CS-405-10861-M01 Secure Coding 2024 C-3

5-3 Activity: Static Code Analysis

Computer Science Department

Southern New Hampshire University

Eric Farkas

6/9/2024

**Approach Details:**

I followed the TODO instructions in the example source file between lines 10 and 26.

**Console Output:**

A screenshot of a computer error message

Description automatically generated

**Warnings and Errors:**

A black surface with white spots

Description automatically generated  
**CPPCheck Results:**

A screenshot of a computer

Description automatically generated

**Comparison of Results and Analysis:**

Visual Studio reported zero errors, two warnings, and one message. CPPCheck reported three errors, seven warnings, and sixteen messages. The uninitialized variable at line 42 returns the same warning from both analysis tools. The out-of-bounds condition value at line 66 returns a warning status from both analysis tools, but CPPCheck provides more specific details about the condition than Visual Studio does. The variable on line 64 returns a message from both analysis tools. CPPCheck reveals that the scope of the variable can be reduced, while Visual Studio shows the variable is not initialized. Both conditions are accurate, revealing that the analysis tools prioritize their responses on varying criteria. These are the common results that can be analyzed using both tools.

CPPCheck provided additional results that were not returned using Visual Studio. Lines 52, 59, and 87 contain errors that CPPCheck defines as high-severity problems. Line 50 contains a function stating exceptions should not be thrown; however, the object in line 52 throws an exception. This is a risk and should be corrected. Line 59 shows an error related to the pointer defined there. The pointer is for a local variable that is invalid. This is a risk and should be corrected. This could be easily accomplished by reassigning the pointer before the end of the function. Line 87 reports an invalid container error. This is due to the use of a vector that can vary in size. The vector could be replaced by a list to resolve this issue. This should address all errors reported by CPPCheck and eliminate the high-severity problems from the source file.

Five unaddressed warnings remain from the CPPCheck results. These warnings occur on lines 109, 127, and 129. Two warnings occur at line 109 and relate to the placement of the variable. The warnings are concerns about the variable not being used beyond the function. This is not a serious concern but could likely be eliminated by moving the variable outside of the function or by not using a variable here. Two warnings occur at line 129 and relate to the expectation of a Boolean value. The constant assigned within the function requires the return of a Boolean value. The operator could be changed to “equal” to obtain the proper true or false result. The warning at line 127 has to do with an assert command being used to assign a value. This is not typical as assert normally checks whether a true or false condition exists. Since this check is not needed, assert could be removed and z could simply be defined. This should address all warnings reported by CPPCheck and eliminate the medium-severity problems from the source file.

The sixteen messages reported by CPPCheck are low-severity problems related mostly to style or syntax. Lines 4 through 8 report missing “include” files which happens because of the way Visual Studio built the project. These messages can be ignored as the include files are correctly located within the Visual Studio project. The remaining messages have to do with variables not being used or shadowing outer variables. This would be helpful in code review to ensure there are no conflicts between the local scope and the enclosing scope, but for this assignment, no action would be required.