Course Syllabus

Course Code	Course Title	ECTS Credits
COMP-434	Secure Systems Programming	6
Prerequisites	**Department**	**Semester**
COMP-212, COMP-354	Computer Science	Spring
Type of Course	**Field**	**Language of Instruction**
Elective	Computer Science	English
Level of Course	**Lecturer(s)**	**Year of Study**
1st Cycle	Prof. Harald Gjermundrød	3rd or 4th
Mode of Delivery	**Work Placement**	**Corequisites**
Face-to-face	N/A	None

Course Objectives:

The main objectives of the course are to:

- describe and discuss the basic structure and environment of the Unix operating system for developing system programs
- recognize the significance of security in low-level system programming and the role of kernel APIs
- apply secure coding practices when developing programs with C/C++, focusing on memory management and preventing vulnerabilities
- implement secure access to system calls and kernel APIs, ensuring the safe handling of kernel resources
- analyze user/group permissions and access control methods, including ACLs, to ensure secure file system access
- examine the structure and security properties of directories and files along with secure I/O operations
- manage memory and resources effectively to prevent leaks and overflows, assuring proper resource cleanup
- evaluate security auditing, logging, and intrusion detection measures in kernel space to maintain system integrity
- understanding of containerization, enabling them to develop and manage applications securely within containerized environments by addressing security challenges and implementing best practices.

Learning Outcomes:

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

1. critically evaluate the Unix operating system environment and its significance in system programming
2. explain the importance of security in low-level system programming, identifying the role of kernel APIs in secure development
3. demonstrate the ability to apply secure coding practices in C/C++, focusing on effective memory management and vulnerability prevention
4. utilize system calls and kernel APIs securely, demonstrating proficiency in managing kernel resources safely
5. analyze and implement access control mechanisms, including user/group permissions and ACLs, to secure file systems effectively
6. critique and perform secure file handling and I/O operations, understanding the security properties of directories and files
7. develop application which manage memory allocation and resource cleanup, effectively preventing memory leaks and buffer overflows
8. conduct security auditing and monitoring in kernel space, developing skills in logging, intrusion detection, and kernel debugging
9. apply secure development practices to containerized applications by understanding containerization principles, evaluating security considerations, and comparing different container image types.

Course Content:

1. Introduce the Unix Operating System
   a. Overview of the Unix Development Environment.
   b. Introduction to Tools Provided for Developing System Programs.
2. Introduction to Secure Low-Level Programming
   a. Overview of Low-Level System Programming.
   b. Importance of Security in System-Level Development.
   c. Introduction to Kernel APIs.
3. Secure Coding with C/C++
   a. Language Features and Security Considerations.
   b. Memory Management Best Practices.
   c. Avoiding Common Vulnerabilities in C/C++.
4. Secure System Calls and Kernel API Usage
   a. Introduction to System Calls and Their Security.
   b. Accessing Kernel APIs Securely.
   c. Handling Kernel Resources Safely.
5. Access Control and Permission Management
   a. User and Group Permissions.
   b. Secure File System Access.
   c. Implementing Access Control Lists (ACLs).
6. Files and I/O
   a. Structure, Organization, and Security Properties of Directories and Files.
   b. Operations on Files and Directories.

c. Buffered and Unbuffered I/O.

7. Secure Memory and Resource Management
    a. Memory Allocation and Buffer Management.
    b. Preventing Memory Leaks and Overflows.
    c. Resource Cleanup and Management.
8. Security Auditing and Monitoring in Kernel Space
    a. Logging and Auditing Kernel Activities.
    b. Detecting and Reacting to Intrusions.
    c. Kernel Debugging and Testing.
9. Developing Secure Applications Using Containers
    a. Introduction to Containerization.
    b. Security Considerations for Containerized Applications.
    c. Comparing OS vs. Serverless Container Images.

Learning Activities and Teaching Methods:

Lectures, Practical Exercises, and Assignments

Assessment Methods:

Final Exam, Midterm Exam, Assignments, and Quizzes

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