5-3 Activity: Static Code Analysis

Gavin M. Bish

Southern New Hampshire University

CS405: Secure Coding

Professor Philomena Ogoh

February 3rd, 2025

5-3 Activity

### **1. “Include file: <cassert> not found” (and similar missing‐include hints)**

* **Found by**: CppCheck only
* **Risk**: NOT RISK
* **Explanation**: CppCheck complains it cannot see certain standard headers in detail (“not found”). This does *not* indicate a genuine bug; it just means CppCheck was not given system include paths or it is skipping standard library parsing. There is no effect on runtime behavior or security.

### **2. “Class ‘A’ defines a copy constructor but does not define a destructor, copy assignment operator, move constructor, or move assignment operator.”**

* **Found by**: CppCheck only
* **Risk**: NOT RISK (mostly a style/best‐practice warning)
* **Explanation**: CppCheck flags that if you provide one special member function (like a copy constructor), you might also want to define (or default) the others. This is not inherently dangerous but can lead to unexpected copy or move semantics if you have special requirements.

### **3. “Throwing an exception whose type ‘const char\*’ is not derived from ‘std::exception’ [hicpp‐exception‐baseclass].”**

* **Found by**: CppCheck only
* **Risk**: NOT RISK (though some coding standards consider it a design risk)
* **Explanation**: This is a style or guideline issue that warns you are throwing a raw C‐style string instead of something derived from std::exception. It is not a direct security bug, but many standards (like CERT C++ or MISRA C++) encourage using types derived from std::exception so the catch blocks work more predictably.

### **4. “Function ‘\_\_\_’ can be made static or moved into an anonymous namespace”**

*(Seen multiple times for various functions: foo, work\_with\_arrays, do\_something\_useless, vector\_test, etc.)*

* **Found by**: CppCheck only
* **Risk**: NOT RISK
* **Explanation**: CppCheck suggests limiting function scope to the translation unit (by marking it static or placing it in an unnamed namespace) to avoid linkage collisions and possibly speed up linkage. It’s purely an optimization / style recommendation, not a correctness or security issue.

### **5. “Member function can be made static”**

*(Seen for MySpecialType::DontThrow and Token::next)*

* **Found by**: CppCheck only
* **Risk**: NOT RISK
* **Explanation**: Similar to the above. CppCheck sees member functions that do not use instance data and suggests making them static. This is a low‐level maintainability point and not a functional defect.

### **6. “Declaration and assignment can be joined”**

* **Found by**: CppCheck only
* **Risk**: NOT RISK
* **Explanation**: A minor style improvement. CppCheck notices that code declares a variable and then assigns it immediately, so it suggests combining them into one statement. This has no runtime or correctness implications.

### **7. “Variable ‘a’ can be made static or moved into an anonymous namespace”**

* **Found by**: CppCheck only
* **Risk**: NOT RISK
* **Explanation**: Same reasoning as with functions—this is just a scope/linkage suggestion that can reduce global namespace pollution.

### **8. “Typo in word ‘myobject’”**

* **Found by**: CppCheck only
* **Risk**: NOT RISK
* **Explanation**: CppCheck has a built‐in spell checker for identifiers and comments. It flags “myobject” as a possible misspelling. There is no security or functional risk here—it is purely advisory.

## **Summary of Differences**

Most of these CppCheck‐only messages are low‐priority style, scope, or naming suggestions that do **not** represent genuine security or correctness issues. The exception about throwing const char\* is also typically categorized as a style/design caution rather than a direct security hole. Consequently, all of the unique findings here are categorized as **NOT RISK** (or “low risk”) compared to the more direct errors (out‐of‐range access, etc.) that both tools flagged.

Screenshot’s

