UNIVERSITY of NICOSIA

Course Syllabus

Course Code	Course Title	ECTS Credits
COMP-385	Defensive Programming	6
Prerequisites	**Department**	**Semester**
COMP-212	Computer Science	Spring
Type of Course	**Field**	**Language of Instruction**
Elective	Computer Science	English
Level of Course	**Lecturer(s)**	**Year of Study**
1st Cycle	Dr Harald Gjermundrød	3rd
Mode of Delivery	**Work Placement**	**Corequisites**
Face-to-face	N/A	None

Course Objectives:

The main objectives of the course are to:

- introduce the concept of defensive programming and its importance for software development
- cover in detail the 3D's (security by design, by default, and in deployment) of software engineering and how they are related to defensive programming
- make aware the importance of the concepts of input validation, buffer overrun, and character encoding
- compare and contrast the different programming languages with respect to inherent security issues
- demonstrate how to perform secure code review and what to focus on
- provide deep knowledge of canonical representation issues and how to prevent canonicalization mistakes
- expose the students to privacy issues (including the GDPR) when building software and how to design privacy-aware applications.

Learning Outcomes:

After completion of the course students are expected to be able to:

1. describe the concept of defensive programming and its importance for software development
2. demonstrated how to apply the 3D's, namely security by design, by default, and in deployment) during the software engineering process

3. develop software where all user input is validated, without any buffer overruns, and with appropriate character encoding
4. compare and contrast the different programming languages with respect to their inherent security issues and limitations
5. perform a code review that focuses on developing secure and privacy preserving applications
6. summarize the issues with canonical representation and how to prevent canonicalization mistakes in the developed software
7. design and develop software which takes conforms to privacy regulations such as the General Data Protection Regulation.

Course Content:

1. Introduction
    a. Why do programmers write insecure code.
    b. The proactive security development process.
2. Security principles
    a. Secure by design, by default, and in deployment.
    b. Security requirements.
3. Input validation
    a. Validate *all* input.
    b. Input validation tools including *regular expressions.*
4. Buffer overrun
    a. Stack overruns, heap overruns, array indexing error
    b. Format string bug.
5. Internationalization issues
    a. Character encoding.
    b. Character set conversion issues.
6. Canonical representation issues
    a. Canonical issues.
    b. Preventing canonicalization mistakes.
7. Language-specific issues
    a. C/C++, Python, Java, and PHP.

8. Performing a security code review
    a. Dealing with large applications.
    b. Extra checks: integer underflow/overflow, returns, pointer code.
9. Building Privacy into your application
    a. Malicious vs. unintentional invasions of privacy.
    b. Designing privacy-aware applications.

Learning Activities and Teaching Methods:

Lectures, Practical Exercises, Project Work, and Assignments.

Assessment Methods:

Final Exam, Midterm Exam, Project, and Quizzes.

Required Textbooks / Readings:

Title	Author(s)	Publisher	Year	ISBN
Secure Programming HOWTO, v3.72 Edition	David A. Wheeler	Published online	2015	
Writing Secure Code, 2nd Edition	Michael Howard, David LeBlanc	Microsoft Press	2002	978-0735617223

Recommended Textbooks / Readings:

Title	Author(s)	Publisher	Year	ISBN
24 Deadly Sins Of Software Security: *Programming Flaws and How to Fix Them*	Michael Howard, David LeBlanc, and John Viega	McGraw-Hill	2010	978-0071626767