Static Code Analysis

Kenneth J Gollaher

CS-405 Secure Coding, Southern New Hampshire

Module 5-3 Activity: Static Code Analysis

Professor Ivan Gappy

October 2nd, 2022

Visual Studio Analysis Output

Text

Description automatically generated

Visual Studio Warnings List

Graphical user interface, text

Description automatically generated

Analysis on Visual Studio and Cppcheck

For this week’s activity on the static code analysis, I tested the provided code on Visual Studio and another test using the static code analysis tool: Cppcheck. There were four warnings found on Visual Studio that shared the same result on Cppcheck. The warnings occurred on Lines 42, 52, 66, and 129. For lines 42 and 66, they are both labeled a warning on both VS and Cppcheck. On line 42, there is a variable not initialized and line 66 it has two issues. There is an array out of bounds and an unused variable. We can limit the limit count to 0 to 9 and remove the unused variable. On line 52 in Cppcheck had a higher severity and labeled as an error vs a warning. There is an exception problem where an object was thrown when it wasn’t expected to do so. Line 52 is a risk and should be corrected. On line 129 there are two warnings. First, there is an assertion fail where the function is called inside of an assert statement. Second, there is an issue with the statement returning an integer instead of a Boolean.

The rest of the lines are shown just on Cppcheck. There are 10 lines labeled with a severity as style. They are lines that need to be analyzed for further review and correct where possible. It may not be an issue now but can be susceptible for exploitation in the future and they need to be tested for vulnerabilities. For example, lines 133, 134, and 135 are shadow variables for lines 117, 118, and 119 and can be condensed.

Also, in Cppcheck, line 59 is categorized as an error and is a risk. The pointer to a local variable is invalid. We can reassign the pointer prior to the end of the function. Line 87 is a risk due to an invalid container. Here we can use a list instead of a vector so resizing doesn’t occur. Line 50 is considered not a risk. If we make the function static this performance issue can be resolved. Lines 109 and 127 are labeled as a warning and not a risk yet. There is possible vulnerabilities and should be assessed. Line 109 is a parm issue and is not used outside of the function. We can test by removing the variable to see where it affects the program. Line 127 assigns a constant within the function and it requires the return of a Boolean. If we use the equal to operator it can return either true or false for that function.