**22pc23XC64 SECURE CODING                                               3 2 0 4**

**Prerequisite:**

* **23XC24 OBJECT-ORIENTED PROGRAMMING**
* **23XC44 OPERATING SYSTEMS**
* **23XC43 CRYPTOGRAPHY**

**INTRODUCTION**:   The Need for Secure Systems, Trustworthy Computing, The Attacker’s Advantage and the Defender’s Dilemma, Vulnerability Cycle, Principles of Security Architecture, Java  Sandbox                                                                                                     (6)

**SOFTWARE DEVELOPMENT PROCESS:** Proactive Security Development Process, Secure Software Development Cycle, Security issues while writing SRS, Process Improvements, Design Phase, Development Phase, Test Phase, Shipping and Maintenance Phases, Security Principles, Secure Product Development timeline                       (12)

**THREAT MODELING** : Secure Design Through Threat Modeling, Decompose the Application, Determine the Threats to the System, Rank the Threats by Decreasing Risk, Choose How to Respond to the Threats, Choose Techniques to Mitigate the Threats, Security Techniques, Authentication, Authorization, Tamper-Resistant and Privacy-Enhanced Technologies, Encryption, Hashes, MACs, and Digital Signatures, Auditing, Filtering, Throttling, and Quality of Service, Least Privilege      (8)

**SECURE CODING TECHNIQUES**:  Buffer Overrun, Stack Overruns, Heap Overruns, Array Indexing Errors, Format String Bugs, Preventing Buffer Overruns                                                                                                                                                                         (7)

**ACCESS CONTROL:**  Determining Appropriate Access Control, Running with Least Privilege                                                                (6)

**CRYPTOGRAPHY CODING**:  Poor Random Numbers, Using Passwords to Derive Cryptographic Keys, Key Management Issues, Creating Your Own Cryptographic Functions, Protecting Secret Data, Creating a Salted Hash                                            (8)

**CANONICAL REPRESENTATION**:  Canonical Filename Issues, Canonical Web-Based Issues, Preventing Canonicalization Mistakes, Web-Based Canonicalization Remedies                                                                                                                                                      (7)

**PROTECTING AGAINST DENIAL OF SERVICE ATTACKS**: Application Failure Attacks, CPU Starvation Attacks, Memory Starvation Attacks, Resource Starvation Attacks, Network Bandwidth Attacks.                                                                                                           (7)

**SECURITY TESTING**: Building Security Test Plans from a Threat Model, Testing Clients with Rogue Servers, Testing with Security Templates, Test the End-to-End Solution, Determining Attack Surface,  Risk Assessment Methodologies                                               (8)

**MAINTENANCE**: Performing a Security Code Review, Secure Software Installation, Building Privacy into Your Application, Writing Security Documentation and Error Messages                                                                           (6)

**TUTORIAL PRACTICE:**

1. Implementation of buffer (stack / heap) overrun
2. Implementation of Stack Randomization
3. Performing Static Code Analysis using open source tool
4. Managing Privileges in a computing system
5. Implementation of Application Logging
6. Practice on memory-safety and thread-safety implementation using RUST Language

**Total L: 45+T:30 = 75**

**TEXT BOOKS:**

1. Michael Howard, David LeBlanc, "Writing Secure Code", Microsoft Press, 2003.
2. Robert C. Seacord, “Secure Coding in C and C++", Addision Wesley, 2013.

**REFERENCES:**

1. C. Warren Axelrod, "Engineering Safe and Secure Software Systems",  Artech House, 2012.
2. Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, David Svoboda, “The CERT Oracle Secure Coding Standard for Java”, Addison-Wesley Professional, 2012.
3. Ric Messier,”Rust Programming”, Wiley, 2021.
4. Code Review: <https://rules.sonarsource.com/>

CO1: Understanding the need for secure systems in software development in various phases.

CO2: Perform threat modeling in the design phase and adhere the secure coding practices.

CO3: Apply access control model for the given software system and prevent canonicalization mistakes.

CO4: Understanding the implementation issues in cryptographic coding.

CO5:Protecting against denial of service attacks and security measures in testing and maintenance.