|  |  |  |
| --- | --- | --- |
|  | Faculty of Computing, Engineering and Science |  |

**Assessment Cover Sheet and Feedback Form** 2018-19

|  |  |  |  |
| --- | --- | --- | --- |
| Module Code:  CS2S562 | Module Title:  Secure Software Development | | Module Team:  Alun King, Janusz Kulon |
| Assessment Title and Tasks:  Practical Coursework 1 | | | Assessment No.  1 |
| Date Set:  24-Sep-2018 00:00 | | Submission Date:  22-Feb-2019 23:59 | Return Date:  22-Mar-2019 23:59 |

**IT IS YOUR RESPONSIBILITY TO KEEP RECORDS OF ALL WORK SUBMITTED**

|  |
| --- |
| **Marking and Assessment** |
| This assignment will be marked out of 100%  This assignment contributes to 60% of the total module marks. |
| **Learning Outcomes to be assessed** (as specified in the validated module descriptor [https://icis.southwales.ac.uk/](https://icis.southwales.ac.uk/studentmodules/13592/studentmodulespecifications) ):  1) To be able to describe the integration of security into the software development life-cycle and reflect on best practice in minimising code vulnerabilities.  2) To be able to apply principles of protection mechanisms, software security and secure design.  3) To be able to conduct static and dynamic security verification and assessment of a software application. |
| *Provisional mark only: subject to change and / or confirmation by the Assessment Board* |

# Your Task

Design and implement a piece of Console application software in to simulate a smart device that could belong to The Internet of Things.

The core functional requirements are:

* The smart service provided must be based on data produced by 3 (simulated) sensors attached to the device (e.g. a weather device might register data from temperature, air pressure and wind speed sensors).
* At least one sensor's input must require some processing of the data to provide the service (e.g. a weather device might use its height above sea level to offset the measured air pressure for the height of its location - i.e. normalise it to sea level)
* At least one sensor must request an action to be performed (by an actuator) that changes the state of the device. (e.g. put the wind sensor blades into neutral when there is a severe storm to protect them from damage)
* Sensor data must be collected in some persistent storage facility - providing historical data and saving current device state. (e.g. saving a log once every 10 minutes with all sensor data plus the device's supply voltage, internal temperature, any faults, etc)
* An interface to the smart device should be simulated that provides the service and allows to manage the smart device.
* Users of the software should only be allowed to view and modify the data that they have permission to access.
* Permissions are determined by the system administrator.
* The core non-functional requirements for the device are:
* It must be designed using Secure Design Principles and using two different Secure Patterns (e.g. Secure Logger and Access Control Patterns)
* It must be implemented by applying secure coding best practices (you must demonstrate in your program the use of: unsigned integers, signed integers, strings, file I/O, user keyboard I/O and pointers)
* Document the design of the application using a UML class diagram. In this document also outline how and exactly where in your code you have applied the secure coding best practices.

Submission instructions

* Check the marking grid on the last page. Does your program tick all the boxes (ideally in the 1st Class / Distinction column)?
* Zip-up the entire Visual Studio project folder and the design document into a single zip file.
* Name the file like this: CS2S562\_CW1\_EnrolmentNumber\_FirstName\_LastName.zip (Replace the placeholders above with your enrolment number, first and last name respectively.)
* Go to the Blackboard pages of this module, select the 'Assessment' link and use the upload facility there to submit the zip file to Blackboard.
* After uploading the system should show you a receipt screen. If that is not the case or in case of problems e-mail peter.plassmann@southwales.ac.uk immediately.

Notes about Implementing the Two Secure Patterns

* You are free to implement any 2 secure patterns such as the Secure Logger, Access Control, Builder (covered in lectures) or any other suitable pattern (e.g. found in literature). If you use the Authenticator Pattern we would be looking for:
* User Logins: several user logins implemented with usernames and passwords. The user is only granted access after entering the correct details. Support for multiple users.
* Persistence: authentication data should be stored persistently.
* Proof of ID: should be fully implemented so that no re-authentication is required during a single program run.
* For the Authorization Pattern we would be looking for:
* Resource Independence: a generic implementation that allows the pattern to be applied to any type of resource.
* Different user types: the program supports several different user types including administrators
* Authorization rules: different users have different access to different resources

# Marking Scheme:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Fail | Narrow Fail | 3rd Class / Pass | Lower 2nd Class / Pass | Upper 2nd Class / Merit | 1st Class / Distinction |
| Sensor data simulation 5% | * No sensor data input simulation was attempted or the input was hard-coded | * Sensor data input simulation was attempted but does not work reliably or is unrealistic | * 1 sensor data simulation is producing data although the data may be not fully realistic | * 1 sensor data simulation is producing realistic data. 1 more sensor provides data that are not fully realistic | * 2 sensor data simulations are producing realistic data. The third sensor provides data that are not fully realistic | * All 3 sensors produce fully realistic data |
| Processing of sensor data 5% | * Data processing from 1 of the sensors was not attempted at all * No attempt was made to change the state of one of the sensors | * Data processing from 1 of the sensors was attempted but it is trivial, unrealistic or faulty * An attempt to change the state of 1 sensor was made but it is trivial, unrealistic or faulty | * Very simple processing of the data of 1 sensor was performed. It may not be entirely realistic * The state of 1 sensor can be changed but the process is very simple somewhat unrealistic or often not working | * Adequate and reasonable realistic processing of the data of 2 sensors was performed * The state of 1 sensor can be changed in a simple process that is realistic and working most of the time | * Complex and realistic processing of the data of all sensors was performed * The state of 1 sensor can be changed in a process that is realistic and fully working | * Very complex and realistic processing of the data of all sensors was performed * The state of 1 sensor can be changed in a complex process that is very realistic and fully working |
| Persistent storage 10% | * No attempt was made to store or retrieve sensor data * No attempt was made to store or retrieve device data | * An attempt was made to load or save at least one set of sensor data but neither of them work * An attempt was made to load or save at least one set of device data but neither of them work | * An attempt was made to load and save at least one set of sensor data but only one of them work * An attempt was made to load and save at least one set of device data but only one of them work | * Both loading and saving of at least one set of sensor data work * Both loading and saving of at least one set of device data work | * Both loading and saving of several sets of sensor data work, providing a history * Both loading and saving of several sets of device data work, providing a history | * Both loading and saving of user defined amounts of sensor data sets work * Both loading and saving of user defined amounts of device data sets work |
| Interface 10% | * No attempt was made to display the sensor data * No attempt was made to allow device management | * An attempt to display sensor data was made but it is faulty, confusing or incomplete * An attempt to manage the device was made but it is faulty, confusing or extremely incomplete | * An attempt to display sensor data was made but it is occasionally faulty or slightly unclear * An attempt to manage the device was made but it is occasionally faulty, slightly unclear or covers only 1 function | * A good attempt was made to display the sensor data. All data are correctly displayed in a good layout * A good attempt to manage the device was made allowing to change 2 functions in a reasonably obvious way | * A very good attempt was made to display the sensor data. All data are correctly displayed in a good layout. Some limited user interaction such as history range setting or searching is possible * A very good attempt to manage the device was made allowing to change 3 functions in an entirely obvious way. Changes made may not be persistent | * An excellent attempt was made to display the sensor data. All data are correctly displayed in a professional layout. Rich user interactions such as history range setting or searching are possible * An excellent attempt to manage the device was made allowing to change 4 or more functions in an entirely obvious and professional way. Changes made are persistent |
| Secure Design Principles I (Environment Settings) 10% | * No compiler flag changes made * No linker flags changed * More than 10 compiler warnings still present * NuGet 'Guideline Support Library' not installed * 'Microsoft all Rules' not set |  |  | * Some compiler flags changed * Some linker flags changed * More than 5 compiler warnings still present |  | * /GS, Gs, guardcf, RTC and Wall used * ASAFESEH, DYNAMICBASE and NXCOMPAT used * No Compiler warnings present any more * NuGet 'Guideline Support Library' installed * 'Microsoft all Rules' set |
| Secure Design Principles II (Secure Coding Rules) 20% | * No checks for unsigned integer wrap or unsigned integers never used * No checks for signed integer overflow or signed integers never used * No attempt at implementing string security or no strings used * No attempt at implementing dynamic memory security or no dynamic memory used * No attempt at implementing keyboard IO security or no keyboard IO used * No attempt at implementing file IO security or file IO not used * No attempt at implementing pointer security or pointers not used | * Only 1 or 2 correct checks for unsigned integer wrap * Only 1 or 2 correct checks for signed integer overflow * String security attempted but faulty, very infrequent or inadequate * Dynamic memory security attempted but faulty, very infrequent or inadequate * Keyboard IO security attempted but faulty, very infrequent or inadequate * File IO security attempted but faulty, very infrequent or inadequate * Pointer security attempted but faulty, very infrequent or inadequate | * About a third of all occurrences of unsigned integers checked for wrap * About a third of all occurrences of signed integers checked for overflow * About a third of the code lines involving strings are secure * About a third of the code lines involving dynamic memory are secure * About a third of the code lines involving keyboard IO are secure * About a third of the code lines involving file IO are secure * About a third of the code lines involving pointers are secure | * About half of all occurrences of unsigned integers checked for wrap using C99 types and constants * About half of all occurrences of signed integers checked for overflow using C99 types and constants * About a half of the code lines involving strings are secure * About a half of the code lines involving dynamic memory are secure * About a half of the code lines involving keyboard IO are secure * About a half of the code lines involving file IO are secure * About a half of the code lines involving pointers are secure | * Almost all occurrences of unsigned integers comprehensively checked for wrap using C99 types and constants * Almost all occurrences of signed integers checked for overflow using C99 types and constants * Almost all of the code lines involving strings are secure * Almost all of the code lines involving dynamic memory are secure * Almost all of the code lines involving keyboard IO are secure * Almost all of the code lines involving file IO are secure * Almost all of the code lines involving pointers are secure | * All occurrences of unsigned integers comprehensively checked for wrap using C99 types and constants * All occurrences of signed integers comprehensively checked for overflow using C99 types and constants * All of the code lines involving strings are secure * All of the code lines involving dynamic memory are secure * All of the code lines involving keyboard IO are secure * All of the code lines involving file IO are secure * All of the code lines involving pointers are secure |
| Secure Pattern 1 20% | * Secure Pattern 1 not present | * An attempt was made to implement the Secure Pattern 1 but it is very faulty and not working | * An attempt was made to implement the Secure Pattern 1 but it is partially faulty and not always working correctly | * Implementation of the pattern present but slightly flawed | * Good implementation with only minor shortcomings | * Perfect implementation of the pattern |
| Secure Pattern 2 20% | * Secure Pattern 2 not present | * An attempt was made to implement the Secure Pattern 2 but it is very faulty and not working | * An attempt was made to implement the Secure Pattern 2 but it is partially faulty and not always working correctly | * Implementation of the pattern present but slightly flawed | * Good implementation with only minor shortcomings | * Perfect implementation of the pattern |
|  | | | | | | |