Validation of a Secure Programming Concept Inventory

Ida Ngambeki
Information Systems
University of Maryland Baltimore County
Baltimore, MD USA
idan1@umbc.edu

Matthew Bishop
Department of Computer Science
University of California, Davis
Davis, CA USA
mabishop@ucdavis.edu

Jun Dai
Department of Computer Science
California State University, Sacramento
Sacramento, CA USA
jun.dai@csus.edu

Phillip Nico
Department of Computer Science
California Polytechnic State University
SanLuis Obispo, CA USA
pnico@calpoly.edu

ABSTRACT

Security failures in software arising from failures to practice secure programming are commonplace. Improving this situation requires that practitioners have a clear understanding of the foundational concepts in secure programming to serve as a basis for building new knowledge and responding to new challenges. We developed a Secure Programing Concept Inventory (SPCI) to measure students' understanding of foundational concepts in secure programming. The SPCI consists of thirty-five multiple choice items targeting ten concept areas of secure programming. The SPCI was developed by establishing the content domain of secure programming, developing a pool of test items, multiple rounds of testing and refining the items, and finally testing and inventory reduction to produce the final scale.

Scale development began by identifying the core concepts in secure programming. A Delphi study was conducted with thirty practitioners from industry, academia, and government to establish the foundational concepts of secure programming and develop a concept map. To build a set of misconceptions in secure programming, the researchers conducted interviews with students and instructors in the field. These interviews were analyzed using content analysis. This resulted in a taxonomy of misconceptions in secure programming covering ten concept areas. An item pool of multiple-choice questions was developed. The item pool of 225 was administered to a population of 690 students across four institutions. Item discrimination and item difficulty scores were calculated, and the best performing items were mapped to the misconception categories to create subscales for each concept area resulting in a validated 35 item scale.

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM ISBN 978-1-4503-9433-8/23/03.
https://doi.org/10.1145/3545947.3576367

VALIDATING THE INVENTORY

Most assessment methods commonly used in computer science target procedural knowledge, focusing on whether students can produce a functional program and ignoring students' understanding of the underlying processes [1]. A concept inventory is an educational assessment tool that can be used to measure the level of a student's understanding of certain content. Concept inventories have two advantages over most standard tests: they are easy to administer and score, and they probe beyond recognition or memorization to examine a student's understanding of a concept [2].

Students each responded to a subset of fifty randomly selected items from the full item pool of 225 items. The scores ranged from 2% to 80% with a mean of 31.6% (M = 31.6, SD = 17.4). The item difficulty and item discrimination index were calculated. The item difficulty index ranges from 0-1 with items ranging from 0- .29 being too difficult, from .3 -.69 considered appropriately difficult, and item difficulty indices above .70 considered to easy. Items considered too difficult or too simple were eliminated. The item discrimination index measures how well a question distinguishes poorly performing students from strongly performing students. Negative item discrimination indices and those below 0.25 were eliminated as not being strong enough discriminators. Items ranging from 0.25 – 0.39 were deemed good discriminators while those with an index above 0.39 were deemed excellent discriminators. The remaining items were then mapped back to the ten misconception categories to create subscales for each. The reliability of each of these scales was calculated using Cronbach's alpha. Items were systematically eliminated from each subscale to leave the items that accounted for the most variance.

REFERENCES

[1.] J. Davis and M.J. Dark, "Teaching students to design secure systems," *IEEE Security and Privacy*, vol. 1, no. 2, pp. 56-58, 2003.

[2.] R. ufresne, W. Leonard and W. Gerace, "Making sense of students' answers to multiple-choice questions," *The Physics Teacher*, vol. 40, pp. 174-180, 2002.