ICS C230: SECURE CODING AND DESIGN

Item	Value
Curriculum Committee Approval Date	11/17/2023
Top Code	070700 - Computer Software Development
Units	3 Total Units
Hours	72 Total Hours (Lecture Hours 54; Lab Hours 18)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

This course is designed to provide students with the knowledge and skills necessary to develop secure software applications and systems. In an era marked by increasing cyber threats and data breaches, secure coding and design have become critical aspects of software development. This course covers best practices, principles, and techniques for identifying and mitigating security vulnerabilities in software. Students will learn how to design and develop applications with security in mind, reducing the risk of exploitation and data breaches. ADVISORY: ICS C120 and ICS C123 and CYBR C101. Transfer Credit: CSU.

Course Level Student Learning Outcome(s)

1. Describe the key principles of secure coding and design in software development.
2. Provide mitigation techniques for common software vulnerabilities such as injection attacks, cross-site scripting (XSS), and cross-site request forgery (CSRF).
3. Design and develop a software application with security in mind to reduce the risk of exploitation.

Course Objectives

- 1. Describe the importance of secure coding and design in software development.
- 2. Discuss and outline common security vulnerabilities and threats in software applications.
- 3. Provide examples of secure coding practices used to prevent vulnerabilities at the code level.
- 4. Describe the concept and purpose of security by design.
- 5. Explain how to apply cryptographic techniques for data protection and authentication.
- 6. Define secure communication and data transmission over networks.
- 7. Provide examples of authentication and authorization mechanisms to control access.
- 8. Discuss examples of real-world security incidents and vulnerabilities in existing systems.
- 9. Explain the purpose of threat modeling and risk assessment to proactively identify security issues.

Lecture Content

Introduction to Software Security The importance of software security. Key principles of secure coding and design. Common Security Vulnerabilities Understanding common vulnerabilities such as injection attacks, cross-site scripting (XSS), and cross-site request forgery (CSRF). Secure Coding Practices Input validation and output encoding. Error handling and logging best practices. Using prepared statements and parameterized queries. Security by Design Threat modeling and risk assessment. Security requirements and secure software architecture. Cryptography and Data Protection Cryptographic algorithms and principles. Data encryption and decryption. Hashing and digital signatures. Secure Communication and Authentication Secure Socket Layer (SSL) and Transport Layer Security (TLS). Authentication mechanisms and multi-factor authentication. Access Control and Authorization Role-based access control (RBAC). Authorization frameworks. Web Application Security Security in web applications, including securing REST APIs. Security headers and content security policy. Real-World Case Studies Analysis of security incidents and vulnerabilities in existing systems.

Lab Content

Code Review and Vulnerability Identification: Analyze a code snippet or an open-source project to identify security vulnerabilities (e.g., SQL injection, XSS). Sanitizing User Input: Write code that accepts user input and implement input validation and output encoding to prevent common injection attacks (e.g., SQL injection, XSS). Authentication and Session Management: Implement secure authentication and session management in a web application, including password hashing and secure session handling. Cross-Site Request Forgery (CSRF) Prevention: Create a web application and implement anti-CSRF measures to protect against CSRF attacks. Securing RESTful APIs: Build a simple RESTful API and implement security mechanisms, such as API key authentication and OAuth. Secure File Upload: Develop a feature for uploading files while implementing security checks to prevent malicious file uploads and execution. Encryption and Decryption: Implement data encryption and decryption using cryptographic libraries and techniques (e.g., AES encryption). Security Headers Implementation: Configure security headers (e.g., Content Security Policy, X-Content-Type-Options) in a web application. Security Testing and Scanning: Use security testing tools to scan a web application for vulnerabilities (e.g., OWASP ZAP or Nessus). Access Control and Authorization: Implement role-based access control (RBAC) for a web application, ensuring proper user access rights. Code Refactoring for Security: Take an existing codebase and refactor it to address identified security vulnerabilities and implement security best practices. Secure Software Design: Collaboratively design a software project with a focus on security by considering threat modeling and risk assessment. Security Incident Response Simulation: Given a simulated security incident, respond to it by analyzing the situa tion and implementing necessary measures.

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)

- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

This course will utilize a combination of lecture, hands-on guided laboratory assignments, classroom/discussion student interactions, problem solving, quizzes, tests, and troubleshooting assignments to achieve the goals and objectives of this course. All instructional methods are consistent across all modalities.

Reading Assignments

Students will be asked to read and review academic concepts in the textbook and other materials while experiencing how working code behaves under different conditions. Thus, practical programming techniques are taught through personal understanding, hands-on discovery, active learning, and discussing concepts and their results with others.

Writing Assignments

Written assignments will focus on analysis and evaluation of real-world case studies of vulnerability exploits and data breaches.

Out-of-class Assignments

The problem-solving exercises will include analysis of programming language for common vulnerabilities. Secure coding skills demonstrations are included in the hands-on projects.

Demonstration of Critical Thinking

Students will be asked to read and review academic concepts in the textbook and other materials while experiencing how working code behaves under different conditions. Thus, practical programming techniques are taught through personal understanding, hands-on discovery, active learning, and discussing concepts and their results with others.

Required Writing, Problem Solving, Skills Demonstration

The problem-solving exercises will include analysis of software code to identify common vulnerabilities. Programming skills demonstrations are included in the hands-on projects.

Eligible Disciplines

Computer information systems (computer network installation, microcomputer ...: Any bachelor's degree and two years of professional experience, or any associate degree and six years of professional experience. Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required.

Textbooks Resources

1. Required Daswani, N.; Kern, C.; Kesavan, A. Foundations of Security: What Every Programmer Needs to Know, 1st ed. Apress, 2007 Rationale: Low-cost textbook Legacy Textbook Transfer Data: Legacy Text

Other Resources

1. Technology related white papers and articles are available at no charge to all students at multiple sites as recommended by the instructor. 2. Coastline Library