Proceedings of the 14th Colloquium for Information Systems Security Education
Baltimore Marriott Inner Harbor
Baltimore, Maryland June 7 - 9, 2010

Injecting Security in the Curriculum – Experiences in Effective Dissemination and Assessment Design

Siddharth Kaza, Blair Taylor, *Towson University*, Harry Hochheiser, *University of Pittsburgh,* Shiva Azadegan, Mike O'Leary, *Towson University*, Claude F. Turner, *Bowie State University*

Abstract – The security injections project at Towson University proposes to "inject" security across the foundational and upper-level courses at universities and community. To achieve this, we and our partner institutions design and develop a series of strategically-placed, security-related, self-contained modules to be used in classes. An easy to use web portal serves as the repository and dissemination medium for modules targeted at computer literacy, CS1, CS2, and other courses. To date, this project has reached over 1000 students and we have held training workshops attended by 45 instructors at 5 institutions. Assessment instruments for student learning have been designed and controlled experiments with 19 classes in three institutions have shown promising results.

Index terms – security injection modules, secure coding, lab modules, security integration

I. Introduction

There is little doubt that security needs to be integrated into the computer science curriculum. The questions of concern are (1) How to integrate security in an already crowded curriculum? (2) What topics need to be included? and (3) Where and how often are the students exposed to security principles? These questions have been addressed by previous research in many innovative ways, including new curriculum [1][2], security tracks [3], thread-based approaches [4,5][6], and single courses. While each approach has its merits and is applicable to certain environments and constraints that a Computer Science (CS) department might face, thread-based approaches have shown much promise as the best trade-off between targeting the maximum number of students and using minimum resources [4,6,7].

The thread-based approach is based on the principle of "security being a process not a product" [8]. The premise is including a small 'module' in each course that addresses security concerns without incurring the cost of creating a special track or degree program [6]. Other advantages of the approach include potentially less instructor training, less time devoted in already tight course schedules, and repeated exposure of students to security concepts. For the past three years at Towson University we have worked with the thread-based approach extensively to inculcate secure coding concepts in our undergraduate CS students. Although we have had a strong security track since 2002, we believe that a thread-based approach to introduce security will prepare all students for the 21st century [5].

However, this approach is not without challenges. Work needs to be done in the development of self-contained, self-explanatory, and experiential learning components that will challenge students without requiring time from instructors. To be broadly applicable, a good approach should be generalizable across two-year institutions, small colleges, and universities alike. However, institutions vary widely in pedagogy methods, languages used, lab environments, and teacher motivation, and developing materials to cater to a broad audience is challenging [9]. Another issue lies in the dissemination and assessment model. Effective dissemination should engage faculty and facilitate active adoption of materials, while increasing student learning and retention. In this paper, we address these issues.

We build upon the "security injections" model at Towson University that proposes to "inject" security across the foundational and higher-level CS courses at Towson along with disseminating materials to several other universities and community colleges in Maryland [10]. The primary goals of this project include:

1. Increasing security awareness among students
2. Improving students' ability to apply security principles
3. Increasing faculty security awareness
4. Increasing the number of security-skilled students

In previous work, we have described the results of an initial study to assess student learning in two classes [4], presented security checklists (one of the cornerstones of our approach) [5], and outlined our preliminary designs to address security in higher-level courses [10].

ISBN 1-933510-99-4

Security Checklist	
Vulnerability: Buffer Overflow **Course:** CS1	
Task - Check each line of code	Completed
1. Finding Arrays:	
1.1 Underline each array declaration	
1.2 For each array, underline all subsequent references	
2. Index Variables – *legal range for an array of size n is 0 <= i < n*	
2.1 For each underlined access that uses a variable as an index, write the legal range next to it.	
2.2 For each index marked in 2.1, underline all occurrences of that variable.	
2.3. Mark with a V any assignments, inputs or operations that may modify these index variables.	
3. Loops that modify index variables	
3.1 Find loops that modify variables used to index arrays. For any index that occurs as part of a loop conditional, underline the loop limit. For example, if *i < max* is the conditional in a for *loop*, underline *max*	
3.2. Write the legal range of the array index next to the loop limit as you did in step 2.1. Mark with a V if the loop limit could exceed the legal range of the array index. Watch out for loop that go until *i <=max* , as the largest valid index is *max-1*	
3.3 If the upper or lower loop limit is a variable, it must be checked just as indices are checked in Step 2	
Highlighted areas indicate vulnerabilities!	

Figure 1. An example checklist for buffer overflow from the CS1 security injection

This paper presents our experiences with disseminating security injections across five institutions, our formal design for assessment, and the initial results of controlled experiments with 19 course sections. The remainder of the paper is organized as follows: Section II presents the project overview and summary of our previous work, Section III describes the dissemination strategy and the web portal. Section IV presents the assessment design and results and Section V concludes and discusses future directions.

II. Project Overview

The project at Towson University is based on the premise that security education reaches too few students, too late into their academic training. We work on integrating security in all classes starting from introductory programming and computer literacy to instill secure coding and other security aware habits in students before they are set in their programming styles. To achieve this, we have designed and developed a series of strategically-placed, security-related, self-contained modules to be used in classes. Our collaborative approach to developing the modules across multiple institutions has led to a formal iterative model that includes: developing materials, piloting across institutions, formal assessment, evaluation, and revision [10] .

1. Security Injection Modules

Security injections modules are lab-based modules that target specific security issues. These modules have evolved with input from three sources: evaluation and extensive review from our independent evaluators, valuable feedback from our partners who are experienced in teaching these courses to a wide range of students, and our own experiences with ongoing pilots. Our project includes an independent evaluator, Robert Seacord of Carnegie Mellon University's Computer Emergency Response Team (CERT), an expert in secure coding, who examines all materials for technical content.

As described in detail in a previous paper [4], each module contains a background section, a lab exercise centered on a security checklist, and related discussion questions. The background section contains a summary of the security vulnerability, a description of the problem and the associated risk. It also includes a documented real-world incident that was caused by the security issue. When appropriate, the modules also contain short code snippets that demonstrate the vulnerability. The lab component includes programs that target the vulnerability by demonstrating the problem and the solution.

An important component of the injection module is the security checklist. The checklists [5] are designed to give students a clear and repeatable procedure to help identify security vulnerabilities in code. As the project progresses, we are improving the checklists to enhance coverage of other security vulnerabilities in other courses. Currently, our checklists target integer overflow, buffer overflow, and input validation for Introduction to Programming Logic (CS0), Introductory Computer Science I (CS1), Introductory Computer Science II (CS2), and phishing, encryption, and passwords for Computer Literacy. Figure 1 shows an example checklist for buffer overflow that is part of a security injection for CS1. Most checklists are language independent.

2. Expansion to other courses and institutions

As the project progresses, the security injection modules are being expanded to include other courses at a range of institutions [10][11]. The courses currently in development and/or planned for the future include –Web Development, Database Design and Development, and Networking. Injections in these courses will use the common template to cover security concerns relevant to specific course materials. Modules for Networking will discuss protocol design and network configuration.

Injections for Database will cover SQL injections, input validation, database design concerns, and access control. Similar issues will be addressed in the Web Development, including cross-site scripting attacks, cross-site request forgery, and other specific web vulnerabilities.

Although initial results have been encouraging [4, 5], our goal of demonstrating a broadly-applicable approach to computer security education requires adaptation to other, different institutions. The project involves faculty members from diverse institutions in the Maryland Alliance for Information Security Assurance (MAISA, a consortium of 15 institutions in Maryland) including Towson (TU), Bowie State (BSU), and community colleges from Baltimore (CCBC), Harford (HCCC), and Anne Arundel (AACC) counties. Materials are being developed primarily at Towson and Bowie State, with faculty collaborators from the other institutions playing an active role in discussing content areas, deploying materials, and assessing results.

III. Dissemination and Web Portal

A. Dissemination to Faculty

The lack of trained faculty is a recognized problem in security education [13]. Since the importance of security has emerged in the last decade, many instructors are untrained in developing secure software. A major goal of this project is to produce a population of more security-aware faculty. The security injection modules 1) provide up-to-date and relevant information to help faculty become more aware of the important security issues and 2) are ready-to-use materials easily incorporated into existing assignments.

To launch the security injections across all five institutions, we implemented a comprehensive set of workshops/ training sessions taking place at the beginning of each semester. We have found that a key factor to a successful workshop is location choice. After the initial kickoff sessions at Towson University, subsequent workshops took place at each partnering institution to maximize participation at each college. Our grant includes small financial incentives to promote workshop attendance. The goal of each workshop is to interest faculty in our project and encourage them to include the security injections in their classes. The agenda includes open discussions and breakout groups to promote feedback, reflection, and sharing of teaching strategies. The format also helps to integrate faculty early in the development process and identify factors that work well across different demographic groups. In addition, we encouraged faculty involvement throughout the process to enhance instructor buy-in and active adoption of materials. Since instructors played a key role in developing the modules, we hope they are motivated to use them in their own classes and recommend to their colleagues. Additionally, we gain vital feedback that we used to revise our materials. Since the inception of this project, 45 instructors at five institutions (see Table 1) have participated in workshops, raising their level of awareness in the critical area of security and secure coding.

Table 1. Faculty workshop attendance

Date	Location	Attendance
Fall 2008	Towson	10
Jan 2009	Towson	17
Fall 2009	AACC	18
Jan 2010	CCBC	12
Feb 2010	Bowie State	11

B. Dissemination to Students

The key to reaching more students begins with the faculty, as one instructor connects with many students. Additionally, by injecting security into the required core courses, we are extending security concepts well beyond the security track to all students in the major. Our collaboration with varied institutions increases our student base and yields highly extensible curriculum materials that will be appropriate for diverse students. By making the modules available to our local security education cohort of MAISA [11] and sharing the results with our colleagues at CISSE and SIGCSE, we can spread the integrated security curriculum to a wide audience.

To increase skills in our students, our modules are introduced in CS1 (or CS0, if applicable). Injection topics are repeated with increased complexity in CS0, CS1, and CS2. Upon completing the core course sequence, a computer science student is exposed up to three times to security injections on the critically important topics. This concept of repeated exposure ensures that students fully synthesize these important concepts.

In addition to expanding security education to reach more students in the major, we have developed a comprehensive set of modules for the computer literacy classes. These classes, which include students of all majors, have a large and varied demographic. Perhaps due a more flexible syllabus, instructors of these classes have been very receptive to expanding the use of the materials.

C. Web Portal

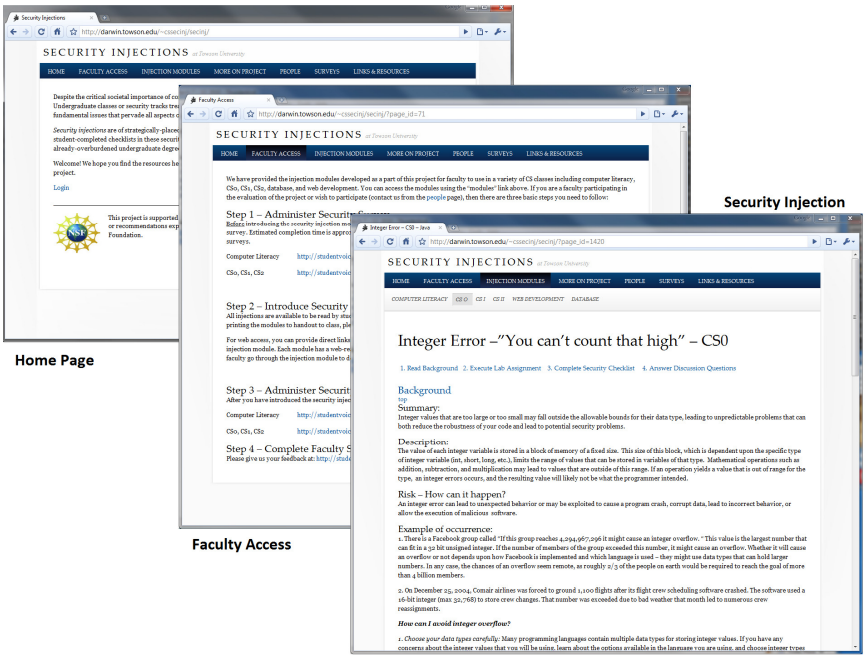

Figure 2. A snapshot of the web portal

The primary vehicle for dissemination and the resource repository for this project is a web portal (http://triton.towson.edu/~cssecinj/). After several iterations of web platforms, we chose the public publishing platform WordPress (http://wordpress.org/) because of its ease of use and professionally styled templates. Establishing one central location for all materials has reduced the need to send frequent emails to participants and allows convenient access for faculty, students, and other interested parties. Our portal includes the following components (Figure 2):

- *Home* – this section briefly describes the project.
- *Faculty Access* – this section outlines the four steps for faculty participation and includes links to the pre-surveys and post-surveys. The aim is to reduce the cognitive load on faculty involved in the project by serving as a one-stop source.
- *Injection Modules* – this section will eventually house all security injections. This section also includes the answers for modules in password protected pages. Faculty can refer students directly to the website or copy and paste portions of the modules directly into their existing materials.
- *More on Project* – this section includes further description of the security injections, workshop information and slides, and the overall project schedule.
- *People* – this section lists project participants, evaluators, and workshop attendees.
- *Surveys* – this section includes links to pre- and post-surveys which are accessible to both students and faculty. Surveys are administered online and results are collected using Student Voice (http://studentvoice.com). These surveys are key evaluative tools used to assess the demographics and security awareness across institutions. Surveys are discussed in more detail later.
- *Links and Resources* – this section includes
    - RSS feed - a graduate project culminated in this live feed which displays current news events relevant to secure coding.
    - Publications and tools – useful links and project publications.

The web portal is designed to facilitate faculty and student participation in the project. Future additions to the portal include: a forum for instructors to provide feedback and suggest enhancements and a repository for quantitative and qualitative results, including summaries from conference group sessions.

IV. Assessment Design and Results

The four primary goals that need to be assessed for this project are: increasing students' security awareness, improving students' ability to apply security principles, increasing faculty security awareness, and increasing the number of security-skilled students. Each of these objectives is assessed using various instruments including surveys, random sampling of assignments, qualitative inputs from faculty, controlled experiments in classrooms, and institutional quantitative data. In this paper, we present the design and the first year results (Spring 2009)

for the first and fourth objectives. As the project progresses, we plan to present further assessment results.

A. Increase in student security awareness – assessment design

We aim to increase student awareness using two strategies 1) exposing students to security injection modules in each of the foundational classes they take, 2) develop the injections such that students are repeatedly exposed to the same security concerns as they progress through their degree. Assessing both these strategies need different but related evaluation plans.

As part of the assessment design, classes were divided into two groups – integrated sections and control sections. Integrated sections are the classes that use security injections and control sections refer to classes that do not use injections but are included for statistical control. In both groups, students are administered a pre-survey at the beginning of the class) and a post-survey (at the end of the class). The survey instruments are designed carefully to assess the general security awareness and secure coding principles students gain from the injections. Our independent evaluator, Dr. Kathryn P. Doherty, an expert in assessment, reviews all assessment materials, including pre-surveys, post surveys, faculty surveys, and coding exercises. The student surveys contain demographic questions and two sets of multiple choice questions – one section targets general security awareness and the other focuses on specific knowledge gained through the injections. The general security awareness questions in the computer literacy and programming classes are the same, while the questions specific to the security injections are different in computer literacy and CS0, CS1, CS2 (a list of sample questions is shown Table 2). An initial version of this survey was piloted earlier and was well accepted by students [4]. As modules are developed for other courses, we plan to develop more targeted instruments.

Table 2. Survey questions from CS0, CS1, and CS2

General Security Awareness
What are the possible consequences of insufficient computer security?
Phishing is …
The conversion of data into a ciphertext that cannot be easily understood by unauthorized people is known as …
When developing secure systems, where does security fit in?
A set of related programs, usually located at a network gateway server, that protects the resources of a private network from other networks, is known as a …
Which of the following is an example of a strong password?
How interested are you in security …
How important do you think security knowledge is to your future career?
Secure Coding
Invalid input can come from the …
Which of the following should your well-designed program do before processing user input?
Which programming mistake is one of the major vulnerabilities in today's applications?
When developing secure systems, where does security fit in?
Security Software and Software Security are the same thing (true/false)
Integer overflow is caused by …
Integer overflow occurs …

Based on the pre-survey and the post-survey, a set of three hypotheses are proposed to test the knowledge gain and the need for security injections:

H1: The post-survey scores will be significantly higher than pre-survey scores in integrated sections.

H2: The post-survey and pre-survey scores in control sections will show no significant difference.

H3: The post-survey scores of the integrated sections will be significantly higher than the post-survey scores of control sections.

H3a: The pre-survey scores of both sections will show no significant difference (this will imply that initial student knowledge in both groups is the same).

Assessing the repeated exposure strategy poses a different set of challenges. The primary issue is identifying students who were exposed to injections in multiple classes. This is logistically difficult task. As the project progresses, each upper-level class will contain a mix of students who were exposed to injections in previous classes and those who were not. We have addressed this using three strategies, 1) assign anonymous id numbers to students (functionally based on their birth dates) on the surveys to identify students who have taken the survey before, 2) ask students about their course history in the survey, and 3) administer the survey in our Senior Seminar (required of all graduating computer science students at Towson). We expect that as students are repeatedly exposed to security injections, they will show higher scores in the security survey as compared to our current graduating class (with less security injection exposure).

H4: The post-survey scores for integrated sections will be higher than scores for the graduating class of Spring 2009.

B. Increase in student security awareness – assessment results

Here we present the results for CS0, CS1, and CS2 sections that used security injections modules across three

institutions in Spring 2009. After data cleaning (removing incomplete surveys, dropped students, etc), a total of 388 student survey responses in the three institutions were used with 308 exposed to the security injections and 80 as control participants. Even though we made attempts to obtain comparable numbers for the integrated sections and non-integrated ones – it was difficult to achieve due to low instructor enthusiasm in the non-integrated sections. We plan to address this issue in future experiments.

1. Comparison for pre-survey and post-survey scores for integrated sections

There were 308 responses to the survey in integrated sections with 182 responses in the pre-survey and 126 in the post-survey. The scores were calculated as the number of questions correctly answers out of a total of 14 questions. The Kolmogorov-Smirnov and Shapiro-Wilk test showed that the scores were not normally distributed, so the Mann-Whitney non-parametric test was used to compare them. It was found that the scores significantly increased ($p < 0.001$) in the integrated sections. This implies that the security injections might be working and replicates previous results [4] although with a larger and more diverse dataset. These results look promising and support H1.

2. Comparison for pre-survey and post-survey scores for control sections

A total of 80 responses were obtained with 61 in pre-survey and 18 in post-survey. Using the same tests, we found that there was no significant difference (p=0.107) between the scores in the control sections. Thus, students did not gain knowledge on security in these classes (without the injections). This points to the need for security injections in the classes and supports H2.

3. Comparison of control and integrated sections

Figure 3 shows the pre-survey and post-survey scores of both control and integrated sections. As can be seen, both sections had higher scores in the post-survey. We also found that the pre-survey scores of the control sections and the integrated sections showed no significance difference (p = 0.331) that supports H3a. However, the post-test scores – which were expected to show a difference – did not show significance either (p = 0.341). H3 was not supported. Further examination of this is needed as the number of respondents to the post-survey was low (18 students). As the number of institutions and courses grow in Fall 2009 and Spring 2010, we expect the number of respondents to increase.

Another challenge in statistical analysis is that the integrated sections may also have variance in number and the details in which the injections were used. This might affect student learning. This problem might be alleviated as the injections become more defined and accepted by instructors. Indeed, repeated exposure may also alleviate this problem as students will see the missing details in future modules on the same topic.

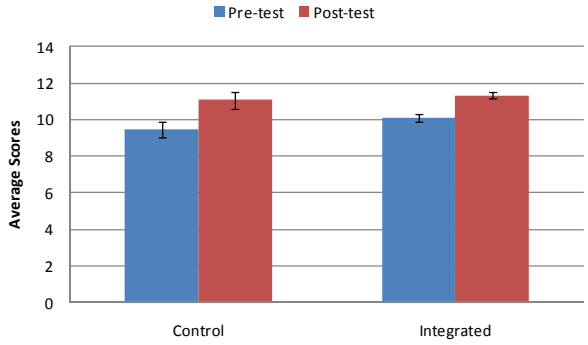

Figure 3. Comparison of control and integrated sections

3. Comparison of integrated sections with graduating class of Spring 2009

There were a total of 187 responses examined for this with 61 from the graduating class and 126 from CS0, CS1, and CS2. As Figure 4 shows, the scores for the integrated sections were significantly higher (p<.006). This supports H4 and shows that security injections may be helping in increasing securing awareness. We expect that as students exposed to injections in their classes reach the Senior Seminar, the results from that course will show an improvement. In addition, we plan to study the results of the survey by categories of questions (general security vs. secure coding) to better examine the difference in scores.

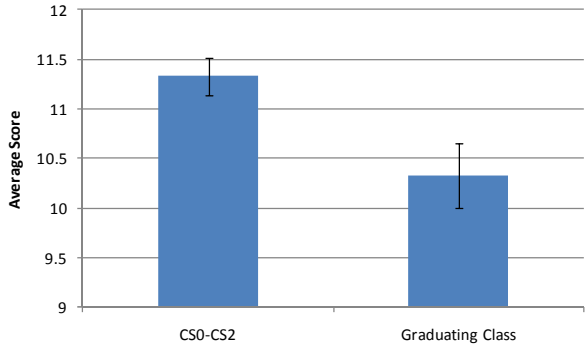

Figure 4. Comparison of integrated sections with graduating class

C. Increase in the number of security-skilled students – assessment results

Our project started in Fall 2008 with the development of the injection modules and the initial portal. In addition, preliminary versions of the modules were deployed at Towson University CS0 and CS1 sections. Previous dissemination results are reported by Taylor and Azadegan [4] . However, full scale pilot testing of the modules across four institutions began in Spring 2009 and Table 3 shows the basic statistics.

Table 3. Dissemination statistics - Spring 2009

Institution	Courses (sections)
Towson University	
	CS0 (3)
	CS1 (4)
	CS2 (4)
Bowie State University	
	CIS0 (3)
	CS1 (1)
Harford Community College	
	CIS0 (3)
	CS1 (1)

As reported in the previous sub-section, a total of 308 students were exposed to the security injections across the three institutions. We are currently analyzing the data for Fall 2009, however, the number of institutions and course participation has increased significantly to encompass approximately 1000 more students.

V. Conclusion and Future Directions

The security injection model at Towson University aims to incorporate security principles across foundational to upper-level CS courses via strategically-placed, security-related, self-contained lab modules. This paper presents our experiences with disseminating security injections,, our formal design for assessment, and the initial results of controlled experiments. Through workshops, we have reached 45 instructors, raising their level of awareness in the critical areas of security and secure coding. The project has also reached over 1000 students across five diverse institutions. We have carefully designed assessment strategies to evaluate the efficacy of our materials. Experiments results from 19 course sections show that repeated exposure to security injections improves knowledge of secure coding principles. All resources from this project are available to faculty, students, and the entire CS community through a web portal (http://triton.towson.edu/~cssecinj).

In the future, we plan to develop additional security injections for high-level courses, reach more faculty through additional workshops, and deploy the modules across various Maryland universities and community colleges. In addition, further assessment and evaluation will be performed.

VI. Acknowledgements

This project is partially supported by the NSF Course Curriculum and Laboratory Improvement (CLLI) grant number DUE-0817267. We greatly appreciate our partners at Bowie State University, Harford County Community College, Community College of Baltimore County, and Anne Arundel Community College. In particular, we are grateful to AC Chaplin, Patricia Gregory, Jack McLaughlin and colleagues at Towson University for their contributions.

VII. References

[1] T. Bacon and R. Tikekar, "Experiences with developing a computer security information assurance curriculum," *Journal of Computing Sciences in Colleges*, vol. 18, 2003, pp. 254 - 267.

[2] B. Bogolea and K. Wijekumar, "Information security curriculum creation: a case study," *Information Security Curriculum Development*, ACM, 2004, pp. 59-65.

[3] S. Azadegan, M. Lavine, M. O'Leary, A. Wijesinha, and M. Zimand, "An undergraduate track in computer security," *8th Annual Conference on Innovation and Technology in Computer Science Education*, ACM, 2003, p. 207–210.

[4] B. Taylor and S. Azadegan, "Moving beyond security tracks: integrating security in CS0 and CS1," *SIG Computer Science Education (SIGCSE)*, ACM, 2008.

[5] B. Taylor and S. Azadegan, "Using Security Checklists and Scorecards in CS Curriculum," *National Colloquium for Information Systems Security Education*, 2007, pp. 4-9.

[6] L. Perrone, M. Aburdene, and X. Meng, "Approaches to undergraduate instruction in computer security," *Annual Conference of the American Society of Engineering Education*, 2005.

[7] A.J. Wang, "A security thread in a thread-based curriculum," *Conference On Information Technology Education (formerly CITC)*, 2008.

[8] B. Schneier, *Secrets & lies: digital security in a networked world*, New York, NY: John Wiley & Sons, 2000.

[9] W. Du and R. Wang, "SEED: A suite of instructional laboratories for computer security education," *ACM Journal on Educational Resources in Computing (JERIC)*, vol. 8, 2008.

[10] B. Taylor, H. Hochheiser, S. Azadegan, and M. O'Leary, "Cross-site Security Integration: Preliminary Experiences across Curricula," *13th Colloquium for Information Systems Security Education (CISSE)*, pp. 158-165.

[11] C. Turner, H. Hochheiser, J. Feng, B. Taylor, and J. Lazar, "Cooperative Information Assurance Capacity Building," *13th Colloquium for Information Systems Security Education (CISSE)*, pp. 44-50.

[12] W.S. Harrison, N. Hanebutte, and J. Alves-Foss, "Programming Education in the Era of the Internet: A Paradigm Shift," *Hawaii International Conference on System Sciences (HICSS)*, IEEE, 2006, p. 219.2.