COURSE OUTLINE

GENERAL

SCHOOL	Sciences and Engineering		
ACADEMIC UNIT	Computer Science		
LEVEL OF STUDIES	1st Cycle		
COURSE CODE	COMP-434	**SEMESTER**	Spring
COURSE TITLE	Secure Systems Programming		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits*	**WEEKLY TEACHING HOURS**	**CREDITS**
	2.5	6
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).		

COURSE TYPE *general background, special background, specialised general knowledge, skills development*	Specialization
PREREQUISITE COURSES:	COMP-212, COMP-354
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	English
IS THE COURSE OFFERED TO ERASMUS STUDENTS	
COURSE WEBSITE (URL)	

LEARNING OUTCOMES

Learning outcomes

The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described.

Consult Appendix A

- *Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area*
- *Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B*
- *Guidelines for writing Learning Outcomes*

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

- critically evaluate the Unix operating system environment and its significance in system programming
- explain the importance of security in low-level system programming, identifying the role of kernel APIs in secure development
- demonstrate the ability to apply secure coding practices in C/C++, focusing on effective memory management and vulnerability prevention
- utilize system calls and kernel APIs securely, demonstrating proficiency in managing kernel resources safely

- analyze and implement access control mechanisms, including user/group permissions and ACLs, to secure file systems effectively
- critique and perform secure file handling and I/O operations, understanding the security properties of directories and files
- Develop applications which manage memory allocation and resource cleanup, effectively preventing memory leaks and buffer overflows
- conduct security auditing and monitoring in kernel space, developing skills in logging, intrusion detection, and kernel debugging
- apply secure development practices to containerized applications by understanding containerization principles, evaluating security considerations, and comparing different container image types.

General Competences

Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology	*Project planning and management*
Adapting to new situations	*Respect for difference and multiculturalism*
Decision-making	*Respect for the natural environment*
Working independently	*Showing social, professional and ethical responsibility and sensitivity to gender issues*
Team work	*Criticism and self-criticism*
Working in an international environment	*Production of free, creative and inductive thinking*
Working in an interdisciplinary environment	*……*
Production of new research ideas	*Others…*
	…….

Search for, analysis and synthesis of data and information, with the use of the necessary technology
Adapting to new situations
Decision-making
Working independently
Project planning and management
Criticism and self-criticism
Production of free, creative and inductive thinking

SYLLABUS

1. Introduce the Unix Operating System
    - Overview of the Unix Development Environment.
    - Introduction to Tools Provided for Developing System Programs.
2. Introduction to Secure Low-Level Programming
    - Overview of Low-Level System Programming.
    - Importance of Security in System-Level Development.
    - Introduction to Kernel APIs.
3. Secure Coding with C/C++
    - Language Features and Security Considerations.
    - Memory Management Best Practices.
    - Avoiding Common Vulnerabilities in C/C++.
4. Secure System Calls and Kernel API Usage

- Introduction to System Calls and Their Security.
- Accessing Kernel APIs Securely.
- Handling Kernel Resources Safely.

5. Access Control and Permission Management
   - User and Group Permissions.
   - Secure File System Access.
   - Implementing Access Control Lists (ACLs).
6. Files and I/O
   - Structure, Organization, and Security Properties of Directories and Files.
   - Operations on Files and Directories.
   - Buffered and Unbuffered I/O.
7. Secure Memory and Resource Management
   - Memory Allocation and Buffer Management.
   - Preventing Memory Leaks and Overflows.
   - Resource Cleanup and Management.
8. Security Auditing and Monitoring in Kernel Space
   - Logging and Auditing Kernel Activities.
   - Detecting and Reacting to Intrusions.
   - Kernel Debugging and Testing.
9. Developing Secure Applications Using Containers
   - Introduction to Containerization.
   - Security Considerations for Containerized Applications.
   - Comparing OS vs. Serverless Container Images.

TEACHING and LEARNING METHODS - EVALUATION

<table>
<tr><td>DELIVERY
Face-to-face, Distance learning, etc.</td><td>Face-to-face</td></tr>
<tr><td>USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
Use of ICT in teaching, laboratory education, communication with students</td><td>Use of ICT in teaching / Χρήση ΤΠΕ
Communication with students / Επικοινωνία με Φοιτητές</td></tr>
<tr><td>TEACHING METHODS
The manner and methods of teaching are described in detail.
Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.

The student's study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS</td><td>
<table>
<tr><th>Activity</th><th>Semester workload</th></tr>
<tr><td>Lectures</td><td>35</td></tr>
<tr><td>Preparation</td><td>30</td></tr>
<tr><td>Homework, quizzes</td><td>45</td></tr>
<tr><td>Exam preparation</td><td>38</td></tr>
<tr><td>Final Exam</td><td>2</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Course total</td><td>150</td></tr>
</table>
</td></tr>
<tr><td>STUDENT PERFORMANCE EVALUATION
Description of the evaluation procedure</td><td>Final Exam, Midterm Exam, Assignments, and Quizzes</td></tr>
</table>

Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other *Specifically-defined evaluation criteria are given, and if and where they are accessible to students.*	

ATTACHED BIBLIOGRAPHY

Required Textbooks / Readings:

Title	Author(s)	Publisher	Year	ISBN
Advanced Programming in the UNIX(R) Environment, 3rd Ed.	R. Stevens, S. Rago	Addison Wesley	2013	978-0321637734
Gray Hat Hacking: The Ethical Hackers Handbook, 6th Edition	A. Harper, R. Linn, S. Sims, M. Baucom, D. Fernandez, H. Tejeda, M. Frost	McGraw-Hill Osborne	2022	978-1264268948

Recommended Textbooks / Readings:

Title	Author(s)	Publisher	Year	ISBN
C Programming Language, 2nd Ed.	Brian W. Kernighan, Dennis Ritchie	Pearson	1988	978-0131103627
Linux System Programming, 2nd Ed.	Robert Love	O'Reilly Media	2013	978- 1449339531