Ty O’Neill

Professor Tam

CS-405

October 6th, 2024

Questionable Code Static Analysis

1. Dangling Pointer in *foo(int\*\* a**)* :
   * Risk: RISK
   * Detected By: CppCheck
   * The function *foo* assigns the address of a local variable *b* to *\*a*. After the function exits, \*a points to a memory location that is no longer valid, leading to dangling pointers.
2. Exception Thrown in *MySpecialType::DontThrow* :
   * Risk: RISK
   * Detected by: CppCheck
   * The method *DontThrow* is declared with *noexcept* but throws an exception, violating the exception specification and potentially causing *std::terminate* to be called.
3. Non-pure Function Call Inside Assert in *my\_function* :
   * Risk: RISK
   * Detected By: CppCheck
   * The function *my\_function* modifies the global variable *a* inside an *assert* statement. Since *assert* can be disabled in release builds, this side effect may be skipped, leading to inconsistent behavior.
4. Assignment of Function Parameter with No Effect *(**foo(int\*\* a)**)* :
   * Risk: NOT RISK
   * Detected By: CppCheck
   * Assigning *\*a = &b;* modifies the parameter *a*, but since *b* is a local variable, the assignment leads to undefined behavior rather than having an effect outside the function.
5. Variable Shadowing in *main* :
   * Risk: NOT RISK
   * Detected By: CppCheck
   * Local variables *x*, *y*, and *z* in the *try* block shadow the global variables of the same name, which can lead to confusion and potential bugs.
6. Unused Variables :
   * Risk: NOT RISK
   * Detected By: CppCheck
   * Multiple variables such as x, y, z, and others are declared and assigned values but never used, indicating redundant code.

After the comparison and analysis of the questionable code, it shows the strengths and limitations of both Visual Studio and CppCheck. CppCheck seems to have a deeper analysis of issues related to dangling pointers, exception misuse, and variable shadowing that Visual Studio’s analyzer did not detect. CppCheck also highlights style issues which overall promotes cleaner and more maintainable code. Visual Studio also had efficient detection in critical issues like buffer overruns and uninitialized variables, while providing this feedback immediately after compiling and running, which allowed for a more real-time issue discovery. However, CppChecks warnings feel more like false positives rather than actual bugs and it took some time to configure the application to get maximum effectiveness out of it. Which might be worth it with larger applications as Visual Studio clearly misses deeper issues and could miss more the larger the application grows.

