ISO/IEC JTC 1/SC 22 N 4432 2009-02-20

ISO/IEC JTC 1/SC 22
Programming Languages

Document Type:	**Summary of Voting/Table of Replies**
Document Title:	**Summary of Voting on SC 22 N 4420, ISO/IEC PDTR 24772, Information technology – Programming languages – Guidelines to avoiding vulnerabilities in language selection and use**
Document Source:	**SC 22 Secretariat**
Document Status:	**As per the results of this ballot, the PDTR has passed. WG 23 is instructed to take the comments received into consideration, prepare a disposition of comments text, and provide a revised text for TR balloting at the JTC 1 level.**
Action ID:	**ACT**
Due Date:	
No. of Pages:	**45**

Secretariat, ISO/IEC JTC 1, American National Standards Institute, 25 West 43rd Street, New York, NY 10036; Telephone: 1 212 642 4932; Facsimile: 1 212 840 2298; Email: lrajchel@ansi.org

Result of voting

Ballot Information:	
Ballot reference:	SC22-N-4420
Ballot type:	CIB
Ballot title:	**ISO/IEC PDTR 24772, Information technology - Programming languages - Guidelines to avoiding vulnerabilities in language selection and use**
Opening date:	2008-11-11
Closing date:	2009-02-11
Note:	**Please submit your vote via the on-line balloting system by the due date indicated.**

Member responses:	
Votes cast (17)	Austria (ON) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) France (AFNOR) Germany (DIN) Italy (UNI) Japan (JISC) Korea, Republic of (KATS) Netherlands (NEN) Romania (ASRO) Spain (AENOR) Switzerland (SNV) Ukraine (DSSU) United Kingdom (BSI) USA (ANSI)
Comments submitted (1)	Belgium (NBN)
Votes not cast (2)	Kazakhstan (KAZMEMST) Russian Federation (GOST R)

Questions:	
Q.1	"Does your National Body approve the PDTR text contained in SC 22 N 4420?"

Answers to Q.1: "Does your National Body approve the PDTR text contained in SC 22 N 4420?"

11 x	**Yes**	**Austria (ON)** **Canada (SCC)** **China (SAC)** **France (AFNOR)** **Germany (DIN)** **Italy (UNI)** **Japan (JISC)** **Korea, Republic of (KATS)** **Netherlands (NEN)** **Romania (ASRO)** **USA (ANSI)**
5 x	**Abstain**	**Denmark (DS)** **Finland (SFS)** **Spain (AENOR)** **Switzerland (SNV)** **Ukraine (DSSU)**
1 x	**No**	**United Kingdom (BSI)**

Comments from Voters		
Member:	Comment:	Date:
Canada (SCC)	***Comment File***	2009-02-03 13:01:11
CommentFiles/Canada(SCC).doc		
France (AFNOR)	***Comment File***	2009-02-02 13:12:37
CommentFiles/France(AFNOR).doc		
Italy (UNI)	***Comment File***	2009-01-29 15:59:12
CommentFiles/Italy(UNI).doc		
Japan (JISC)	***Comment File***	2009-02-10 02:53:40
CommentFiles/Japan(JISC).doc		
Netherlands (NEN)	***Comment File***	2009-02-09 15:44:13
CommentFiles/Netherlands(NEN).doc		
United Kingdom (BSI)	***Comment File***	2009-02-11 14:54:56
CommentFiles/UnitedKingdom(BSI).doc		

Comments from Commenters		
Member:	Comment:	Date:
Belgium (NBN)	***Comment***	2009-02-11 11:45:25

YES - NBN approves the PDTR text contained in SC22 N4420

Template for comments and secretariat observations Date:2009-02-20 Document: ISO_PDTR_24772

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
CA-3		6 title	Te	Title should reflect “weakness”	Change title to Weaknesses and Vulnerabilities	
NL-4		Sections 2, 6 and Blibliography	Ed	The system of literature references and cross references would probably need a review. It is assumed that Section 2 Normative references will be completed at a later stage. Bibliographic references are sometimes incomplete, sometimes erroneous; e.g. the names of Holzmann and Jones are spelled inconsistently. Apart from the bibliographic references in section 6.x.7 there exists a bibliography in an unnumbered section at the end of the document.		
JP	Bibliography	[14]	Ed	A punctuation mark should be given after “Seacord”.		
JP	Bibliography	[18]	Ed	A punctuation mark should be given after “Siek”.		
JP	Bibliography	Footnote	Ed	The footnote is numbered 3, but there is no footnote numbered 2. This footnote should be number 2.		
UK	Blank page		ed	There should be a blank page following the cover page so that page i is a recto page.	Add blank page after cover.	
UK	Cover page	title	ge	The title on the cover says "Guidelines ... vulnerabilities in language selection and use". But the title on page i is "Guidance ... Vulnerabilities in Programming Languages through Language Selection and Use"	Make them the same.	
CA-73	Cross references		Ge	There are no cross-references for Ada, yet C, C++, Java have references all throughout this document. Should there be an Ada cross-reference, such as the Ada style guide, Ada RM, etc?	Consider adding cross-references to Ada-style guide or other Ada related documents throughout this document.	
JP	Foreword		Ed	The title of SC22 itself is not correct. It is not “Programming Languages”, but “Programming languages, their environments and system software interfaces”.		
CA-1	general	3.1, 5. title 5, para 1 6, par 1 line2	Ge	The term Vulnerability should follow the custom in use now and refer to Application Vulnerability.	Change the title to ISO/IEC PDTR 24772, Information technology – Programming languages – Guidelines to avoiding application vulnerabilities through programming language selection and use	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
UK	general		te	This document does not seem to discuss problems caused by parallelism at all.	Add material on the problems of parallel programming.	
JP	Introduction		Ed	The word “behavior” is spelled in this way here. It is spelled as “behaviour” in this Technical Report.		
IT	Introduction	1	Ed	Term “vulnerability” is introduced with only an implicit correlation to its interpretation in the report: use of an explicit definition is preferable.	All programming languages contain constructs that exhibit undefined behavior, are implementation-dependent, or are difficult to use correctly. The use of those constructs may therefore give rise to *vulnerabilities*, as a result of which, software programs can execute differently than intended by the writer.	
UK	Introduction	para 1	ge	"All programming languages" is maybe too severe. One can conceive of a simple language that did not have the quoted properties. But maybe Gödel says that it would be too simple to be useful.	Change to "All practical programming languages" or "All serious programming languages" perhaps.	
UK	Introduction	para 2	ge	The title of this standard says " ...through language selection and use". But the introduction implies that the language has been chosen since it says "... in their chosen language". This is a serious inconsistency.	Clarify the purpose of this document. Is it to aid language selection or not?	
CA-2	pervasive	3.1 title 3.1 note	Te	The term Vulnerability should follow the custom in use now and refer to Application Vulnerability.	Change the term “language vulnerability” to “language weakness” and “programming language vulnerability” to “programming language weakness” as noted.	
CA-7	Whole document		Ge	Need Programming Language Annexes	The document cannot move forward until annexes for C, C++, Fortran, COBOL, Ada, Java, and EcmaScript are finished.	
UK	1.1	para 1	ed	Is this a list of three of four items? It is not clear.	If it is a list of three then write "security-critical, safety-critical, and mission and business critical". If it is a list of four then write ""security-critical, safety-critical, mission-critical, and business-critical".	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
UK	1.2	para 1	ed	It would read better if it said "... use configuration management tools, ..."	Add "tools" after "configuration management"	
CA-59	Section 1.3	Para 1	Ed	First two sentences do not read well, “The impact ... are likely to affect ... more people ... that worked on them”	The goal of this Technical Report is to provide guidelines that have a broad scope for usage and have the potential for larger savings at a smaller cost.	
UK	1.4	para 4	ed	There is a current trend in the misuse of English by inserting "would" everywhere. It's probably some sort of wimpish political correctness. There is no condition in this sentence for the would to apply to. Be positive -tell the truth. Do not hide behind an implied condition.	Change to "It is hoped .that such developers will use this document..."	
CA-60	Section 1.4	Para 4	Ed	It may not be possible to remove all vulnerabilities, so perhaps even improving an application would justify this document.	Replace :”are removed” with “are removed or at least minimized”	
UK	1.4.3		ge	I was expecting this section to be followed by a section entitled Business critical applications to match my understanding of the list in the first sentence of section 1.1	Either delete Business critical from 1.1 or add a section explaining it after 1.4.3.	
JP	2.		Te	No reference documents are cited here. This is not appropriate in this Technical Report. For example, language standards mentioned in the main text should be cited in this clause.		
IT	3	All	Ed	A note is missing on typographical convention that represent programming language keyword: the report seems to use the `courier` font for all terms that may be keywords or syntactic tokens in programming languages. The report is also ambivalent in the correct or only exemplary use of such terms (for example, it uses `inout` instead of `in out`). The conventions in use in this regard should be declared explicitly.		
IT	3.1	1	Ed	Term “property” does normally have a mathematical connotation, which is too rigid for the meaning intended in this report.	Replace property by feature.	
UK	3.1	Note	te	Another vulnerability that occurs because of the absence of a garbage collector is the simple one of running out of	add at the end "... or, on the other hand, by not freeing storage can result in the program failing by	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				storage.	running out of storage."	
UK	3.2		ed	This definition introduces the terms "security vulnerability", "safety hazard", and "defect". The first two are then defined in 3.3 and 3.4 but "defect " is not defined.	Define "defect".	
IT	3.4	1	Ed	Incorrect spelling of reference to IEC 61508-4. Subsequent uses of the reference should be corrected likewise.	IEC 61508-4 defines	
UK	3.4	second Note	ed	Spelling error "materiel".	Change to "material".	
FR	3.5	Note	ed	Typo: is some domains, a distinction is make	in some domains, a distinction is made	
JP	3.5	Note	Ed	We cannot understand the word “is” after “Not withstanding that”. The word “make” after “a distinction is” would be a misspelling of “made”.		
IT	3.5	Note	Ed	Various typos and poor structure of paragraph.	Notwithstanding that in some domains a distinction is made between safety-related (may lead to any harm) and safety-critical (life threatening), this Technical Report uses the term safety-critical for all vulnerabilities that may result in safety hazards.	
UK	3.5	sentence 1	ed	Saying "human injury or death" seems to imply that death is not an injury.	Change to "such as human injury and even death".	
UK	3.5	Note	ed	"Not withstanding that is some domains" reads oddly. Presumably it should be "in some domains"	Change to "... in some domains".	
CA-61	Section 3.5	Note	Ed	Second sentence doe not make sense	I think “is some domains” was supposed to be “in some domains”	
CA-62	Sect 3.6	Para 1	Te	This definition does not seem to be complete. A program can fully feet the requirements, yet be poorly written and unmaintainable with inherent security vulnerabilities. Not to mention that the requirements themselves may be poorly written or vague.	Replace “by its specification” with “by its specification and by the degree to which the software is maintainable, understandable, and free from undesirable behaviours and vulnerabilities.”	
JP	3.7	Note	Te	The example given in the definition of “Implementation-defined behaviour” is not correct. It says “using the value of a variable before it has been assigned”. This case		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				should be interpreted as an instance of “undefined”. 5.1.4 refers to this case in the definition of “undefined”.		
IT	3.7	Note (4)	Ed	Poor phrasing. The latter part (in yellow background in the proposed change) reads so unclear that it may possibly be taken off altogether.	This notion is related to neither unspecified behaviour, which is a characteristic of an application, nor the language used to develop the application.	
UK	3.7	Note 1	ed	Here is another pointless "would". Use present indicative.	Change to "this raises issues...".	
UK	3.7	Note 2	ed	This reads better with "to" inserted before "approach". Or maybe "in approaching" or "to move towards". The last is best. However, maybe the whole sentence is weird. The programmer doesn't have predictable execution it's the program that the programmer has written.	Change to ""a reasonably competent programmer to move towards the ideal of creating programs with predictable execution".	
UK	3.7	Note 3, bullet 1	ed	It would be preferable not to mention particular languages	Change to "in an expression in many languages".	
UK	3.7	Note 3, bullet 2	ed	Use subjunctive "be" after "that"	Change to "that this choice be documented".	
CA-63	Sect 3.7	Note 2	Te	Last sentence does not read well and appears to be missing a verb.	Replace “programmer approach” with “programmer to approach”	
UK	5	para 1	ed	Many will be aware of the danger of the use of the word sophisticated with its original meaning of adulterated. I guess that a clearer alternative would be cumbersome.		
UK	5	para 1	te	It's not always so much that programmers fail to understand the requirements but that the requirements are incomplete.	Rephrase to cover the possibility that the requirements are incomplete or wrong. This might need a new paragraph. It is an important issue.	
UK	5	para 5	ed	Rephrase first sentence	Change to "which can result in the use of a complex sequence...".	
UK	5	para 6	te	The line of code is one of several chunks sizes used by programmers, others include blocks, functions and even subexpressions within complicated expressions. Also paragraph appears disjoint and does not appear to add anything to the argument being made.	Delete.	
JP	5.		Ge	We cannot understand the objective or raison detre of this		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				clause in this Technical Report. Is it a summary of clauses 6 and 7? The second paragraph seems to be a part of the definition of scope of this Technical Report. The second last paragraph only gives a general statement and has no connection to paragraphs before and after it.		
UK	5.1	para 1	ed	This paragraph shows an instance of the fact that this document does not seem to address real-time or parallel programming much if at all. The third sentence says "Programming involves selecting and sequentially combining features...". This ignores the possibility of the combination being in parallel. Just omit "sequentially".	Change to "Programming involves selecting and combining...".	
UK	5.1	para 1	ed	The last sentence leads into the thought that difficulties are usually about the properties of a particular program rather than the world of all possible programs.	Incorporate this thought at the end.	
UK	5.1	para 2	ed	The phrase "to be an inconsistency" in the last sentence might perhaps be phrased more strongly. The word "wart" comes to mind.		
UK	5.1	complete subsection	te	While compiler selection is an important issue it is not within the scope of this TR.	Delete.	
UK	5.1.1	para 1	ed	The mention of "digital signature" seems unnecessary. The point is that the source has to be trusted. Moreover, compilers should be developed according to agreed standards. A reference to 7.4.4 would be a good idea.	Rewrite thus "..., unless coming from a trusted source and developed according to agreed standards, should ..." Add at end of paragraph "See 7.4.4".	
UK	5.1.1	para 2	ed	The verb "get" should be avoided in serious text. Use the passive.	Change to "After the source has been compiled...".	
UK	5.1.1	para 3	ed	Here is another unnecessary mention of "digital signature"	Delete "with a digital signature".	
UK	5.1.1		te	Compilers often have multiple options. If a compiler has 40 options say, then it may well be that the options used for a specific critical project have never been used by the compiler vendor for regression testing.	Add new paragraph "If a compiler has many options then developers should check with the vendor that the compiler has been validated with the combination of options to be used for a specific project. The vendor should supply the user with evidence that this is so."	
UK	5.1.2	para 1	ed	The third sentence uses "translator" whereas everywhere	Consistently use either the term "translators" or	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				else it has been "compiler". One wonders why. Did the author have interpreters in mind?	"compilers".	
UK	5.1.2	para 1	ed	The fourth sentence might read better if it said "to specify one particular behaviour" at the end.	Change to "... one particular behaviour."	
UK	5.1.3	para 1	ed	Maybe "a range of behaviours for a given language feature". We talk about a range of mountains with plural.	Change to " a range of behaviours for a given language feature".	
UK	5.1.3	para 3	ed	Probably "variations" in first sentence.	Change "variation" to "variations".	
UK	5.1.3	para 3	ed	In second sentence "code-checking tools" seems mundane. Code analysis sounds posher	Change to "code analysis tools".	
UK	5.1.4	para 2	te	It is not clear whether the phrase "has not yet been assigned" means assigned to or assigned from. More pedantic wording might help.	Change to "the use of the value of a variable to which there has not yet been an assignment".	
UK	5.2	complete subsection	te	Replace paragraphs 2,3 and 4 by (paragraph 2 is a good example but does not tie in with the following material and paragraph 4 only indirectly deals with the topic being covered by the clause). Suggested replacement wording:	Possible human cognitive factors include the following: • Cognitive failure, external pressures on readers and writers results in them failing to invest the time and effort needed to fully comprehend the code, • Knowledge failure: o people reading source code having incomplete and incorrect knowledge of the appropriate language semantics, o people reading source code having incomplete and incorrect knowledge of how it will be executed by a particular implementation, o people reading source code having incomplete and incorrect knowledge of the interaction between its various components, It is intended that this technical report identify issues that will enable a greater level of predictability to be achieved for the same level of investment of time and money. The following are some of the mechanisms used to achieve this goal:	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
					• reducing the amount of cognitive effort that needs to be invested by readers of the source code, • reducing the amount of knowledge needed by readers of the source code, • reducing the probability that incorrect developer knowledge will result in incorrect prediction of behavior, Aside from a propensity to introduce faults, another characteristic of humans is the tendency to look for solutions that reduce the amount of work required. Therefore, language constructs that make is possible to quickly write code that is more likely to contain faults are more likely to be used than the more time consuming constructs that are less likely to result in faults. Where a language construct has this property, mechanisms need to be put in place to discourage the use of constructs that are more likely to cause faults to be created.	
UK	5.3		ed	This whole discussion is a bit strange. Although the general idea is clear, the phrase "state of a program" sounds more like the dynamic behaviour than the lexical understanding. Perhaps it would be clearer if it simply said "intent of a program" everywhere.	Change "state" to "intent" perhaps.	
UK	5.3	para 2	ed	In the last line "interpreted" has a strange flavour. It is being used here in a linguistic understanding sense but in computer science an interpreter has a specific technical meaning which is not that required here. I would use the word "understood" again. And rephrase to get only out of parens.	Change to "It is only to a first approximation that code is read and understood line by line."	
JP	5.4	4th paragraph	Ed	There would be a grammatical error in the phrase “the implemented the library” .		
UK	5.4	para 1	ed	fixed point does not have a hyphen but floating-point does. Please treat them the same.	Put hyphen in "fixed-point".	
UK	5.4	para 3	ed	Third sentence needs restructuring.	Change to "...other sources and these are..." or	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
					perhaps "...other sources – these are ...".	
UK	5.4	para 4	ed	Third sentence, second bit in parentheses, delete "the" before "implemented"	Change to " ..may not have implemented ...".	
CA-4	6		Te	The document does not adequately address scripting languages	Investigate causes of the numerous and extremely hazardous vulnerabilities caused by scripting languages and develop guidelines to counter these effects.	
CA-5	6		Te	The document does not adequately address security domains	Languages such as Java have notions of security domains and explicit security and verification methodologies applied to the byte code and the byte code interpreter. This document should consider recommending to language implementers, especially of scripting languages, the creation and invocation of similar strategies.	
CA-6	Section 6		Ge	Need sections on concurrency	Will be supplied	
NL-5	Section 6		Ed	The 'arbitrarily generated three letter codes' for referencing descriptions are probably of little value to the user of this document given the non-mnemonic nature of these codes.		
CA-8	6.1.1		Te	The paragraph does not describe an application vulnerability	Add another paragraph The most obvious vulnerabilities are those associated with misunderstood code – erroneous results, exceptions or traps, timing difficulties, but can also result in more classic vulnerabilities such as arbitrary code execution or buffer overruns.	
JP	6.1.2		Ge	Subclauses called “Cross reference” are too terse. The convention for these subclauses is not explained anywhere in this Technical Report. No information is given about the referenced documents, and it is hard to access them without reference information.		
UK	6.1.2		ed	This remark applies to all these cross reference sections. There seems to be no proper list of the documents being referenced. The first three might be documents 5, 15, 20, of the Bibliography. But what is the fourth CERT/CC?	Add proper references.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				It is interesting to note that there is no reference to such a document for Ada. That is presumably partly because one is less likely to go astray in Ada, and if one uses Spark then most unlikely to go astray. Would the Ada 95 Quality and Style guide be relevant?		
CA-9	6.2.1		te	The only paragraph in this section does not specify that unspecified behaviour can result n different results each time a program is rebuilt using different tool options, such as optimization level,, different tools, or different versions of the same tool set.	Change "The external behaviour of a program whose source code contains one or more instances of constructs having unspecified behaviour may not be fully predictable when the source code is (re)compiled or (re)linked." To The external behaviour of a program whose source code contains one or more instances of constructs having unspecified behaviour may not be fully predictable when the source code is (re)compiled or (re)linked. The program can show different behaviour when built with different tool options such as optimization level, different tools, or different versions of the same tool set.	
JP	6.2.3	1st paragraph	Ed	The word "analyse" is spelled in this way here. However, all other appearances of the same word are "analyze".		
JP	6.2.3	3rd paragraph	Ed	The word "produces" in the phrase "the set of possible behaviours always produces" should be "produce".		
CA-10	6.2.6 6.3.6			Undefined orunspecified behaviours should be treated with even more caution	Add a bullet to each section that language designers should consider making these behaviours implementation-defined behavious (if they cannot eliminate them) to force them to be documented.	
JP	6.3.6		Ed	The term "language specifiers" is used here. However, other clauses use "language designers". It seems to be unnecessary to use different terms.		
JP	6.4.3	3rd paragraph	Ed	The word "behavior" appears twice in this paragraph. It is spelled as "behaviour" in this Technical Report.		
JP	6.4.5	6th bullet	Ed	We cannot parse the sentence "… by and for …".		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
CA-86	6.4.5		Te	Add a bullet to describe issues pointed out above in 6.14.4	Add a new bullet, “When a language requires full coverage of an enumeration in a switch/case statement, a default choice should not be provided. For languages that do not require full coverage, then a default choice with a possible error generation should be provided to ensure that there is full coverage”	
JP	6.4.6	1st bullet	Ed	Is the term “implementation behaviours” correct? Shouldn’t it be changed to “implementation-defined behaviours”?		
JP	6.5.4		Ed	The first line says “This vulnerability description is …”. The word “description” did not appear In the corresponding position of previous clauses.		
JP	6.5.5	3rd bullet	Ed	We think that the phrase “should not be used or used with caution” is not appropriate. The word “not” also applies to the latter half of the phrase.	Insert “should be” after “or”.	
CA-11	6.6			An issue that has been missed is that preprocessor directives are often included in a number of files, may agugment or replace a directive with another one	Give direction to application (6.6.4) that directives should always test to see if a variant has already been defined, and should not rely upon symbols defined outside of the file being processed.	
UK	6.6	complete subsection	te	This guideline is overly broad in recommended against the use of a preprocessing functionality. There are many situations where use of preprocessing (e.g., macros) is perfectly reasonable and there are situations where usage leads to faults. Significant instances of the fault causes should be highlighted and recommended against (the side-effects example given in this clause in one such usage).	Delete.	
CA-64	Sect 6.6.1	Para 3	Ed	Feels like there is a missing preposition at the end of the sentence, and there shouldn't be on there in the first place.	Replace “in the programming language that the code is written” with “in a given programming language”	
CA-65	Sect 6.6.1	Para 1	Ed	Avoid using the technical term "regular expressions" in this context.	Change "from the regular expressions programmers expect" with "from the expressions programmers regularly expect"	
JP	6.6.3		Te	The first two paragraphs are not the description of		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				"mechanism of failure".		
CA-65	Sect 6.6.3	Para 1	Te	The assertion that macros greatly decrease readability and maintainability is not a true statement. While it is true that macros **may** greatly decrease readability/maintainability, it is also true that macros can greatly increase readability/maintainablity. A macro is functionally similar to a function call in that it can create an abstraction for a grouping of statements. The wording should be more to the effect that macros are dangerous and more error prone because they can present a view of the source code that is very different from the view of the source code that the compiler sees. For the case of macros and the purpose of increased efficiency by avoiding a function call, we should recommend instead that languages provide mechanisms to inline functions and procedure calls, since that is a safer view, and because the compiler can check rules for such a construct. For the case of conditional compiling, I see the problem being that this is not scalable. The more conditional switches embedded in the code, the more unreadable and unmaintainable the code becomes. Another important point is that by having one source for multiple targets, a change to the source for one target could inadvertently break a compilation for another target. Such a change would ideally need to be tested for every combination of switches. Instead, having the build environment select where to pull source from, it is possible to make a change for a target, without affecting other targets, providing a large cost savings.	Change "maintainability is greatly" with "maintainability may be greatly" Add another sentence "Macros are error prone and dangerous to use because they can make the source code look very different from the preprocessed code that the compiler sees. Text substitutions are applied without regard to the syntax rules of the language, and a macro substitution may create a substitution that is legal for the compiler, but different than what the programmer had intended. Such errors may be difficult to detect in code reviews or at run time." Add "Conditional compilation directives do not scale well, as the more such switches exist in a software application causes the application to be more and more unreadable and unmaintainable. Making a software change for one target has the potential to break a compilation for another target. To fully test such a change would involve examining the effects of generating executables for every possible combination of switches. If instead separate copies of the source are maintained for each target, then it becomes possible to make a change for a target without having to worry about the effects for other targets, thus creating large savings in cost and time. Further, the code for a particular target is easier to understand and maintain."	
CA-67	Sect 6.6.3		Ed	Missing brackets in source code line	Replace "#define CD(x, y) ((x) + (y) - 1) / (y)" with "#define CD(x, y) (((x) + (y) - 1) / (y))" Note: This allows CD(x, y) to be used in an expression.	
CA-66	Section 6.6.3	Para 2	Te	While pre-processor can cause problems for static analysis tools, we should note that it should be possible to run the static analysis tools on the pre-processed source code.	Replace "analysis tools." with "analysis tools, although it should be possible to apply the static analysis tools to to the intermediate pre-processed source code."	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
JP	6.6.4	2nd bullet	Ed	We understand that most languages call this notion “arithmetic expressions”, not “arithmetic statements”.		
CA-68	Sect 6.6.4	Bullet 3	Te	What is an “improperly nested language construct”?	Suggest deleting this bullet or provide a better explanation or example of an improperly nested language construct.	
CA-69	Sect 6.6.6		Te	Should mention providing support for inlining capabilities	Add “Standards should consider providing capabilities to inline functions and procedure calls, to reduce the need for preprocessor macros.”	
CA-12	6.7.1			Does not specify an actual vulnerability	Add near the beginning the following: Name confusion can lead to the application executing different code or accessing different objects than the writer intended, or than the reviewers understood. This can lead to outright errors, or leave in place code that may execute some time in the future with unacceptable consequences.	
CA-70	Se 6.7.1	1st bullet	Te	Should mention that this sort of problem could also apply to linking errors. C for example typically puts an underscore in front of global symbols. There is potential that an application written in another language will not understand such a convention, or may simply link to a similar name other than the expected global symbol. This could be a difficult to detect problem. Are there other problems than linking problems related to this issue? Should we be more specific here and mention the problem with linkage?	Replace bullet with; “Large projects often have mixed languages and global symbols exported to the linker may have language specific prefixes, suffixes, or mangling applied. It is possible that the name of a symbol being imported into another part of the program may map to a different global symbol than the intended symbol. This may lead to unexpected software behaviour that could be difficult to detect or diagnose.”	
CA-71	Sect 6.7.1	Para 4	Ed	The second sentence starts with an opening square bracket, “[“, but there is no closing bracket.	Replace “computer languages.” with “computer languages.]”	
CA-72	Sect 6.7.1	Para 5	Te	This whole paragraph is confusing and has questionable value. It says there are similar situations, but doesn't say what they are, only what they are not. Either there should be some examples of these other “similar situations” or this paragraph should be deleted.	Possibly delete this paragraph, or provide examples of these “similar situations”	
JP	6.7.4	2nd bullet	Ed	We suspect that “invariances” should be “invariants”.		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
JP	6.7.7		Ed	Is the URL www.coding-guidelines... complete? Aren't last few characters missing?		
CA-13	6.8 (all)	AJN		This is a mess. May not even belong here.Specifics: 6.8.1 is a writep of the problem, not the application vulnerability. The vulnerability is what happens to the program when it encounters such errors. 6.8.3 is needs a description of more typical failures, such as creating badly named external objects, inaccessible external objects, and accessing the wrong object, and the usual effects on the program, such as denial of service or erroneous executions. 6.8.4 bullet 2 is not a language characteristic	Return to the committee for a rewrite of AJN	
JP	6.8.1	3rd paragraph	Ed	The case of "n" is not consistent. It is sometimes spelled in lowercase, and sometimes in uppercase.		
CA-74	Sect 6.8.5		Te	Should also add that the programmer should avoid creating names longer than the minimal length supported by potential ports of the application to various operating systems.	Add a new bullet, "Avoid creating resource names that are longer than the minimal unique length of all potential target platforms."	
CA-14	6.9.1			Unused variable – app vulnerability, This does not give a vulnerability	Add: Unused variables by themselves are innocuous, but can be combined with other vulnerabilities such as index bounds errors and buffer overflows and may mask errors or provide hidden channels.	
JP	6.9.2		Ed	The font of "563" is not correct.		
CA-75	Sectin 6.9.3	Para P3	Te	Ignoring the return status of a function call is one common cause of an unused variable. The call may require a variable to be declared, but if the status is not checked, this could lead to erroneous behaviour. E.g., Not checking the return status of a memory allocation call. The programmer may feel that a memory allocation is not likely to fail, so don't bother writing code to handle that case. This should be made more explicit in the wording.	Replace, "coding error;" with "coding error such as ignoring the return status of a function call;"	
Ca-	Sect 6.9.5		Te	Add suggestion to provide handling of all function call	Add a bullet, "Add handling for any return status of	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
76				results, or output values from a procedure call.	a function call, or any output values from a procedure call.	
JP	6.10.1		Ed	The phrase “without a diagnostic being issues” seems to be incorrect.	Change “issues” to “issued”.	
CA-15	6.10.1		Te	Identifie reuse – does not give an actual application vulnerability.	Add a paragraph: When human do not recognize which identifier is being used, the program will behave in ways that were not prediced by reading the code. This is usually found quickly in test, but circumstances can arise (such as the values of the same-named objects being mostly the same) where harmful consequences occur. This weakness can also lead to vulnerabilities such as hidden channels where humans believe that important objects are being rewritten or overwritten when in fact other objects are being manipulated.	
CA-77	Sect 6.10.1	P1	Ed	Typo	Replace “being issues” with “being issued”	
CA-16	6.10.4		Te	Missing notion that systems with different ranges for names (ex compiler vs linker) need a guide	Add: Languages where unique names can be transformed into nonunique names as part of the normal toolchain.	

Ca-78	Sect 6.10.4	3rd last para	Te	It states that the situation only occurs in languages that allow multiple declarations of the same identifier. This is true for the example given involving global symbols, but its a bigger problem if you have a nested scope using local variables. In this case the situation occurs in languages that support nested scopes.
				eg.
				int a_long_symbol_definition_lookup_table_a = 3;
				{
				int a_long_symbol_definition_lookup_table_b;
				a_long_symbol_definition_lookup_table_b = 4;
				}
					Suggest adding this example, because it applies to more languages than the first example.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				if the second variable declaration is removed, the second assigment may still assign the value to the first variable, even though the name is different. This is bad.		
CA-17	6.10.5		Te	Missing guidance for reduced name length situations	Develop or use tools that identify name collisions or reuse when truncated versions of names cause conflicts.	
UK	06/10/07	para 1	ed	Citations should be the same as in 6.7.7.		
JP	6.10.7		Ed	The reference “Jones 2007 (sentence 792)” does not give any information to ordinary readers.		
CA-79	Sect 6.11.3	2nd last para	Ed	The use of the term “coder” is inconsistent with “software programmer” used elsewhere in the document.	Replace “coder” with “software programmer” or just “programmer'	
JP	6.11.5	4th bullet	Ed	The term “tooling” seems to be unusual. Shouldn’t it be changed to “tools”?		
JP	6.11.5	5th bullet	Ed	We cannot parse the phrase “without further analysis a cast”.		
CA-80	Sect 6.11.5		Ed	Typo	Replace “further analysis a cast;” with “further analysis;”	
JP	6.12.1		Ed	Shouldn’t a word “a” be inserted between “it is” and “common practice”?		
CA-18	6.12.1		Te	This is not a statement of an application vulnerability	Move the discussion of 6.12.1 into 6.12.3. New 6.12.1: Interfaceing with hardware, other systems and protocols often requires access to to one or more bits in a single computer word, or access to bit fields that cross computer words for the machine in question. Mistakes can be made as to what bits are to be accessed because of the “endianness” of the processor (see below) or because of miscalculations. Access to those specific bits may affect surrounding bits in ways that compromise their integrity. This can result in the wrong information being read from hardware, incorrect data or commands being given, or information	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
					being mangled, which can result in arbitrary effects on components attached to the system.	
CA-81	Sect 6.12.1	Para 1	Te	The sizes of the integer bit sets supported is more often related to the target platform and the implementation of the compiler rather than the language itself.	Replace "by a particular language" with "by a particular target platform".	
CA-19	6.12.5			Incomplete	Add the following bullets: - placing all code associated with explicit manipulation of bits and bit fields into isolated program units to contain the specific access issues and provide interfaces that work with whole words as provided by the language system.	
UK	06/12/07	para 1	ed	While the cited web page contains some useful information it is not sufficiently informative or specific to the aims of the TR to act as a reference.	Change to: "Hacker's Delight" by Henry S. Warren Jr. published by Addison Wesley ISBN 0-201-91465-4.	
CA-20	6.13.1		Te	Does not give an application vulnerability	Add a paragraph: Floating point suffers from inexactness of representation and inprecision (the precision depends upon the relative size of the number). As a result, algorithms that use floating can have anomalous behaviour when used with certain values. The most common results are erroneous results or algorithms that never terminate for certain segments of the numeric domain, or for isolated values.	
UK	6.13.1	para 1	te	In the first sentence, the proportion mentioned is zero! So better reverse the sense to say that most cannot.	Change to "Most real numbers cannot be represented exactly in a computer."	
JP	6.13.3	3rd paragraph	Ed	We cannot parse the sentence "… when using other representations as can …".		
JP	6.13.3	3rd paragraph	Ed	Two occurrences of the word "uncertainly" in the last sentence should be "uncertainty".		
CA-21	6.13.3	Para 2	Te	Floating point is rarely used as a loop counter	Change sentence 1 to say: Using a floating point value as the loop termination condition can propogate rounding ...	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
UK	6.13.3	para 2	te	The third sentence is dubious and encourages the view that avoiding equality of floating-point values solves the problem which it does not.	Delete third sentence.	
CA-82	Sect 6.13.3	Para 2	Te	The choice of using a particular floating point precision representation may also affect the choice of algorithm used.	The choice of using a particular floating point precision representation may affect the choice of algorithm used. For example, a 32-bit floating point number, regardless of implementation, has a 6-digit precision. This is insufficient for the arc-cosine term near the zeros in a great circle calculation when the law of cosines algorithm is used. The Haversine algorithm is used in this case without the need for an arc-cosine term. Reference: R.W. Sinnott, "Virtues of the Haversine", Sky and Telescope, vol. 68, no. 2, 1984, p. 159.	
CA-22	6.13.5		Te	Lacking recommendations for algorithms that are iterative	Add a bulllet: - for algorithms that perform floating point calculations that depend upon small deltas for termination, provide analysis that the algorithm terminates under all possible inputs	
UK	6.13.5	bullet 1	te	This bullet item makes no sense from the point of view of numerical analysis. For a given algorithm and accuracy of the floating-point system, the acceptable tolerance will vary.	Replace by "Unless the use of floating-point is very simple an expert in numerical analysis should check the stability and accuracy of the algorithm employed."	
CA-83	Sect 6.13.5		Te	New bullet	Understand the implication of an algorithm when different precision floating point representations are used.	
JP	6.13.6	2nd bullet	Te	ISO/IEC 10967-3 is referred to here, but it seems inappropriate. 10967-3 defines types and operations for complex numbers. Those for usual floating-point numbers, which is the theme of this subclause, are given in 10967-1 and -2.		
CA-23	6.13.6	Bullet 1	Te	IEEE 754:2008 is now a standard	Remove the reference to IEEE 754R, and discussion of the revision.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
CA-84	Sect 6.13.6		Te	It seems that it would be desirable to have a compiler issue warnings for attempts to test equality for floating point values.	Add a bullet, “Languages should consider providing a means to generate warnings for code that attempts to test equality of two floating point values”	
UK	6.13.7	para 2	ed	The references given, although useful, do not provide the best detailed practical advice.	Add a further reference "N J Higham, Accuracy and stability of numerical algorithms, Siam, 1996".	
CA-83	Sect 6.14.1	2nd para	Ed	Typo	Replace “have the wrong maps.” with “have the wrong mappings.”	
JP	6.14.2		Ed	“Holzmann rule 6” is referred to here. 6.6.2 and 6.38.7 refer to “Holtzmann …”. “Holzmann” and “Holtzmann” seem to be the same document.		
JP	6.14.3	2nd paragraph	Ed	The term “undefinable values” is quite unusual.		
CA-84	6.14.3	Para 3	Te	Does "lost material" mean "unreachable memory" here? There is the possibility of creating large array objects in this scenario.	Clarify "lost material", or add " possibility of creating large array objects" to the list.	
JP	6.14.4	1st bullet	Ed	The sentence is unreadable after “and coverage analysis …”. It seems to be an erroneous copy of the second bullet of 6.14.5.		
CA-24	6.14.4	Bullet 4	Te	This is not a bullet, but the rest of bullet 3	Make part of bullet 3	
CA-85	6.14.4	1st bullet	Te	Part of this bullet should be moved to 6.14.5. It is more about how to avoid the vulnerability than a language characteristic. It also seems that the issue is not correctly captured. The real issue is that some languages do not require full coverage of an enumeration in a switch/case statement. This is a problem because values can be added to the enumeration, but the software programmer may fail to add handling for the new value in every switch/case statement, which may lead to erroneous behaviour. The current wording suggests that Ada is unique in that it has a problem in this area, whereas in fact other languages are worse in this regard. The optional use of “others” in Ada at least provides full coverage for the enumeration, as does the “default” keyword in C, C++, etc. The interesting point	Replace the first bullet with; “Languages that do not require full coverage of an enumeration in a switch/case statement” Add, “Languages that provide a default choice in a switch/case statement.”	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				is that languages that require full coverage should avoid having a default choice, since it improves maintainability (adding a new value to the enumeration causes the compilation to break and points out where changes need to be made), whereas languages that do not require full coverage should recommend using a default choice, because at least that gives you full coverage.		
CA-25	6.15.1		Te	This section does not give an application vlnerability	Add a paragraph: Type conversion errors can lead to erroneous data being generated, algorithms that fail to terminate, array bounds errors, and arbitrary program execution.	
JP	6.15.3	2nd paragraph	Ed	There is no verb in the sentence “the failure of Ariane 5 launcher …”.		
CA-87	6.15.4	Ist para after bullets	Ed	Sentence can be misread	Replace “Verifiably in range operations” with “Verifiably in-range operations”	
JP	6.15.5	2nd paragraph	Ed	The word “Verifiably” seems to be a misspelling of “Verifiability”.		
CA-26	6.15.5	Last para	Te	The discussion is too C-specific for this part of the document	Move paragraph to the C language Annex	
CA-88	6.15.6	2nd bullet	Te	Suggest the alternative of at least generating compiler warnings.	Replace “explicit.” with “explicit, or at least generating warnings for implicit conversions.”	
CA-27	6.16.1		Te	Section missing application vulnerability	Add a sentence: The results of an exploitation can be buffer overflows, data corruption, unlanned program termination, and arbitrary code execution.	
FR	6.16.3		ed	Unclear formulation: String termination errors occur when the termination character is solely relied upon to stop processing on the string when the termination character is not present	… and the termination character is not present	
FR	6.16.5	2nd bullet	te	This is too language specific	Move to the C language annex.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
CA-28	6.17.1		Te	This section does not give an application vlnerability	Add: As for all out-of-bounds accesses, this can result in corrupted data, premature program termination, non-terminating algorithms and arbitrary code execution.	
FR	6.17 6.18 6.19 6.20		te	These four clauses describe the same vulnerability. 6.17 (XYX) addresses access outside an array from the "low" end, while 6.18 (XYZ) is general, and 6.19 (XYW) addresses improper access through some library functions peculiar to C. 6.17 is the only one to assess that the vulnerability "may modify internal runtime housekeeping information". This assumes that the array is pushed on the stack on top of other information, and that the lower bound of the array is the end that is closest to the internal information. This is both language and implementation dependent, and is no reason to treat underflows differently from overflows. 6.19 is peculiar to the C family of languages. Most high level languages use simple assignment to copy arrays. It is hard to tell how 6.20 differs from the preceding clauses.	Merge the three clauses, move C peculiarities to the C language annex.	
JP	6.17.3	4th bullet	Ed	An apostrophe should be inserted in “functions” and “programs”.		
FR	6.17.3		ed	(in both cases … There are more than two bullets following this phrase.	(in all cases	
FR	6.17.3	4th bullet	ed	when the array …	when an array…	
FR	6.17.4	1st bullet	ed	Remind how it can happen	add at the end of the sentence: (either by means of an index or by pointer	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
					arithmetic).	
FR	6.17.5		te	The first paragraph after the bullets (Some guideline document…) talks about a non-recommended way of avoiding the vulnerability, and is therefore useless. Moreover, it is applicable only to the C family of languages, since other languages may have arbitrary bounds, including negative values.	Delete paragraph, or move to the C language annex.	
FR	6.17.5		te	The second paragraph after the bullets (In the past…) is about language implementation, not avoiding the vulnerability	The paragraph should be moved to "language standardization" if necessary to justify a recommendation that all bounds should be checked, or removed.	
JP	6.17.6		Ed	The meaning of the term “standard” is not obvious in “consider specifying a standard for a pointer type”. Shouldn’t it be changed to “standardized feature” or something like this?		
CA-29	6.18.1		Te	This section does not give an application vlnerability	Add: As for all out-of-bounds accesses, this can result in corrupted data, premature program termination, non-terminating algorithms and arbitrary code execution.	
FR	6.18.3	1st paragraph	ed	The whole paragraph is very confused, and addresses various issues	Restructure and make the various mechanisms clearer, by using bullets for example.	
JP	6.18.3	3rd paragraph	Ed	The word “accessed” in “against out of bounds accessed” seems to be a misspelling.	Change it to “accesses”.	
JP	6.18.4	1st bullet	Ed	Is the term “bounds check” a proper verb?		
JP	6.18.6	1st bullet	Ed	The word “Language” should be changed to “Languages”.		
JP	6.19.1		Ed	The word “being” appears twice in sequence.		
CA-30	6.19.1		Te	This section does not give an application vlnerability	Add: This can result in corrupted data, premature	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
					program termination, non-terminating algorithms and arbitrary code execution.	
FR	6.19.3	2nd paragraph	ed	Missing period after first sentence	The arguments to these library functions include the addresses of the contents of the two storage areas and the number of bytes (or some other measure) to copy**.** Passing the…	
	6.19.3	2nd paragraph	Ed	The word “is” should be deleted in “makes it is possible”.		
	6.19.3	2nd paragraph	Ed	3-letter code “XYZ” should be enclosed by square brackets.		
	6.19.4	2nd bullet	Ed	3-letter code “XYZ” should be enclosed by square brackets.		
FR	6.19.6	2nd bullet	te	The canary technique does not provide bounds checking, it detects (some) buffer overflows after the fact.	Remove improper recommendation. Recommend that languages provide true assignment for arrays.	
	6.19.6	2nd bullet	Ed	What is the meaning of “canary style”?		
CA-31	6.20.1		Te	This section does not give an application vlnerability	Add: As for all out-of-bounds accesses, this can result in corrupted data, premature program termination, non-terminating algorithms and arbitrary code execution.	
FR	6.20.3		te	"Overwriting adjacent data (or data at arbitrarily computed locations) outside the area allocated for an array leads to *value* failures of the application." is the term "value" the intended one ? all failure mechanisms described in the previous clauses are also applicable.		
CA-32	6.20.4		Te	Missing applicability to languages that permit check suppression	Add a bullet: - Languages that provide bounds checking but permit the check to be suppressed	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
CA	6.20.4		Te	Missing applicability to languages that permit check suppression	Add a bullet: - Languages that provide bounds checking but permit the check to be suppressed	
CA-33	6.21.1	Para 1	Te	Missing an application vulnerability	Add before the final sentence: Improper access via a function or method pointer can result in program termination or in in arbitrary code execution.	
JP	6.21.2	Title	Ed	The title line is unintentionally indented.		
FR	6.21.3		te	The vulnerability described here is more about pointer types, not values.	If a pointer's type ~~or value~~ is not appropriate for the data or function being accessed,	
FR	6.21.4	3rd bullet	te	This has nothing to do with casting and is covered in the next vulnerability	Remove	
FR	6.21.4	4th bullet	te	This has nothing to do with casting and is covered in the next vulnerability	Remove	
JP	6.21.5	2nd bullet	Ge	Cross-referenced documents "JSF AV, CERT/CC, Hatton, or MISRA C" are referred to here. Such documents are not referred to in other places of this Technical Report. Is it appropriate?		
CA-34	6.21.6		Te	Missing implications for standardization	Add: Languages should consider creating a mode that provides a runtime check of the validity of all accessed objects before the object is read, written or executed.	
JP	6.21.7		Ed	The document is also referred to in 6.21.2, so this is a double reference.		
CA-35	6.22.1		Te	Section does not provide sufficient application vulnerability.	Delete the last portion of the sentence (after the comma) and replace with: ..., which in turn can cause corrupted data, unplanned application termination, and arbitrary code execution.	
FR	6.22.1		te	The view that pointer arithmetic is used to index buffers is	Change to:	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				purely C-centric	… can lead to miscalculations that can result in addressing arbitrary locations …	
FR	6.22.3	3rd and 4th bullet	te	Buffer overflows and underflows are just special cases of addressing arbitrary memory locations (the 3rd bullet)	Remove	
CA-36	6.22.5		Te	Insufficient guidance	Add: - Use array notation instead of pointer arithmetic - Statically show the correctness of all operations for all possible input values	
FR	6.22.5	1st bullet	te	The recommendation should be inverted, since languages that allow pointer arithmetic to access array elements also have a proper syntax	Use proper indexing for accessing array elements rather than pointer arithmetic	
FR	6.22.5		te	Pointer arithmetic is justified only when addressing raw memory is needed (memory mapped devices for example)	Add a bullet: Use pointer arithmetic only when addressing raw memory is necessary	
FR	6.22.5	2nd bullet	te	The recommendation is not clear. It should state what is forbidden rather than what is allowed.		
FR	6.23		te	It is very strange to have a clause for null pointer dereference, and not for the general case of invalid pointer (as can result from an uninitialized pointer variable), although the description states that “this is a special case of accessing storage via an invalid pointer”	Rewrite the clause for the general case of invalid pointer values	
CA-37	6.23.1		Te	Missing implications for standardization	Move the final sentence of 6.23.3 to this section.	
FR	6.23.3		te	Many languages do *not* initialize pointers to null	Remove the sentence that states that “pointers are typically are initialized to null”.	
FR	6.23.3		te	Null dereference can also access random memory places (typically at address 0) without causing runtime errors	Add at the end of the last sentence: or accessing arbitrary memory locations	
FR	6.23.4	2nd bullet	te	All languages with pointers allow “the use of the null pointer”. How could it be otherwise?	Remove	
FR	6.23.6		te	The recommendation makes little sense if the language	Change the beginning of the bullet to:	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				does not automatically initialize all pointers to null	Pointers could be initialized automatically to null, and checked for the null value…	
JP	6.24.1	2nd paragraph	Ed	3-letter code “DCM” should be enclosed by square brackets.		
FR	6.24.1	3rd paragraph	te	A pointer is not necessarily an address	Change the end of the first sentence to: … twice on the same pointer value	
FR	6.25.1	1st paragraph	te	Generics can be parameterized by other elements than types. For example, in Ada, they can be parameterized by values, objects, subprograms, packages and interfaces. Constructs other than objects and functions can be generic (procedures and packages in Ada).	Change the first sentence to: Many languages provide a mechanism that allows language constructs to be parameterized by other language constructs. (there are also other mentions of types as parameters in other subclauses)	
JP	6.25.3	5th paragraph	Ed	There would be a grammatical error in the phrase “being designed the C++ committee”, probably a missing “by”.		
FR	6.25.3	4th paragraph	ed	typo	In the second sentence: a generic class defines a serie~~s~~ …	
FR	6.25.3	5th and 6th paragraphs	te	These paragraphs are specific to C++	Move to the C++ language annex	
FR	6.25.4	1st bullet	te	The vulnerability applies only to languages that do not enforce a contract model for generics at compile time. Parameterization is not limited to types.	Change bullet to: Languages that permit definitions of constructs that can be parameterized without enforcing the consistency of the use of the parameters at compile time Remove Ada from the list, since the properties of generic parameters are checked at compile time (To be verified: case of Java)	
FR	6.25.5	2nd bullet	te	This is a recommendation for language design, not for software developers	Move to 6.25.6	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
JP	6.25.6		Ed	The word “that” appears twice in sequence.		
FR	6.25.6		te	Add recommendation to enforce the contract model at compile time or run time.	Add a bullet: Language specifiers should design generics in such a way that any attempt to instantiate a generic with constructs that do not provide the required capabilities results in a compile-time error. For properties that cannot be checked at compile time, language specifiers should provide an assertion mechanism for checking properties at run-time. It should be possible to inhibit assertion checking if efficiency is a concern.	
JP	6.26.1	1st paragraph	Ed	The word “behavior” should be “behaviour” which is the spelling used in this Technical Report.		
CA-38	6.26.1		Te	Missing implications for standardization	Add: Implications for the application include inadequately tested code with errors, and code that missed review and provides arbitrary vectors for code execution	
JP	6.26.3	1st bullet	Ed	The word “behavior” (two occurrences) should be “behaviour” which is the spelling used in this Technical Report.		
JP	6.26.4		Ed	This subclause does not follow the standard template given in C.1.4. It should not start with “This is”.		
JP	6.26.7	[3]	Ed	“Contraint” would be a misspelling.	It should be changed to “Constraint”.	
CA-39	6.27.1		Te	Missing implications for standardization	Add after para 2: Tnis situation is difficult to exploit for an attack, but can cause premature program termination, corruption of data, or livelock (runaway program).	
CA-	6.27.1	Paras 3-5	Te	This is a discussion of how the vulnerability occurs	Move to section 6.27.3	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
40						
JP	6.27.5	2nd bullet	Ed	The word “elaboration” is used here. We suspect that this is a term used only in a limited number of languages, and is not a general technical term in programming language definitions.		
JP	6.27.5	8th bullet	Ed	The first sentence does not have a main verb.		
CA-41	6.28.1		Te	Missing implications for standardization	Add: Implications for the application include livelock, data corruption, and potentially arbitrary code execution.	
JP	6.28.6		Ed	The term “Language standards-writers” is not used in other places of this Technical Report.		
CA-42	6.29.1		Te	Missing implications for standardization	Add: Implications for the application are the same as for Buffer Overflow (XZB)	
JP	6.29.6		Ed	The term “Languages definitions” should be “Language definitions”.		
CA-43	6.30.1		Te	Missing implications for standardization	Add: Implications for applications include erroneous program execution, premature program termination or livelock.	
CA-44	6.30.5		Te	Missing a mechanism to avoid vulnerabilities associated with precedence	Add: Break up complex expressions and use temporary variables to make the order clearer.	
UK	6.30.7	para 1	ed	Add citation to experiments using professional developers which provide evidence of high error rates.	"Developer beliefs about binary operator precedence" by Derek M. Jones CVu Vol. 18, No. 4. (August 2006), pp. 14-21.	
CA-45	6.31.1		Te	Missing implications for standardization	Add: Implications for applications include erroneous calculations, premature program termination, livelock.	
JP	6.32.1		Te	We cannot understand what is meant by “or even be		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				exploited as a vulnerability under certain conditions".		
CA-46	6.32.1		Te	Missing implications for standardization	Add: Implications for applications include erroneous calculations, premature program termination, livelock.	
JP	6.33.3		Ed	The sentence "fun_b is dead code, as only fun_a can ever be executed" is wrong. "fun_b" is called, and "fun_a" is not called.		
JP	6.33.3		Ed	The sentence "Or is there a legitimate reason for its presence" does not have a question mark.		
JP	6.33.5	4th bullet	Ed	The word "recognised" is spelled in this way here, while it is spelled as "recognized" in 5.1.4, 7.4.4, and 7.6.3.		
JP	6.33.6		Ed	This subclause is empty.		
CA-47	6.34.1		Te	Missing implications for standardization	Add: Implications for applications include erroneous calculations, premature program termination, livelock.	
JP	6.34.3		Te	More detailed explanation should be given. Current wording says nothing about "mechanism of failure".		
CA-48	6.35.1		Te	Missing implications for standardization	Add: Implications for applications include erroneous calculations, premature program termination, livelock.	
CA-49	6.35.6		Te	Missing an implication for standardization	Add a bulet (preferrably first): Languages of languages should consider adding a mode that strictly enforces compound conditionl and looping constructs with explicit termination, such as "end if" or a closing bracket.	
IT	6.36.3	1	Ed	Poor phrasing.	A common assumption is that a loop control variable is a constant since such variables are not usually modified in the body of the associated loop.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
IT	6.36.4	Bullet 1	Ed	Redundant and inelegant parentheses.	Languages that permit a loop control variable to be modified in the body of its associated loop (some languages, e.g., Ada, treat such usage as an erroneous construct and require translators to diagnose it).	
JP	6.36.7		Ed	The titles of reports “Guidelines for the use of the C language in critical systems” etc. are given here, but no other places in this Technical Report.		
JP	6.37.1	2nd bullet	Ed	The word “or” in “the length or a structure” would be a misspelling.	Change it to “of”.	
CA-50	6.37.1		Te	Missing implications for standardization	Add to penultimate paragraph: Implications for applications include erroneous calculations, premature program termination, livelock.	
IT	6.37.1	Bullet 2	Ed	Poor phrasing.	Confusion as to the index range of an algorithm, such as: beginning an algorithm at 1 when the underlying structure is indexed from 0; beginning an algorithm at 0 when the underlying structure is indexed from 1 (or some other start point); or using the length of a structure as its bound instead of the sentinel values.	
IT	6.37.1	4	Ed	Spurious parenthesis.	The existence of this possible flaw can also be a serious security hole as it can permit someone to surreptitiously provide an unused location (such as 0 or the last element) that can be used for undocumented features or hidden channels.	
IT	6.37.3	Bullet 2	Ed	Syntax errors.	incomplete comparisons or calculation mistakes,	
IT	6.37.6	Bullet 1	Ed	Incorrect use of colon at end of sentence, instead of standard period.	Prevent the need for the developer to be concerned with explicit sentinel values.	
UK	6.38	complete subsection	te	Although entitled "Structured Programming" the body of the text deals with the use of various forms of jump statements. Use of jumps has benefits as well as costs. Studies (including ones done for Ada source) have found	Delete.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				that goto is widely used. There are certain kinds of jump that tend to lead to faults (e.g., those that cause the flow graph to be irreducible). The recommendation against the use of jumps is a hang over from the days when programming languages did not contain constructs that made it easy to group sequences of statements together into blocks. These days jump statements are less commonly used and only tend to be used when needed.		
CA-51	6.38.1		Te	Missing implications for standardization	Add to penultimate paragraph: Implications for applications include erroneous calculations, premature program termination, livelock.	
JP	6.38.4	1st bullet	Ed	The font of “goto” is different from that of “continue” and “break” in the following bullet.		
UK	6.38.7	para 1	ed	Cited article is not a particularly informative. A better citation would be Knuth's paper on the use of structured goto.		
JP	6.39.3	3rd paragraph	Te	The name of the parameter-passing mechanism “the values of the locals corresponding to formal parameters are copied to the corresponding actual arguments” is not “call by value”. It is “call by result”.		
JP	6.39.3	9th paragraph	Ed	The title “Order of Evaluation section” is different from the title of the subclause.		
IT	6.39.3	2-3	Ed	The break between paragraph 2 and paragraph 3 is at an illogical place.	In call by reference, the calling program passes the addresses of the arguments to the called subprogram. When the subprogram references the corresponding formal parameter, it is actually sharing data with the calling program. If the subprogram changes a formal parameter, then the corresponding actual argument is also changed. If the actual argument is an expression or a constant, then the address of a temporary location is passed to the subprogram; this may be an error in some languages. In call by copy, the called subprogram does not	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
					share data with the calling program. Instead, formal parameters act as local variables. Values are passed between the actual arguments and the formal parameters by copying. Some languages may control changes to formal parameters based on labels such as in, out, or inout. There are therefore three cases to consider: call by value for in parameters; call by result for out parameters and function return values; and call by value-result for inout parameters. For call by value, the calling program evaluates the actual arguments and copies the result to the corresponding formal parameters that are then treated as local variables by the subprogram. For call by value, the values of the locals corresponding to formal parameters are copied to the corresponding actual arguments. For call by value-result, the values are copied in from the actual arguments at the beginning of the subprogram's execution and back out to the actual arguments at its termination.	
IT	6.39.6	Bullet 1	Ed	Verbs erroneously at singular.	Programming language specifications could provide labels – such as in, out, and inout – that control the subprogram's access to its formal parameters, and enforce the access.	
CA-52	6.40.1		Te	Missing implications for standardization	Add a sentence: Using a dangling reference to the stack can result in corrupted data, an application exception, premature application termination, or in rare cases, arbitrary code execution.	
JP	6.40.6		Ed	The sentence "Language designers can avoid …" did not appear in "Implications for standardization" subclauses. Moreover, it does not follow the standard template given in C.1.6.		
JP	6.40.6	2nd bullet	Ed	Subclause 6.44.6 is referred to here. We understand that this Technical Report avoids the reference using such explicit subclause numbers, and prefers 3-letter codes.		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
IT	6.40.6	Bullet 2	Ed	Erroneous (forward) reference. As a general practice, hard-coded references should be avoided to prevent the risk of them inadvertently becoming obsolete.	Define implicit checks to implement the assurance of enclosed lifetime expressed in 6.40.5. Note that, in many cases, the check is statically decidable, for example, when the address of a local entity is taken as part of a return statement or expression.	
IT	6.41.3	1 (Period 2)	Ed	Verb erroneously at singular.	If the number and type of the actual arguments do not match the number and type of the formal parameters, then the push and the pop will not be commensurable and the stack will be corrupted.	
JP	6.41.4		Ed	The first sentence says “applicable to implementations or languages”. Other corresponding subclauses did not say “implementations”. It does not follow the template given in C.1.4.		
JP	6.41.5	4th bullet	Ed	The word “and” in “Intensively review and subprogram calls” should be deleted.		
JP	6.42.5	2nd bullet	Ed	There is no verb in the first sentence.		
IT	6.43.1	1 (Period 1)	Ed	Double dash instead of “em dash”.	Unpredicted error conditions–perhaps from hardware (such as an I/O device error), perhaps from software (such as heap exhaustion)–sometimes arise during the execution of code.	
IT	6.43.5	Bullet 7	Ed	Failed compliance to style convention (missing comma).	In applications with the highest requirements for reliability, defense-in-depth approaches are often appropriate, i.e., checking and handling errors thought to be impossible.	
IT	6.43.6	Bullet 1	Ed	Failed compliance to style convention (missing comma).	A standardized set of mechanisms for detecting and treating error conditions should be developed so that all languages to the extent possible could use them. This does not mean that all languages should use the same mechanisms as there should be a variety (e.g., label parameters, auxiliary status variables), but each of the mechanisms should be standardized.	
CA-	6.44.1	Para 2	Te	This paragraph is more the mechanisms of failure, rather	Move to 6.44.3	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
53				than the application vulnerability		
CA-54	6.44.1		Te	The description of the application vulnerability is incomplete.	Add a new para 2: When the software does not terminate in the planned mechanism, safety or security is compromised, as failing in an unspecified way interferes with the alternative recovery features. In safety-related systems the results can be catastrophic: for other systems the result can mean failure of the complete system.	
CA-55	6.45.1		Te	Missing implications for standardization	Add: Implications for the application include corruption of data, application termination, denial of service (application runaway), covert channels, and arbitrary code execution.	
JP	6.45.3	5th paragraph	Ed	The reference “CSJ, 6.39” is mysterious. We first thought this is a name of a document. 3-letter code CSJ should be enclosed by square brackets. Explicit clause numbers should not be given. There is an unintentional space between “6.39” and “Passing”.		
IT	6.45.5	Bullet 3	Ed	Failed compliance to font convention (bold italic instead of courier).	`Unchecked_Conversion`	
IT	6.45.6	Bullet 1	Ed	Failed compliance to font convention (italic instead of courier).	`Unchecked_Conversion`	
JP	6.45.7	[2]	Ed	A comma should be inserted between “10426-4” and “John Wiley & Sons”.		
UK	6.46.1	para 1	ed	At first sight, the third sentence is a bit muddled. The key problem is that a program can stop because it runs out of storage. This can happen with a long running program just because of the elapse of time. Moreover, a program that would not normally run out of storage because it only runs for a short time can be attacked by repeatedly causing it to execute a sequence that triggers the leak thus causing a denial of service. And why does this refer to safety-critical systems specifically? It is a problem for any critical system.	Rewrite sentence 2 thus "... of available memory and eventually lead to the shutdown of the program." Rewrite sentence 3 thus "A memory leak can be exploited by an attacker to cause a denial-of-service by causing the program to execute repeatedly a sequence that triggers the leak. Add sentence 4. "Moreover, a memory leak can cause any long-running critical program to	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				Maybe a good solution is to mention shutdown in the second sentence.	shutdown prematurely."	
UK	6.46.2			It is regrettable that a respectable document should use deplorable junk slang such as aka.	Replace "aka" by "alias".	
UK	6.46.3	para 1	te	Second sentence: it would be better to start "Moreover" and it seems unnecessary to include "partially". Fragmentation often occurs even if all memory is returned. The last part of the sentence would read more smoothly as "... in the inability to obtain storage of the required size."	Change second sentence to "Moreover, memory claimed and returned can cause the heap to fragment, which will result eventually in the inability to obtain storage of the required size."	
UK	6.46.3	para 2	ed	The whole point is that the attacker can make it leak more quickly. So the comparative adverb should be used. Either "more quickly" or "quicklier"; however the latter form is rarely used these days probably because nobody is taught English properly anymore.	Change to "... application to leak more quickly...".	
UK	6.46.4	bullet	te	The problem also arises even if there is a garbage collector since it might still result in fragmentation.	Extend sentence "...under program control or through the use of a garbage collector."	
	6.46.5		Ed	Text lines in bullets are not well aligned. Indentations are not uniform.		
UK	6.46.5	bullet 1	ed	Why upper case G in Garbage?	Use lower case g.	
UK	6.46.5	bullet 1	te	This bullet is unclear. Why distinguish garbage collectors that are intrinsic to those that are add-ons? Is it trying to say that add-ons are not 100% effective? Should likelihood be probability? They are technical terms and should be used correctly.	Clarify. No wording suggested since it is not clear what it is attempting to say.	
UK	6.46.7			No bibliography given	Perhaps refer to MISRA C as an example of limiting usage of dynamic memory.	
UK	6.47		te	Should not this section be headed "Argument Passing to Library Subprograms"? In most languages function is used in the sense of a subroutine that returns a result as in mathematics. But clearly these sections should apply to all subroutine calls. And throughout this section it refers to "calling function" but the calling code might not itself be a function. Better to say	Change section heading to "... to Library Subprograms" or to "... to Library Subroutines". Throughout this section change "function" in the sense of the called routine to "subprogram" or "subroutine". And change "calling function" to "calling code".	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				"calling code"		
CA-56	6.47.1		Te	Missing implications for standardization	Add: Implications for the application include corruption of data, application termination, denial of service (application runaway), covert channels, and arbitrary code execution.	
UK	6.47.1	para 1	te	How do libraries supply objects and how does one pass parameters to an object? By a subprogram/method! This should also discuss the question of the number of parameters being correct as well.	Discuss number of parameters as well.	
UK	6.47.1	para 1	ed	Second sentence is confusing.	Change to "When parameter validation is required, it should be demonstrated that the library function performs such validation or the application should undertake it."	
UK	6.47.3	para 2?	te	The question of the number of parameters being the same needs to be addressed.	Add "If the number of parameters supplied is not the same as the number expected and the implementation does not check this then this will typically result in the integrity of the execution stack being destroyed. The final result is likely to be quite unpredictable. For example an arbitrary item of data could be interpreted as a return address and the program could jump into outer space."	
UK	6.47.5	bullets	ed	The first two bullets are statements, the last three are imperative commands. They should be rewritten to be uniform.	Unify grammar.	
CA-57	6.48.1		Te	Missing implications for standardization	Add: Implications for the application include corruption of data, application termination, denial of service (application runaway), covert channels, and arbitrary code execution.	
JP	6.48.4		Ed	Is “Language” an entity that “run”s?		
UK	6.48.4	bullet 2	te	It seems incorrect to say "i.e. the stack" because that implies that all data space is in the stack which is not true.	Either delete "i.e. the stack" or replace by "e.g. the stack"	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				One could change it to e.g. or simply remove it.		
UK	6.48.5	bullet 3	te	it seems surprising that self-modifying code should ever be necessary in critical code. And "heavily" is a curious word in this context.	Change to "Self- modifying code should never be used in the most critical applications. In those extremely rare instances ... should be very limited and thoroughly documented."	
UK	6.48.6	bullet	ed	The term digital signature is used here as well. That is surely just one means to the desired end.	Change to "...check that the library used in the application environment is identical to that used in the compile/test environment"	
UK	7		te	Remove Clause 7	It is out of scope of the document	
UK	7		ge	Spoofing of data is not mentioned, e.g. Somali pirate ship sends spoof AIS data claiming to be a trawler.	Mention spoofed data.	
CA-58	Section 7		Ge	Category does not fit the work statement for vulnerabilities of programming languages	Move section 7 to another document. We would prefer 24772-2	
UK	7.2		ed	This use of the word sandbox is outside our domain of understanding and that of the Shorter Oxford Dictionary which gives the following possible meanings. 1) A box with a perforated top for sprinkling sand on to wet ink; 2) A box or receptacle holding sand, used for various purposes, esp a) a box used on a locomotive for sprinkling sand on slippery rails; b) *(golf)* a container for sand used in teeing; c) a small low-sided children's sandpit; d) a box kept indoors and filled with sand or other material for a cat to urinate or defecate in. We assume it is none of those meanings. Note that there is a definition in Wikipedia, perhaps refer to that or summarise it.	Please clarify or refer to a definition of sandbox.	
JP	7.2.3		Ed	Bullets are not well aligned. The first and fifth bullets are more indented than other bullets.		
JP	7.2.3	6th bullet	Ed	The slash before “etc.” should be deleted.		
UK	7.3.4	bullet 1	ed	Here is another use of digital signature.	Delete "with a digital signature"	
UK	7.3.4	bullet 2	te	How does validating a library determine whether it is required?	Split into “All native libraries should be validated” and “Determine whether the application requires the use of the native library” (presumably omitting it if not required).	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
UK	7.4.1	para 1	ed	It would be nice to have a definition of an Easter Egg in this context.	Define "Easter Egg" perhaps by writing something like "...may be no more than an amusing additional and irrelevant piece of functionality (such an addition is often called an Easter Egg), like the flight simulator..."	
JP	7.4.4		Ed	The apostrophe in “End user’s” seems to be inappropriate here.		
UK	7.4.4	para 1	ed	Spurious apostrophe in "End user's".	Replace by "End users".	
UK	7.4.4	bullet 1	te	Observe that this means that all compilers and similar software tools used in developing critical applications should be from such a developer. Moreover, it is not enough for the developer to have such a process but it must be used as well!	Rewrite thus "Programs and development tools such as compilers that are to be used in critical applications should come from a developer who uses a recognized and audited development process for the development of those programs and tools. For example ...".	
UK	7.5.4	para 2	te	The OS might have an option to clear the swap file on shutdown (Windows does).	Add "If the OS allows, clear the swap file on shutdown."	
UK	7.6.3	para 2	ed	Why Perl? Unless choosing Perl specifically for this can be justified then it should be deleted.	Delete “Perl”.	
UK	7.6.3	para 3	ed	The example of a switch is a bit low-level. Another example might be that if you accept data from a public source, such as an RSS feed, then you have to be prepared to throttle it in case someone deliberately sends too much data.	Perhaps illustrate with another example as described.	
JP	7.7.3	5th bullet	Ed	The reference “path manipulation description” is not sufficient. Does it refer to “7.13 Path Traversal”? 3-letter code should be given.		
UK	7.7.4	bullet 8	ed	"refractor" should be "refactor".	Change "refractor" to “refactor”.	
UK	7.8	para 7 bullet 3	ed	"browsers" was probably meant to be singular.	Change “browsers” to “browser”.	
UK	7.8.3	para 2	ed	Is “inner-office” correct?	Possibly “inter-office” was intended.	
UK	7.9.3	para 1	ed	It seems likely that “filenames or folder names” was intended rather than “files or folders”.	Change “files or folders” to “filenames or folder names”.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
UK	7.11		te	OS fingerprinting often makes use of knowing which OS's IP stacks fail to follow the RFCs correctly.	In the avoidance section add selection of an OS that confirms to international standards.	
JP	7.11.2		Ed	The word "Behavorial" in 207 seems to be a misspelling.		
JP	7.11.4	1st bullet	Ed	This bullet does not make sense. It simply repeats the sentence before the bullets.		
JP	7.11.4	2nd bullet	Ed	The word "your" is not appropriate here. Other parts of this Technical Report do not say "you" or "your".		
JP	7.12.3	2nd paragraph	Ed	The term "can not" should be changed to "cannot".		
JP	7.13.3	3rd paragraph	Ed	An empty parentheses pair appears after "a backslash absolute path".		
JP	7.13.3	4th paragraph	Ed	The indentation of the first line is not correct.		
JP	7.13.3	5th paragraph	Ed	The indentation of the first line is not correct.		
JP	7.13.4	3rd bullet	Ed	We cannot understand the word "fir" in "may be required fir form".		
JP	7.13.4	10th bullet	Ed	What is the meaning of square brackets?		
UK	7.13.4	bullet 3	ed	Typo – "fir" should be "for".	Change "fir" to "for".	
JP	7.17.3	2nd paragraph	Ed	The word "apps" seems to be an inappropriate abbreviation.		
JP	Cover page of N4420		Ed	Document Title is not correct. It says "Guidelines to avoiding vulnerabilities in language selection and use". The correct title is "Guidelines to avoiding vulnerabilities in programming languages through language selection and use".		
JP	Cover page of PDTR 24772		Ed	French title is not given. The current one seems to be a copy of the title template.		
JP	B.1	2nd paragraph	Ed	The word "favor" is spelled in this way here. It is spelled as "favour" in 6.38.6.		
JP	C.1		Ed	The phrase "No [additional] text should appear here"		

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				appears twice in this subclause.		
JP	D.1		Ed	The reference “Clause 6” is not correct. Clause 6 speaks about programming language vulnerabilities not application vulnerabilities.	Change it to “Clause 7”.	
JP	D.1.4	2nd paragraph	Ed	The term “programming language vulnerability” is not appropriate here. This Annex speaks about “application vulnerability”.		
JP	F.3.0		Ed	A period should be inserted between “repeat the general material” and “The text should be”.		
JP	F.3.5		Ed	There is an unintentional vertical gap in the first paragraph.		
JP	Annex A		Ge	The status of this Annex in this Technical Report is not explained anywhere. For example, we cannot understand when and how this Annex is applied. Title itself is not sufficient explanation.		
UK	Annex A		ge	We do not see how this annex relates to the rest of the document. It is referred to in Annex B but not elsewhere. The same applies to Annex B.	Clarify use of and need for Annexes A and B.	
NL-3	Annex A - F		Ed	It seems that (the purpose of) Annexes A .. F are not introduced in the main text of this PDTR.		
NL-1	Annex B		Ed	The contents of Annex B seems so central to this Report that this material should be covered in the main text of this PDTR.		
NL-2	Annex B		Ed	In Annex B a list of four 'major sources of evidence' is produced. The latter three of these sources are based upon measurements.	It would be helpful to have literature references to these measurements.	
UK	Annex E		ge	This annex describes 66 vulnerabilities and indeed these are covered in Sections 6 and 7 but not quite in the same order. The structured list in the annex is laid out in a helpful way and one wonders why sections 6 and 7 did not follow the same structure.	Align order and structure of this annex with sections 6 and 7. Or delete this annex.	
UK	Annex E		te	We find it very surprising that there is no discussion whatsoever about vulnerabilities in multitasking. Many, perhaps almost all, critical systems in areas such as avionics involve parallel processing at some level. There are many ways in which these programs are vulnerable to errors such as deadlock and interference due to multiple	Add vulnerabilities on parallelism and multitasking to this annex and corresponding sections in the body of the document.	

1	2	(3)	4	5	(6)	(7)
MB[1]	**Clause No./ Subclause No./ Annex** (e.g. 3.1)	**Paragraph/ Figure/ Table/ Note** (e.g. Table 1)	**Type of comment**[2]	Comment (justification for change) by the MB	Proposed change by the MB	**Secretariat observations** on each comment submitted
				access to data. If this document is meant to be comprehensive then these areas really must be addressed. We suggest a new section E.12 could be added and then the existing E.12 and E.13 would become E.13 and E.14. And then of course Sections 6 and 7 need expanding to match. The new E.12 could be subdivided thus E.12.1 Concurrency E.12.2 Communication E.12.3 Scheduling See below for suggestions for detailed vulnerabilities.		
UK	Annexes C & D		ge	Section 6 on language vulnerabilities is written in the style described by the template in Annex C and section 7 on application vulnerabilities is written in the style of Annex D. But it is confusing that the sections in 6 have the first subsection entitled application vulnerability rather than language vulnerability.	Maybe change headings of sections in 6 from "Description of application vulnerability" to simply "Description of language vulnerability" or maybe to something like "Language vulnerability and implication for application".	