**Module Five Static Code Analysis**

Joseph Veneski

Department of Computer Science, Southern New Hampshire University

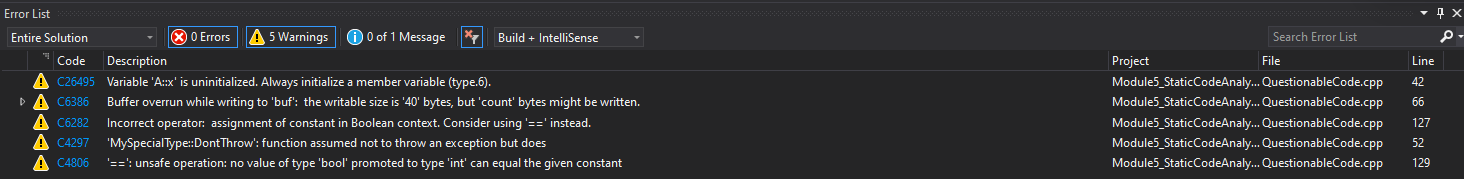
CS-405 Secure Coding

Professor Kaan Esendemir

April 7, 2024

**Module Five Static Code Analysis**

**Static Code Analysis:**

****

**A screenshot of a computer

Description automatically generated**

The first picture is the full list of warnings from Visual Studio. The second picture features the full list of warnings and errors from CPP Check with matching reports from Visual Studio highlighted.

Below are the non-matching entries with description, risk factor, system that identified the issue, and the line number in table format.

|  |  |  |  |
| --- | --- | --- | --- |
| RISK FACTOR | SYSTEM IDENTIFIED | SUMMARY | LINE NUMBER |
| RISK | CPP Check | Address of local auto-variable assigned to a function parameter.  Undefined behavior due to variable ‘b’ being deallocated after return of function foo(). The pointer \*a will point to a deallocated pointer. | 59 |
| NO RISK | CPP Check | Using iterator to local container ‘items’ that may be invalid.  Vector items is not verified, but use of begin() and end() prevent out of bounds issues. | 87 |
| RISK | CPP Check | Non-boolean value returned from function returning bool.  Function specifies returning Boolean value, this is likely unexpected behavior which will cause issues. | 98 |
| RISK | CPP Check | Assignment of function parameter has no effect outside the function.  while(tok); will infinitely run if tok is not a nullptr since the loop only checks if tok is non-null and nothing happens inside the loop. This is connected to the error below. | 109 |
| RISK | CPP Check | Either the condition ‘tok’ is redundant or there is possible null pointer dereference: tok.  After the above loop, there is an attempt to access tok->next() without verifying whether there exists another entry. This can result in attempting to dereference a null pointer and undefined behavior. | 109 |

**Process Summary:**

The images above clearly show the difference between how many errors Visual Studio compiler catches versus CPP Check. The static analysis tool CPP Check caught all errors mentioned by Visual Studio and then some. Many entries on CPP Check were ‘Information’ or ‘Style’ issues that were redundant due to already identified errors and thus were not listed in the risk assessment. Out of the list from CPP Check, five issues of concern were identified in addition to the issues caught by both systems, one of which was two issues relating to the same error (line 109).

Four of the five additional vulnerabilities were identified as potential risk factors. Assignment of pointers to variables that will become out of scope, returning a non-boolean value from a function with return type Boolean, a loop with no update logic, and attempting to dereference a null pointer are four major issues identified in the QuestionableCode.cpp file.

Verifying scope when working with pointers is essential to avoid a pointer to dereferenced memory. Always double check return types of functions as in the case of the function that specifies returning a bool, but the integer value ‘3’ is returned which can be confusing to others reading the code. The assert statement fails since returning ‘3’ from a function specifying return type Boolean leads to the return of ‘true’ for any non-zero value, not an integer. Ensure loops always have some sort of logic to update, and a method of exiting the loop. Finally, when working with containers, always ensure that operations are within bounds and values exist in indices or there will be risk of dereferenced pointers or bounds issues. Use of smart pointers, proper bounds checking, and loop constructs can greatly improve this code.