Utilizing Real-World Software Vulnerabilities to Enhance Secure Programming Education

Denise Daniels
Virginia State University
Petersburg, Virginia, U.S.A.
dcdaniels@vsu.edu

Joon Suk Lee
Virginia State University
Petersburg, Virginia, U.S.A.
jlee@vsu.edu

Hui Chen
CUNY Brooklyn College
Brooklyn, New York, U.S.A.
hui.chen@brooklyn.cuny.edu

Kostadin Damevski
Virginia Commonwealth University
Richmond, Virginia, U.S.A.
kdamevski@vcu.edu

***Abstract*—This research paper describes a study of using real-world vulnerabilities to motivate computer science students towards learning secure programming. Given the rise in cybersecurity incidents due to programming errors, there is a pressing need to improve programmers' secure programming skills. Despite educators' numerous efforts towards this goal, communicating the importance of this training to students remains a challenge. Grounding on the theory of intrinsic motivation, we propose that exposing students to authentic, relatable vulnerabilities can significantly enhance their learning orientation towards secure programming. Our approach involves selecting vulnerabilities from the National Vulnerability Database that are both relatable to students and understandable without extensive external context. These vulnerabilities are transformed into comprehensive course modules, each featuring a demonstrative video, source code snippets of the vulnerability and its patch, and associated developer communications about the vulnerability. We assess the impact of one of our course modules on students' learning disposition through a study conducted in two universities in an identical setting. The study results indicate that students appreciate seeing real-world vulnerabilities in detail, especially the video we recorded reproducing the vulnerability, and that they gain in self-efficacy after completing the module.**

***Index Terms*—5.b.vii. Computer science; 8.c. Computing skills; 8.u. Student perception; 12.d.iii. Mixed methods research**

I. Introduction

Cybersecurity attacks and incidents are increasing at an alarming rate [1], [2]. The Software Engineering Institute attributes nearly 90% of reported cybersecurity incidents to programming errors, i.e., defects in the design or implementation of software [3]. Improving programmers' secure programming skills can lead to more robust programs [4], with significantly fewer attack surfaces.

In recent years, the integration of secure programming into the introductory curriculum of computer science courses has become increasingly prevalent [1], [2]. While this integration marks a positive step towards enhancing programmers' abilities to create more secure software [4], a significant challenge remains in ensuring that students not only learn these principles but also effectively apply them in practical programming situations and real-world contexts. Despite some initial exposure to secure programming, many students struggle to internalize and utilize this knowledge beyond the classroom. This gap is evident as students often fail to continue improving and applying their secure programming skills, particularly when faced with new programming tasks [5]–[7]. To address this issue, it is essential to develop an educational model that not only imparts secure programming knowledge but also fosters a strong learning disposition. This involves cultivating the will or orientation to continually learn and apply secure programming principles, thereby bridging the gap from scaffolded classroom examples to novel programming challenges.

The theory of intrinsic motivation indicates the factors like contextualization, personalization, and learning choices can enable learners to improve the depth of engagement in learning, the amount learned in a fixed time period, and the perceived competence and levels of aspiration [8], [9]. Following this, we focus on increasing students' exposure to real-world vulnerabilities and connecting these to students' prior life experiences or daily lives, in order to make the learning experience more meaningful and directly applicable. We hypothesize that this strategy can increase the students learning orientation towards secure programming and increase their self-efficacy towards this task (and the overall cause of secure programming) due to highly contextualized and personally relevant learning experiences.

More specifically, in this paper we aim to answer the following research question (RQ): *Does the authenticity and personal relevance of the real-world software vulnerabilities improve the students' 1) self-efficacy in secure programming and 2) awareness of the importance of security in programming?* Support for our hypothesis comes from the theory of learning engagement and the competency-based education model [8]–[11], i.e., we expect learners to develop a strong learning disposition for secure programming from personally relevant content.

To answer this RQ, we first curate real-world security vulnerabilities from the National Vulnerability Database (NVD) that are: 1) relatable and likely to be personally relevant to students; 2) accompanied by source code for both the vulnerability and the patch; 3) understandable without excessive context about the project or external software libraries. Based on each curated vulnerability, we create course modules[1] that highlights the real-world impact of the vulnerability by providing the following content: 1) a video recording reproducing the vulnerability and showing how it is exploited; 2)

[1] https://realvulnerabilityedu.github.io/modules/.

snippets of source code showing the vulnerable and fixed code with explanations; and 3) additional documents from the software developers that reported the vulnerability highlighting their communications to other software users. We study the impact of the course module via "pre- and post-test" quasi-experimental design method [12] where we use a pre/post survey of students in two different universities. The results of our study indicate that learning more about real-world vulnerabilities that are relatable to students improves their self-efficacy (e.g., improving their agreement with the statement *"I understand what secure programming is."*)

II. Related Work

There has been a growing importance of cybersecurity education in computer science [13]. Cybersecurity represents a broad spectrum of knowledge and skills, encompassing both technical and non-technical areas [14]. This paper focuses on secure programming, a technical area within the broader fields of robust or defensive programming, which focuses on designing and implementing resilient software. For secure programming, robustness is defined with regard to security policies [4].

There is a rich body of literature on teaching secure programming [6], [15]–[21]. A few studies demonstrate approaches to enhance computer science courses and curriculum, such as integrating secure programming concepts into the introductory sequence, elective and core computer science courses, and the entire curriculum [15]–[19]. Accompanying this is the creation of secure programming modules, labs, or tools aimed at improving students' awareness, knowledge, and skills in secure programming [22]–[26]. Table I summarizes some of these resources. Beyond this are the development of novel pedagogy in secure programming. For instance, out-of-classroom secure programming "clinics" where experts provide diagnostic reviews of students' code are shown to be effective in improving secure programming awareness and practice [6]. Also, automated methods to provide in-time feedback to students about security problems in their program code have been explored as effective tools for learning secure programming [24], [27]. Segmenting the modules and incorporating interactive elements could enhance student completion rates, engagement, and motivation by reducing cognitive overload that can result in students' skipping valuable information [20].

In addition, researchers have also made progress in understanding the students and in improving the students' readiness to learn secure programming. For instance, Siraj et al. [28] suggest that we should teach students to have a security mindset in addition to secure programming knowledge and skills. Lam et al. [29] found gaps in students' knowledge and skills in securing programming. A particular gap is the lack of awareness of security vulnerabilities or their effect on their code. Slusky et al. [30] demonstrated that students' lack of security awareness originates from their struggle to apply security knowledge in real-world scenarios.

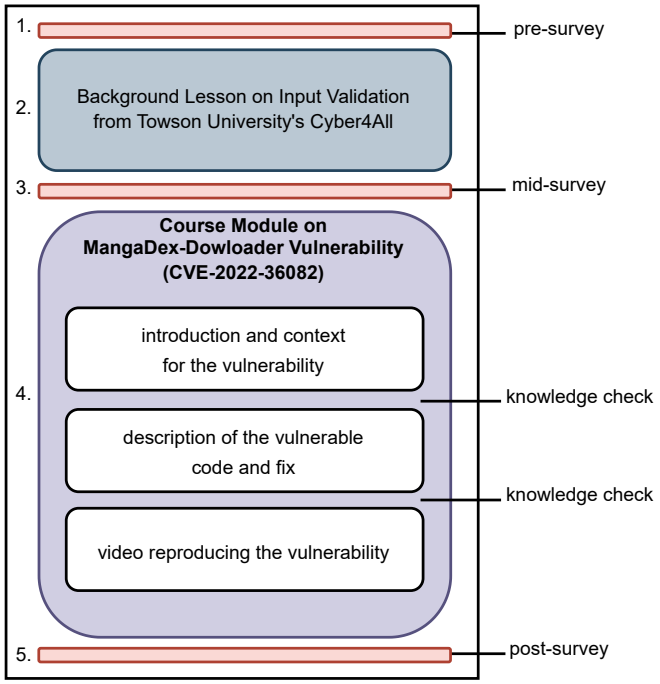


Fig. 1: Structure of the course module based on CVE-2022-36082.

As discussed in Section I, students suffer from the inability to improve and apply secure programming knowledge and skills [5], [6]. Interestingly, this observation can also be extended to professional developers [7]. In order to address this, we set out to develop a pedagogical approach and course modules that can cultivate the will or orientation to continually learn and apply secure programming principles by taking advantage of real-world vulnerabilities and connecting these to students' prior life experiences or daily lives. This approach distinguishes our paper from prior work.

We argue that by exposing students to real-world security vulnerabilities that are relevant to their daily experiences, we can instill in students the importance of secure programming practices, enrich their security mindset, and empower them to apply this knowledge in their programming.

III. Course Module Design

We selected a specific course module to use in our study that focuses on one of the most common software security problems: improper input validation. Vulnerabilities based on improper input validation allow an attacker to take advantage of software that does not check if potentially dangerous inputs are safe for processing. We developed a Python script that examined the publicly disclosed vulnerabilities assigned to the category: *CWE-20: Improper Input Validation* [31] for the following criteria: 1) the vulnerability has a publicly-accessible GitHub link to the patch (i.e., fix); 2) had a severity score (based on CVSS-3.0 [32]) of Medium, High, or Critical; 3) was published in the last 3 years. From the retrieved set, we manually examined the vulnerability descriptions searching for

TABLE I: Overview of several popular computer security educational resources.

Name	Description
SEED Lab [22]	Over 30 publicly-available labs targeted at upper-level undergraduate and undergraduate computer and information security courses based on several known exploits, among which 6 are about software security
Cyber4all / Security Injections [23]	Security modules for integrating security across the curriculum; model for teaching secure coding to introductory programming students, and SPLASH (Secure Programming Logic for college credit to high school girls). A number of lessons can be found in their publications and on their Website.
Nice Challenge Project [25]	Projects with a narrative-driven scenario, a business environment (workspace), and a set of technical objectives and/or a written deliverable.
Cyber Threat Hunting [26]	4 cyber threat hunting labs
ES-IDE [24], [27]	1 SQL injection exercises and tutorials for EIDE, an Eclipse IDE plugin for teaching secure programming

projects or application domains that students could relate to (e.g., games, popular mobile apps). We finally selected *CVE-2022-36082* [33] as it concerned Manga, which are Japanese comic books that we believed students would find relatable. More specifically, CVE-2022-36082 has the following description in the National Vulnerability Database:

> *mangadex-downloader is a command-line tool to download manga from MangaDex. When using 'file:<location>' command and '<location>' is a web URL location (http, https), mangadex-downloader between versions 1.3.0 and 1.7.2 will try to open and read a file in local disk for each line of website contents. Version 1.7.2 contains a patch for this issue.*

We framed the course module around both *CWE-20: Improper Input Validation* and *CVE-2022-36082*. The structure of the module is shown in Figure 1. We hosted the secure programming course module on Notion, an online platform commonly used for note-taking, project management and organization, and website creation. The course module consisted of 5 parts: 1) pre-survey, 2) lesson on Improper Input Validation from Cyber4All (first 3 out of 5 sections: Background, Code Responsibly and Laboratory Assignment) [23], [34], 3) mid-survey; 4) our course module focused on CVE-2022-36082; and 5) post-survey. All of the content was either directly on Notion[2] or embedded content within a Notion page, i.e., participants did not need to leave Notion to access any content, including the surveys. All survey pages consisted of an embedded Google form for each survey to be completed by the participants.

Our content focused on the MangaDex-Downloader vulnerability (CVE-2022-36082) was laid out as follows. We started our module with a brief introduction providing context for the vulnerability, followed by a detailed narrative description of the vulnerability, including the vulnerable code and its fix, and, finally, by a 4 minute video where we reproduced the vulnerable version of the software and showed its exploit. We provided brief knowledge checks, consisting of 1-2 multiple choice question, between each sections to break up the content and provide some interactivity. The responses to these knowledge checks as well as the 3 surveys provide the means to answer our research question.

[2]Notion is a Web host service that we used (see https://www.notion.so/)

IV. Study Design

We performed a study with the goal of measuring the effects of the course module on student self-efficacy towards secure programming and their awareness and perceived importance of secure programming. The study follows a pre- and post-test quasi-experimental design [12].

a) Participants: For this study, participants were recruited from two distinct universities, all of whom were enrolled in a programming course (on the level of CS2 or CS3 courses). The validity of the pre- and post-test design can suffer from confounding factors, such as history, maturation, and test effects [35]. The study design of involving two universities is to offset partially the effect of the confounding factors and thereby to improve the robustness of the study by increasing the randomness due to the differences of the two groups of the students and the uncontrollable factors in carrying out the study. A total of 60 participants successfully completed the provided secure programming course module. Of these, 21 were from one university, and the remaining 39 from the other. The cohort consisted of 5% freshmen, 55% sophomores, 33% juniors, and 7% seniors. Notably, 90% of the participants expressed an interest in pursuing a career as a computer programmer, while the remaining 10% had different career aspirations.

In general, the study participants appreciated the importance of secure programming. On the pre-survey statement *Secure programming is important.*, 39/60 (65%) of the participants indicated they Strongly Agree, 16/60 (27%) of the participants indicated they Agree, and 5/60 (8%) of the participants indicated Neither Disagree nor Agree. However, students were somewhat less confident in their secure programming skills. In the study pre-survey, for the statement *"I understand what a security vulnerability is when programming"*, 14/60 (23%) indicated they Strongly Agree, 12/60 (20%) indicated Agree, 23/60 (38%) Neither Disagree nor Agree, 7/60 (12%) Disagree, and 4/60 (7%) indicated Strongly Disagree.

b) Study Protocol: Students worked on the secure programming course module independently during class time. The course module was distributed through the respective university's online course platform, where each participant received a link to the course module. During class, participants were directed to log in to their course portal, navigate to the module, and complete each part of the module. The participants completed the assigned work during one 75-minute class session.

We opted to focus only a single session, i.e., not to conduct multiple study sessions with multiple course modules. This study design choice was made to avoid maturation, i.e., students' learning outcomes tend to improve simply due to their increased learning maturity over time. Past studies have observed that the potential effects due to maturation become greater as the time difference between the pre- and post-tests increases [36], [37].

c) Data Collection: Data was collected through surveys embedded within the course module. We positioned three separate surveys at different stages — Pre, Mid, and Post — along with the two distinct yet interconnected sections: "Input Validation" and "Analysis of Input Validation." The survey instruments consisted of open-ended and closed-ended questions, addressing various aspects of secure programming and software security vulnerabilities. The closed-ended questions are rated on a Likert scale. The pre-survey consisted of 2 Likert questions, and the mid and post-survey consisted of 17 Likert questions, ranging from 1 (Strongly Disagree) to 5 (Strongly Agree). For qualitative support, the pre-and post-surveys contained open-ended questions addressing a participant's comprehension of secure programming before and after completing the entire module. For the ease of the discussion, we include the common questions in the mid- and post-surveys in Table II.

We group the Likert-scale questions in a few topics: 1) *Learning and Educational Methods (Q.1 to Q.6):* These questions are about the effectiveness of using real-world examples to learn secure programming. They explore different approaches to teaching and understanding secure programming, particularly through practical examples. Included is also a question about the likelihood of recommending the course module, which directly pertains to the course itself. 2) *Self-Efficacy (Q.7 to Q.12):* These questions assess the respondents' confidence in their understanding and abilities regarding secure programming and vulnerability management. 3) *Importance and Awareness of Security in Programming (Q.13 to Q.16):* These questions focus on the respondent's recognition of the importance of secure programming and their awareness of security concepts like vulnerabilities and input validation. This topic gauges the respondent's understanding and acknowledgment of security in the context of programming.

TABLE II: Common questions in mid- and post-surveys

No.	Question
Course Module and Educational Method:	
Q.1	I need a real world example to learn secure programming.
Q.2	Seeing a real-world example of a software security vulnerability helps me develop good secure programming skills.
Q.3	Being shown real-world security vulnerabilities in software makes you want to program securely.
Q.4	I learn better when provided with a real-world example of a problem.
Q.5	Exploring a real-world vulnerability help me to better understand the importance of secure programming.
Q.6	How likely are you to recommend this secure programming course module to another student?
Self-Efficacy:	
Q.7	I know how an input validation vulnerability can affect my program.
Q.8	I believe programs that I have written in the past may contain security vulnerabilities.
Q.9	I am aware of various types of security vulnerabilities.
Q.10	I will take into account potential security vulnerabilities when coding.
Q.11	I understand what secure programming is.
Q.12	I know how to program securely.
Awareness of the Importance of Security in Programming:	
Q.13	Secure programming is important.
Q.14	A security vulnerability can affect my code.
Q.15	I am aware of different security vulnerabilities that have affected real-world software.
Q.16	I know what input validation is.

A. Data Analysis

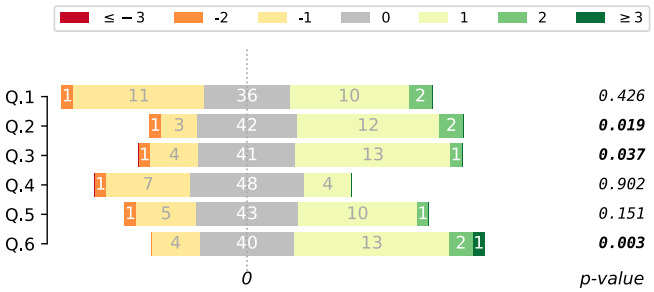
In order to answer the research question (Section I), we adopt a mixed approach to analyze the participants' responses to the surveys. We analyzed the quantitative survey data obtained from the Likert scale questions to reveal the difference before and after the participants' studying the course module. The participants' free-text answers to the open-ended questions were subjected to a comprehensive thematic analysis to reveal common patterns of the participants' self-efficacy and awareness of secure programming and software vulnerabilities. Thematic analysis is a commonly used method to identify patterns in data to extract notable themes [38].

V. Results

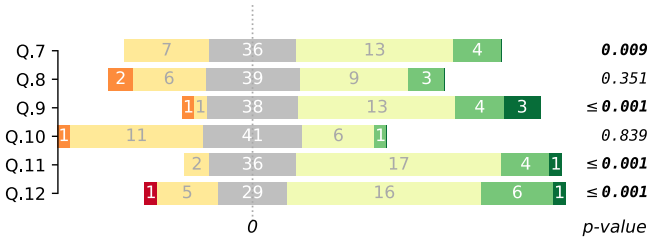
Based on the study method in Section IV-A, in this section we present the results of quantitative and qualitative analysis.

A. Quantitative Analysis

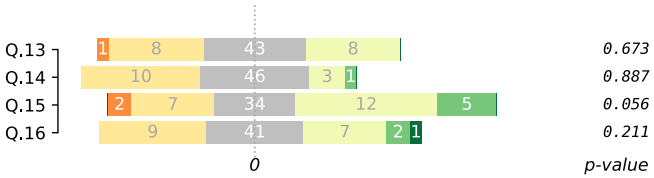
To evaluate whether the authenticity and personal relevance of the real-world software vulnerabilities improve the students' learning disposition toward secure programming, we compare the participants' responses to the common questions in the mid- and post-surveys. For each shared survey question, we calculate the difference between each student's responses, e.g., if a participant answered 3 for survey question Q.1 in the mid-survey, but 4 for the same question in the post-survey, the difference is 1; however, if the response changed to 4 in the mid-survey, and then 3 in the post-survey, the difference is -1. The difference captures the change in a participant's response to survey questions before and after taking the real-world



(2.a) Course Module and Educational Method



(2.b) Self-Efficacy



(2.c) Awareness of the Importance of Security in Programming

Fig. 2: Difference between participant's responses to mid- and post-surveys. Bar blocks corresponding to 0 differences in responses is shrunk by a factor of 5 while the others are to scale.

vulnerability part of the course module. Figure 2 summarizes the comparison in a bar chart. Each bar characterizes the differences of the responses to a single question, indicating the number of participants whose response difference to the question is $\leq -3, -2, \ldots, \geq 3$. To assess the strength of the evidence, we conducted a Wilcoxon signed-rank test, a distribution-free nonparametric statistical hypothesis test method that is frequently used to analyze Likert scale data with small sample size [39]. In our test, the null hypothesis posits that the participants' responses show no improvement. Our observations are as follows:

a) **Course Module and Educational Method.:** Figure 2.a reveals that for most questions in this group, i.e., Q.2, Q.3, Q.5, and Q.6, the response differences tend to favor a positive trend with a small p-value. Particularly, the p-values for Q.2, Q.3, and Q.6 are below 0.05. The evidence is stronger for these survey questions for us to reject the null hypothesis. The participants' responses to Q.2 suggest that there should be a benefit of using the real-world example of a software security vulnerability in developing secure programming skills. Furthermore, the changes in the participants' responses to Q.3 and Q.5 from mid- to post-surveys suggest that there should be an improvement of the participants' willingness towards learning secure program due to the real-world example presented in the module. However, according to the participants' responses to questions Q.1 and Q.4, the participants seem to believe that they can learn secure program without real-world vulnerability, despite the stronger willingness to learning secure programming indicated by the other questions.

b) **Self-Efficacy:** In this group (Figure 2.b), the participants had a positive difference in their responses to Q.7, Q.8, Q.9, Q.11, and Q.12. Among these, except for Q.8, the hypothesis testing yields small p-values (≤ 0.05), suggesting a statistically significant trend in the evidence. Given these results, after observing the real-world vulnerability, the participants appear to have formed a stronger belief that input validation vulnerability can affect their own code (Q.7), to have developed an improved awareness of security vulnerabilities (Q.9), and to have gained improved confidence in their ability to program securely (Q.11 and Q.12). However, the participants do not believe that they can take into account security vulnerabilities when coding, given their responses to Q.10. We posit that the discrepancy in the students' responses exhibited by Q.10 indicates that students do not believe that they will need to focus on ensuring their code is secure in their classwork, as they are typically graded only on satisfying an assignment's required functionality.

c) **Awareness of the Importance of Security in Programming:** As shown in Figure 2.c, although the evidence is weaker than the previous two groups of survey questions, the participants gave positive responses to Q.15 and Q.16, which suggests that the participants believe that they understand the concept of improper input validation and also believe that security vulnerability affect real-world software. However, their responses to Q.13 and Q.14 appear to indicate that they don't believe security vulnerabilities are important to their own code, i.e., their homework assignments and class projects, likely because they believe that these are not considered real-world software.

B. Qualitative Analysis

In this study, two of the paper's authors focused specifically on the thematic analysis of the student answers to the open ended questions (Table III). They analyzed the responses to each question, categorizing and interpreting the underlying themes and patterns observed across related groups of questions. Following their individual analyses, they engaged in a comprehensive comparison and discussion of their findings, combining similar categories and agreeing on labels for the most prevalent themes. The final themes focus on the student perspectives on the course module, on their self-efficacy in secure programming, and how students perceive the importance of secure programming.

a) **Course Module and Educational Method.:** Participants had a positive response to the use of real-world examples in the course module. They appreciated the practicality of learning through relatable scenarios, citing increased attention

TABLE III: Post-survey open-ended questions

No.	Question
Course Module and Educational Method:	
OQ.1	What did you like about the module?
OQ.2	What did you dislike about the module?
OQ.3	How has the real-world example of MangaDex-Downloader influenced your understanding of input validation vulnerabilities, and what insights did you gain from it?
OQ.4	In what ways do you believe the MangaDex-Downloader Demo video in Part 3 aids in your understanding of the impact of the input validation vulnerability on the program?
Self-Efficacy:	
OQ.5	How do you believe real-world vulnerabilities emphasize the significance of your efforts in secure programming?
Awareness of the Importance of Security in Programming:	
OQ.6	Do you think that exploring a real-world vulnerability is beneficial for gaining a better understanding of the importance of secure programming? Explain, why?

and a deeper understanding of the concepts presented. Participant 7 stated, *"Being given real world examples helps me relate to the topic. It makes me think if I have ever been in any of these situations, thus making me more intrigued in the topic."* Similarly, Participant 14 stated, *"It showed what the vulnerability actually looks like in real life, and when you yourself are coding and doing testing, you're more likely to spot something that's not secure, such as that."*

Likewise, participants expressed that using a video demo to present the real-world vulnerability was beneficial. They found this multimedia approach was effective, providing a visual representation that complemented their learning experience. Participant 57 stated, *"The MangaDex Demo video allows viewers to see the real-time exploitation of a vulnerability. This can be more impactful than reading about it in text, as it provides a step-by-step illustration of how an attacker can take advantage of a weakness in input validation."*, while Participant 7 noted that *"A video always helps me understand something more rather than reading."*

One shortcoming of the current version of the module was the lack of hands-on activities related to the real-world vulnerability. Some of the student participants desired more in-depth content or practical, hands-on coding experiences to enhance their learning, e.g., for the question on what they disliked about the course module Participant 47 simply indicated: *"Lack of hands-on coding opportunities"*.

b) **Self-Efficacy:** Several students emphasized the importance of implementing secure coding practices to prevent vulnerabilities and security risks. For instance, Participant 4 noted *"I know I need to write code that can be full-proof so there are no potential security risks"*. After completing the course modules participants also recognized the real-world impact and severe consequences of vulnerabilities, e.g., Participant 10 stated *"I believe real world vulnerabilities are crucial to understand and prevent because a small error can have a negative impact"*. Some provided responses that reflect a sense of personal responsibility and the need for individual effort in secure programming practices. More specifically, Participant 13 said, *"Real-world vulnerabilities emphasized my significance in my efforts of secure programming because I now know the dangers of how a small error on my part could affect the life of someone else"*.

c) **Awareness of the Importance of Security in Programming:** The real-world vulnerability explored in the module focused on input validation. We provided the MangaDex-Downloader software vulnerability as an example to demonstrate this vulnerability in the real-world. Participants explained that the example showed how important it is to ensure that input accepted by users is validated. Participant 4 stated, *"I now see how dangerous input validation vulnerability errors can be. I know that a lot of code that I have written has plenty of errors like these, and I should go out of my way to fix these errors in the future."* Similarly, inspired by the real-world example, participants developed a heightened awareness of security vulnerabilities when coding. They have gained insight into the dangers of improper input validation and expressed their motivation to program with this security concern in mind. Participant 33 stated, *"MangaDex serves as a valuable reminder of the significance of robust input validation in preventing security breaches and underscores the need for a proactive approach to security in software development."*

C. Implications

Our quantitative and qualitative evaluation revealed an overall positive trend in the improvements in student self-efficacy due to our course module. In addition, students appreciated the course module, particularly our efforts in recording a video that reproduces a vulnerability and demonstrates how it can be exploited. However, we observed a smaller positive difference in the students' perception of the importance of secure programming and their awareness of security issues in their code. This was because students ranked those characteristics highly even prior to the module. This observation aligns with findings from other researchers, who noted that students recognize the importance of secure coding but often lack the necessary skills to write secure code [7], [29]. Our course module specifically enhanced learning self-efficacy concerning improper input validation, the vulnerability we focused on. This suggests that employing real-world vulnerabilities can be an effective way to learn secure programming.

D. Threats to Validity

There are several key threats to validity of our study that might affect the interpretation and generalizability of our findings.

External validity concerns the extent to which our results can be generalized. Our study's participants were drawn from only two universities and all were enrolled in a programming course (CS2 and CS3), which may not be representative of all computer science students. The risk is mitigated by the fact

that both universities have a large and diverse population of students.

The design of the surveys and the choice of questions could also impact our findings and is a source of construct validity. The reliance on self-reported measures, especially in assessing understanding of secure programming, might not accurately reflect the participants' actual skills and knowledge in the subject. This threat is mitigated by the fact that used both qualitative and quantitative measures to get a more complete picture of the students perceptions and that the goal of the study was to measure student perceptions and not their actual learning gain.

Internal validity concerns include how well the study was conducted and whether the results can be attributed to the interventions made. The single-session format of the course module might not be sufficient to instigate significant changes in understanding or attitudes toward secure programming. Additionally, by using a single-session format, we may observe a reduced effect size. However, we note that multi-session format may introduce its own set of confounding factors, such as maturation (i.e., a natural improvement in student knowledge over time) [35]. Furthermore, we did not ensure that the participants fully read the course module content and watched the recorded video. This risk is mitigated by the fact that the study was performed in a classroom setting where students were generally focused on their task and had few distractions. Finally, an internal threat to validity arises from the absence of a random control group in the study. We mitigate this threat by recruiting numerous participants and enlisting two distinct universities in the study [35].

VI. Conclusions

Teaching secure programming is crucial for mitigating software security threats. However, students often struggle to maintain and apply their secure programming skills in new tasks. The availability of large vulnerability databases offers a chance to introduce real-world software vulnerabilities into educational settings. These real-world examples are authentic and can resonate with students' experiences. Our study, one of the first of its kind, investigates whether the authenticity and relevance of these vulnerabilities affect students' self-efficacy and awareness in secure programming.

We designed course modules centered on a real-world input validation vulnerability. Using a module, we designed a quasi-experiment study and implemented it at two universities. Using Likert scale questionnaires and open-ended questions, we assessed the module's impact quantitatively and qualitatively. The results indicate that incorporating real-world vulnerabilities positively influences students' learning disposition and self-efficacy in secure programming.

However, this study has limitations, including its single-experiment nature and mixed student responses to certain survey questions. These mixed responses might reflect students' views of their assignments as lacking real-world security significance, underscoring the need for a stronger focus on secure programming in curricula.

This research is a preliminary step in developing pedagogy and materials that use real-world vulnerabilities to enhance secure programming education. Future research will explore different security vulnerabilities and educational methods, aiming to not only assess self-efficacy but also examine directly students' mastery of secure programming skills and knowledge and assess the extent to which design factors of the pedagogy and materials affect students' learning disposition towards secure programming.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No.'s 2235224 and 2235976. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] T. E. Gasiba, U. Lechner, M. Pinto-Albuquerque, and D. Mendez, "Is secure coding education in the industry needed? an investigation through a large scale survey," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 2021, pp. 241–252.

[2] T. E. Gasiba, U. Lechner, M. Pinto-Albuquerque, and D. M. Fernandez, "Awareness of secure coding guidelines in the industry-a first data analysis," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 345–352.

[3] US-CERT, "Software assurance," September 2020, available: https://www.cisa.gov/uscert/sites/default/files/publications/infosheet_SoftwareAssurance.pdf, retrieved on June 20, 2022.

[4] M. Bishop, I. Ngambeki, S. Mian, J. Dai, and P. Nico, "Measuring self-efficacy in secure programming," in *IFIP World Conference on Information Security Education*. Springer, 2021, pp. 81–92.

[5] M. Almansoori, J. Lam, E. Fang, K. Mulligan, A. G. Soosai Raj, and R. Chatterjee, "How secure are our computer systems courses?" in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 2020, pp. 271–281.

[6] M. Bishop, "A clinic for "secure" programming," *IEEE Security & Privacy*, vol. 8, no. 2, pp. 54–56, 2010.

[7] T. Yilmaz and Ö. Ulusoy, "Understanding security vulnerabilities in student code: A case study in a non-security course," *Journal of Systems and Software*, vol. 185, p. 111150, 2022.

[8] D. I. Cordova and M. R. Lepper, "Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice." *Journal of educational psychology*, vol. 88, no. 4, p. 715, 1996.

[9] S. J. Priniski, C. A. Hecht, and J. M. Harackiewicz, "Making learning personally meaningful: A new framework for relevance research," *The Journal of Experimental Education*, vol. 86, no. 1, pp. 11–29, 2018.

[10] S. Buckingham Shum and R. D. Crick, "Learning dispositions and transferable competencies: pedagogy, modelling and learning analytics," in *Proceedings of the 2nd international conference on learning analytics and knowledge*, 2012, pp. 92–101.

[11] J. A. Fredricks, *Eight myths of student disengagement: Creating classrooms of deep learning*. Corwin Press, 2014.

[12] L. Cohen, L. Manion, and K. Morrison, *Research methods in education*, 8th ed. routledge, 2002.

[13] V. Švábenskỳ, J. Vykopal, and P. Čeleda, "What are cybersecurity education papers about? a systematic literature review of sigcse and iticse conferences," in *Proceedings of the 51st ACM technical symposium on computer science education*, 2020, pp. 2–8.

[14] S. Furnell and M. Bishop, "Addressing cyber security skills: the spectrum, not the silo," *Computer fraud & security*, vol. 2020, no. 2, pp. 6–11, 2020.

[15] C. E. Irvine and S.-K. Chin, "Integrating security into the curriculum," *Computer*, vol. 31, no. 12, pp. 25–30, 1998.

[16] C. E. Irvine, "What might we mean by "secure code" and how might we teach what we mean?" in *19th Conference on Software Engineering Education and Training Workshops (CSEETW'06)*. IEEE, 2006, pp. 22–22.

[17] K. Nance, "Teach them when they aren't looking: Introducing security in CS1," *IEEE Security & Privacy*, vol. 7, no. 5, pp. 53–55, 2009.

[18] M. Bishop, "Teaching security stealthily," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 69–71, 2011.

[19] J. Cappos and R. Weiss, "Teaching the security mindset with reference monitors," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 523–528.

[20] S. Raina, S. Kaza, and B. Taylor, "Segmented and interactive modules for teaching secure coding: A pilot study," in *E-Learning, E-Education, and Online Training: First International Conference, eLEOT 2014, Bethesda, MD, USA, September 18-20, 2014, Revised Selected Papers 1*. Springer, 2014, pp. 147–154.

[21] S. Kaza and B. Taylor, "Introducing secure coding in undergraduate (cs0, cs1, and cs2) and high school (ap computer science a) programming courses," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 1050–1050.

[22] W. Du, "SEED: hands-on lab exercises for computer security education," *IEEE Security & Privacy*, vol. 9, no. 5, pp. 70–73, 2011.

[23] B. Taylor and S. Kaza, "Security injections@Towson: Integrating secure coding into introductory computer science courses," *ACM Transactions on Computing Education (TOCE)*, vol. 16, no. 4, pp. 1–20, 2016.

[24] M. Whitney, H. R. Lipford, B. Chu, and T. Thomas, "Embedding secure coding instruction into the ide: Complementing early and intermediate cs courses with eside," *Journal of Educational Computing Research*, vol. 56, no. 3, pp. 415–438, 2018.

[25] V. Nestler, T. Coulson, and J. D. Ashley, "The nice challenge project: providing workforce experience before the workforce," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 73–78, 2019.

[26] J. Wei, B.-T. Chu, D. Cranford-Wesley, and J. Brown, "A laboratory for hands-on cyber threat hunting education," in *Journal of The Colloquium for Information Systems Security Education*, vol. 7, no. 1, 2020, pp. 7–7.

[27] J. Zhu, H. R. Lipford, and B. Chu, "Interactive support for secure programming education," in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013, pp. 687–692.

[28] A. Siraj, N. Sridhar, J. A. D. Hamilton Jr, L. Khan, S. Kaza, M. Gupta, and S. Mittal, "Is there a security mindset and can it be taught?" in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 335–336.

[29] J. Lam, E. Fang, M. Almansoori, R. Chatterjee, and A. G. Soosai Raj, "Identifying gaps in the secure programming knowledge and skills of students," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, 2022, pp. 703–709.

[30] L. Slusky and P. Partow-Navid, "Students information security practices and awareness," *Journal of Information Privacy and Security*, vol. 8, no. 4, pp. 3–26, 2012.

[31] MITRE Corporation, "CWE-20: Improper input validation," available: CWE-20:ImproperInputValidation, retrieved on June 20, 2022.

[32] FIRST.Org, Inc., "Common vulnerability scoring system v3.0: Specification document," available: https://www.first.org/cvss/v3.0/specification-document, retrieved on May 25, 2023.

[33] NIST, "Cve-2022-36082 detail," available https://nvd.nist.gov/vuln/detail/CVE-2022-36082, retrieved on May 25, 2023.

[34] Cyber4All Team, "Cyber4all input validation module," available: https://cisserv1.towson.edu/~cyber4all/modules/nanomodules/Input_Validation-CS0_Java.html, retrieved May, 2023.

[35] E. Marsden and C. J. Torgerson, "Single group, pre-and post-test research designs: Some methodological concerns," *Oxford Review of Education*, vol. 38, no. 5, pp. 583–616, 2012.

[36] R. Hyman, "Quasi-experimentation: Design and analysis issues for field settings (book)," *Journal of Personality Assessment*, vol. 46, no. 1, pp. 96–97, 1982.

[37] T. D. Cook, D. T. Campbell, and W. Shadish, *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin Boston, MA, 2002, vol. 1195.

[38] V. Braun and V. Clarke, "Thematic analysis." in *APA handbook of research methods in psychology, Vol 2: Research designs: Quantitative, qualitative, neuropsychological, and biological.* American Psychological Association, 2012, pp. 57–71.

[39] J. F. de Winter and D. Dodou, "Five-point likert items: t test versus mann-whitney-wilcoxon (addendum added october 2012)," *Practical Assessment, Research, and Evaluation*, vol. 15, no. 1, p. 11, 2019.