Entailing Security Mindset in Foundational CS Courses: An Interactive Approach

Vahab Pournaghshband
Network and Security Laboratory (Nets Lab)
Computer Science Department
University of San Francisco
vahab.p@usfca.edu

Introduction and Motivation

- Cyber attacks are prevalent
- The annual cost of global cybercrime is now estimated to be $600 billion
  - McAfee's "Annual report on economic impact of cyber crime."
- Data breach affecting 37 million T-Mobile customers
- Downtime of Vanuatu's government digital networks
- Intersection of geopolitics and cyber-attack landscape: the Russia-Ukraine war

Why Teaching Security in Foundational Courses?

- Software security vulnerabilities are the most common cause of software security breaches
- Inability to anticipate how these failures can be potentially exploited, the absence of a security mindset significantly hinders the early detection of exploitable bugs in software development
- Computer security is a skill and cannot be taught in a single course
- Early programmers should be introduced to secure coding practice

Challenges

- The foundational courses are already packed with programming concepts
- Understanding security implications requires advanced CS knowledge
- Computer security is a skill and cannot be taught in a single course
- Computer security should be taught through practice, e.g., programming assignments

Methodology: A Structured Process (1-3)

1) Release the assignment specification to students. Students implement the assignment according to the given specifications.

2) Instruct students on exploiting inherent security vulnerabilities in their code, causing program crashes or undesired behavior.

3) Facilitate a discussion forum with students to explore their understanding of the program’s failure and identify the parts of their code responsible for the abnormal behavior.

Methodology: A Structured Process (4-6)

4) Educate students on the security consequences stemming from the identified vulnerabilities, enhancing their awareness of security issues.

5) Conduct a class discussion on effective strategies to mitigate or eliminate these vulnerabilities in their code.

6) Conclude the process by presenting and demonstrating widely recognized practices for mitigating or eliminating these vulnerabilities.

An Example: Banking Software

- Use a security-sensitive software
- Primary objective is to teach traditional programming concepts in CS1
  - Through implementation of routine banking operations
- Language used was C++

Security Topics Covered

- Security tools
  - One-way hash functions
  - Encryption
- Attacks
  - Buffer overflow
  - Denial-of-Service
  - Integer Overflow
  - Memory Leaks
  - Data Scrubbing

Evaluation Methodology

- Measuring Students' Attitudes Towards Computer Security
- Students' Test Scenarios
- Examining the Log File

Evaluation Methodology

- Measuring Students' Attitudes Towards Computer Security
- Pre/Post Linkert test *(strongly agree, agree, neutral, disagree, and strongly disagree)*
- To evaluate the individuals' confidence, interest, usefulness, and professional constructs
- Example:
  - I am eager and willing to understand computer security concepts.
  - I value understanding the rationale behind cyber attacks and defense.
  - I am confident I can learn to understand computer security concepts.

Evaluation Methodology

- Measuring Students' Attitudes Towards Computer Security
- Students' Test Scenarios
- Examining the Log File

Summary

- We advocate for integrating computer security into early computer programming courses.
- We outlined a six-step process to teach the security mindset interactively.
  - Through a self-contained security-sensitive programming assignment.
- We introduced an ongoing evaluation methodology.