CS 405

Module 5 Static Code Analysis

Caleb Ewer

01/20/2025

Visual Studios Static Code Analysis Results:

A screenshot of a computer program

Description automatically generated

Cppcheck Static Code Analysis Results:

A screenshot of a computer

Description automatically generated

Summary of static code analysis results:

CppCheck identified several errors not detected by Visual Studio. While both tools agreed on the errors in the following lines, their descriptions differed slightly:

* **Line 64**: Accessing an array index out of bounds.
* **Line 52**: Throwing an exception in a function marked noexcept.
* **Line 129**: Side effects in an assert involving my\_function and unsafe operations.

CppCheck stands out by identifying more errors, including some critical ones, that Visual Studio did not flag. Below is a detailed comparison of the errors exclusively found by CppCheck:

|  |  |  |  |
| --- | --- | --- | --- |
| **Type of Error** | **Description of Error** | **VS or Cppcheck** | **Risk or not** |
| unusedStructMember | Stylistic issue that the member variable ‘x’ in class ‘A’ is defined but not utilized anywhere. Unnecessary code clutter. | Cppcheck | No Risk |
| uninitMemberVarPrivate | The member variable ‘x’ of class ‘A’ is not initialized in the copy constructor, leading to undefined behavior. Critical issue. | Cppcheck | Risk |
| functionStatic | The method ‘DontThrow; does not access ant instance-specific members of ‘MySpecialType” and can be marked static. Helps to improve performance. | Cppcheck | No Risk |
| autoVariables | Assigning the address of a local variable to a pointer parameter leads to dangling pointers after the function exits. Unpredictable results or potential crashes. | Cppcheck | Risk |
| arrayIndexOutOfBoundsCond | Accessing out-of-bounds indices in an array leads to undefined behaviors including memory corruption or crashes. Compromises program stability and security. | Cppcheck | Risk |
| unreadVariable | A potential oversight where the value assigned to ‘buf[count]’ is not utilized. Redundant or incomplete logic. | Cppcheck | No Risk |
| invalidContainer | Modifying a container invalidates iterators, and accessing such iterators can lead to undefined behavior. Causes crashes or corrupt data. | Cppcheck | Risk |
| returnNonBoolInBoolean  Function | Returning a non-boolean value from a function declared to return bool leads to implicit conversions. Lgical errors or unintended results. | Cppcheck | Risk |
| uselessAssignmentPtrArg | Assigning a value to a pointer parameter without dereferencing it has no effect on the original variable. Incomplete or incorrect functionality. | Cppcheck | Risk |
| nullPointerRedundantCheck | A null pointer check is redundant if a dereference is unavoidable. Unsafe code practice and risk of crashes. | Cppcheck | Risk |
| shadowVariable | A local variable redefines a variable in an outer scope. Leads to confusion or unintended bugs. | Cppcheck | No Risk |

CppCheck identified several critical issues that Visual Studio missed, such as uninitialized member variables, dangling pointers, invalid iterators, and unsafe null pointer practices. These are significant because they can lead to crashes, data corruption, and undefined behavior, posing substantial risks to the software's stability and security.

While some issues, such as unused variables and shadowed variables, are stylistic, addressing them can improve readability and maintainability. Visual Studio, while robust, may overlook nuanced issues like performance optimizations or subtle bugs, emphasizing the need for complementary tools like CppCheck.

CppCheck is a valuable addition to the static analysis process, as it highlights both critical and stylistic errors that Visual Studio may miss. By using both tools, developers can create safer and more robust code, reducing the likelihood of runtime issues and improving code quality.