COMP7024- Operating Systems Security and Development
Coursework- Part 2: Understanding and modifying an OS- File Encryption
Semester 2- 2022-23

Learning outcome

(This exercise assesses LO (Learning Outcomes) 2 and 4.)
LO 2: Create system-level software that modifies and extends existing operating systems. Conduct experiments designed to evaluate the performance, security and reliability of their modifications and additions.
LO 4: Demonstrate a thorough understanding of multi-threaded/process systems through the design and implementation of communicating, multi-threaded systems software.

Task

A File Encryption prevents unauthorized access to files to those who physically access a computer/drive. This is especially important to protect lost/stolen devices' data/information against unauthorised accesses.

For this exercise, you will be designing software (/OS patch) that protects data while sitting on systems' storage, and then adding and implementing this on your selected Operating System.

Follow the steps below in order, as you might find it harder if you don't. Complete each step before moving on to the next.

- List the essential features that a File Encryption should have. Look at similar products for this (e.g., macOS FileVault).
- Design your software/patch to be added to your selected. This should involve communications with the existing parts of the OS.
- Implement and test your code using an incremental method.
- Embed your code in your selected Operating System and perform an integration test that includes designing and running an experiment to evaluate the performance of the implemented functionalities.

Deliverable, word limit, and deadline:

This exercise is worth 50% of the total marks for the module.

Your report covers

1. A **description** of your File Encryption (5% of the assessment component mark).
2. A list of functional and non-functional **requirements** and security features of a File Encryption (10%).
3. **Design** of your software/patch that includes communications with the OS (10%).
4. **Implementation** of your File Encryption including annotated C code (20%).
5. **Testing** plan for validating your software (10%).
6. Description of **integrating**/adding the implemented component/patch to OS (10%).
7. **Integration testing** plan for integrating your component/patch into the system. This includes designing and running an experiment that evaluates the performance of the implemented functionalities (10%).
8. Reporting the possible **limitations**, failures, and/or difficulties you experience in your work (5%).
9. A **conclusion** section that includes recommendations for extending the conducted work and personal reflection (5%).
10. **References** (using Harvard or Numerical style of referencing) and proper citation (5%).
11. Demonstrate a thorough understanding of multi-threaded/process systems through the design and implementation of your system. This is not a separate section in your report. Instead, it has to be addressed and present in the other sections, e.g., 2, 3, 4, and 6 (10%).

Submission

- Submit your final report via Moodle by 28th April 2023, 5:00 PM.
- Your report must at most be 1600 words (excluding tables, figures, annotated codes, and references).
- The reports longer than 10% of the word limit will be penalised; the extra words will not be marked.
- Marks and feedback will be available on Moodle 3 weeks after submission.
- This coursework is an individual piece of work. The University rules concerning plagiarism, syndication and cheating apply.
- Version Control: You'll need to use a version control platform that records your report development history i.e., GitHub. Your report will include the link to your GitHub repository. If you use any other repository, you must justify this in your report Appendices. Reports without a valid version control history will not be acceptable.

Marking rubric

Weighting	Section	Mark distribution				
		0	**1 to <50**	**50 to <60**	**60 to <70**	**70 to 100**
5%	A description of File Encryption	Not presented	Inadequately addressed: Poor quality content; incomplete /irrelevant.	Satisfactory: Provides an acceptable overview of File Encryption and some of its roles and features.	Good: Provides a good overview of File Encryption and good coverage of the majority of its roles and features.	Excellent: Provides a great overview of File Encryption and almost all its essential roles and features.
10%	A list of functional and non-functional requirements and security features of File Encryption			Satisfactory: A satisfactory list of some of the main functional and non-functional requirements and security features of File Encryption.	Good: A good list of most of the main functional and non-functional requirements and security features of File Encryption.	Excellent: An excellent list of almost all the main functional and non-functional requirements and security features of File Encryption.

10%	Design of your software/patch that includes communications with the OS			Satisfactory: Satisfactory design of File Encryption and its communications with the OS that addresses some of the basic parts of it.	Good: Good design of File Encryption and its communications with the OS that addresses most of the main parts of it.	Excellent: Excellent design of File Encryption and its communications with the OS that addresses almost all the essential parts of it.
20%	Implementation of your File Encryption including annotated C code			Satisfactory: Satisfactory implementation of the designed File Encryption and its communications with the OS including annotated code for some of the basic parts of it.	Good: Good implementation of the designed File Encryption and its communications with the OS including annotated code for most of the main parts of it.	Excellent: Excellent implementation of the designed File Encryption and its communications with the OS including annotated code for almost all the essential parts of it.
10%	Testing plan for validating your software			Satisfactory: Satisfactory testing plan that covers basic parts of the developed product.	Good: Good testing plan that covers the main parts of the developed product including details of test cases.	Excellent: Excellent testing plan that covers almost all the essential requirements of the developed product along with details of test cases.
10%	Description of integrating/adding the implemented component to the OS			Satisfactory: A fair high-level description of how to integrate the developed product to the OS and its communication.	Good: A good detailed description of how the developed product integrated into the OS and its communication.	Excellent: A clear, correct, detailed, and almost complete description of how the developed product integrated into the OS and its communication.
5%	Integration testing plan for integrating your component into the system			Satisfactory: Satisfactory integration testing plan that covers testing the basic functionalities after integration of the developed product.	Good: Good integrations testing plan covering the main functionalities after integrating the developed product, including details of test cases.	Excellent: Excellent integration testing plan that covers almost all the essential functionalities of the product after integration with details of test cases.
5%	Reporting the possible limitations, failures, and/or difficulties you experience in your work			Satisfactory: A fair report of the limitations, failures, and/or difficulties during the completion of the task.	Good: A good and well-presented report of the limitations, failures, and/or difficulties during the completion of the work with reference to the presented product.	Excellent: An excellent, clear, sound, and very well presented report of the limitations, failures, and/or difficulties during the completion of the work with reference to the presented product.

5%	A conclusion section that includes recommendations for extending the conducted work			Satisfactory: A fair conclusion that summarises some parts of the conducted work and gives some recommendations for extending it.	Good: A good conclusion that summarises most parts of the conducted work and gives good recommendations for extending it.	Excellent: An excellent conclusion that summarises almost all parts of the conducted work and gives clear, feasible, and sound recommendations for extending it.
5%	References (using Harvard or Numerical style of referencing)			Satisfactory: A satisfactory list of some references provided.	Good: Good list of some related valid references provided, which some cited inside the report.	Excellent: Excellent well-structured reference list containing valid resources/scholars, which all were cited inside the report.
15%	Demonstrate a thorough understanding of multi-threaded/process systems through the design and implementation of your system. This is not a separate section in your report. Instead, it has to be addressed and present in the other sections, e.g., 2, 3, 4, and 6			Satisfactory: Demonstrate a fair understanding of multi-threaded/process systems in the basic parts of the design and implementation of the system.	Good: Demonstrate a good understanding of multi-threaded/process systems in the main parts of the design and implementation of the system.	Excellent: Demonstrate an excellent understanding of multi-threaded/process systems in almost all the essential parts of the design and implementation of the system that needs concurrency/efficiency.