Liverpool John Moores University

Title:	Secure Software Development
Status:	Definitive
Code:	**5218COMP** (127994)
Version Start Date:	01-08-2021
Owning School/Faculty:	Computer Science and Mathematics
Teaching School/Faculty:	Computer Science and Mathematics

Team	Leader
Nathan Shone	Y

Academic Level:	FHEQ5	**Credit Value:**	20	**Total Delivered Hours:**	44
Total Learning Hours:	200	**Private Study:**	156		

Delivery Options
Course typically offered: Semester 2

Component	Contact Hours
Lecture	22
Practical	22

Grading Basis: 40 %

Assessment Details

Category	Short Description	Description	Weighting (%)	Exam Duration
Artefacts	AS1	Group software development and security task	80	
Presentation	AS2	Group presentation on software development and security task	20	

Aims

To familiarise students with common software security problems and vulnerabilities, and the methods, tools and techniques that can be used during software development to prevent them, including formal techniques.
To provide students with an understanding of techniques that should be applied throughout the software development lifecycle in order to improve software security.

Learning Outcomes

After completing the module the student should be able to:

1 Analyse software security vulnerabilities and apply best-practice practical techniques to prevent them.
2 Apply wide-ranging technical and conceptual security skills to the software development lifecycle.
3 Use mitigation techniques to fix vulnerabilities that exist in complex software.
4 Apply group-based development and testing principles to address a broad range of security issues.

Learning Outcomes of Assessments

The assessment item list is assessed via the learning outcomes listed:

Software Development	1	2	4
Presentation on security task	3	4	

Outline Syllabus

Characteristics of large-scale software systems projects, team membership and activities.
Common software vulnerabilities.
Programming languages and security characteristics, decompilation and obfuscation.

Integrating security into the software development lifecycle.
Threat modelling.
Formal techniques for vulnerability analysis.
Testing, including practical experience of unit testing and fuzz testing.
Networking vulnerabilities.
Random number generation and cryptography.
Secure deployment.
General rules and guidelines; secure coding policies.
Recent examples from computing are used throughout and practical exercises used to illustrate the applications of these concepts.

Learning Activities

Students will participate in lectures, practical tutorials / lab sessions and work in groups using collaboration tools to manage development (e.g., GitHub)

Notes

Students will undertake a group software engineering task involving the application of secure software development lifecycles to a software development task. As part of

this task, students will be expected to undertake a variety of roles as seen in a secure software development teams (i.e., developer, software tester, vulnerability researcher, report & documentation author, etc). Students will be expected to complete a report that demonstrates an understanding of how software should be designed, implemented, and tested to reduce the risk of security vulnerabilities. Students will also be expected to discover and mitigate vulnerabilities in software provided to them as part of this activity.