DEPARTMENT OF COMPUTER SCIENCE COURSE SYLLABUS

CS 43401/53401 SECURE PROGRAMMING 3 credit hours

Instructor’s Name: Kambiz Ghazinour


Course Content:
Code vulnerabilities, static code analysis, error handling, secure I/O, race conditions and mediation, handling buffer and integer overflow, handling vulnerabilities in web and database programming, random number generators.
Prerequisites or co-requisites: C (2.000) or better in CS 23001 and junior standing.
Required, elective, or selected elective

Topics to be covered (45 hours):

1. The Software Security Problem (4 hours)
2. Introduction to Static Analysis (4 hours)
3. Running Static Analysis Tools (4 hours)
4. Static Analysis (4 hours)
5. Handling Input (4 hours)
6. Buffer Overflow (5 hours)
7. Bride of Buffer Overflow (4 hours)
8. Errors and Exceptions (5 hours)
9. Web Applications (5 hours)
10. XML and Web Services (4 hours)
11. Race Conditions (2 hours)

Learning Outcomes:
1. Explain why input validation and data sanitization is necessary in the face of adversarial control of the input channel.
2. Explain why you might choose to develop a program in a type-safe language like Java, in contrast to an unsafe programming language like C/C++.
3. Classify common input validation errors, and write correct input validation code.
4. Demonstrate using a high-level programming language how to prevent a race condition from occurring and how to handle an exception.
5. Demonstrate the identification and graceful handling of error conditions.
6. Explain the risks with misusing interfaces with third-party code and how to correctly use third-party code.
7. Discuss the need to update software to fix security vulnerabilities and the lifecycle management of the fix.
8. List examples of direct and indirect information flows.
9. Explain the role of random numbers in security, beyond just cryptography (eg password generation, randomized algorithms to avoid algorithmic denial of service attacks).
10. Explain the different types of mechanisms for detecting and mitigating data sanitization errors.
11. Demonstrate how programs are tested for input handling errors.
12. Use static and dynamic tools to identify programming faults.
13. Describe how memory architecture is used to protect runtime attacks.
14. Describe the requirements for integrating security into the SDL.
15. Apply the concepts of the Design Principles for Protection Mechanisms, the Principles for Software Security, and the Principles for Secure Design (Morrie Gasser) on a software development project.
16. Develop specifications for a software development effort that fully specify functional requirements and identifies the expected execution paths.
17. Describe software development best practices for minimizing vulnerabilities in programming code.
18. Conduct a security verification and assessment (static and dynamic) of a software application.

**Learning Outcomes Assessment:**
1- Four assignments during the semester
2- Midterm exam
3- Final exam
4- Class discussion