Building a serious game for teaching secure coding in introductory programming courses

Nicolleta Adamo-Villani Department of Computer Graphics Technology, Purdue University; nadamovi@purdue.edu
Steve Cooper Department of Computer Science, Stanford University; cooper@cs.stanford.edu

Why a game to teach secure coding

- Reports of software-related security breaches are a weekly occurrence in
  the news.
- There is a tremendous need for computing graduates with a background in
  Information Assurance (IA)
- Given that the vast majority of practicing programmers do not have
  advanced degrees, there is a need to give beginning programming
  students early exposure to secure coding.
- The CS2008 document identifies ‘secure coding’ (their term for IA) to
  be one of the three major new focal areas of computing.

Background: Secure Coding

Secure coding can be defined as writing code without bugs or vulnerabilities.
Many of the most dangerous vulnerabilities result from a reasonably small set of
programming errors.
Current IA courses and textbooks are targeted at the advanced undergraduate or graduate level. The intent is to “un-teach” students the bad
habits they have previously learned.
An approach is needed that is appealing to both students and instructors, and
could help to teach beginning programming students secure coding principles.

Background: Serious Games

In Mike Ziya’s definition, a serious game is “a mental contest, played with a
computer in accordance with specific rules that uses entertainment to
a mental contest, played with a
computer in accordance with specific rules that uses entertainment to
"...” (Ziya 2005).
“Play”, an important contributor to human development, maturation, and
learning, is the main component of serious games (Derryberry 2010). Another
fundamental element is “fun”.
The player of a serious game is motivated to play the game, and continues
the lesson much longer with greater attention than he or she would using traditional learning techniques.
The founders of the Education Arcade at MIT stated that there are many
intrinsinc motivations for learning associated with games. "The threat of failure
is lowered. Games allow players to try, make mistakes or fail, and then by
again without being face. Discovery and application of learned skills in
new contexts encourages exploration and experimentation. A sense of
engagement continues during gaming. Computer games allow players to be
shareholders in the events that occur on the screen” (Klopfer et al. 2010).

Technical Details

The platform for the game is based on Autodesk Maya and Unity3D. We used Maya software to model and texture the virtual environments, props and characters and to animate its functionality. Interactivity with the 3D components is programmed in C# using the Unity game development platform.
The choice of the Unity platform was based on the following considerations:
- Unity has an optimized graphics pipeline that supports interactive rendering of complex geometrical meshes and advanced lighting and feedback even on computer with limited graphics capabilities.
- Unity interfaces seamlessly with major 3D animation tools (i.e. Autodesk Maya and 3D Studio Max) and the formats, and allows for instantaneous import and update of asset files and animations.
- It supports a wide range of publishing platforms, including standalone builds for Mac OS and Windows, web delivery through the Unity Web Player Plug-in; Java and iPhone publishing.

The game is deliverable via web or as an exe or app file, and is being designed to run on hardware and software infrastructure that is already widely deployed in universities. Students will be able to use the game on their own personal computers (PC/laptop) with text and graphic cards.

Project Outcomes

Outcome 1: A fully functional, usable and engaging serious game for undergraduate students’ learning of secure coding (including game levels and accompanying lab modules) is available to the public on the project websites for free.

Outcome 2: Undergraduate students can improve their knowledge of secure coding principles and practices as a result of using the serious game and accompanying lab exercises. We have collected data that support the educational effectiveness of the game and lab modules for undergraduate students’ learning of secure coding concepts, specifically specific precedence, buffer overflow, input validation and arithmetic overflow. Findings of the summative studies are currently in progress and review.
Study 1: Focus 1: Focus on operational precedence and buffer overflow. Subjects: 5 undergraduate students. Findings: Results show that playing the game and completing the accompanying lab modules led to an increase in subjective content learning by 25%. In particular, students who played the game increased their declarative knowledge by 25% and procedural knowledge by 5%. Findings also show that these differences in learning game between Group A (control) and Group B (treatment) is significant for all subjects. Overall, playing the game and completing the accompanying lab tasks led to higher learning gains from traditional learning methods.
Study 2: Focus 2: Focus on operational precedence and arithmetic overflow. Subjects: 54 undergraduate students. Findings: Results show that playing the game and completing the accompanying lab modules led to an increase in subjective content learning by 25%. In particular, students who played the game increased their declarative knowledge by 25% and procedural knowledge by 5%. Findings also show that these differences in learning game between Group A and Group B. Overall, playing the game and completing the accompanying tasks led to higher learning gains from traditional learning methods.

Outcome 3: A group of CS educators have been exposed to the game and its lab modules, and have been given the opportunity to incorporate the serious game and accompanying lab modules in their CS courses to improve teaching of secure coding principles and practices.

Published:
Adamo-Villani, N. (2013). Building a serious game to teach secure coding to introductory programming students. Proc. of 2013 TUES PI Conference - Transforming Undergraduate Education in STEM, Washington DC. [An abstract of the paper is uploaded to this conference.]

Project websites
Visit the project websites at
http://i2012.tuespiconference.org/adamo-villani
Here for playing the game

Game levels

The IA game: Level 1 – Data Validation

Level 2 – Returning Values and Handling Errors

The IA game: Level 4 – Operator Precedence

The IA game: Level 5 – Array Checking

The IA game: Level 6 – Arithmetic Overflow

The IA Game

The IA Game is a role-playing serious game (RPG) in which the student plays as the main protagonist - Data Detective (DD). 2D games through computer technology-inspired environments (IA concept rooms) and in each environment he/she learns a different IA concept. After playing one level, the student completes the related CS educational module. Each module includes a theory lesson and one or several lab exercises. The game includes seven levels; each level is an engaging IA room in which the player is exposed to a specific IA concept.

IA concepts: We chose the following seven security-related coding issues: Validating User Input, Array Range Checking, Buffer Overflow, Operator Precedence, Rounding Errors, Returning Values and Handling Errors. Numeric Overflow/Underflow. The selection of the IA concepts was based on criteria. First, each issue represents a common coding practice that has direct or indirect security concern. Second, each issue is easily understandable and accessible by CS1/CS2 students as well as by advanced high school students. Third, each issue supports a lesson/learning/problematic/ test paradigm for student learning. Fourth, by providing varying levels of difficulty, each issue provides the opportunity for differentiated instruction and learning for a wide range of student abilities. Finally, the essence of each coding issue is suitable to be encapsulated in an engaging and compelling game scenario.

The IA Game: Level 1 – Data Validation

The IA Game: Level 2 – Returning Values and Handling Errors