Module Specification

Operating Systems Security and Defensive Programming

Version: 2023-24, v2.0, 19 Jul 2023

Contents

Part 1: Information

Module title: Operating Systems Security and Defensive Programming

Module code: UFCFHU-30-2

Level: Level 5

For implementation from: 2023-24

UWE credit rating: 30

ECTS credit rating: 15

Faculty: Faculty of Environment & Technology

Department: FET Dept of Computer Sci & Creative Tech

Partner institutions: None

Field:

Module type: Module

Pre-requisites: Operating Systems and Architecture 2022-23, Programming 2022-23

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body requirements: None

Part 2: Description

Overview: This module introduces students to the tasks of operating systems such as controlling and allocating memory, prioritising system requests, controlling input and output devices, facilitating data networking and managing files, including security and protection.

Students will learn the concepts of security protection in operating systems, such as

hierarchical protection domains, and how they are employed to resist malware threats.

A design pattern is a description of how to solve a problem that can be used in many different situations and can help deepen the understanding of object-orientated programming and help improve software design and reusability. They can also be used for secure programming and students will learn how to apply them along with other methods and tools.

Features: Not applicable

Educational aims: This module contributes to cyber knowledge. It strengthens and deepens the computing underpinnings developed at earlier levels of study.

Outline syllabus: security and protection

kernel security and protection

typical OS security features and how they may be exploited

approaches to defensive programming, for example input validation, least privilege, defence in depth, data sanitization, etc

authentication, authorisation and access control

resistance to malware techniques such as memory corruption, code injection, user/kernel space vulnerabilities, privilege escalation, etc.

design patterns for developing secure software

use of compiler features to support the creation of secure code

static and dynamic code analysis techniques

sources of secure programming practices, including employer or software development organisation, for different types of software systems (e.g., OWASP,

CERT, etc.)

at least 1 formal method that may be applied to software development and its strengths and weaknesses when applied to development of software with security properties

Part 3: Teaching and learning methods

Teaching and learning methods: Lecture sessions cover the technical knowledge required. Designated practical work is included to ensure that apprentices have absorbed and understood the key principles involved.

Module Learning outcomes: On successful completion of this module students will achieve the following learning outcomes.

MO1 Configure an Operating System in accordance with security policy.

MO2 Identify threats and the features that mitigate the threats.

MO3 Identify appropriate secure programming principles and design patterns and analyse their fitness to address security issues

MO4 Research sources of secure programming practices and apply them

Hours to be allocated: 300

Contact hours:

Independent study/self-guided study = 135 hours

Placement = 75 hours

Face-to-face learning = 90 hours

Total = 300

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://rl.talis.com/3/uwe/lists/20F18CAC-2867-6DAC-83CA-F36CCC443E5E.html

Part 4: Assessment

Assessment strategy: At both first sit and resit, this module is assessed by a combination of techniques: a report (3,000 words) and a portfolio which includes a practical demonstration.

Students will be given a specification from which they have to produce a solution. This must incorporate the use of secure design patterns. They must show the research they have done to select the secure programming practices they will employ. As well as demonstrating the solution, apprentices will have to produce evidence of the design, development, implementation, test and debug.

Students will write a 3,000-word report that will require them to research the most prevalent threats to operating systems. They will then describe how operating system security features protect against these threats. Finally, they will show how operating systems should be configured to take the most advantage of these features.

Assessment tasks:

Portfolio (First Sit)
Description: Portfolio showing evidence of design, development, implementation, testing and debugging of a solution to a security problem.
Weighting: 60 %
Final assessment: No
Group work: No
Learning outcomes tested: MO3

Report (First Sit)
Description: Report on contemporary operating system threats.
Weighting: 40 %
Final assessment: Yes
Group work: No

Learning outcomes tested: MO1, MO2

Portfolio (Resit)
Description: Portfolio showing evidence of design, development, implementation, testing and debugging of a solution to a security problem.
Weighting: 60 %
Final assessment: No
Group work: No
Learning outcomes tested: MO3, MO4

Report (Resit)
Description: Report on contemporary operating system threats.
Weighting: 40 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO2

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Cyber Security Technical Professional {Apprenticeship-GLOSCOLL} [GlosColl] BSc (Hons) 2022-23