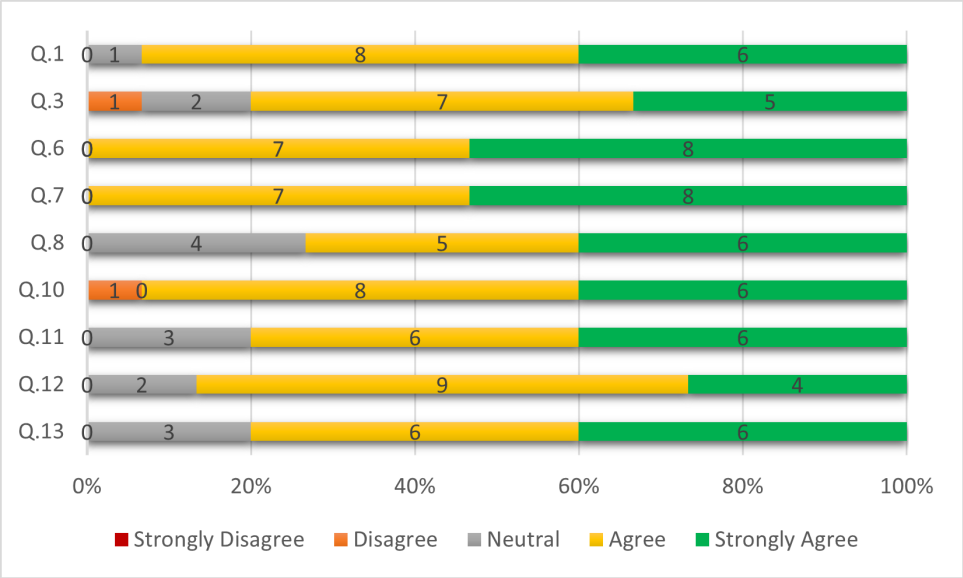
Additionally, we can analyse the results of the first sub-set of questions regarding the challenges, platform and concept as a whole, Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11. Figure 5.10

Figure 5.9: Cycle Four - Hint Related Results



shows the results related to this sub-set.

Figure 5.10: Cycle Four - Results from Initial Sub-set



Overall, this first sub-set of questions remains positive. In Q.1, we can see that 53.33% of participants agrees with the statement (4/5), 40% strongly agrees (5/5), and the remainder 6.67% feels neutral about the statement (3/5). In Q.3, 46.67% agrees with the statement (4/5), 33.33% strongly agrees (5/5), 13.33% gave neutral answers (3/5) and the remainder 6.67% disagrees with the statement (2/5). Q.6 and Q.7 gathered the same results. In both statements, 46.6% of participants agreed (4/5), and the remainder 53.33% strongly agrees with the statement (5/5). In Q.8, 33.33% of participants agreed with the statement (4/5), 40% strongly agreed (5/5), and the remainder 26.67% provided neutral answers regarding the statement (3/5). Q.10 shows that 53.33% of participants agreed with the statement (4/5), 40% strongly agreed with the statement (5/5) and the remainder 6.67% disagreed (2/5). In Q.11, both the agree (4/5) and the strongly agree (5/5) metric gathered the same results of 40%. The remainder 20% gave neutral answer about the corresponding statement (3/5). On Q.12 60% of participants agreed with the statement

(4/5), 26.7% strongly agreed (5/5), and the remainder 13.3% felt neutral (3/5). Finally, on Q.13, both the agree (4/5) and strongly agree (5/5) gathered the same results from participants 40%. The other 20% provided neutral answers regarding the statement (3/5).

Each workshop had a feedback round where participants could voice their opinions on the experience. The following statements are part of the recorded feedback.

"The exercises were very good and nice"
"Initially I thought that I could not play the game, but I was positively surprised"
"The feedback the platform gives guided me when I got stuck, very nice"

5.4.5 Discussions

The results gathered in this cycle points to one major conclusion: that the majority of the participants that interact with the Sifu-Platform find the hints and hint-system useful in aiding the player while solving the challenge. When comparing the results of this cycle with the ones of the previous one we can clearly see two additional findings. Firstly, that the players do, in-fact, prefer hints more succinct and specific. This can be observed when comparing the percentage of results within each value. Secondly, we also observed that the feedback given by the participants had also slightly improved. In the last cycle, participants told that the platform, challenges and concept were fun and appropriate, yet the hints, although good, could still be improved. In this cycle, the participants voiced their opinions by once again highlighting the quality of the concept and by stating that the hints and feedback employed helped reaching a solution and solving the issue. The main goal of this cycle was to understand how participants felt about the newly developed hints and if these were good to employ in a final version of the artifact.

5.4.6 Reflection

At the end of this cycle, and based on our results and the continuous development of the artifact, we can state that the artifact produced is viewed as a good way to raise cycbersecurity awareness. Moreover, the participants agree that our developed artifact is capable of giving meaningful feedback that aids them in solving the exercises. The major finding in this cycle is that software developers find the hint system part of the platform useful, and prefer a more direct and succinct hint system in contrast to one with more hints per issue. Lastly, we can also conclude that the two major aspects of the artifact (backend analysis process and hint generator system) compose a good way of raising cybersecurity awareness. Furthermore, both these aspects were iteratively improved until the quality of the overall artifact met both the researchers, and the hosting organization expectations. This evaluation was performed by participants that tested both these aspects of the platform and, in the end, gave positive feedback. In the four distinct interventions held in this cycle, we observed a saturation in the feedback provided by participants regarding the developed artifact. As such, in this cycle we conclude that our design has reached maturity and was accepted by both the players and industry. Therefore, no further design cycles were necessary since the developed artifact was deemed useful.

Chapter 6

Overview and Reflection on the Design

6.1 Iterative Cycles Overview

In this section an overview and comparison of the overall results gathered will be performed. Furthermore, we will also preform a reflection on our design effort as required by the ADR methodology. Due to the employed methodology each cycle of development generated its own set of results. This last section compares the results of each iterative cycle, in order to better identify the progresses made along the research, and reflects on the undertaken development. To pinpoint the evolution of the artifact we analysed the percentage points of the results gathered. The results were gathered using a questionnaire with a total of 16 questions. Of these, 13 were based on a 5-point Likert Scale, with 1 being the lowest (Strongly Disagree) and 5 being the highest (Strongly Agree). Along the research we divided the questionnaire questions into two main sub-sets, one regarding the concept and platform as a whole and the other linked to the hints and feedback given by the platform. By checking the percentage of participants that gave positive answers, either agreed (4) or strongly agreed (5) with the statements of the questionnaire, we identified the overall progress the artifact suffered along the development cycles of the research. Figure 6.1 shows the evolution of the percentage of positive results regarding each sub-set of questions across the development cycles.

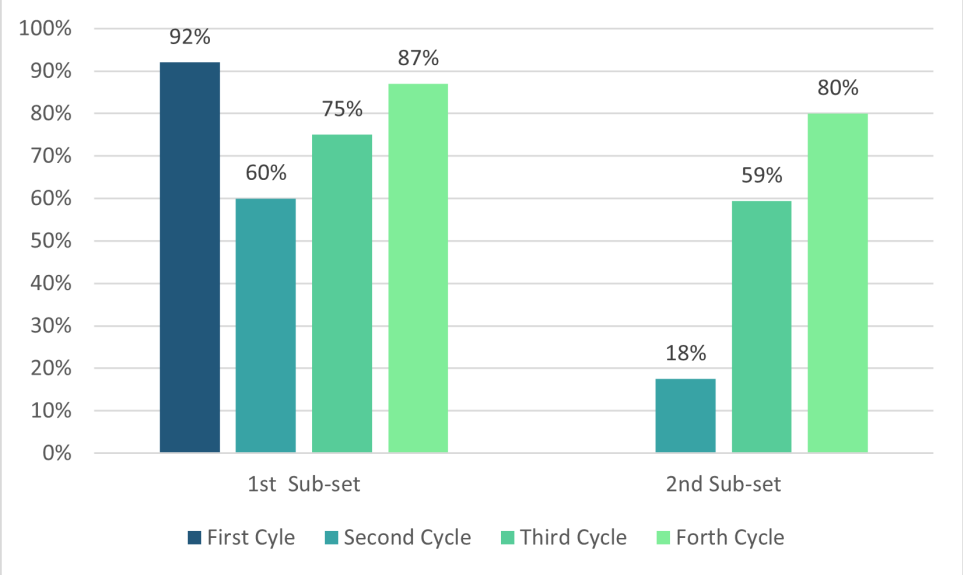


Figure 6.1: Evolution of Results

In the first cycle of development (Proof-Of-Concept), an initial version of the artifact was

developed and tested. This version had one Java challenge and lacked the capability of generating instant hints. Due to this we restricted the analysis of the results to the subset of questions directly related to the concept of the research (Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11). Since the remaining questions were linked to the hints provided, we did not take these results into consideration. This cycle of development gathered very positive results thus validating our approach. We can see that 92% of results gathered were either agreed (4) or strongly agreed (5). At the time we theorized that a possible positive bias could have occurred mainly due to one aspects, the proximity of the coaches. In this trial run the coaches provided extra help and were overall more active in the game. This was due to the fact that at the time the intelligent coach was undeveloped and the feedback the player received was very simple and rudimentary. The coaches tried to mimic the help that the intelligent coach might give in the future to give players a better sense of the concept.

The second cycle of the research was defined by a series of major developments. Firstly, we selected new rules from the SEI-CERT coding standards that would latter be used as basis for the development of more challenges. Secondly, we further improved the backend analysis process. In addition, we also performed small adjustments to the tools used in the analysis of submitted code. Lastly, we also developed an initial version of the intelligent coach. The artifact tested in this cycle contained a total of 4 challenges and an initial hint generation that still provided very simple hints to the player. In this cycle we chose to analyse both the previously defined subsets of questions. Regarding sub-set one (Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11), 60% of the results gathered were of positive nature (either agreed (4) or strongly agreed (5)). By comparing these results to the ones gathered in the previous cycle one major aspect is observable, the overall results are less positive. This 32% decrease in positive results can be explained by two major causes. Firstly, the results of the first cycle have indeed suffered from a positive bias, as suggested by our results. Secondly, in this intervention the coaches didn't participate and help as much, taking an overall more passive approach. In addition, the intelligent coach developed at the time, only gave few, simple hints. This could have lead to an overall sense of frustration from the participants since they didn't have a good helping system ainding them in solving the exercises. Although these results are less positive they remain majoritively positive and so, the approach is validated as a good, fun environment to learning secure coding.

In this cycle we also analysed the results gathered from the sub-set of questions related to the hints and hint-system (Q.2, Q.4, Q.5 and Q.9). The hint-system employed in this version of the artifact was very primitive. The development performed in this aspect of the artifact lead to some rudimentary and simple hints being provided to the players. Furthermore, the amount of hints per vulnerability was limited. Consequently, from the results gathered we observe that of the total results gathered from this sub-set, only 18% were positive in nature (either agreed (4) or strongly agreed (5)). To address this aspect, we decided to continue the research and change our focus to improving the hints and hint-system to give a better experience to the players.

The third cycle focused on further developing the hints and hints-system. It was in this cycle that the hint generation algorithm was improved and the hints were created for each one of the developed challenges. In this cycle two different interventions were held, one in an industry context and another in an academia setting. The results gathered from this latter one were discarded and were not analysed or used in this research. Although they were majoritively positive, the participants were exclusively from an academia background, and so, the results were not included in this research. The industry focused intervention however gathered interesting results. Beginning from the second sub-set we observe an improvement when comparing with the results from the second cycle. In this cycle, 59% of the results were of positive nature (either agreed (4) or strongly agreed (5)). This represents a 41% increase in positive results. The intelligent coach and hints employed in this version of the artifact were extensively more developed than the ones incorporated in the previous cycle. Although the participants found the hints received helpful, some improvement could still be made to the hints and hint-system. As

stated by the participants the provided hints could be more succinct and "straight to the point". Nevertheless, the improvement of the hint generation process performed in this cycle also improved the perception and enjoyment of the platform by participants. This can be seen by the increase in the first sub-set of questions (Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11). We observe that 75% of results are of positive nature (either agreed (4) or strongly agreed (5)). This 15% increase in positive results is linked to the improvement performed in the hints and hint-system. This conclusion arises from the fact that no modifications in the backend analysis process or challenges were performed in this version of the artifact.

We performed one additional design cycle with two major goals. Firstly, to performed the changes suggested by the participants of the third cycle regarding the provided hints. Secondly to further validate the theory that having an appropriate intelligent coach that gives meaningful and useful hints to the players increases their perception and enjoyment regarding playing in the platform. The overall provided hints suffered an improvement. We started by diminishing the amount of hints given per vulnerability. While the previous version of the artifact had a maximum of nine hints, the version of this cycle employs a maximum of six. We reached this number by eliminating some more generic hints and leaving the more specific ones. Furthermore, after this adaptation process was performed, we also adapted some hints, rephrasing them to better convey the necessary information. The results we gathered from the interventions of this cycle were very positive. Regarding the hint related sub-set of question (Q.2, Q.4, Q.5 and Q.9), 80% of results were positive in nature (either agreed (4) or strongly agreed (5)). This represents a 21% increase in positive results when comparing to the same sub-set of the third cycle.

In the development part of the cycle, one more challenge was developed. The final version of the artifact contained a total of five programmable challenges. This was the only other change performed to the artifact in this design cycle.

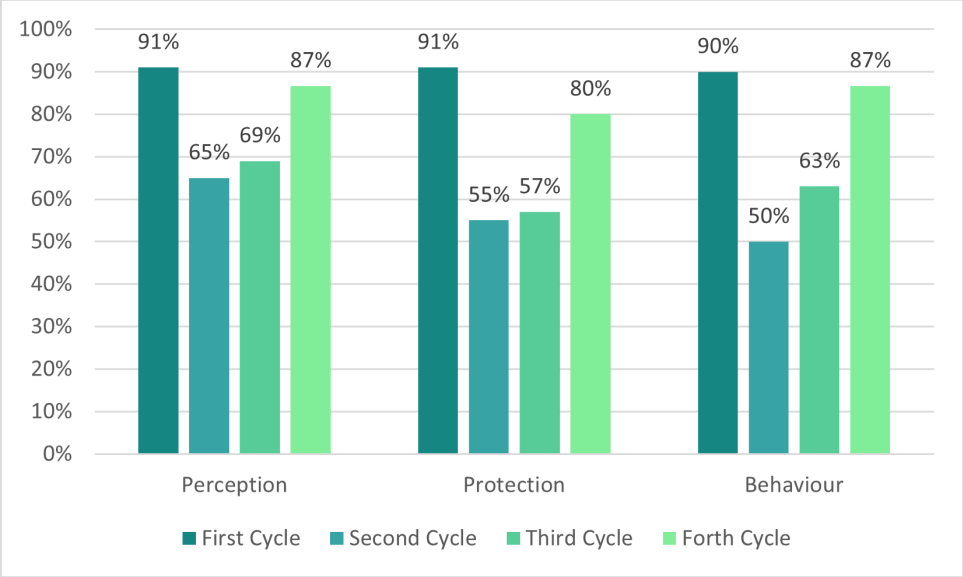
Regarding the first sub-set of questions (Q.1, Q.3, Q.6, Q.7, Q.8, Q.10 and Q.11) the results also showed positive results. Of our results 87% were of positive nature (either agreed (4) or strongly agreed (5)). This accounts for an 12% increase from the third cycle. We can state that the improvements to the hints and hint-system had an impact in the improvement of the overall results.

By analysing the results of the final version of the artifact we conclude that both the aspects of the platform (code analysis and hint generation) are viewed in a positive way by the large majority of players. To reach this final version of the artifact and answer this work's research question, we defined more concrete steps to be performed in each cycle of development. By tackling the problems defined in each cycle we created a useful solution to the encountered problem. The final version of the developed artifact consists of a serious game, embedded in the Sifu-Platform, that is viewed by players as a fun, positive and as a useful approach to raising cybersecurity awareness. This serious game consists of two major sections. Firstly, it employs the code analysis process that defines the step-by-step method used to analyse submitted code. This analysis process also contains a subset of tools that analyse the submitted code and report on the errors found. Secondly, the game also employs an intelligent coach that aims at helping the players solving the challenges. The intelligent coach designed in our work is viewed by the participants as a good and useful tool in aiding the players solving the programmable challenges. We can also see an improvement in results when comparing each version of the artifact. Apart from the first cycle, where coaches had an active role in helping the participants, all the remaining iterative cycles show an improvement in both analyzed subsets of questions.

6.2 Awareness Results

Some of the questions presented to the players in the form of the questionnaire of table 5.1, were mapped to one awareness aspects defined by Gasiba et al. [30]. We can acknowledge the

Figure 6.2: Evolution of the Awareness Related Results



usefulness of the developed artifact in terms of the observed awareness increase by analysing the overall results of these particular questions. In total, of the 16 questions asked, five were mapped to a particular aspect of the awareness theory. Q.3 and Q.13 were mapped to the protection dimension. Q.12 relates to the behavioral vector and lastly, Q.10 and Q.11 represent the perception dimension. Figure 6.2 shows the evolution of the results gathered by these five questions.

Following the overall results gathered, this awareness sub-set of questions shows a similar increasing behaviour in results. Each development performed in the artifact led to the increase of positive results, regarding these questions and, consequentially, the increase in software developers awareness. In the *Perception* and *Behaviour* dimensions the final positive results were of 87%. The results regarding the *Protection* aspect of the platform were of 80% . Combining the results of the three aspects of awareness, we observe that 84% of the results gathered by this sub-set were of positive nature. This further confirms that the developed serious is a useful way to raise Java cybersecurity awareness of software developers in the industry. We, therefore, conclude on the success of our research.

Chapter 7

Conclusion

Today's world is heavily dependent on technology services and software. With the overall increase in digitalization it is only normal that the usage of software products has also seen a large increase. However, the number of reported cybersecurity incidents has also been at rise. This fact indicates a possible problem with the quality developed software. Cybersecurity is, nowadays, gaining more and more attention in the software programming field. To mitigate potential security issues and meet the requirements of the industry, companies have developed several potential ways to deal with this issue.

In this research we focus on the human factor of software development. By increasing the awareness of software developers to this problem, we can tackle the root of the issue itself and ultimately improve the security aspect of developed software. Our approach to raise Java cybersecurity awareness is based on both the concepts of serious games and Capture-the-Flag events. This work extends previous work on Cybersecurity Challenges to the Java programming language. In this research we present the design of the *Java Cybersecurity Challenges*. These challenges are embedded in the Sifu-Platform and represent a type of programmable exercises that players can actively interact with. Each exercise contains at least one security vulnerability. To solve these challenges, the players must mitigate the security issues by changing the given code, fixing the present vulnerability and making it secure. The platform employs an intelligent coach that aids the players in solving the exercises. Towards this solution, we developed a method to automatically perform a security assessment of a players Java code (based on code analysis) and generate meaningful hints to the players. The algorithm in charge of creating the hints intends to mimic the behaviour of a real life coach by providing the players with concrete feedback about the encountered problems in their code.

This research was conducted using a four cycled approach based on the Action Design Research methodology as defined by [22]. Each cycle of development aims at solving encountered problems in an iterative way. We started this research by developing and testing an initial version of the artifact that served as a proof-of-concept. The positive results gathered from this initial version confirmed the potential of our approach and gave us motivation to continue the research. The second cycle of development focused on improving the code analysis aspect of the artifact and developing more challenges for the platform. Both these angles served to improve the overall experience of the players. With several custom built challenges employed in the Sifu-Platform, we changed our focus to the hint generation algorithm. This topic was the focal point of the third cycle of development which generated hints for each vulnerability encountered in the previously developed challenges. The final cycle of development focused on improving the hint generation algorithm.

By defining smaller problems and solving them in each cycle we developed and tested an artifact that was capable of answering the initially defined research question. Our artifact encompasses all the development and improvements performed along this ADR-based research. Analysing the data gathered from each cycle we have shown improvements in the results gath-

ered. With each refinement of the artifact, the players perspective of the whole experience also improved. The overall results gathered in this research are in line with previous work on the subject, which further validates our approach. The final developed artifact was deemed useful and was accepted by the industry. The produced artifact is now part of the official software developer training curriculum of Siemens.

Initially we defined a research question that would be the focus point of the present research: *How to adapt CyberSecurity Challenges to create a useful serious game to raise Java Cybersecurity awareness of industrial software developers?* The positive results we gathered on the players perspective of the game, in addition to the awareness mapped results, leads us to conclude that the approach presented in this work is a useful way to create a serious game that raises Java cybersecurity awareness of software developers in the industry. This work contributes with a step-by-step explanation of the development of Java challenges to aid researchers and industry practitioners who want to develop a serious game with the intent to raise cybersecurity awareness on the Java programming language.

By completing this research we open opportunities to develop further work regarding the specific topic. Since this research was focused on an industry context, an academia research could also be performed in order to understand if the proposed approach is viable in different contexts. Furthermore, we also hypothesize that the same method can be used to raise awareness to other coding issues (e.g., code-smells, naming conventions, code format), which also presents a possibility for future research opportunities. Lastly, some future research regarding other programming languages could be performed to further broaden the cybersecurity training present in the Sifu-Platform.

References

[1] ABS Group, "The rise of industrial cyber attacks," Accessed February 2021. [Online]. Available: https://www.abs-group.com/Knowledge-Center/Insights/The-Rise-of-Industrial-Cyber-Attacks-/

[2] K. Beckers and S. Pape, "A Serious Game for Eliciting Social Engineering Security Requirements," *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference, RE 2016*, pp. 16–25, 2016.

[3] PurpleSec, "2020 Cyber Security Statistics: The Ultimate List Of Stats, Data & Trends," Accessed 2021. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/

[4] Department of Homeland Security, "Cybersecurity and Critical Infrastructure," Accessed November 2020. [Online]. Available: https://www.dhs.gov/coronavirus/cybersecurity-and-critical-infrastructure

[5] Department of Homeland Security, "Department of Homeland Security, US-CERT. Software Assurance," Online, Accessed 12 January 2021. [Online]. Available: https://us-cert.cisa.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf

[6] World Health Organization, "Coronavirus disease (covid-19) pandemic," Accessed, November 2020. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019

[7] Center for Strategic and International Studies, "Significant Cyber Incidents," Online, Accessed 12 January 2021. [Online]. Available: https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents

[8] Departmentof Homeland Security, US-CERT, "Software Assurance," Accessed November 2020. [Online]. Available: https://tinyurl.com/y6pr9v42

[9] Columbia Broadcasting System (CBS) News, "Wannacry" ransomware attack losses could reach $4 billion," Accessed November 2020. [Online]. Available: https://www.cbsnews.com/news/wannacry-ransomware-attacks-wannacry-virus-losses/

[10] T. Espinha Gasiba, K. Beckers, S. Suppan, and F. Rezabek, "On the requirements for serious games geared towards software developers in the industry," *Proceedings of the IEEE International Conference on Requirements Engineering*, vol. 2019-September, pp. 286–296, 2019.

[11] Suri Patel, "2019 Global Developer Report: DevSecOps Finds Security Roadblocks Divide Teams," Accessed December 2020. [Online]. Available: https://tinyurl.com/3z57t32d

[12] T. Espinha Gasiba, U. Lechner, M. Pinto-Albuquerque, and D. Mendez, "Is Secure Coding Education in the Industry Needed? An Investigation Through a Large Scale Survey," *43rd International Conference on Software Engineering*, pp. 1–12, 5 2021, . [Online]. Available: https://arxiv.org/abs/2102.05343

[13] ISO/IEC 2500n – Quality Management Division, "Iso 25000 series of standards," Accessed February 2021. [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards

[14] Siemens, "Siemens AG: Charter of Trust," Accessed 24 February 2021. [Online]. Available: https://www.charteroftrust.com/

[15] Software Engineering Institute, Carnegie Mellon, "SEI CERT Oracle Coding Standard for Java," Accessed November 2020. [Online]. Available: https://tinyurl.com/ypm4mnj8

[16] T. Espinha Gasiba, U. Lechner, and M. Pinto-Albuquerque, "Sifu - a cybersecurity awareness platform with challenge assessment and intelligent coach ," in *Special Issue of Cyber-Physical System Security of the Cybersecurity Journal*, vol. 3, no. 1. SpringerOpen, 2020, pp. 1–31.

[17] Yasin Affan, Liu Lin, Li Tong, Fatima Rubia, Jianmin, Wang, "Improving software security awareness using a serious game," in *IET Software*, vol. 13, no. 2, 2019, pp. 159–169.

[18] Dörner Ralf, Göbel Stefan, Effelsberg Wolfgang, Wiemeyer Josef, "Foundations, Concepts and Practice," in *Serious Games*, 2016, pp. 421– 475.

[19] T. E. Gasiba, U. Lechner, and M. Pinto-Albuquerque, "Cybersecurity challenges for software developer awareness training in industrial environments," 2021.

[20] Oracle, "Java programming language," Accessed October 2020. [Online]. Available: https://www.java.com/en/

[21] Alison DeNisco Rayome, "The 3 Least Secure Programming Languages," Accessed February 2021. [Online]. Available: https://www.techrepublic.com/article/the-3-least-secure-programming-languages/

[22] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, "Action design research," *MIS Quarterly*, vol. 35, no. 1, pp. 37–56, 2011. [Online]. Available: http://www.jstor.org/stable/23043488

[23] Tim Rietzand and Alexander Maedche, "LadderBot: A Requirements Self-Elicitation System," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 357–362.

[24] L. A. Casqueiro, T. Espinha Gasiba, M. Pinto-Albuquerque, and U. Lechner, "Automated Java Challenges' Security Assessment for Training in Industry - Preliminary Results," in *Second International Computer Programming Education Conference (ICPEC 2021)*, ser. Open Access Series in Informatics (OASIcs), P. R. Henriques, F. Portela, R. Queirós, and A. Simões, Eds., vol. 91. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 10:1–10:11. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2021/14226

[25] I. 2021, "2nd International Computer Programming Education Conference," May 2021. [Online]. Available: https://www.icpeconf.org

[26] University of Oxford, "Oxford reference - methodology," Accessed March 2021. [Online]. Available: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100153801

[27] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009, special Section - Most Cited Articles in 2002 and Regular Research Papers. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584908001390

[28] Catrien Termeer, Art Dewulf and Robbert Biesbroek, "A critical assessment of the wicked problem concept: relevance and usefulness for policy science and practice," *Policy and Society*, vol. 38, no. 2, pp. 167–179, 2019. [Online]. Available: https://doi.org/10.1080/14494035.2019.1617971

[29] Norman Hansch and Zinaida Benenson, "Specifying IT Security Awareness," in *25th International Workshop on Database and Expert Systems Applications, Munich, Germany*, 09 2014, pp. 326–330.

[30] T. Espinha Gasiba, U. Lechner, M. Pinto-Albuquerque, and D. M. Fernandez, "Awareness of Secure Coding Guidelines in the Industry - A First Data Analysis," *TrustCom 2020: International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1–8, 12 2020, IEEE, Guangzhou, China. [Online]. Available: https://arxiv.org/abs/2101.02085

[31] Fausto Spoto, Elisa Burato, Michael Ernst, Pietro Ferrara, Alberto Lovato, Damiano Macedonio and Ciprian Spiridon, "Static identification of injection attacks in Java," *ACM Transactions on Programming Languages and Systems*, vol. 41, no. 3, 2019.

[32] K. Goseva-Popstojanova and A. Perhinschi, "On the capability of static code analysis to detect security vulnerabilities," *Information and Software Technology*, vol. 68, pp. 18–33, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2015.08.002

[33] J. Li, "Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST)," *Annals of Emerging Technologies in Computing*, vol. 4, no. 3, pp. 1–8, 2020.

[34] A. Calderón, M. Ruiz, and R. V. O'Connor, "A multivocal literature review on serious games for software process standards education," *Computer Standards and Interfaces*, vol. 57, pp. 36–48, 2018.

[35] S. Ros, S. González, A. Robles, L. Tobarra, A. Caminero, and J. Cano, "Analyzing students' self-perception of success and learning effectiveness using gamification in an online cybersecurity course," *IEEE Access*, vol. 8, pp. 97 718–97 728, 2020.

[36] T. Espinha Gasiba, U. Lechner, M. Pinto-Albuquerque, and A. Zouitni, "Design of secure coding challenges for cybersecurity education in the industry," *Communications in Computer and Information Science*, vol. 1266 CCIS, pp. 223–237, 2020.

[37] T. Chothia and C. Novakovic, "An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education," *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education, 3GSE 2015*, 2015.

[38] M. Milne-Ives, C. de Cock, E. Lim, M. H. Shehadeh, N. de Pennington, G. Mole, E. Normando, and E. Meinert, "The Effectiveness of Artificial Intelligence Conversational Agents in Health Care: Systematic Review," *Journal of Medical Internet Research*, vol. 22, no. 10, 2020.

[39] S. M. Nguyen, P. Tanguy, and O. Remy-Neris, "Computational architecture of a robot coach for physical exercises in kinaesthetic rehabilitation," *25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016*, pp. 1138–1143, 2016.

[40] A. Tironi, R. Mainetti, M. Pezzera, and N. A. Borghese, "An Empathic Virtual Caregiver for assistance in exer-game-based rehabilitation therapies," *2019 IEEE 7th International Conference on Serious Games and Applications for Health, SeGAH 2019*, 2019.

[41] R. Rodrigues, R. Silva, R. Pereira, and C. Martinho, "Interactive Empathic Virtual Coaches Based on the Social Regulatory Cycle," *2019 8th International Conference on Affective Computing and Intelligent Interaction, ACII 2019*, pp. 69–75, 2019.

[42] J. Allen, "Why is security a software issue?" *EDPACS*, vol. 36, no. 1, pp. 1–13, 2007. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/07366980701500734

[43] S. Chung, L. Hansel, Y. Bai, E. Moore, C. Taylor, M. Crosby, R. Heller, V. Popovsky, and B. Endicott-Popovsky, "What approaches work best for teaching secure coding practices," in *Proceedings of the 2014 HUIC Education and STEM Conference*, 2014.

[44] H. S. Freehills, "Are you keeping pace with cyber security rules and regulations?" Accessed July 2021. [Online]. Available: https://www.herbertsmithfreehills.com/latest-thinking/are-you-keeping-pace-with-cyber-security-rules-and-regulations

[45] OWASP Organization, "OWASP Top Ten Web Application Security Risks," Online, Accessed January 2021. [Online]. Available: https://owasp.org/www-project-top-ten/

[46] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," *ACM International Conference Proceeding Series*, 2014.

[47] P. Gestwicki and K. Stumbaugh, "Observations and opportunities in cybersecurity education game design," *Proceedings of CGAMES 2015 USA - 20th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational and Serious Games*, pp. 131–137, 2015.

[48] M. Albahar, "Cyber Attacks and Terrorism: A Twenty-First Century Conundrum," *Science and Engineering Ethics*, 8 2019. [Online]. Available: https://doi.org/10.1007/s11948-016-9864-0

[49] S. Jariwala, M. Champion, P. Rajivan, and N. J. Cooke, "Influence of team communication and coordination on the performance of teams at the iCTF competition," pp. 458–462, 2012.

[50] Shellphish, "iCTF - the International Capture The Flag Competition," Accessed 12 January 2021. [Online]. Available: https://shellphish.net/ictf/

[51] J. Mirkovic, M. Dark, W. Du, G. Vigna, and T. Denning, "Evaluating Cybersecurity Education Interventions: Three Case Studies," *IEEE Security and Privacy*, vol. 13, no. 3, pp. 63–69, 2015.

[52] S. Hart, A. Margheri, F. Paci, and V. Sassone, "Riskio: A Serious Game for Cyber Security Awareness and Education," *Computers and Security*, vol. 95, 2020.

[53] T. Espinha Gasiba, U. Lechner, M. Pinto-Albuquerque, and A. Porwal, "Cybersecurity Awareness Platform with Virtual Coach and Automated Challenge Assessment," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12501 LNCS, pp. 67–83, 2020.

[54] Kent Beck, Erich Gamma, David Saff and Kris Vasudevan, "JUnit5," Accessed March 2021. [Online]. Available: https://junit.org/junit5/

[55] Bill Pugh and David Hovemeyer, "Spotbugs," Accessed March 2021. [Online]. Available: https://spotbugs.github.io/

[56] David Dixon Peugh, David Craine and Tom Copeland , "PMD Source Code Analyzer," Accessed March 2021. [Online]. Available: https://pmd.github.io/

[57] SonarSource, "SonarQube," Accessed March 2021. [Online]. Available: https://www.sonarqube.org/

[58] Facebook, "FB-Infer," Accessed March 2021. [Online]. Available: https://fbinfer.com/

[59] Douglas Crockford, "JSON - JavaScript Object Notation," Accessed April 2021. [Online]. Available: https://www.json.org/json-en.html

[60] MITRE Corporation, "Common Weakness Enumeration," Accessed 4 November 2020. [Online]. Available: https://cwe.mitre.org/

[61] MITRE Corporation, "Common Weakness Enumeration - 772," Accessed November 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/772.html

[62] Wikipedia, "YAML," Accessed January 2021. [Online]. Available: https://en.wikipedia.org/wiki/YAML

[63] Carnegie Mellon University - Software Engineering Institute, "SEI CERT Oracle Coding Standard for Java - FIO04-J: Release resources when they are no longer needed," Accessed November 2020. [Online]. Available: https://wiki.sei.cmu.edu/confluence/display/java/FIO04-J.+Release+resources+when+they+are+no+longer+needed

[64] R. Likert, "A technique for the measurement of attitudes / by rensis likert." *Archives of psychology ; no. 140*, 1985 - 1932.

[65] Carnegie Mellon University, "Secure Coding Standards," https://wiki.sei.cmu.edu/confluence/display/seccode, 2019, [Online; accessed 19-March-2019].

[66] T. Espinha Gasiba, U. Lechner, J. Cuellar, and A. Zouitni, "Ranking secure coding guidelines for software developer awareness training in the industry," in *First International Computer Programming Education Conference (ICPEC 2021)*, R. Queirós, F. Portela, M. Pinto, and A. Simões, Eds. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2020/12298/pdf/OASIcs-ICPEC-2020-11.pdf

[67] MITRE Corporation, "Common Weakness Enumeration - 180," Accessed November 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/180.html