Aligning a Serious Game, Secure Programming and CyBOK-Linked Learning Outcomes

Citation for published version:

Digital Object Identifier (DOI):
10.1109/EuroSPW55150.2022.00058

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Peer reviewed version

Published In:
2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)

General rights
Copyright for the publications made accessible via Heriot-Watt Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
Heriot-Watt University has made every reasonable effort to ensure that the content in Heriot-Watt Research Portal complies with UK legislation. If you believe that the public display of this file breaches copyright please contact open.access@hw.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Aligning a Serious Game, Secure Programming and CyBOK-Linked Learning Outcomes

Léon McGregor, Sheung Chi Chan, Szymon Wlodarczyk, Manuel Maarek
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh, United Kingdom
{LM356,Sheung_Chi.Chan,SW2029,M.Maarek}@hw.ac.uk

Abstract—The increased need for cybersecurity professionals and the necessity to raise the general security awareness of software developers underlines the importance of exploring new approaches for secure programming education. In this paper, we present the Citadel Programming Lab which comprises a GitLab instance for simulated secure programming tasks and a tower defence game. Whithin the lab we integrated guidance linked to CyBOK, a knowledgebase which organises and presents cybersecurity topics, to introduce learners to fundamental security and secure programming concepts. We discuss the design approach of using the game as a motivation to engage with the materials but also as the vehicle for introducing key cybersecurity concepts to assist students in tackling the secure programming tasks embedded in the lab. We present the result of a focus group discussion evaluation of the approach which confirms the pertinence of the CyBOK linkage and the suitability of the serious game to support the lab’s progressive introduction to complex cybersecurity topics.

Index Terms—secure programming, serious game, virtual lab, CyBOK, learning outcomes

1. Introduction

Cybersecurity has become a fundamental topic with the ubiquity of our computer-based infrastructures. The shortage of trained experts in cybersecurity and the limited awareness of cybersecurity in computing and wider cohorts of graduates, has led to the development of cybersecurity curriculum, programmes, courses and certification schemes. These developments are taking place in parallel with efforts, such as those by the Cyber Body Of Knowledge (CyBOK), to “codify the foundational and generally recognised knowledge on cyber security” [1]. As such, CyBOK’s knowledge base provides a grounding for relating resources and materials for cybersecurity teaching.

In a general undergraduate study of computer science, one obstacle in learning security concepts are the absence of scenarios to understand how a sounded attack is working on specific vulnerabilities, and thus how applying additional security considerations could help to defend against possible attacks. There is a steep learning curve from transforming book knowledge into real practical defence concepts to eventually build a critical security mindset.

One solution is to provide examples through real hands-on exercises for the students to experience security attacks and defences for themselves. The hands-on approach to cybersecurity education, as exemplified by the SEED labs [2], [3], consists in providing virtual machines and practical exercises with learning materials such as slides, lab sheets or videos. These materials are useful for allowing students to individually explore different security concepts. They specifically target to fix real scenarios but do not necessarily help to understand the cybersecurity context or to understand or visualise the vulnerability/attack impacts. In most cases, the attack and defence in a real-world context are not obvious or directly visible as many happen at the system level without observable changes at the user interface level. This could decrease the incentive and interest in understanding those hidden details. Attacks with obvious results, like “Web page hijacking”, tend to raise more general public interest than simply “altering database records” or “theft of personal credentials”.

The approach we explore and present in this paper is to couple hands-on learning with a serious game. We build a secure programming virtual computer lab to fulfil the coupling and bring a gentler learning pathway to complex cybersecurity concepts. The serious game approach balances reality, meaning and play [4] to harvest the motivational aspect of gaming for a learning purpose. The approach goes beyond a simple gamification of a cybersecurity practice or training as the game design is influenced by the context. In that sense, our serious game approach implements the vision described in [5] of going beyond points and badges in blending games and programming environments.

The serious game we co-designed [6] is a tower defence game which embodies cybersecurity processes to help developers consider security during software development. The game and its coding platform were developed [7] as part of a project on Developer Centred Security (DCS) which is concerned with focusing on the developer to generally improve software security. The base story-line of the game sees a bank (vulnerable asset) under threat by a continuous flow of potentially adversarial vehicles (attackers) that the player needs to protect using different kinds of defensive towers (defenders). The vehicles, towers and user controls relate to software cybersecurity concepts. They specifically target to fix real scenarios allowing students to individually explore different security concepts. They specifically target to fix real scenarios allowing students to individually explore different security concepts. They specifically target to fix real scenarios allowing students to individually explore different security concepts.
metaphors within the game. We derived CyBOK-linked learning outcomes which we aligned with the secure coding exercises of the lab and the in-game explanations and metaphors. The coupling of the coding tasks and gameplay in the lab provides both a context/embedding of the tasks with the game, and a playful motivation and meaningful introduction to take up the tasks. The educational lab uses the game as a tangible attack and defence scenario on a vulnerable asset, and embeds real defensive coding to activate gamified defence within the game. The aim is to make the players gain awareness and knowledge of secure programming techniques throughout the playful process.

To evaluate the educational design of the alignment of the serious game, the hands-on secure programming with CyBOK-linked learning outcomes, we ran a focus group discussion. At this stage, we have not yet evaluated the use of the platform in the context of a course. The focus group discussion focused on alignment. Although the sample of participants was small, the discussion highlighted the meaningfulness of the linkage and how the platform could be extended to cover further mobile security topics, and the suitability of the serious game to support the lab’s progressive introduction to complex cybersecurity topics. The participants also pointed at ways to improve the visibility of the CyBOK in-game content, and suggested a linkage to public vulnerability databases as an addition to CyBOK. They also pointed out the imbalance between the high time commitment the tasks require and the relatively low in-game upgrade gains they bring. This would be compensated in the context of a course with the game and tasks benefiting the student in the course.

We named this platform the “Citadel Programming Lab” and have released it under free software licence1. The platform can be self-hosted. It relies on a GitLab instance for the coding environment and includes a WebGL Unity version of the game (as illustrated in Figure 1). The contributions of the papers are as follows.

1) We present an approach to align with CyBOK the learning outcomes of serious game-based cybersecurity educational labs.
2) We describe how we incorporated a serious game for self-motivated and gradual secure programming learning.
3) We demonstrate how we embed secure programming exercises within a serious game educational platform.
4) We introduce the Citadel Programming Lab, a self-contained, deployable, open source, GitLab-based online secure programming game-based lab.

In the reminder of the paper, we detail the objectives of the Citadel Programming Lab (Section 2), present the design and implementation of the learning in the lab (Section 3), and our evaluation through focus groups with early users of the lab (Section 4). We finally discuss related work (Section 5) and conclude (Section 6).

2. Objectives

In this section we detail the guiding objectives we followed when designing and implementing the Citadel Programming Lab.

1. Information and setup files for the lab can be found at: https://citadel-programming-lab.gitlab.io

2.1. CyBOK Alignment

The CyBOK’s knowledge base attempts to cover all practical and theoretical topics of cybersecurity. These topics are structured into a tree to help distil and organise cybersecurity knowledge from different sources of information, including learning materials, publications and foundational and practical materials. It is therefore natural to be eliciting the content and learning outcomes of a cybersecurity educational activity in terms of CyBOK’s Knowledge Areas and Topics. In the context of the Citadel Programming Lab, the cybersecurity content and learning is articulated through both the gameplay and programming tasks. Thus, an objective of this work is to make explicit the learning scope of the lab by aligning with CyBOK the learning outcomes of the tasks as well as the cybersecurity metaphors the game includes.

2.2. Gradual and Motivational

The main reason to employ a game in an educational context is to provide motivation to engage with the materials or tasks of the course via the playfulness of the game. Employing a serious game approach implies that the game does not solely have the play-provider role but also carries and distils the knowledge. As the cybersecurity topics covered by the lab are complex, the objective of the game, in addition to be motivational, is to introduce these topics in a more approachable way than a set of coding tasks could. The purpose of the game is therefore to exemplify the processes at play in the coding tasks to make them less difficult to grasp and to offer a smoother or more gradual learning curve.

2.3. Contextual and Embedded

Secure coding exercises provide a practical and technical experience necessary for cybersecurity learning. But practical exercises can be difficult to directly relate to the successful outcome of an attack or a defence, and the live processes and cybersecurity scenarios in which they would play a role can be hidden or difficult to observe. To make the cybersecurity context of the programming exercises more tangible, and highlight the adversarial nature of the task, we embedded these real coding interactions into the gameplay. Therefore, one objective of the lab is to provide a context for concrete and practical secure coding exercises and to embed these tasks within the gameplay to provide a rationale for moving from the game to the task, and then back to the game to tackle other tasks.
This bridge between the tasks and the game is not an actual visualisation of the effect of the coding but a simulation of the processes at play in real life approaches to secure development. The lab provides coding exercises to help the student learn real defensive strategies and gain awareness on secure coding practices while exhibiting through the game the implication in a fictional context.

3. Implementation

In this section, we present the implementation of our lab. The presentation is articulated along the relations between three main aspects of the lab: the game, the hands-on tasks and the CyBOK Learning Outcomes.

Once the lab platform is deployed by an educator, students log in to the GitLab-based Web interface to start the lab (students only need a modern Web browser to access the lab). The intended students’ interaction with the lab follows three main steps with a main game loop as shown in Figure 2.

1) Play tutorial level of game, be exposed to purpose of game and gameplay mechanics;
2) Play main level, be exposed to security metaphors, develop motivation to defend your goal, earn points;
3) Spend points unlocking upgrades, see that some upgrade tiers require solving a programming task and reviewing other solutions.

The platform is not prescriptive on the way a teacher has to use the lab. The teacher can review code submissions as they need, make uses of the various mechanisms available in GitLab to comment or raise issues directed to the students. A teacher interface visualises students progress through the lab (such as high scores, commits, completed tasks) for the whole cohort, allowing them to monitor students’ engagement.

In the remainder of this section, we will present the rationale between the linkage of each aspect of the implementation.

<table>
<thead>
<tr>
<th>Task</th>
<th>Learning Outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
<td>Understand the necessity of storing securely passwords</td>
</tr>
<tr>
<td>Storage</td>
<td>Learn the importance not to blind trust user inputs</td>
</tr>
<tr>
<td></td>
<td>Make use of cryptography APIs</td>
</tr>
<tr>
<td></td>
<td>Understand the different between hashing and secure hashing</td>
</tr>
<tr>
<td>URL Shortener</td>
<td>Understand the concept of reversibility</td>
</tr>
<tr>
<td></td>
<td>Experience generating secure randomness</td>
</tr>
<tr>
<td></td>
<td>Recognise that features can have security trade-offs</td>
</tr>
<tr>
<td>String Encryption</td>
<td>Understand how cryptography helps protecting data</td>
</tr>
<tr>
<td></td>
<td>Make use of cryptography APIs</td>
</tr>
<tr>
<td></td>
<td>Learn about different types of ciphers, their strength and weaknesses</td>
</tr>
<tr>
<td>Public Key Certificate</td>
<td>Understand the necessity of the bundle between identity and public key</td>
</tr>
<tr>
<td></td>
<td>Experience the hierarchy tree of trust in public key infrastructure</td>
</tr>
<tr>
<td></td>
<td>Make use of cryptography APIs</td>
</tr>
<tr>
<td></td>
<td>Understanding the long list of factors affecting the trust of a certificate</td>
</tr>
<tr>
<td>PGP</td>
<td>Understand the necessity of message authentication</td>
</tr>
<tr>
<td></td>
<td>Experience the operations involved in public key infrastructure</td>
</tr>
<tr>
<td></td>
<td>Make use of cryptography APIs</td>
</tr>
<tr>
<td></td>
<td>Compare implementations of digital signature methods</td>
</tr>
<tr>
<td>SSL Connections</td>
<td>Understand the SSL/TLS communication channels</td>
</tr>
<tr>
<td></td>
<td>Experience the handshaking and negotiation process of SSL/TLS</td>
</tr>
<tr>
<td></td>
<td>Make use of SSL/TLS libraries to build up trusted communication channels</td>
</tr>
<tr>
<td></td>
<td>Understand the linkage of X.509 certificate and the SSL/TLS authentication</td>
</tr>
</tbody>
</table>

3.1. Programming Tasks & CyBOK Learning Outcomes

This linkage is between the choice and design of the programming tasks, and the learning outcomes for the lab as a whole. The learning outcomes are given in Table 1 and cover a range of introductory concepts of CyBOK as shown in Table 2.

Based on the underlying topics of the course we wanted to use the lab in, we developed 3 programming tasks, and repurposed 3 tasks from an earlier secure development study [8]. Each task aims to support learning outcomes backed by CyBOK Guidance. The 6 tasks are:

1) Credential Storage [8] — Store password in a database
2) URL Shortener [8] — Shorten a long URL
3) String Encryption [8] — Encrypt a text message
4) Public Key Certificates — Validate an X.509 Certificate
5) PGP — Verify the signature of a message signed via PGP
6) SSL Connections — Implement an SSL/TLS handshake

The learning outcomes for each of these tasks are presented in detail in Table 1.

When a student wishes to complete a programming task, they visit the private GitLab instance of the task (see Figure 4). Given a skeleton file with a method signature and explanation of what needs to be completed,
TABLE 2. LINKAGE OF CyBOK v1.1 KNOWLEDGE AREAS AND TOPICS TASKS AND GAME ELEMENT

<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial Behaviours</td>
<td>Attribution</td>
</tr>
<tr>
<td></td>
<td>Hacktivists</td>
</tr>
<tr>
<td>Applied Cryptography</td>
<td>Authenticated Encryption/AE schemes</td>
</tr>
<tr>
<td></td>
<td>Binding Public Keys and Identities via Certificates</td>
</tr>
<tr>
<td></td>
<td>Cryptographic Libraries</td>
</tr>
<tr>
<td></td>
<td>Diffie-Hellman Key Exchange</td>
</tr>
<tr>
<td></td>
<td>Digital Signatures</td>
</tr>
<tr>
<td></td>
<td>Hash functions</td>
</tr>
<tr>
<td></td>
<td>Managing Public Keys and Public Key Infrastructure</td>
</tr>
<tr>
<td></td>
<td>Public-key schemes with special properties</td>
</tr>
<tr>
<td>Authentication, Authorization &amp; Accountability</td>
<td>Authentication</td>
</tr>
<tr>
<td>Distributed System Security</td>
<td>Reliable and secure group communication</td>
</tr>
<tr>
<td>Law &amp; Regulation</td>
<td>Electronic signatures and identity trust services</td>
</tr>
<tr>
<td></td>
<td>Prescriptive jurisdiction and data protection</td>
</tr>
<tr>
<td>Network Security</td>
<td>Cloud and Data Center Security</td>
</tr>
<tr>
<td></td>
<td>Networking Infrastructure Security</td>
</tr>
<tr>
<td></td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td></td>
<td>TLS (Transport Layer Security)</td>
</tr>
<tr>
<td></td>
<td>Device fingerprints</td>
</tr>
<tr>
<td>Privacy &amp; Online Rights</td>
<td>Cryptography-based access control</td>
</tr>
<tr>
<td></td>
<td>Oblfuscation-based inference control</td>
</tr>
<tr>
<td></td>
<td>Privacy engineering</td>
</tr>
<tr>
<td>Risk Management &amp; Governance</td>
<td>Risk assessment</td>
</tr>
<tr>
<td>Secure Software Lifecycle</td>
<td>Motivations for secure software lifecycle</td>
</tr>
<tr>
<td>Security Operations &amp; Incident Management</td>
<td>Cyber-threat intelligence (CTI)</td>
</tr>
<tr>
<td>Software Security</td>
<td>SQL injection</td>
</tr>
<tr>
<td></td>
<td>Coding practices</td>
</tr>
<tr>
<td></td>
<td>Query generation</td>
</tr>
<tr>
<td>Web &amp; Mobile Security</td>
<td>SQL injection</td>
</tr>
<tr>
<td></td>
<td>Input sanitisation</td>
</tr>
<tr>
<td></td>
<td>Password leaks</td>
</tr>
<tr>
<td></td>
<td>Web PKI and HTTPS</td>
</tr>
</tbody>
</table>

students work on the task and commit their code through the Web IDE. Within the explanation given, the skeleton file also offers prompts for what kinds of security the player should consider as they complete their solution. The programming tasks are developed as GitLab projects, each with their own continuous integration tests, so it is possible to adapt and change individual tasks to match the intended learning outcomes for the course in which the lab is deployed.

Within the lab there is support for both Java and Python programming. Students would have a choice between the two for implementation of each task. The aim of the lab was not to teach one particular language or API, but to teach the concepts behind them. Students can edit package requirement files to use additional libraries or APIs for these languages if they wish.

When they make a commit, our lab uses pre-configured unit tests that run using the GitLab continuous integration system. This gives feedback on the functional correctness of the solution a student produces. These tests need to pass in order for the task to be considered “complete”.

Students can complete the tasks, and the lab, in any order that they wish. We envision a teacher would issue the lab as a coursework exercise, and students work to complete all tasks before a deadline or following a prescribed order.

By implementing our programming tasks and deriving learning objectives from CyBOK, we are able to ensure that our lab is aligned with the goals and knowledge of CyBOK, and we are able to use the game to motivate completion of tasks, while offering a graduated introduction to security concepts.

3.2. Programming Tasks & The Serious Game

The game is designed as a tower defence game, where the player aims to prevent enemies from getting into a bank to steal money (see Figure 3). Its design incorporates many links relating to security and secure programming. This section details the linkage between the game and the programming tasks, specifically how the game prepares players for later completing programming tasks.

Within the game, players organise their towers to attack enemies that are attempting to gain access to a bank. They can command towers to fire at enemies, issue commands to launch missiles at enemies, and manage communication links to extend the range of towers. Through a command line interface the players can maintain the security of their towers and target named characters. Upgrades purchasable using virtual currency allow the player to enhance their abilities.

The primary link between the tasks and the game is through the security metaphors in the context and design of the game. The link to security through metaphor helps to visualise security concepts, and presents possible vulnerabilities, attacks and defences before the player encounters them in programming tasks. The enemies in
Figure 3. Screenshot of the gameplay featuring enemies, including those carrying injection attack names; and towers such as the X509 watchtower

Figure 4. Screenshot of the GitLab-based coding environment

Table 3 includes a detailed reference of the intent of each metaphor in the game’s design, as it applies to the programming tasks and learning. Many game elements also carry a specific importance to one programming task, this is also presented in the table with the task number given, or “*” for elements that apply equally to all tasks.

The main motivator to complete the tasks is through the game’s upgrades system. Players can purchase the first tier of an upgrade immediately, but in order to access higher tiers they must complete programming tasks. Each upgrade is associated with a programming task, and in addition to presenting a coding challenge that implements a concept represented by that upgrade, it acts as a motivator for the player to complete that particular task.

We also use the upgrade system as a motivator for completing peer reviews. The top tiers of upgrades are only unlocked once the task has been completed, and a peer review completed. Peer reviews are completed by a student reviewing the code submitted by another student, and describing their agreement with the following prompts:
1) This solution is secure
2) This solution is correct
3) This solution is good

The intent behind these 3 questions is to prompt the reviewer to focus the scope of the review, and to reduce the burden of completing many reviews by simplifying what to consider. When the review is sent back to the programmer, this gives an indicator of what the programmer needs to prioritise going forward. The review questions need to be answered before the player can fully unlock certain upgrades. Being presented with a generic criteria for a peer review will also motivate the student when they start completing their next task.

As an additional motivator for playing the game and progressing, we add a leaderboard. The leaderboard and scoring mechanics to encourage the player to do better each round they play, encouraging them to purchase the upgrades, thus encouraging them to complete programming tasks. Players can see other classmate’s scores. If players wish, they can change their display name to maintain anonymity, or show off, in their scores.

Through a strong link between the programming tasks, and the game we offer an effective motivation for completing the tasks, while leveraging the game as a means to introduce technical concepts in a softer more gradual way. The game also allows us to embed security concepts through a visual metaphor, and we can better set the context for why security is important.

3.3. The Serious Game & CyBOK Learning Outcomes

Through playing the game, the student will be encouraged to complete the tasks, and through exposure to the metaphors in the game, will be aware they need to consider security. The lab, and the game, embed concrete knowledge as well as links to CyBOK topics to open up additional resources and guidance (the coverage is shown by Table 2).

By embedding the knowledge, we ensure that playing the game becomes a vital part of the lab, and that students are then prepared with relevant knowledge before they start to complete programming tasks. This is one of the major differentiations between our serious game, and a simple gamification of programming tasks. The learning outcomes associated with each of the game elements are presented in Table 4.

We considered how to present this knowledge within the game and settled on a question-and-answer format which mimicked the natural train of thought of a user playing the game. We chose questions such that they were likely to be asked by a user during play, and so that each one was connected to a metaphor within the game. This transformed the answers from simple explanations of cyber-security concepts, to both conceptual explanations, and subtle descriptions of in-game mechanics. This link between knowledge and play is designed to be an important incentive to engage with the material.

In order to organise the Q&A items, we grouped them under 7 categories: Input Sanitisation [9], Encryption [10], Network Safety [11], Passwords [12], Security Education [13], Detecting Malicious Activity [14], and Building Defences [15]. Each of the questions in a category is focused on either the defence, attack and vulnerability angle.

The Q&A information is presented in two locations, both with the same information, but with different ways of targeting and accessing the information. The main help menu just offers the knowledge as-is, organised by category (Figure 5). The in-game help menu offers a manual of what each game element is and how the player interacts with it, and then any relevant topic areas are included as an aside / related help article (Figure 6).

The process we used to build up the Q&A information started with writing and consolidating the help information about how to play the game. In addition to the tutorial
TABLE 4. LINKAGE BETWEEN GAME ELEMENTS AND LEARNING OUTCOMES

<table>
<thead>
<tr>
<th>Game Element</th>
<th>Learning Outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicles</td>
<td></td>
</tr>
<tr>
<td>Standard</td>
<td>Recognise that no user should be implicitly trusted</td>
</tr>
<tr>
<td>Tank</td>
<td>Be aware that some attackers target the system vulnerabilities</td>
</tr>
<tr>
<td></td>
<td>Experience the method and effects of injection attack</td>
</tr>
<tr>
<td>Hacker</td>
<td>Be aware that some attackers target communication channels</td>
</tr>
<tr>
<td>Interceptor</td>
<td>Be aware that some attackers target to intercept confidential information</td>
</tr>
<tr>
<td>Towers</td>
<td></td>
</tr>
<tr>
<td>Standard Turret</td>
<td>Be aware of the role of primitive passive defence</td>
</tr>
<tr>
<td>Communication Tower</td>
<td>Recognise the importance of safe communication channels</td>
</tr>
<tr>
<td>Missile Turret</td>
<td>Understand the trade-off between strong defence and service availability</td>
</tr>
<tr>
<td>Laser Turret</td>
<td>Be aware of the role of directed active defence</td>
</tr>
<tr>
<td>Watch Tower</td>
<td>Recognise the importance of verifying legitimate actors</td>
</tr>
<tr>
<td>Radar</td>
<td>Recognise the role of authentication</td>
</tr>
<tr>
<td></td>
<td>Recognise the role of signatures to guarantee authorship</td>
</tr>
</tbody>
</table>

we have also created a help system with guidance for each game element the player can interact with. Then, we went through the CyBOK knowledge tree and for each of these game elements, we identified which topics were appropriate (see Table 2).

This game-level integration of knowledge helps to ensure alignment with CyBOK and makes it more accessible to the player, providing an additional help resource if needed. It also helps to strengthen the theory before putting it into practice in the programming tasks, deepening the embedded nature of the lab.

4. Evaluation

We conducted an evaluation of the lab through a focus group discussion. We made the platform available to students and then invited them to discuss their thoughts. The aim of the focus group was to investigate if the lab met our objectives.

A total of 3 participants took part in an over-90-minute session. Participants and facilitators discussed over online video, while participants edited an online word document together to record their answers.

4.1. CyBOK Alignment

We wanted to see if using CyBOK for the alignment of the lab, and as a basis for learning outcomes, was appropriate.

Integration of CyBOK for Learning. CyBOK itself was praised for being well structured and clear when reading them. Participants felt CyBOK would help them “if they had done something wrong”.

Positive reactions to the CyBOK linkage included each of the tasks not duplicating CyBOK concepts, well-relation game metaphors and CyBOK concepts (such as hacker vans as intruders and the game mechanics of passwords). Cryptography and Network Security stood out as CyBOK topics within the game, though it was more relevant for some tasks than others, namely, it was found by one participant to be less relevant to the SSL task.

Participants suggested the lab could be improved by introducing more mobile security topics. One participant proposed making the references to CyBOK more prominent, as this would place focus more on learning topics from CyBOK than just playing the game.

Useful data sources other than CyBOK. During the focus group, there was discussion around what other information sources participants were familiar with, or which they turned to when seeking help.

OWASP® was one such source, their “how to prevent” guidance was seen as useful for secure development and fixing vulnerabilities. However, another participant cautioned the use of an additional source of information if it did not add anything new, simply duplicating information from CyBOK. A participant suggested that CVSS scores could be used to prioritise completing the different tasks.

Stack Overflow (SO) was frequently mentioned. In addition to the main code solutions which can be copied and adapted as needed, the comments section on SO answers were seen as useful. The comments would show security issues in solutions, point to documentation, and “offer theory behind the code”. SO was mentioned as being less useful for less popular languages like Spark or Ada.

When we asked what techniques and sources are used when problem-solving, participants often mentioned searching for the solution on Google. One identified that though CyBOK has a lot of useful information, it requires knowing how the topics are organised before you can get to it whereas Google will “take you directly to what you want to know”. Wikipedia would give a basic introduction to unknown theories. Another student noted their approach was to write code in whatever way used the “shortest API”.

4.2. Gradual and Motivational

Within the focus group, we also aimed to see if it helped gradually introduce concepts and motivate players to complete the programming tasks and to learn.

Audience Targeted by the Lab. There was consensus that newer learners, such as 18–25 years old, would be most motivated by the game. The game itself would work well when incorporated in a module, with suggestions it is either an introduction or that it be a coursework requirement. The focus group agreed that the group that would benefit most would be developers and testers — roles that work with code. One focus participant felt it would be of particular benefit to people who would copy and paste code from online, so they learn not to copy code with bugs in it. The focus group praised the game as being engaging and motivating.

Gradual and Motivational Aspects of Lab. The multi-user nature of the lab was well received. The idea of the leaderboard was seen as fun, positive, competitive and

2. https://owasp.org/
more motivating than just playing alone. Though not all players would be motivated by a leaderboard, including this gameplay feature "would not bring any negatives". There was a proposal that a game pitting "developers against testers" would work well. The peer-review system, which allows seeing what others do, was motivating.

One participant felt that the tasks required a high time commitment, so the reward for completing tasks ought to be higher than just a single unlocked upgrade, to improve motivation.

A participant felt the lab should introduce security jargon more gradually.

4.3. Contextual and Embedded

We wanted to see if the lab offered an appropriate contextualisation of the issues at stake, and that the knowledge embedded facilitated learning.

Embedding Guidance in the Lab. The in-game help manual explaining each of the game mechanics was highly praised, and it was suggested that CyBOK guidance should be included in the same way for each task.

However, the in-game presentation of CyBOK guidance was criticised. The main button that opens the guidance, simply titled "CyBOK", was poorly named and unclear. CyBOK guidance should have been included in the tutorial. The CyBOK guidance that was included for each game mechanic should have been refined to be more specific to where it was included.

The peer-review aspect was praised as allowing for "sharing of ideas"; and "helping when you forget to do something or don’t use the most secure code". However, there was a suggestion that the best way to do peer review is with code comments on GitLab, or if time is not an issue, with a teams (virtual) meeting.

Contextualisation in the Lab. One student felt that the programming tasks were realistic and of the right sort of difficulty, and that if "the tasks don’t make you think about cybersecurity, I don’t know what would" — indicating that the lab offers a strong contextualisation of the concepts.

Within the game, some of the attack mechanisms such as injection attacks and interceptions were "funny". These attacks added a sense of realism, and the focus groups suggested this is good as too much abstraction was identified as a barrier to learning. One participant identified ransomware as an interesting new attack that could be added, given the target in the game is a bank with money.

In the game, the number of minutiae that need to be managed in the game can be overwhelming and a large number of mechanics means "it is easy to forget which ones can be used". The text prompt interface also needed to be improved to explain what text could be entered.

Regarding the languages chosen within the lab exercises, participants agreed that Python was a "solid choice", and easier than Java, and easier to research online. One participant suggested substituting JavaScript for Java.

4.4. Limitations

There are two limitations to this evaluation. First, our focus group discussions had a low number of participants which means that we lack views from the whole student body. The second limitation is that we have yet to gather usage metrics for whole cohort of students using the lab in the context of a course, which prevents from drawing conclusions on the usefulness and effectiveness of the lab as a teaching environment.

5. Related Work

Within cybersecurity education, and computer programming education in general, many new trends are emerging in how technology is used to assist with teaching. In this section we explore the related work along four different focuses.

Designing Cybersecurity Learning Outcomes. As we mentioned in the rest of the paper, we built our approach on a linkage with the knowledge base that the CyBOK project has produced [16]. CyBOK is used as part of the undergraduate and postgraduate programme certification process that NCSC has put in place in the UK to map the cybersecurity coverage of applications. In [17], the authors present the design of the learning outcomes of an applied cyber security course and its evaluation with a cohort of students. Evaluation often rely on grades and questionnaires such as in [18], which used gamification to boost the student’s engagement and the effectiveness of the delivery of the course content. In our case, we focused on a qualitative evaluation of the linkage before deploying and evaluating the lab with a cohort of students in a course.

Self-Contained Labs. Within cybersecurity education, the use of self-contained labs and tutorial environments is a trend that has been developing for a long time. One of the first such labs, the SEED Labs [2], [3] provided virtual machine images with linux machines pre-configured for network security teaching. These labs have since been adapted for use with other portable formats like docker [19], which could allow for advanced labs that are tailored to be unique to each participating student [20].

More recently, labs have been developed that teach such topics as malware detection and analysis [21], [22], safely configuring networked machines [23], and teaching cyber physical security [24]. Many of these modern labs can now be hosted in the cloud, eliminating the need for students to manage their own VMs, which can often be resource heavy, focusing more on doing the actual labs.

Online Platforms for Education. One of the more noticeable changes is a move towards using industry standard development platforms within a teaching environment. Platforms like GitHub and GitLab offer Git-based version control, giving students an early taste of a real world development tool and encouraging collaboration while also facilitating course delivery [25]. Project management tools and workflows like pull requests can enable improved collaboration amongst students and course facilitators [26]. Metrics collected by these platforms can be used to assist students by revealing comparisons of how students interacted with the platform [27]. Alongside these
development platforms, other open source tools can be used to make learning and code more accessible, such as Jupyter Notebooks, which have been integrated into course delivery platforms [28] offering a lighter touch approach to code, while allowing flexibility to link in other teaching tools like small games or quizzes. These online platforms also allow for increased easy collaboration between students, such as through code reviews. Code reviews are used in higher education programming courses as a means of encouraging peer interaction [29], which brings benefits such as fostering collaboration, sharing of knowledge and critique, and developing skills in code review which will be useful in their careers.

Game-based Learning. In education and within industrial practice, games and serious games have found a foothold as a fun and engaging way to explore and present programming and security concepts. Serious games may be deployed as an introduction to security topics, as well being the output of development projects to focus learning [30]. Games and general visualisation tools help to provide a visual context to abstract concepts like buffer overflow attacks [31]. Gamification is also being used, both in education such as to encourage attention to bug analysis tools [32] or evaluate and analyse student’s learning performance [33], as well as in industry, supporting code quality [34]. It is also used with tailor-made platforms to maximize the engagement with and effectiveness of cyber security training to learners with different levels of background knowledge [35]. The link between gamification and cyber security learning has been researched through a study on students’ self-perception of success while learning certain cybersecurity topics [36]. This was investigated through a gamified course which contains metaphors of security concepts, showing that using serious games in course does improves student engagement and consolidates their knowledge on cybersecurity. Security contests such as Build it Break it Fix it [37], [38] or Capture The Flag (CTF) exercises [39] are being organised to offer a practical environment to learn how to both develop securely and gain experience in testing and attacking a target. In contrast to these competitions, our approach focuses on secure development only, and the competitive aspect is driven by the game rather than attacking code.

6. Conclusion

In this paper, we propose the “Citadel Programming Lab” designed for use in undergraduate security courses. It comprises a tower defence game with six secure programming tasks which are solvable in Java or Python. The lab aims to deliver hands-on secure programming exercises for the learner to gain experience and awareness towards the different fundamental topics in cybersecurity. The lab employs a serious game approach to motivate engagement, to provide a gentler learning curve, and to contextualise the practical exercises. We detailed the alignment of the game and programming tasks to learning outcomes linked to Knowledge Areas and Topics of CyBOK. We evaluated the design approach with a focus group composed of three student participants who discussed the CyBOK linkage and the educational approaches of the lab. They also mentioned the meaningfulness of the linkage and the suitability of the game to support a progressive introduction to complex cybersecurity topics. The “Citadel Programming Lab”, available under free software licences, is a self-contained, self-motivated, game-based learning platform for secure programming.

7. Acknowledgements

The original game was developed as part of a joint project supported by NCSC-RISCs between Heriot-Watt University and the Glasgow School of Art. The work presented in this paper received funding through a Cyber Security Body of Knowledge (CyBOK) small grant, and was supported by the EPSRC Secrious Project (EP/T017511/1 Serious Coding: A Game Approach To Security For The New Code-Citizens).

References


