SecTutor: An Intelligent Tutoring System for Secure Programming

Ida Ngambeki[1], Matt Bishop[2(✉)], Jun Dai[3], Phillip Nico[4], Shiven Mian[2], Ong Thao[3], Tran Ngoc Bao Huynh[3], Zed Chance[3], Isslam Alhasan[1], and Motunrola Afolabi[1]

[1] Purdue University, West Lafayette, IN, USA
{ingambek,ialhasan,mafolabi}@purdue.edu
[2] University of California, Davis, CA, USA
{mabishop,smian}@ucdavis.edu
[3] California State University, Sacramento, CA, USA
{jun.dai,ongthao,tranngocbaohuynh,zchance}@csus.edu
[4] California Polytechnic State University, San Luis Obispo, CA, USA
pnico@calpoly.edu

Abstract. SecTutor is a tutoring system that uses adaptive testing to select instructional modules that allow users to pursue secure programming knowledge at their own pace. This project aims to combat one of the most significant cybersecurity challenges we have today: individuals' failure to practice defensive, secure, and robust programming. To alleviate this, we introduce SecTutor, an adaptive online tutoring system, to help developers understand the foundational concepts behind secure programming. SecTutor allows learners to pursue knowledge at their own pace and according to their own interests, based on assessments that identify and structure educational modules based on their current level of understanding.

Keywords: Secure programming · Tutoring · Intelligent system

1 Introduction

Secure programming is one of the most fundamental elements of a software development life-cycle and it's crucial to develop robust secure coding practices and procedures. According to a recent survey of professional developers, seventy percent of companies emphasized the importance of learning secure programming practices right from the early stages of writing code [6]. This high percentage indicates that secure programming is becoming synonymous with high quality code within the software development life cycle. According to a study conducted by IBM System Science Institute, software defects detected in later phases cost anywhere from six to fifteen times more than if the same defects were found in earlier phases [4].

Published by Springer Nature Switzerland AG 2022
L. Drevin et al. (Eds.): WISE 2022, IFIP AICT 650, pp. 17–28, 2022.
https://doi.org/10.1007/978-3-031-08172-9_2

SecTutor is a self-directed learning tool driven primarily by the learner. This learning tool focuses on the educational aspect of teaching students proper practices of improving the security and robustness of programs from the early stages. The fundamental goal of SecTutor is to instill good coding practices into learning and facilitate teaching secure programming in academic institutions. Learners can practice and develop robust programming skills and concepts to determine their current level of knowledge and understanding of secure coding and programming techniques.

This paper will first examine the background of self learning, as well as the benefits of teaching secure programming practices to students in computer science. Next, the focus and need of a tool to address these practices, SecTutor, will be described, as well as a comparison of SecTutor to a previously existing security tool. The layout of SecTutor is given in diagram form, as well as an example question that a student may see is shown. Implementation details like how the tool is intelligent and the question approval system are described at a high level. Finally, the paper will conclude with how SecTutor's data can help find common security misconceptions in education.

1.1 Background

SecTutor is based upon self-directed learning theory, known as the attainment of knowledge partially or entirely driven by the learner. Self-directed learning comprises four paradigms; self-modification, self-motivation, self-monitoring, and self-management [5]. Self-modification allows students to change their outlook on learning and take responsibility for changing their learning behaviours based on feedback. Self-motivation gives a sense of responsibility for learning and improving. Self-monitoring evaluates behaviours in learning and identifies current progress. Self-management controls learning behaviours and allows students to follow up on goals and complete assignments. The goal is to enable a self-directed learning approach to encourage students to overcome reluctance and present guidance in three different strategies; multiple entry points, gamification, and adaptive testing. Self-directed learning allows students to take control of their learning behaviors and provides flexibility in enhancing their skills through new methods of learning to meet their specific learning needs. SecTutor uses the principles of self-directed learning to allow students to learn secure programming practices using effective learning tools.

According to the National Research Council (2000), students who focus on the memorization of topics, rather than taking the time to understand and make sense of the topic, often have limited opportunities to learning [2]. SecTutor guides the learning process by developing practice questions and provide feedback to identify the learner's performance and contribute to the student knowledge model, which will inform the intelligent tutoring system the selection of content and misconceptions that the learner needs to spend more time on. The educational aspect of SecTutor aims to improve elements of proper secure programming practices and prepare learners to apply new skills and concepts. The

educational assessment deals with measuring the learner's abilities in robust programming, assist students in learning and offer suggestions on areas of improvement. Currently, there is a gap in secure programming education that seeks to pinpoint knowledge areas to better prepare students the skills of secure coding. A recent study found that many students majoring in computer science, lack necessary fundamental knowledge in their abilities to read and write secure code and graduate without being introduced to any secure programming practices [9]. Furthermore, research has shown that basic yet important secure programming topics are not covered in the required programming courses [1]. One of the strengths of this tool is the ability to target the misconceptions students have regarding secure programming concepts. To build good coding practices, students need to have a solid understanding of how to identify and develop secure software. These primary concepts should be a required practice in all computer science courses.

Researchers of the SecTutor tool previously collaborated on a project to develop a secure concept inventory to measure a student's understanding of concepts in a specific knowledge domain was also developed by the same researchers to assess how well students were learning secure programming [10]. The goal of this project is to use the developed assessments to diagnose misconceptions and structure personalized instruction based on the learner's current level of understanding in secure programming. This will be accomplished through constructing an adaptive test, constructing the intelligent tutorial system, integrating the learning analytic space and testing the system. The identified areas of misconceptions and foundational knowledge can be seen in Table 1.

The three main categories in Table 1 are the overall flow of writing a program, looking at the way programs evolve, the principles to guide the software development, and the artifacts handled during development through execution. The topics covered in Table 1 are targeting at both undergraduate and graduate students. For any class in which there is programming, where security misconceptions may arise, SecTutor would be a great tutoring tool.

1.2 The Focus of the Tool

SecTutor focuses on the educational aspect of teaching robust coding practices from the beginning of writing programs rather than making programs robust after they are written. The key is to inculcate good coding practices into the teaching and practices of programming in institutions where it is taught. Researchers have developed a concept map that allows users to view the primary concepts of secure programming practices. The concept map is an excellent starting point to target specific concepts that will help guide a user's progress through different learning modules. SecTutor will also provide practice questions clustered around knowledge areas calibrated by different difficulty levels. Based on a user's selection of questions and performance, SecTutor will guide the user to appropriate content. Lastly, SecTutor uses a psychometric designed test that will assess a user's understanding of secure programming concepts while providing individualized feedback on performance across specific domains and

Table 1. The identified areas of misconceptions and foundational knowledge in secure programming are broken down into three main categories: Principles, development, and execution.

Principles	Assurance
	Complexity/Simplicity
	Requirements/Design
	Implementation
	Programming languages
	Representation
Development	Threat modeling
	C Strings
	Crypto algorithms
	Random number generation algorithms
	Interdependency
	Error Handling
	Compiling
	Linking
	Testing/Debugging/Prototyping/Evaluation
	Tools
	IDE (Integrated development environments)
Execution	Library/API/Third party functions
	Input
	Memory
	Runtime

identifying the areas the user is struggling with. The focus will be achieved in 4 stages.

1. The first stage - Establishing the content domain. During this phase, the primary research questions are: What are the concepts of secure programming and their relationship? What are the critical/foundational concepts in secure programming?
2. The second stage - Developing the item pool. In this phase, the primary research questions are: How do students understand concepts in secure programming? What are common misconceptions in secure programming? What concepts in secure programming do students find difficult?
3. Third stage - Pilot testing and refining items: The primary aim of this stage is to identify which questions from the item pool best target conceptual understanding.
4. The fourth stage - Field testing: Are the scale items valid and reliable across the target populations? The research question at this stage would be seeking

to know how effective and reliable the Sec Tutor is by testing with a large number of participants.

1.3 Why Create the Tool?

There have been several concept inventories in the past, such as:

1. The force concept inventory developed by David Hestenes [7] and his graduate student between the late 1980 s s and early 1990 s s at the Arizona State University. In the early version of the concept inventory, students were made to write out answers and were not multiple choice questions. Instead, multiple choice wrong answers were built based on common wrong answers, which Hestenes tagged as distractors.
2. Computer science concept inventory for introductory programming developed in 2016 [3].
3. The CATS hackathon - cybersecurity inventories in 2019 [11].

This tool was created to successfully implement the development of secure programming self-efficacy amongst students in a secure programming clinic. One of the ways to successfully make self-efficacy is from constant practise and exposure, as indicated by results showing a correlation between self-efficacy and increased secure programming knowledge.

The objectives of this tool are as listed below [8].

1. Defining the content domain in secure programming and creating a concept map to describe that domain.
2. Identifying the concepts in the content domain that are foundational/critical.
3. Identifying challenging topics and common misconceptions held by students in secure programming.
4. Developing a pool of items(questions) that specifically target complex concepts and misconceptions in secure programming.
5. Testing and refining the collection of items to establish a draft secure programming concept inventory.
6. Test the scale for validity and reliability.

1.4 What Does This Project Propose?

This project is a development of a dual-purpose testing and tutoring system which will aid students in learning about secure programming at their own pace while in an extra-curricular environment. This will be done with continuous access to secure programming knowledge through an online system called SecTutor. SecTutor uses an assessment-driven approach for individuals to learn about secure programming through a personalized learning system with rigorous assessments to determine a learner's level of knowledge and skill, used to personalize instructions for the learner. It will create a learning guide for students and give them access to an adaptive learning platform with a concept map that has been defined. The platform will also assist teachers with better analysis

and adaptation of teaching techniques by identifying, managing and correcting erroneous beliefs once they manifest.

The primary focus of the results from concept inventories is to improve pedagogy while also achieving the below.

1. Helping instructors compare teaching over time.
2. Assisting institutions to rank instructors.
3. Helping other stakeholders make comparisons across institutions.

1.5 The Purpose of the Tool

The design of SecTutor enables it to identify students' misconceptions through a unique test tailored to each user and designed so that the questions, administration, scoring procedures and interpretations are consistent and in adherence to laid down standards and guidelines. They do not replace examinations or grading of students' learning; instead, they diagnose areas of programming misconceptions and help students overcome the challenges. Like textbooks, the students are motivated to use SecTutor because it will increase their knowledge about secure programming and make their performance (such as grades) and job skills better. We will promote the tool, and host workshop(s) to scale up the amount of questions.

Concept inventories are designed to measure the following. The generated scores indicate how well a student understands a concept, where low scores may be indicative of a misconception.

1. Core concepts of a topic.
2. The extent to which students have achieved expert-level thinking in a domain.
3. A concept map of secure programming which will define the content domain in secure programming and identify the major and minor concepts, while portraying the relationships among these concepts.
4. Concepts ranked based on their criticality and difficulty.
5. Misconceptions in secure programming better understood.
6. Collection of multiple choice questions designed to identify misconceptions.

1.6 Related Tools

A related tool that aims to close the gap on insecure programming is the Assured Software Integrated Development Environment (ASIDE) [13]. ASIDE is a interactive static code analysis plugin built for Java in Eclipse. ASIDE attempts to provide secure programming support to developers during the actual development phase. So, ASIDE will statically analyze code during development and look for common security mistakes, and provide solutions to fix those mistakes. This differs from SecTutor in that it is only used during development, and ASIDE is geared only toward Java insecurities. SecTutor, on the other hand, is a quiz based learning site that is programming language independent, and can be used in conjunction with regular computer science curriculum (similar to using a

normal tutor, mainly outside of class time to increase areas where students are slipping) to help find and address insecure programming practices before sending students off into industry.

2 Layout

The layout of SecTutor, including how the tool is intelligent, how the users interact with SecTutor, and how SecTutor's model is implemented follows in this section.

SecTutor is implemented as a web app, built using the Python web framework Django. The account model is split into two distinct roles: teachers and students. The general account layout of SecTutor can be seen in Fig. 1. In short, the teacher accounts create questions and view results, and the student accounts take quizzes. An example of a typical question seen in SecTutor follows in Fig. 2.

2.1 How is the Tool Intelligent?

SecTutor uses item response theory [12] to recommend what subject the student should study. By using past quiz scores, an ability level θ is determined for a given interest. This ability level is used in a three-parameter model defined as:

$$P(\theta, a, b, c) = c + (1 - c)\frac{\exp(\theta - b)}{1 + \exp(a(\theta - b))} \tag{1}$$

where a is item discrimination, which is how well the question can discriminate between students of low ability and students of high ability. b is item difficulty, where students with lower ability will have a harder time answering questions with high difficulty. Finally c is item guessing, which accounts for the student merely guessing the answer. The range of a, b, c is between 0 and 1.

Using this value, SecTutor is able to predict a probable score for a student in a given interest. The interest that has the lowest predicted score is the next area of study that SecTutor will recommend for the student.

Initial item difficulty and discrimination has been determined by testing a large and diverse population of students. Question difficulty and discrimination will change over time as the system is used. Newly added questions will determine their difficulty and discrimination when the question has been answered by enough students. Question difficulty and discrimination will change over time with more data.

2.2 The Student's Point of View

When a student account is newly created, the student picks their *interests* and takes a placement quiz. Interests are the main categories that each question belongs to, see Table 1, and the student will only see questions from their interests. The placement quiz is populated by 2 random questions from each of the student's interests. A student can always add or remove more interests.

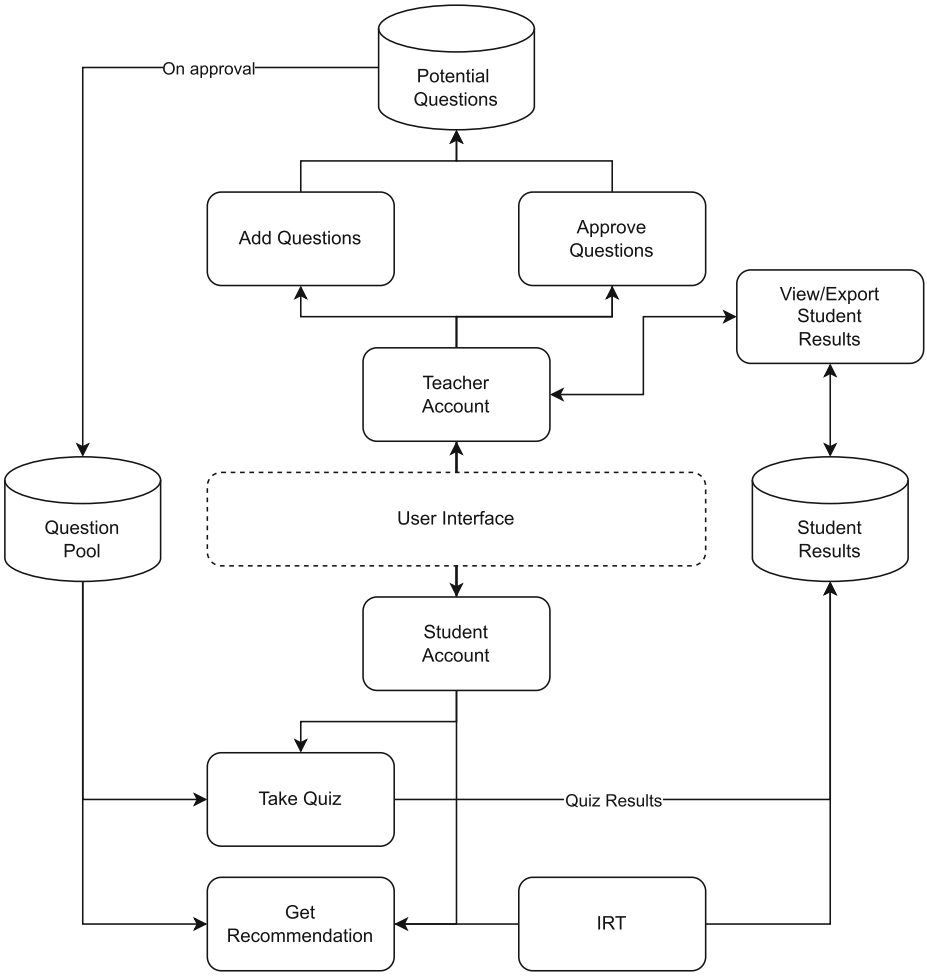

Fig. 1. The layout of the SecTutor system, from the point of view of the user interface. As seen here, teacher accounts may add or approve question to the "potential questions" database. A question is added to the pool after enough approvals, to be used in the student's quizzes. When a students gets a recommendation, item response theory (IRT) is used to determine which area should be studied.

Taking Quizzes. When a student takes a quiz, they first start off by choosing an interest. The quiz is populated with 10 random questions, starting with questions that the student hasn't taken yet. There is no time limit, but the student may not go back to a previous question. Once the student is finished, a score is shown along with quiz results.

In the example question seen in Fig. 2, the last answer is correct. This highlights the fact that if a buffer overflow occurs, both the contents of memory and the control flow may be altered unexpectedly. So, students must understand the

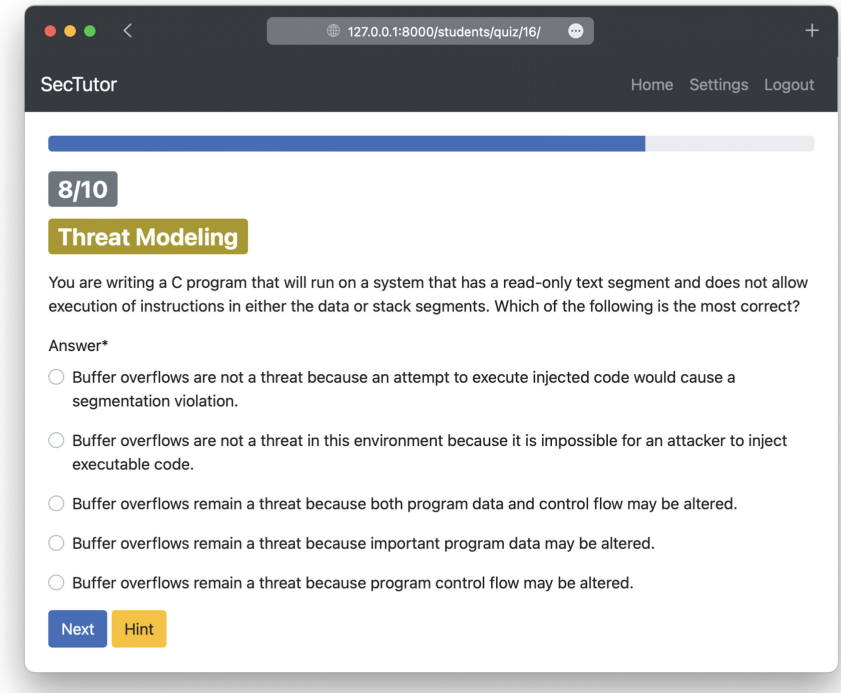

Fig. 2. The student's view while taking a quiz.

attack surface of the program or system to ensure security. This question falls under the "Threat Modeling" misconception, seen in Table 1.

SecTutor is designed in a way where each wrong answer can have custom feedback to further explain to the student why said answer is wrong.

Quiz Subject Recommendation. Each question has both a difficulty and a discrimination value. These values are used in the quiz recommendation functionality of SecTutor. The student's ability level (called θ in the item response theory model, see Eq. 1) is calculated based on a running average of previous question scores in a given interest. So, SecTutor is able to determine what interests the student needs to study next by recommending the interest with the lowest predicted score. If a student decides to add a new interest, and they haven't taken any questions that fall in that interest, then SecTutor will immediately recommend that they take that quiz.

More Resources. SecTutor can link to external resources for each of the student's interests. This external resource takes the form of a concept map created during a past funded project called the secure programming concept inventory

(SPCI) [10]. This provides more reading material for students to study outside of taking quizzes.

2.3 The Teacher's Point of View

The questions that the student accounts see in their quizzes are added by teacher accounts. A teacher account can add new questions, approve potential questions, and view question score performance.

Viewing Student Performance. Teacher accounts have the ability to see question performance on a per question basis. A low average score can be indicative of a difficult question, or of a common misconception. This helps teachers change parts of their curriculum to address low score areas. This data can be viewed or exported.

Question Approval System. Newly created teacher accounts can always add potential questions to the pool. However, until an account has permissions, this newly created account cannot view any other questions. Once the account is granted permissions, they have the ability to view "potential questions", that being questions that have not yet been approved. If the potential question is approved by 2 separate teacher accounts, then the question will be used in the generation of quizzes for students. A question will not appear in a quiz for a student unless it has these 2 approvals.

To help legitimatize a teacher approval, each teacher account has a profile page with stats about their contributions: amount of questions added and amount of questions approved. A teacher may add a short bio to their profile page as well, where teachers are encouraged to add their skills.

The bulk of our questions are added by experts among the field of secure programming. Another round of question brainstorming and approvals is scheduled to happen soon, and we'll be using the SecTutor system to gather and approve these questions.

3 Conclusion

With security being arguably the most important part of software today, SecTutor hopes to understand where students are failing to learn. SecTutor attempts to address the lack of curriculum for common security practices by identifying the weak points.

Since the tool is implemented as a web app via the internet, we hope to reach an audience of thousands of students and assess their secure programming knowledge. With a higher volume of students, our tutoring system will result in a more accurate determination of what misconceptions lie in the field of secure programming. This data can be very valuable to instructors in this field, as they can tune their curriculum to match the most common misconceptions.

With our question approval system, we aren't limited by the current inventory of questions, and can slowly expand the database of questions. This also allows the ability to test out new concepts on a large group of users.

SecTutor can then employ machine learning to understand common behaviors that students have. With more data, SecTutor becomes more calibrated to identify common mistakes employed by students that lead to insecure software.

Acknowledgements. This work was supported by grants DGE-1934279 and DGE-2011175 from the National Science Foundation to the University of California Davis, grant DGE-1934269 from the National Science Foundation to Purdue University, and grant DGE-1934285 to the California State University Sacramento. The opinions, findings, and conclusions, or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the California State University, Purdue University, and the University of California Davis.

References

1. Almansoori, M., et al.: How secure are our computer systems courses? In: Proceedings of the 2020 ACM Conference on International Computing Education Research, pp. 271–281. ACM, New York (2020). https://doi.org/10.1145/3372782.3406266
2. Bransford, J.D., Brown, A.L., Cocking, R.R. (eds.): How People Learn: Brain, Mind, Experience, and School. National Academy Press, Washington DC, USA, expanded edn. (2000)
3. Caceffo, R., Wolfman, S., Booth, K.S., Azevedo, R.: Developing a computer science concept inventory for introductory programming. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pp. 364–369. ACM, New York (2016). https://doi.org/10.1145/2839509.2844559
4. Dawson, M., Burrell, D.N., Rahim, E., Brewster, S.: Integrating software assurance into the software development life cycle (sdlc). J. Inf. Syst. Technol. Plann. **3**(6), 49–53 (2010). https://www.researchgate.net/publication/255965523_Integrating_Software_Assur-ance_into_the_Software_Development_Life_Cycle_SDLC
5. Garrison, D.R.: Self-directed learning: Towards a comprehensive model. Adult Educ. Q. **48**(1), 18–33 (1997). https://doi.org/10.1177/074171369704800103
6. Help Net Security: 70% of organizations recognize the importance of secure coding practices, March 2021. https://www.helpnetsecurity.com/2021/03/26/secure-coding-practices/
7. Hestenes, D., Wells, M., Swackhamer, G.: Force concept inventory. Phys. Teach. **30**(3), 141–158 (1992). https://doi.org/10.1119/1.2343497
8. Hyder, J.: Electronics systems concept inventory. http://www.esyst.org/PDF/Concept%20Inventory%20Presentation.pdf
9. Lam, J., Fang, E., Almansoori, M., Chatterjee, R., Soosai Raj, A.G.: Identifying gaps in the secure programming knowledge and skills of students. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education, vol. 1, pp. 703–709. ACM, New York (2022). https://doi.org/10.1145/3478431.3499391
10. Ngambeki, I., Nico, P., Dai, J., Bishop, M.: Concept inventories in cybersecurity education: an example from secure programming. In: Proceedings of the IEEE Frontiers in Education Conference (FIE), pp. 1–5 (2018). https://doi.org/10.1109/FIE.2018.8658474

11. Sherman, A.T., et al.: The cats hackathon: creating and refining test items for cybersecurity concept inventories. IEEE Secur. Priv. **17**(6), 77–83 (2019). https://doi.org/10.1109/MSEC.2019.2929812
12. Tay, L., Huang, Q., Vermunt, J.K.: Item response theory with covariates (IRT-C): assessing item recovery and differential item functioning for the three-parameter logistic model. Educ. Psychol. Meas. **76**(1), 22–42 (2016). https://doi.org/10.1177/0013164415579488
13. Zhu, J., Xie, J., Lipford, H.R., Chu, B.: Supporting secure programming in web applications through interactive static analysis. J. Adv. Res. **5**(4), 449–462 (2014). ISSN 2090–1232. https://doi.org/10.1016/j.jare.2013.11.006