Document N0603 **Dated 23 November 2015**

Top 10 rules to avoid programming language vulnerabilities

1. Validate input.

2. Check return values from subprograms.

3. Enable compiler static analysis checking and resolve compiler warnings and/or run static analysis tools to identify vulnerabilities in the code.

4. Perform range checking.

5. Allocate and free memory at the same level of abstraction. (*need to think about pointers and 1 or 2 sensible guidance that cover most languages. This a(llocate and free…) works as an e.g.)*

6. Ensure that the use of constructs whose effects are not deterministically specified does not affect the correctness of the application.

7. Ensure that undefined or deprecated language features are not used.

8. Make error detection, reporting, correction, and recovery an integral part of a system design.

9. Use only those features of the programming language that enforce a logical structure on the program.

10. Sanitize, erase or encrypt data that will be visible to others (for example, freed memory, transmitted data).

Meta-rule: Develop and use a coding standard based on this document that is tailored to your risk environment.