**Secure Software Coursework**

Username: Admin – Password: 321

Username: User – Password: 123

Functional requirements:

1. The three sensors are on the Sensors.cpp. There is Temperature, Depth and Current.
2. Depth sensor takes the input of speed, then depending on which selection of temp is chosen, then multiplies them together and divides by 100.
3. Temperature sensor gets the input of the temperature and checks to see if it is below 10 degree C. If so it turns on the heater, calling the function Sensor::Heater().
4. File I/O is within the Temperature sensor in Sensor.cpp. It saves the temperature that was recorded to the file. However only runs once and creates the file every time.
5. My interface class WaterType.cpp. Using the WaterTypeFactory.cpp to ask the user which water type they want to measure.
6. The users can only view the sensors and can’t choose which display they would like, just get the default it is set to which is Celsius.
7. The system Admin can choose if they want to display temperature in Celsius or Fahrenheit. The program gets the ID from the AuthInfo.cpp and calls the Sensor::checkAdmin function and passed through the ID. The sensor::checkAdmin function then stores whatever was passed into a global string variable. I know this is not secure, however this is the only way that I could figure out how to do it. This variable is then checked in Sensor::Temperature to see if it holds “Admin” if so then it allows the admin to choose between Celsius and Fahrenheit. In Sensors.cpp on lines 6, 16-19,35-57.

Secure Principles:

Integer Security: Sensors.cpp. Lines 26-32checking if random number generated is within the integer size, however it should never be out of bounds) and 194-204(to check that the formula to work out the distance is still within the integer size).

String Security: Sensors.cpp. Lines 85(strnscpy\_s is the secure version of strncpy to copy to a string) and line 115-131(making sure the pointer doesn’t return a NULL pointer and passing the character array into a string to check the validation)

Memory Management: Sensors.cpp. Lines 115-131(taking a pointer to a character user input using malloc to allocate memory, then freeing the memory after use.)

Formatted I/O: Sensors.cpp. Line 180(setprecision, to set it to 2 decimal places only.)

File I/O: Sensors.cpp. Lines 68-172(Saving the temperature to a file, asking the user if they want to display the file, hwoever this creates the file everytime.

Pointer Vulnerabilities: Main.cpp. Lines 34-39(deleting the pointers after use) and Sensors.cpp. Lines 120-124(to check if the array is a nullptr, it should never be.) Also in Main.cpp. Lines 22 to check for null pointer, if ProofofID doesn’t work, however, it should always work.

**UML Class diagram on next page.**

