Segmented and Interactive Modules for Teaching Secure Coding: A Pilot Study

Sagar Raina(✉), Siddharth Kaza, and Blair Taylor

Department of Computer and Information Sciences, Towson University,
7800 York Road, Towson, MD, USA
{sraina, skaza, btaylor}@towson.edu

Abstract. Learners can experience content disorientation in web based learning modules. The security injection modules developed by Towson University have increased students' secure coding awareness and ability to apply secure coding principles, but feedback from instructors indicate that students tend to skim or skip the module contents and proceed directly to the laboratory assignment. In this paper, we describe the factors that cause cognitive overload in hypertext readers and address the pertinent issues and describe the process we used to enhance the effectiveness of the modules. Security Injections 2.0 incorporates principles of segmentation - breaking large module content into smaller sections and presenting each section one at a time, dialoguing - answering questions and receiving corrective or explanatory feedback, and controlling - reading and learning content at learners own pace. Segmentation, dialoguing, and controlling engage learners and retain concepts. Pilot study results indicate 77 % of the students scored above 70 % in concept retention assessment.

Keyword: Computers and education

1 Introduction

Addressing the crucial need for cybersecurity learning materials, the Security Injections@Towson project (towson.edu/securityinjections) has developed modules for Computer Science 0 (CS0), Computer Science 1 (CS1), Computer Science 2 (CS2) and Computer Literacy courses that target key secure coding concepts including integer error, buffer overflow, and input validation. Assessment results indicate that these modules have led to an increase in students' security awareness and their ability to apply secure coding principles [7, 15]. The modules, developed on the cognitive learning principles of Bloom's Taxonomy, adopt a uniform structure. Each module begins with a background section to describe the problem, including examples, followed by a code responsibly section which includes methods to avoid security issues, a laboratory assignment with a security checklist, and a discussion questions section. The module content is presented as hypertext on a single webpage [15]. The module structure is designed to help students to first *remember* and *understand* the problem through the background and code responsibly sections, *apply* the concepts learned through laboratory assignments and *analyze*, through discussion questions [15]. Students have to turn in the laboratory assignment and discussion question answers to their

G. Vincenti et al. (Eds.): eLEOT 2014, LNICST 138, pp. 147–154, 2014.
DOI: 10.1007/978-3-319-13293-8_18

instructors as text document to receive the grades and feedback. Over 160 instructors from various community colleges and universities are using security injection modules in CS0, CS1, CS2 and Computer Literacy courses. Although assessment results indicate significant improvement in students' secure coding awareness and their ability to apply secure coding principles, instructors using these modules have observed that students tend to skim or skip the module contents. The previous research suggests that skimming or skipping important content might hamper students' knowledge about the topic they are learning [6]. In this paper, we investigate this problem and propose a solution.

2 Literature Review

2.1 Hypertext Reading

The security injection modules are web-based, that is, students can access the modules as hypertext using a uniform resource locator (URL). Hypertext is defined as a document that contains a variety of media resources such as text, audio, video, graphics, presented in a non-linear fashion unlike printed textbooks. The media resources in hypertext, may link to other documents, giving flexibility to a reader to click any of the links and acquire knowledge [4, 9]. To acquire knowledge from hypertext, readers do not follow a specific reading order. Depending upon the readers' goals and interest, the hypertext readers may adopt a specific reading strategy [4, 8, 13]. The reading strategy allows readers to decide what to read and what to skim [13]. In addition, the amount of content presented on a screen is also considered as a deciding factor for skimming hypertext. Huge amount of hypertext on a screen may cause readers to skim the text in comparison to lesser amount of hypertext [5]. The readers, in skimming process, read the first half of the paragraphs and skip the rest if they think information gain is low and, start reading the next paragraph [5]. During this process, readers might skip important content [5, 13]. In addition, this selective reading might result in less in-depth reading, less concentration and less attention towards the content [9]. Overall, skimming content results negatively towards knowledge consumption [6]. In a study conducted among 113 participants, to observe their reading behavior, 78 % of participants reported they read more selectively because of the large amount of information available on web [9]. The process of selecting content to read hypertext requires readers to make decisions, which induces cognitive overload in a reader [4, 13]. There are several other factors that might induce cognitive overload in a reader including - lack of prior knowledge or domain knowledge about the content, complexity of the concept being taught, structure of the content, and lack of motivation to read the content [1]. Readers lacking prior knowledge or domain knowledge, have difficulty processing and understanding the content, thus inducing cognitive overload [4, 8, 12]. Additionally, if a concept itself is complex to teach and present, readers find it difficult to process and understand, resulting in cognitive overload in a reader. Well-structured hypertext is easy to process by human brains as compared to unstructured content. Unstructured content may distract readers while reading, and induce cognitive overload [1, 13]. Lack of motivation in a reader affects their cognitive ability to understand the content and thus inducing cognitive overload [1, 8]. Cognitive overload make reader's

become disoriented and lose attention towards the content, resulting readers to skim or skip the content [2, 13, 16]. Also, because of the large amount of content presented on a single computer screen, readers scroll windows up and down, and during the process skip some lines of text and lose their place. In one study, which assessed 73 students reading on the web, 25 % of the students reported they skipped lines and lost their place while scrolling the window up and down on a computer screen [16]. In conclusion, there are several factors that induce cognitive overload in a reader while reading a hypertext, including domain knowledge/prior knowledge, concept complexity, content structure, amount of content presented on a single screen and lack of motivation. Cognitive overload cause readers to become disorient and inattentive towards the content, which results in skimming and skipping of content. In the next section, we discuss cognitive load theory.

2.2 Cognitive Load Theory

As discovered in the literature review section, one of the factors that might cause readers to skim or skip hypertext content is cognitive overload. According to cognitive load theory, cognitive load is the amount of information the working memory in a human brain can process at a given time [2, 4]. Human beings have a working memory which can hold or process a limited amount of new information at a given time [4, 10, 11]. When this working memory receives information to process beyond its limited capacity, it leads to cognitive overload in a human brain [11].

Working memory could be affected by different ways while processing information, resulting in different types of cognitive loads. If the working memory load is affected due to the complexity of the nature of learning task itself, it is called as intrinsic cognitive load. If the cognitive load is affected due to the way information is presented on the screen, it is called as extraneous cognitive load.

In order to avoid cognitive overload and induce learning in a reader, we need to either reduce the intrinsic or extraneous cognitive load or both [10, 11].

3 Proposed Solution

To improve the effectiveness of the secure coding modules, we analyzed the security injection modules for the factors that lead to cognitive overload in readers, which lead them to skip and skim the content. We found that domain knowledge/prior knowledge, concept complexity, and content structure are addressed by the security injection modules. The security injection modules did not address amount of content and lack of motivation factors. Section 3.1 provides detailed analysis of how we mapped cognitive overload factors with the security injection modules.

3.1 Mapping Cognitive Overload Factors with Security Injection Modules

1. *Amount of information presented on a single computer screen*: Security injection modules are presented on a single webpage which includes content related to background, and code responsibly information, laboratory assignments and

discussion questions. A single web page accommodates a large amount of information where students have to scroll window up and down for selecting any information. We believe there are higher chances that students might skim or skip the content because of large amount of text and scrolling phenomena.

2. *Domain Knowledge*: Security injection modules are given to students as a part of laboratory assignment. Before attempting the modules, students are taught basic programming concepts during lecture hours, for example: knowledge about data types, variables, arrays etc. Therefore, we assume students have enough domain knowledge to read and understand the security injection module content. Additionally, the modules are structured to provide foundational knowledge first.
3. *Complexity of content*: The security injection modules are designed based on the principles of Bloom's Taxonomy and are structured to help students methodically understand, remember and apply the concepts learned. The assumption is, by adhering to this taxonomy, content complexity is reduced.
4. *Structure of content*: The security injection modules follow a uniform format through all CS0, CS1, CS2 and Computer Literacy courses. Each module, across all courses, contains background, code responsibly, laboratory assignment and discussion questions sections. Each section has a heading and sub-heading that reader can easily process and understand.
5. *Lack of motivation*: The security injection modules are presented in a linear and non-interactive format and include a large amount of information on a single page. It is possible students are less interested and motivated to read the content because of little or no interactivity between the reader and the system.

3.2 Possible Solution

To increase their effectiveness in teaching secure coding content, the security injection modules need to address the amount of content and motivational factors to prevent students from skimming and skipping the content. The large amount of information presented at once can increase extrinsic cognitive load [11]. The research suggests, this type of cognitive load can be reduced by breaking large information into small chunks and present only one idea at a time on a single screen. This principle is called segmentation [1, 3, 12]. Segmentation improves processing of information in the working memory and makes recalling/retention of concepts easier [12].

Motivation can be increased by engaging learners or readers with the system they are interacting with. Engaging learners with the system can possibly be introduced by adding elements of interactivity within the learning system. The interactivity motivates learners to learn [12]. Interactivity in the context of learning is the "responsiveness to the learner's actions during learning" [12]. The types of interactivity in e-Learning environments include: *dialoguing*, *controlling*, *manipulating*, *searching* and *navigating* [12]. We focus on dialoguing and controlling types of interactivity, the remaining three are addressed by the platform where security injection modules are currently hosted. Dialoguing occurs when the learner answers questions and receives feedback to his/her input. Controlling means the learner can determine or control the pace of the presentation. Dialoguing help students to learn better, through the feedback provided by the learning

environment [12]. Feedback reduces extraneous cognitive load in the working memory. Controlling also help students learn better by allowing them to control the pace of the presentation. Controlling reduces the extraneous load by allowing learners to process smaller chunks of information in the working memory at their own pace.

We propose to enhance the security injection modules by incorporating principles of segmentation and interactivity (dialoguing and controlling). We propose to implement segmentation by breaking the module content per section (background, code responsibly, laboratory assignment, discussion questions) and present each section one at a time on a computer screen. This way readers have to read small amount of content one at a time and there will be less processing load on the working memory, which will help readers to retain the concept's learned for the longer period [1, 3, 12, 14]. We propose to implement dialoguing by including a set of questions at the end of each section. Based on the student's response, they will receive both corrective and explanatory feedback. Feedback will increase student learning [12]. We propose to implement controlling by not allowing students to proceed to the next section until they answer all of the questions correctly, which might encourage students to refer back to the content, on answering incorrectly [3, 11, 12].

4 System Implementation

4.1 System Design

To implement the proposed solution, several solutions were considered (including writing the system from scratch) before determining that a modified version of Stanford University's class2go web-based application (https://github.com/Stanford-Online/class2go/) was most appropriate. Class2go is built using the django framework and is a well-tested open-source framework that provides core functionality including user registration, course creation, test administration, and components for auto-grading. The application creates modules and sections within those modules. Each section in a module is auto-graded using built-in functionality for text and multiple choice questions. Additional components were added to allow students to find and correct software vulnerabilities in code segments using security checklists until all questions are answered correctly (Refer Fig. 1).

4.2 Module Design

Applying the segmentation principle [1], the learning modules were organized into four subsections: background, code responsibly, laboratory assignment, and discussion questions. In the background and code responsibly sections, students are required to go through the content and answer a series of questions. Each question provides feedback upon submit, thus satisfying the dialoguing principle. The student cannot advance to the next section until all questions are answered correctly, which satisfies the controlling type of interactivity. In laboratory assignments and discussion questions, students answer text-based and multiple choice questions, and identify vulnerabilities based on a security checklist. The student is able to highlight portions of the code according to steps in the checklist. These are also auto-graded until completion.

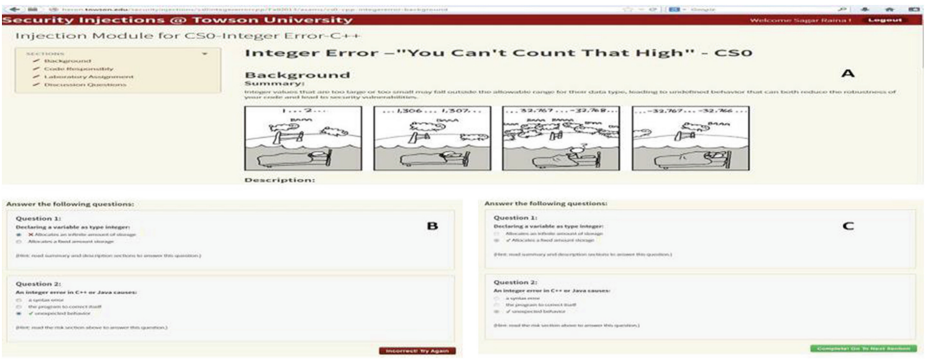

Fig. 1. The (a) security injection module, (b) content-based questions, (c) instant feedback.

5 Pilot Evaluation

5.1 Method

A pilot study was conducted with the newly built system using a C++ module focused on integer error in a CS0 class at Towson University. The class had 60 in-class undergraduate computer science students and 9 online students. The students were given security injections 2.0 module URL and were asked to follow the instructions. While in-class students were interacting with the newly built system, their behavior was observed by the instructors, giving an idea of students' engagement towards the system. A week later, a paper-based test was conducted to assess the students' retention of integer overflow concepts. Students in a paper-based test were asked to "Explain integer overflow". The students were evaluated on the scale out of 10 on the paper-based test.

5.2 Assessment

The data was evaluated for 43 students who took the post-test. The assessment results indicate that students scored an average of 7.7 out of 10 with a standard deviation (SD) of 3.18. Approximately 77 % of students scored 7 and above (falling on the higher end, on the scale of 0–10) in their post-test, reflecting their good understanding and retention of integer overflow concepts including 45 % students scoring 10 out of 10 and 25 % students scoring 8 out of 10 (Refer Fig. 2). We also assessed the correlation between the number of attempts student made to complete the module and their post-test scores to find out if increase in number of attempts improves learning and retention of integer overflow concepts. The results indicate weak pearson's correlation coefficient (–2.63), thus giving us a hint that multiple attempts does not contribute to learning and retention of concepts. We intend to study the data to further analyze this claim.

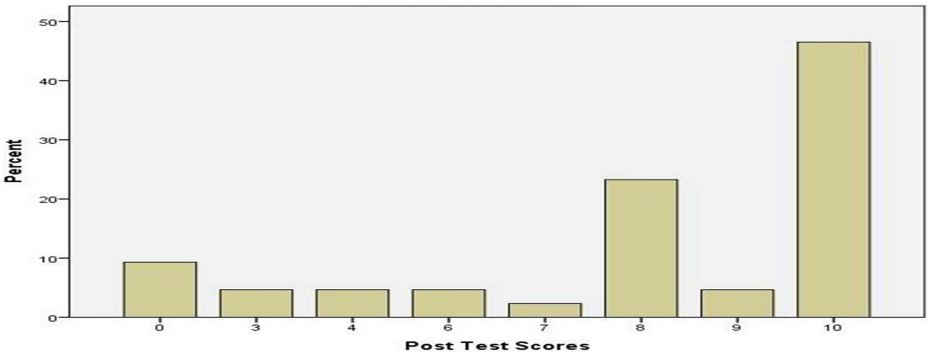

Fig. 2. Post-test scores versus students (percent)

6 Conclusion and Future Work

In this study, we addressed students' tendency to skip and skim content in order to improve the effectiveness of secure coding learning modules. We enhanced the modules by incorporating segmentation and interactivity principles to create segmented, feedback-oriented and self-paced modules to improve student learning by engaging them with content and increase their motivation. Assessment of the results indicated higher scores for the majority (77 %) of the students. Though the design of this study lacks a control group and has a small sample size, therefore, the results cannot be generalized to entire population, but, the descriptive statistics gives us an idea of students' good performance on security injections 2.0 modules. To further expand the study, we are designing an experiment and instruments which includes the deployment of multiple modules in CS0, CS1, and Computer Literacy to measure student learning, student concept retention and student engagement in two e-learning modalities (interactive modules and non-interactive modules) and plan to pilot them this summer. Future work will include auto-grading for full-text questions, which is currently limited to matching keywords. In addition, the system architecture requires modification to support large-scale dissemination and assessment.

References

1. Al-Samarraie, H., Teo, T., Abbas, M.: Can structured representation enhance students' thinking skills for better understanding of E-learning content? Comput. Educ. **69**, 463–473 (2013)
2. Chalmers, P.A.: The role of cognitive theory in human–computer interface. Comput. Hum. Behav. **19**(5), 593–607 (2003)
3. Clark, R.C., Mayer, R.E.: E-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning (Google eBook). Wiley, New York (2011)
4. DeStefano, D., LeFevre, J.-A.: Cognitive load in hypertext reading: a review. Comput. Hum. Behav. **23**(3), 1616–1641 (2007)

5. Duggan, G.B., Payne, S.J.: Skim reading by satisficing: evidence from eye tracking. In: CHI 2011 (2011)
6. Dyson, M., Haselgrove, M.: The effects of reading speed and reading patterns on the understanding of text read from screen. J. Res. Reading **23**(2), 210–223 (2000)
7. Kaza, S., Taylor, B., Hochheiser, H., Azadegan, S., O'Leary, M., Turner, C.F.: Injecting security in the curriculum – experiences in effective dissemination and assessment design. In: The Colloquium for Information Systems Security Education (CISSE), 8 (2010)
8. Lawless, K.A., Brown, S.W., Mills, R.: Knowledge, interest, recall and navigation: a look at hypertext processing. J. Literacy Res. **35**, 911–934 (2003)
9. Liu, Z.: Reading behavior in the digital environment: changes in reading behavior over the past ten years. J. Documentation **61**(6), 700–712 (2005)
10. van Merrienboer, J.J.G., Ayres, P.: Research on cognitive load theory and its design implications for e-learning. Educ. Tech. Res. Dev. **53**(3), 5–13 (2005)
11. van Merrienboer, J.J.G., John, S.: Cognitive load theory in health professional education: design principles and strategies. Med. Educ. **44**, 85–93 (2010)
12. Moreno, R., Mayer, R.: Interactive multimodal learning environments. Educ. Psychol. Rev. **19**(3), 309–326 (2007)
13. Protopsaltis, A., Bouk, V.: Towards a hypertext reading/comprehension model. In: SIGDOC'05 (2005). http://delivery.acm.org/10.1145/1090000/1085349/p159-protopsaltis.pdf
14. Singh, A.-M., Marcus, N., Ayres, P.: The transient information effect: investigating the impact of segmentation on spoken and written text. Appl. Cogn. Psychol. **26**(6), 848–853 (2012)
15. Taylor, B., Kaza, S.: Security injections: modules to help students remember, understand, and apply secure coding techniques. In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education. **99**, 3–7, ACM (2011)
16. Tseng, M.: The difficulties that EFL learners have with reading text on the web. Internet TESL J. **14**(2) (2008). Available at http://iteslj.org/Articles/Tseng-TextOnTheWeb.html