National University of Engineering (UNI)
School of Cybersecurity
Syllabus 2024-II

1. COURSE

CY221. Software Security (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CY221. Software Security
2.2 Semester	:	8^{th} Semester.
2.3 Credits	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS3I1. Computer Security. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course addresses the principles and practices for secure software development, enabling students to build applications resistant to vulnerabilities. Design, implementation, and testing techniques are explored, considering ethical and legal responsibilities.

5. GOALS

- Apply principles and practices to design and implement secure software.
- Identify and mitigate common vulnerabilities in software development.
- Understand the ethical and legal impact of secure software development.

6. COMPETENCES

2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

4) Recognize professional responsabilities and make informed judgments in computing practice based on legal and ethical principles. (Usage)

6) Apply security principles and practices to maintain operations in the presence of risks and threats. (Assessment)

7. TOPICS

Unit 1: Principios fundamentales (12 hours)	
Competences Expected: 2,6	
Topics	**Learning Outcomes**
• Mínimo privilegio – Esta unidad de conocimiento presenta los principios que subyacen tanto al diseño como a la implementación. Los primeros cinco son principios de restricción, los tres siguientes son principios de simplicidad y el resto son principios de metodología. • Valores predeterminados a prueba de fallos – El estado inicial debería ser denegar el acceso a menos que se requiera explícitamente. Entonces, a menos que al software se le dé acceso explícito a un objeto, se le debe negar el acceso a ese objeto y el estado de protección del sistema debe permanecer sin cambios. • Mediación Completa – El software debe validar cada acceso a los objetos para garantizar que el acceso esté permitido. • Separación – El software no debe otorgar acceso a un recurso ni realizar una acción relevante para la seguridad basándose en una única condición. • Minimizar la confianza – El software debe verificar todas las entradas y los resultados de todas las acciones relevantes para la seguridad. • Economía del mecanismo – Las funciones de seguridad del software deben ser lo más simples posible • Minimizar el mecanismo común – Reducir los recursos compartidos al máximo • El menor asombro – Las características de seguridad del software y los mecanismos de seguridad que implementa deben diseñarse de manera que su funcionamiento sea lo más lógico y simple posible. • Diseño abierto – La seguridad del software, y de lo que ese software proporciona, no debería depender del secreto de su diseño o implementación. • Capas – Organice el software en capas de modo que los módulos de una capa determinada interactúen solo con los módulos de las capas inmediatamente superiores y inferiores. Esto le permite probar el software una capa a la vez, utilizando	• Analice las implicaciones de confiar en el diseño abierto o el secreto del diseño para la seguridad [Usar] • Enumere los tres principios de seguridad [Usar] • Describa por qué cada principio es importante para la seguridad. [Usar] • Identificar el principio de diseño necesario [Usar]

Unit 2: Diseño (8 hours)	
Competences Expected: 2	
Topics	**Learning Outcomes**
• Derivación de requisitos de seguridad. – Comenzando con el negocio, la misión u otros objetivos, determine qué requisitos de seguridad son necesarios para tener éxito. Estos también pueden derivarse o modificarse a medida que evoluciona el software. • Especificación de requisitos de seguridad. – Traducir los requisitos de seguridad a una forma que pueda usarse (especificación formal, especificaciones informales, especificaciones para pruebas). • Ciclo de vida de desarrollo de software/Ciclo de vida de desarrollo de seguridad – Incluya los siguientes ejemplos: modelo en cascada, desarrollo ágil y seguridad. • Lenguajes de programación y lenguajes de tipo seguro – Analice los problemas que introducen los lenguajes de programación, qué hace la seguridad de tipos y por qué es importante.	• Analice las implicaciones de confiar en el diseño abierto o el secreto del diseño para la seguridad. [Usar] • Enumere los tres principios de seguridad. [Usar] • Describa por qué cada principio es importante para la seguridad. Identificar el principio de diseño necesario [Usar]
Readings : [**McGraw2006**]	

Unit 3: Implementación (10 hours)

Competences Expected: 2,6

Topics	Learning Outcomes
• Validar la entrada y comprobar su representación. – Verifique los límites de los buffers y los valores de los números enteros para asegurarse de que estén dentro del rango – Verifique las entradas para asegurarse de que sean las esperadas y que se procesen/interpreten correctamente. • Utilizando las API correctamente – Verifique los resultados del uso de la API para detectar problemas – Asegúrese de que los parámetros y entornos estén validados y controlados para que la API aplique la política de seguridad correctamente. • Uso de características de seguridad – Utilice aleatoriedad criptográfica – Restrinja adecuadamente los privilegios del proceso. • Comprobación de relaciones de tiempo y estado. – Compruebe que el archivo sobre el que se actúa es aquel para el que se comprueban los atributos relevantes – Verifique que los procesos se ejecuten. • Manejar excepciones y errores adecuadamente – Bloquear o poner en cola señales durante el procesamiento de señales, si es necesario – Determine qué información se debe brindar al usuario, equilibrando la usabilidad con cualquier necesidad de ocultar cierta información, y cómo y a quién reportar esa información. • Programación de robusta – Solo desasignar la memoria asignada, – Inicializar variables antes de usarlas – No confíe en un comportamiento indefinido. • Encapsulación de estructuras y módulos. – Procesos de aislamiento. • Teniendo en cuenta el medio ambiente – Ejemplo: no incluya información confidencial en el código fuente.	• Analice las implicaciones de confiar en el diseño abierto o el secreto del diseño para la seguridad. [Usar] • Enumere los tres principios de seguridad. [Usar] • Describa por qué cada principio es importante para la seguridad. Identificar el principio de diseño necesario [Usar]

Readings : [**Seacord2005**]

Unit 4: Análisis y pruebas (8 hours)	
Competences Expected: 2,6	
Topics	**Learning Outcomes**
• Análisis estático y dinámico. – Este tema describe los diferentes métodos para cada uno de ellos, incluye cómo funcionan juntos el análisis estático y dinámico, y los límites y beneficios de cada uno, además de cómo realizar estos tipos de análisis en sistemas de software de gran tamaño. • Pruebas unitarias – Este tema describe cómo probar componentes del software, como módulos. • Pruebas de integración – Este tema describe cómo probar los componentes de software a medida que se integran. • Pruebas de software – Este tema describe cómo probar el software en su conjunto y colocar las pruebas unitarias y de integración en un marco adecuado.	• Explique por qué los requisitos de seguridad son importantes [Usar] • Identificar vectores de ataque comunes [Usar] • Describir la importancia de escribir programas seguros y robustos [Usar] • Describir el concepto de privacidad, incluida la información de identificación personal [Usar]
Readings : [**Whittaker2012**]	

Unit 5: Implementación y mantenimiento (10 hours)	
Competences Expected: 2,6	
Topics	**Learning Outcomes**
• Configurando – Este tema cubre cómo configurar el sistema de software para que funcione correctamente. • Parches y ciclo de vida de la vulnerabilidad – Este tema incluye la gestión de informes de vulnerabilidad, la reparación de las vulnerabilidades, la prueba del parche y la distribución del parche. • Comprobando el entorno – Este tema cubre cómo garantizar que el entorno coincida las suposiciones hechas en el software, y si no, cómo manejar el conflicto • DevOps – Este tema combina desarrollo y operación, y la automatización y monitoreo de ambos. • Desmantelamiento/Retiro – Este tema describe lo que sucede cuando se elimina el software y cómo eliminarlo sin causar problemas de seguridad.	• Explique por qué son necesarias la validación de entradas y la desinfección de datos [Usar] • Explica la diferencia entre números pseudoaleatorios y números aleatorios [Usar] • Diferenciar entre codificación segura y parcheo y explicar la ventaja de utilizar técnicas de codificación segura [Usar] • Describa un desbordamiento del búfer y por qué es un posible problema de seguridad [Usar]
Readings : [**Humble2010**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY