Guidelines to Editors of the Annexes

(V0.95 EP, 6/19/2015, draft for second review by WG23)

- Please keep the text short and simple and stay with the main issues. Likely readers are project managers and composers of coding guidelines, not language lawyers.

- Please use the general template of the Annexes (Annex B of the main TR), so that readers can easily migrate between Annexes. There have been changes to the section layout to conform to ISO standard layout. Please redistribute existing text accordingly; advice on content is given in the template.

- When addressing a core vulnerability, do not add your own vulnerabilities that are not represented in the core document. If your concerns address a vulnerability that is not unique to your language, please let the editing team know as they would like to hear about them for possible inclusion in the next version of the core document. If the vulnerability is unique to your language, add it to section 7 of your Annex.

- Avoid any kind of unsubstantiated general claims ("X is a safe language") in section 6. In the argument why a vulnerability does not apply, facts and concrete arguments are necessary.

- Place general guidance that applies to many vulnerabilities in section 5 rather than repeating it often in section 6.

- If terminology or concepts need to be explained, do it in section 3 or 4, respectively, of the Annex, not as part of the subsections for an individual vulnerability. (Apply this guidance with a grain of salt.) Section 4 is intended for general encompassing concepts and principles; section 3 is for technical definitions and terms.

- Use imperative style for giving advice (as these guidelines do). Advice should never be expressed like in this sentence. Express it by an imperative sentence like this one.

- As a lead-in for the discussion of each vulnerability, provide a statement or explanation to what extent the vulnerability described in the main document applies to your language. Several cases (and interim shades of grey) arise:
    1. The vulnerability does not arise in your language, for example because the feature under examination (e.g., arrays, pointers, references) does not exist in the language, or because the language has legality rules or run-time checks that protect against the vulnerability. In this case, write: *"This vulnerability does not apply to X, because <<< provide a very brief argument (1-3 lines) why this is so>>>*
       Caveats:
        - Often several variants of the vulnerability are described in the main document. Use this sentence only if indeed all of them are avoided. Otherwise go to alternative 2.

- If the mechanism of avoidance is the use of a tool whose use is not mandatory for all users of the language, the vulnerability does apply to your language! Go to alternative 2 or 3.
- Sometimes the vulnerability is described in terms that do not apply to your language, but the vulnerability exists nevertheless in a somewhat different guise. Please deal with the latter, and do not simply claim absence of the vulnerability, merely because you call a reference a pointer, or vice-versa, while indirection is the real cause of the vulnerability.
- The argument *"because the programmer can avoid it"* in all its variations is unacceptable "*because the hacker does not avoid and instead exploits it*"! Admit to the vulnerability and provide constructive advice to the benevolent user on how to avoid it. If you have constructive advice to prevent the malicious exploitation as well, by all means add it.

2. Most of the variants of the vulnerability or of their bad consequences are avoided in your language, but not all are. In this case, write:
   "This vulnerability is mitigated by <<<whatever language rule or principle avoids which problem>>. " Provide advice or rules on how to avoid the rest of the vulnerabilities. Also include advice to engage the mitigation mechanism if it requires user action or program code.
   Only language features (and rule-enforcing tool chains required by the language standard) constitute mitigation of a vulnerability in the language-specific description of the vulnerability. Naturally, users can, by their actions, mitigate the vulnerability as well, e.g. by using non-mandatory tools for checking or by avoiding features. For user actions, please do not use the term "mitigation" to avoid confusion. Put such actions in the guidance section instead as mechanism for avoiding an existing vulnerability. Whether or not a tool is mandatory, is either obvious (e.g. compiler or interpreter) or is described by you as part of the general concepts defining the language in section 4.

3. The vulnerability applies to the language X. In this case write: "This vulnerability applies to X." Provide advice or rules on how to avoid it in all the variants described in the main document.

4. Sometimes you will find yourself torn between alternatives 1 and 2 for cases where almost all variants are indeed excluded. It is o.k. to use "*Because <<< provide a very brief argument (1-3 lines) why this is so>>>, this vulnerability does not apply to X, except in the case <<<describe your exceptional remaining problem and, later, provide advice on avoidance>>>"*

5. Similarly, if your language avoids only a small part of the vulnerability, do not use the "mitigate"-version. Write: *"This vulnerability applies to X, except <<< describe your small good case>>>"*

- Please make sure that the user can understand why you are giving the advice that you give, so she or he can make an intelligent choice whether or not to follow the advice.