| | |
|---|---|
| **COURSE TITLE** | **SYSTEM-SECURE SOFTWARE DESIGN** |
| **COURSE CODE** | **01CC0706** |
| **COURSE CREDITS** | **3** |

**Objective:**

1 The course aims to provide students with a foundational understanding of system security by emphasizing the integration of security principles early in the software development lifecycle and identifying common design vulnerabilities. It develops the ability to perform threat modeling using methodologies such as STRIDE and LINDDUN to identify and prioritize risks during the design phase. The course also focuses on applying secure design patterns and best practices for authentication, authorization, and session management, along with designing secure mechanisms for data protection, input validation, and error handling.

**Course Outcomes:** After completion of this course, student will be able to:

1 Apply secure design principles and practices within the software development lifecycle to identify and mitigate design-level vulnerabilities.

2 Analyze software designs using threat modeling techniques such as STRIDE and LINDDUN to identify risks and trust boundaries.

3 Apply secure design patterns for authentication, authorization, and session management in software applications.

4 Evaluate data security mechanisms, input validation strategies, and error handling techniques to prevent common vulnerabilities.

5 Design secure software architectures and conduct design reviews using security principles and analysis tools to ensure system security and maintainability

**Pre-requisite of course:**Solid understanding of software engineering principles and practices. Proficiency in object-oriented analysis and design. Experience with designing and implementing software modules or components. Basic knowledge of common software security vulnerabilities (e.g., OWASP Top 10)

**Teaching and Examination Scheme**

| Theory Hours | Tutorial Hours | Practical Hours | ESE | IA | CSE | Viva | Term Work |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 2 | 50 | 30 | 20 | 25 | 25 |

DR. MUNINDRA HASMUKHBHAI LUNAGARIA
**Digitally signed by (Name of HOD)**

DR. RAJENDRASINH BAHADURSINH JADEJA
**Digitally signed by (Name of Dean/ Principal)**

| Contents : Unit | Topics | Contact Hours |
|---|---|---|
| 1 | **Foundations of Secure Software Design**<br>The Importance of Secure Design (why security must be designed in, not bolted on),, Secure Design within the Software Development Lifecycle (SDL) (integrating security activities into requirements, design, and architecture phases), Common Software Design Vulnerabilities (examples of how design flaws lead to security issues), Core Secure Design Principles (a deeper dive into principles like Least Privilege, Defense in Depth, Fail-Safe Defaults, Separation of Concerns, Complete Mediation as applied to software components), Introduction to Security Requirements Engineering | 6 |
| 2 | **Threat Modeling for Software Design**<br>Threat Modeling Concepts (defining the scope, identifying assets, threats, and vulnerabilities at the design level), Common Threat Modeling Methodologies (e.g., STRIDE, LINDDUN) (practical application to software modules and features), Identifying Trust Boundaries in Software Design (understanding how data and control flow across different levels of trust within the application), Creating Data Flow Diagrams (DFDs) and other models for threat analysis, Prioritizing Threats based on Risk (likelihood and impact) | 6 |
| 3 | **Secure Design Patterns and Practices**<br>Secure Design Patterns (implementing common security controls using established patterns (e.g., Access Object, Gatekeeper, Validator), Avoiding Insecure Design Patterns and Anti-Patterns, Designing for Authentication (integrating authentication mechanisms into the software design, secure handling of credentials (excluding OS-level details)), Designing for Authorization (implementing access control logic within the software based on roles, permissions, or attributes), Designing for Secure Session Management within the application | 7 |
| 4 | **Designing for Data Security and Input Handling**<br>Designing for Secure Data Storage (protecting sensitive data within the software's data structures and persistence mechanisms (encryption, access control within the application)), Designing for Secure Input Validation and Sanitization (determining where and how to validate and clean user inputs at the design level to prevent injection attacks), Designing for Secure Output Encoding (preventing injection attacks by encoding output correctly based on context), Designing for Secure Error Handling and Logging within the application logic to avoid information leakage | 7 |

DR. MUNINDRA HASMUKHBHAI LUNAGARIA
**Digitally signed by (Name of HOD)**

DR. RAJENDRASINH BAHADURSINH JADEJA
**Digitally signed by (Name of Dean/ Principal)**

| Contents : Unit | Topics | Contact Hours |
|---|---|---|
| 5 | **Design Review, Architectural Security, and Assurance**<br>Secure Software Architecture Design (designing multi-layered applications, separating concerns for security), Security Considerations in Specific Software Architectures (e.g., microservices security design patterns, client-server design), Security Design Review Techniques (conducting structured reviews of design artifacts to identify flaws), Introduction to Static Analysis Tools for Design and Architecture (concepts and types of analysis), Designing for Maintainability and Evolvability from a Security Perspective | 6 |
| **Total Hours** | | **32** |

**Suggested List of Experiments:**

| Contents : Unit | Topics | Contact Hours |
|---|---|---|
| 1 | **Practical 1**<br>Analyzing Software Design Case Studies: Reviewing documentation (diagrams, specifications) of real or hypothetical software modules to identify potential security design flaws | 2 |
| 2 | **Practical 2**<br>Applying Secure Design Principles: Given a simple insecure design scenario, propose improvements based on core secure design principles. | 2 |
| 3 | **Practical 3**<br>Performing Threat Modeling on a Software Feature: Applying a threat modeling methodology to a specific feature of a software application (e.g., password reset functionality, file upload module). | 2 |
| 4 | **Practical 4**<br>Identifying Trust Boundaries and Data Flows in a Software Module: Analyzing the design of a software module and mapping its trust boundaries and data interactions. | 2 |
| 5 | **Practical 5**<br>Designing a Secure Authentication Module: Creating a detailed design for a software module responsible for handling user authentication, including state transitions and error handling. | 2 |
| 6 | **Practical 6**<br>Designing a Secure Data Handling Component: Designing a software component responsible for storing and retrieving sensitive user data securely. | 2 |
| 7 | **Practical 7**<br>Analyzing Code-Level Design Artifacts: Reviewing design artifacts like class diagrams or sequence diagrams for security weaknesses related to interactions and responsibilities. | 2 |

**Suggested List of Experiments:**

| Contents : Unit | Topics | Contact Hours |
|---|---|---|
| 8 | **Practical 8**<br>Implementing a Secure Design Pattern (Simple): Writing a small piece of code that implements a basic secure design pattern (e.g., a simple validator or access controller). | 2 |
| 9 | **Practical 9**<br>Comparing Design Alternatives for Security: Evaluating two different design options for a software feature based on their security properties and threat mitigation. | 2 |
| 10 | **Practical 10**<br>Mini-Project: Secure Design of a Software Module: Designing the secure architecture and detailed design for a significant software module based on given requirements, including threat modeling and documentation of design decisions. | 2 |
| **Total Hours** | | **20** |

**Textbook :**

1 Software Security: Building Security In, Gary McGraw, Addison-Wesley Professional, 2006

**References:**

1 Secure Coding in C and C++, Secure Coding in C and C++, Robert, Addison-Wesley Professional, 2005

2 Threat Modeling: Designing for Security, Threat Modeling: Designing for Security, Adam Shostack, Wiley Publishing / John Wiley & Sons, 2014

**Suggested Theory Distribution:**

The suggested theory distribution as per Bloom's taxonomy is as follows. This distribution serves as guidelines for teachers and students to achieve effective teaching-learning process

| Distribution of Theory for course delivery | | | | | |
|---|---|---|---|---|---|
| **Remember / Knowledge** | **Understand** | **Apply** | **Analyze** | **Evaluate** | **Higher order Thinking / Creative** |
| 0.00 | 7.00 | 30.00 | 33.00 | 20.00 | 10.00 |

**Instructional Method:**

1 The course delivery method will depend upon the requirement of content and need of students. The teacher in addition to conventional teaching method by black board, may also use any of tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc

2 The internal evaluation will be done on the basis of continuous evaluation of students in the laboratory and class-room.

3 Practical examination will be conducted at the end of semester for evaluation of performance of students in laboratory.

**Instructional Method:**

4 Students will use supplementary resources such as online videos, NPTEL videos, e-courses, Virtual Laboratory.

**Supplementary Resources:**

1 NPTEL Course: Software Security (IIT Kanpur)

2 Online Videos: OWASP Foundation Webinars

3 E-Course: Secure Software Development (Coursera)