			Balloted	document:	PDTR 24772	
				Vote:	Approve, Disapprove, Abstain	
NB	**No.**	**Category**	**Clause, Sub-clause**	**Paragraph, Figure, Table**	**Comment and rationale**	**Proposed new text**
JWM	1	TL	6.30.3	EOJ	This sentence doesn't parse: "Testing of the software may not reveal that statements thought to be included in an if-then, if-then-else, or loops that are not in reality a part of the if statement."	Testing of the software may not reveal that statements that appear to be included in a construct actually lay outside of it, or vice versa, because of an incorrect placement of a terminator.
JWM	2	E	D.30.1	EOJ-C	Wording of this sentence can be improved: "C is a block-structured language, while languages such as Ada and Pascal are comb-structured languages."	C is a block-structured language (as contrasted to comb-structured languages like Ada and Pascal).
JWM	3	TL	F.30	EOJ-Ruby	The claim is made that "This vulnerability is not applicable to Ruby since control constructs require an explicit termination symbol."	Isn't this situation, the same as C? Hence, shouldn't the description be similar? Alternatively, maybe it's similar to Ada and deserves a similar description (C.30).
JWM	4	TH	6.7 and annexes	FLC	This is a general comment on FLC. The general description and the annexes all neglect the problem of a numeric value falling outside the range that is semantically treated by the program. Strongly typed languages have mechanisms that can be used to raise exceptions but weakly typed languages require explicit checking.	I'm not sure if FLC should be extended to deal with this situation or if a new vulnerability should be written. The argument for extending FLC is that, for many languages, the remedy of both FLC and the extended problem are the same--write explicit range checks.
JWM	5	TL	C.7	FLC-Ada	The sheer length of the explanation leads the casual reader to believe that Ada has a big problem here.	Insert a new first paragraph: "Ada has mechanisms to mitigate most forms of numeric conversion errors as explained by the following text."
JWM	6	TL	D.37	GDL-C	Too many words. They add little, if anything, to the general guidance.	Replace D.37.1 with: "C permits recursion, hence is subject to the problems described in 6.37." Replace D.37.2 with: "Apply the guidance in 6.37.5."
JWM	7	TL	E.37.2	GDL-Python	Neglects the general guidance.	Replace the first bullet with: "Apply the guidance in 6.37.5."
JWM	8	TL	F.37	GDL-Ruby	Says nothing that is not in the general guidance.	Replace F.37.1 with: "Ruby permits recursion, hence is subject to the problems described in 6.37." Replace D.37.2 with: "Apply the guidance in 6.37.5."

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
JWM	9	TH	6.24.5	LAV	The description neglects to describe the ill effects of junk initialization.	Add to second bullet: "However, the initial value must be a sensible value for the logic of the program. So-called "junk initialization", for example, setting every variable to zero, simply defeats the use of static analysis without providing any benefit."
JWM	10	E	6.36.3	OTR	4th line: "Commensurable" does not mean "consistent".	Change "commensurable" to "consistent".
JWM	11	E	D.36.1	OTR-C	At top of page 215, there is an appropriate line feed after the second line of text.	Remove line feed.
JWM	12	E	D.36.2	OTR-C	The second sentence of the first bullet is unneeded and serves only to teach the language syntax.	Remove it.
JWM	13	TL	E.36.2	OTR-Python	Incomplete guidance	Add this bullet (paraphrased from the language-independent description): "Intensively review subprogram calls to/from non-Python modules."
JWM	14	TH	F.36.1	OTR-Ruby	This description neglects the case of inter-language calls.	Add a paragraph (paraphrased from the Python annex) to F.36.1: "Signature mismatches in calls to/from non-Ruby modules could cause a call stack problem." Replace the two bullets of F.36.2 with the following :" - Intensively review subprogram calls to/from non-Ruby modules. // - Analyze any error messages from the Ruby interpreter indicating an incorrect number of parameters."
JWM	15	E	6.52	SKL	Third paragraph, first line.	Change "functionally" to "functionality".
JWM	16	TH	D.52	SKL-C	For inherently unsafe operations, the claim is made, "Does not apply to C". This is incorrect. In some sense, all of C is unsafe.	Change to: "C is intended as a language for implementing systems programs where unsafe operations are inherent and common."
JWM	17	TL	6.31	TEX	Consider the loop control statement, "For I = J to K" This description treats J and K as loop control variables, but not I. (For example, 6.31.3 says, "a common assumption is that a loop control variable is a constant". That obviously does not apply to I.) Is that intended?	Consider generalizing the vulnerability to deal with the case where code inside the loop makes changes (or attempts to change) I.
JWM	18	TL	D.31	TEX-C	The only case treated here is when I is changed inside the loop. That does not agree with the general description, which doesn't treat I as a loop control variable.	Make TEX and TEX-C consistent.

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
JWM	19	TL	F.31.1	TEX-Ruby	"This is usually not performed, as the exact results are not clear." The passive construction causes confusion of whether the actions of the processor or the human are described.	Change to "This practice should be avoided as the exact results are not always clear."
JWM	20	TL	6.45.5	TRJ	Do bullet one and bullet two say the same thing?	Delete bullet one.
JWM	21	TL	6.45.6	TRJ	As worded, bullet one seems to be unrelated to the problem.	Rewrite bullet one as follows: "Languages that define a support library should ensure that unvalidated parameters cannot lead to undefined behaviour."
JWM	22	TL	C.45.2	TRJ-Ada	The third bullet is puzzling. Does "specify" mean "document" or "code"?	Delete bullet three.
JWM	23	TH	E.45	TRJ-Python	The python annex equates TRJ (checking parameter values) with OTR (checking parameter types). This is an incorrect equation and the guidance given is appropriate.	Change E.45 to be similar to C.45 (the C annex).
JWM	24	TL	F.45	TRJ-Ruby	One bullet of the guidance states "Use only libraries known to have consistent and and validated interface requirements." Of what relevance are the "requirements"?	Change to "Use only libraries known to validate parameters."
JWM	25	TL	C.41.1	XYL-Ada	This final sentence mentions the issues of garbage collection. However, those issues are not described in the general description.	Make XYL and XYL-Ada consistent.
JWM	26	TL	D.42	XYM-C	D.42 reads: "Does not apply to C." This is correct but leaves the reader wondering why.	Replace with text adapted from E.42. "This vulnerability is not applicable to C because C does not implement these mechanisms." Make a similar change to D.43 and D.44.
JWM	27	E	6.28.1	XYQ	1st line says, in part: "(the distinction is addressed in [XYQ])". But this *is* [XYQ].	Delete the phrase.
JWM	28	E	6.28.1	XYQ	Final sentence contains multiple errors.	Replace final sentence with "Dead and Deactivated Code is considered separately from Unused Variable, which is covered in [YZS]."
JWM	29	E	6.28.3	XYQ	The paragraph beginning "The presence of dead code" seems to contain cut-and-paste errors.	Rewrite as intended.
JWM	30	TL	6.28.5	XYQ	The second and fourth bullets make no sense to me.	Remove them.

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
JWM	31	TL	6.28.5	XYQ	Bullets are overlapping, redundant, and lack parallelism	Replace with: "- The developer should identify any dead code in the module and analyze its purpose. Code lacking purpose should be removed. - The developer should apply branch coverage tools and ensure that all branches are neither dead nor deactivated."
JWM	32	TL	D.28	XYQ-C	The description doesn't add much to the LI description.	Rewrite (similarly to the Ada description) as follows: D.28.1 "C allows the usual sources of dead code (described in 6.28) that are common to most conventional progamming languages. // [Keep the paragraph beginning 'C Uses some operators'.]" D.28.2 "- Apply the guidance provided in 6.28.5. - Use the "//" comment syntax instead of the "/*...*/" comment syntax to avoid inadvertent syntactic inclusion of code segments within comments."
JWM	33	TL	E.28	XYQ-Python	The description doesn't add much to the LI description.	Replace first paragraph of E.28.1 with the following: "Python allows the usual sources of dead code (described in 6.28) that are common to most conventional progamming languages." Insert a new first bullet in E.28.2 "Apply the guidance provided in 6.28.5."
JWM	34	TL	F.28	XYQ-Ruby	The description doesn't add much to the LI description.	Replace first paragraph of F.28.1 with the following: "Python allows the usual sources of dead code (described in 6.28) that are common to most conventional progamming languages." Replace the bullet in F.28.2 with "Apply the guidance provided in 6.28.5."
JWM	35	E	Contents		The editor did a great job of generating PDF bookmarks. It really helps a lot in navigating the document.	The editor should consider adding a bookmark for the Table of Contents.
CA-1	36	GE			While we appreciate the work that SC 22/WG 23 has gone to to develop language-specific Annexes for the document, we believe that the document is incomplete without annexes for the major languages, such as C++, COBOL, Fortran, Java and PHP	Add Annexes for the suggested languages. Retard the publication schedule of the TR, or publish a 3rd edition as soon as these annexes become available.
CA-2	37	GE			Annex numbering	

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
CA-3	38	TL	6.39.3		This section relating to termination should address only sequential termination. Termination of multiple threads or of concurrent programs is addressed in section 8.6 [CGY] and 8.4 [CGT]	Add to section 6.39.1 (at the end): For termination issues associated with multiple threads, multiple processors or interrupts see 8.4 Concurrency - Directed Termination [CGT] and 8.6 Concurrency - Premature Termination [CGT]. Situations that cause an application to terminate unexpectedly or that cause an application to not terminate because of other vulnerabilities are covered in those vulnerabilities.
CA-4	39	TL	G.3	Paragraph 2	Do not compare Spark to Ada.	It is acceptable to be informative , but not to be critical of another language. Change the sentence “SPARK’s type system is a simplification of that of Ada” to Spark's type system derives from Ada's type system but is simplified to remove dynamic properties and undefined and implementation dependent properties.
CA-5	40	TL	G.12		This section refers to Ada's sliding and subrange capabilities as vulnerabilities. These are not identified in C.11.	Either add a discussion of vulnerabilities associated with “slicing and sliding” in C.11 or remove the discussion from G.11.
CA-6	41	TL	G.27		Be explicit when referencing Ada or comparing to Ada	Indicate which “likely incorrect” expressions are not possible, eg. conditional entry calls and timed entry calls.
CA-7	42	TL	G.34		The writeup says that Spark “mitigates” but is not clear why it does not “prevent”.	Document what steps a user must take in addition to ensure that the mitigation is successful.
CA-8	43	E	G.36	Last	Reference to “Annex Ada” is wrong.	Replace with “C.36”
CA-9	44	TL	C.39, D.39		The writeup for this section must change if the CA comment for 6.39 is adopted.	
CA-10	45	TL	D.39		The writeup for this section must change if the CA comment for 6.39 is adopted.	
CA-11	46	TL	E.39		The writeup for this section must change if the CA comment for 6.39 is adopted.	
CA-12	47	TL	F.39		The writeup for this section must change if the CA comment for 6.39 is adopted.	
CA-13	48	TL	G.39		The writeup for this section must change if the CA comment for 6.39 is adopted.	

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
CA-14	49	TL	G.43		The writeup says that Spark “mitigates” but is not clear why it does not “prevent”.	Document what steps a user must take in addition to ensure that the mitigation is successful (such as is done for G.56).
CA-15	50	TL	G.45	Last	Remove comparative references to Ada.	Document the “expressive power” and show what it provides.
CA-16	51	TL	G.45		The writeup says that Spark “mitigates” but is not clear why it does not “prevent”. In this case, it is likely that a correct formal statement of the preconditions and postconditions will permit formal verification of the subprogram body, but to weak conditions will show a “correct” proof that is meaningless (I.e TRUE => TRUE)	Document what steps a user must take in addition to ensure that the mitigation is successful (such as is done for G.56).
CA-17	52	TL	G.53		The writeup says that Spark “mitigates” but is not clear why it does not “prevent”.	Document what steps a user must take in addition to ensure that the mitigation is successful (such as is done for G.56).
CA-18	53	TL	G.1		The section does not address partial proof of correctness vs total proof of correctness.	Add a paragraph to explain that the Spark tools generate partial proof of correctness, and it is also incumbent upon the verification team to show that constructs complete or terminate to make the partial proof of correctness complete.
CA-19	54	TL	G.1-G.58		The Annex uses a different layout than all other language specific annexes. In particular, there is no G.x.2 Guidance to Language User sections anywhere in the section. In places where the language “prevents” the vulnerability or where the vulnerability and mitigations are the same as Ada, the section sub-subsection is not needed, but in the other places, it is needed.	Explicitly identify and label in G.x.2 the steps that users need to take to achieve the mitigations claimed.

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
CA-20	55	TL	D.3.2		The section has ignored the use of static analysis tools, some of which are quite good, to help users identify problematic breakages of the type system in code. This is true for many other sections, such as D.15.2(6.15.).	Since the main section already has specific recommendations on tool usage, it is sufficient to say, as a lead in to D.3.2, "in addition to the mitigations identified in section 6.3.2, ...". Add the general recommendation in other sections as appropriate. In fact, we would like to see the general statement made almost everywhere, and explicit statements made where the general recommendations do not apply. If the general statement is not accepted, then put explicit recommendations to use analysis tools wherever they make sense.
CA-21	56	TL	D.6.2	Final Bullet	Add a recommendation to never iterate over enums with gaps or that repeat.	
CA-22	57	TL	6.18		None of the languages that currently have an annex admit to having a sign extension error problem. This means that either the vulnerability does not exist as written, or thee Annex authors do not understand the problem.	Remove the vulnerability [XZI] or modify it to be meaningful.
CA-23	58	TL	D.23		(Namespace issues) – statement that this does not apply to C should be explained, perhaps with an additional sentence.	The fact that C has a separate space for macro names should be explained somewhere, or explained why it is not a "namespace issue".
CA-24	59	TL	D.25		We are troubled that this section provides so little guidance to users. The number of operator precedences is well defined, but there are so many, and they not match the normal mathematics that humans learn in school, so humans had difficulty remembering the order, and correctly applying the logic.	Be more up front with the acknowledgement of the problem. Make recommendations such as the following: Only use 2 or 3 operators together in a single statement, i.e. break up complex expressions to simplify the logic, and Add to the existing bullet in D.25.2 to include mixed arithmetic/logical operators, mixed arithmentic/shift operators.
CA-25	60	TL	D.27.2		(likely incorrect). The adoption of a coding style that forbids the use of the assignment operator except as the singular ultimate result of the expression would permit analysis to identify all mistaken uses of "=" for "==".	Add a bullet that says: - consider the adoption of a coditng standard that forbids the use of the assignment statement within an expression, except as the ultimate result of that expression.

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
CA-26	61	E	D.45.2		The ".2" is missing in D.45.2	
CA-27	62	TL	D.45.2		The discussion of efficiency in this bullet is misguided and should be removed. This document is only concerned about avoiding vulnerabilities, and in this case, to avoid vulnerabilities due to input into a library routine one must check for variable ranges, variable format (if a struct) and number of variables, or must be able to show that all cases are handled.	
CA-28	63	TL	E.3.1		We dispute the statement that Python is strongly typed. By the common definitions of strong typing, such as strong guarantees about the runtime behaviour of a program, fixed and invariable typing of data objects, or the absence of unchecked run-time type errors, Python fails these tests and cannot be considered to be strongly typed.	Remove the statemant.
CA-29	64	E	E.8		Be consistent in language stating that the language does have a vulnerability.	Begin the clause with “This vulnerability is not applicable to Python since...”
JP-1	65	E	8.5.2	CWE:	There are two CWE 821’s. 821. Missing Synchronization and 821. Incorrect Synchronization. The original CWE Version 2.1 assigns 820 to Missing synchronization. Therefore, the first 821 should be changed to 820.	Correct as follows: 820. Missing Synchronization and 821. Incorrect Synchronization
JP-2	66	E	C.20.1	1st Paragraph	The line break in the last sentence of the first paragraph, between "thread" and "communication" should be removed.	
JP-3	67	E	C.33.1	1st Paragraph	We cannot understand what the term "parent report" means.	
JP-4	68	E	C.38.1	1st Paragraph	The section reference "6.OYB" is not consistent with other similar references. "6.38" is the usual style.	
JP-5	69	GE	F.1	1st Paragraph	The Annex F is currently written based on the old WG draft of the Ruby specification. The Annex F should be updated based on the latest specification (FDIS 30170).	

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
JP-6	70	TL	F.54.1	2nd Paragraph, 1st example	The first example can be misleading - as if the standard does not specify the behaviour for a use of break-expression even in a do-clause. Only the case the standard does not specify is the behaviour for /expression/ in a for-expression, which ordinarily represents a collection object, is terminated by those jump expressions. (see 11.5.2.3.4 of FDIS 30170). We recommend that you remove this esample or replace it with a better example.	
JP-7	71	GE	Annex F	Annex F	The technical report has only the standard Ruby specification as a reference documents while it mentions the behaviours and the programming entities which are not specified in the standard Ruby specification. All the referenced materials besides the standard Ruby specification should be listed.	
DN	72	TL	Introductio n	1st Paragraph	Include simulation in motives	Add at the end of the paragraph: "Where a program is a simulation, correct results may not be known. Thus, erroneous results may be difficult to detect."
DN	73	TL	Introductio n	1st Paragraph	Mention concurrency and/or parallelism	Add a bullet to the bullet list. - Concurrency and parallel execution.
DN	74	TL	4.1	1st Paragraph	Mention scientific and engineering computation	At the end of the first sentence, change '.' to ", or the consequences of misleading results."
DN	75	TL	4.2	5th bullet	Simulation results lead to actions	At the end of the sentence, change '.' to ", or the consequences of misleading results."
DN	76	TL	4.3	bullet list at mid page	Not all programmes are software engineers	Add a bullet at the end of the list. "Scientists, engineers, economists, statisticians, or others who write computer programs as tools of their chosen craft can read this document to become more familiar with the issues that may affect their work."

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
DN	77	TL	6.3.3	3rd Paragraph	Correctly characterize real -> integer	Change "... the inverse conversion risks the loss of any fractional value ..." to "... the inverse conversion imposes the loss of any fractional value ..." {at best, the fraction is all zeros} {if the float is so large that there is no mathematical fraction, the low order portion of the integer is likely gibberish}
DN	78	TL	6.5.1	1st Paragraph	2nd and 3rd sentences contradict each other	Change "The bit representation for a floating point number can vary from compiler to compiler and on different platforms." to "The bit representation for a floating point number actually used in arithmetic operations can vary when different instruction sequences are used to implement the same operations." {for example, MMX vector versus x87 stack}
DN	79	TL	6.5.1	1st Paragraph	Correctly state the issue	Change "... using a binary representation would require ..." to "... using a binary representation may well require ..." {some floats can be represented exactly in few bits}
DN	80	TL	6.5.1	2nd Paragraph	Correctly state the issue	Change "Algorithms that use ..." to "Many algorithms that use ..." at the end of the paragraph add "Those without training or experience in numerical analysis may not be aware of which algorithms, or, for a particular algorithm, of which domain values, should be the focus of attention."
DN	81	TL	6.5.3	2nd Paragraph	Correctly state the issue	Change "..., particularly relatively small values, ..." to "..., whenever the ratio of two addends or the ratio of an addend to the sum is very large or very small, ..."
DN	82	TL	6.5.3	Page 32 1st paragraph	Correctly state the issue	Change "... due to propagation or conversion errors." to "... due to rounding or truncation errors, which may propagate far from the operation of origin. Even comparisons of constants may fail when a different rounding mode was employed by the compiler and by the application."
DN	83	TL	6.5.3	Page 32 2nd paragraph	The sign bit is not part of the mantissa	Change "... (including the sign bit) ..." to "... (including a hidden bit) ..."

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
DN	84	TL	6.9.5	Page 40, paragraph after the bullet list	Include more langauges	Add two sentences to the end of the paragraph. "Some languages support arbitrary bounds of arrays, so a priori categoric assertions of bounds values cannot be made. Some languages support zero-sized arrays, so any reference to a location within such an array is invalid."
DN	85	TL	6.15.3	Page 47, 3rd paragraph 1st bullet	What consequence causes the problem	Change "... circumstances" to "circumstance unexpectedly causes an object to become undefined"
DN	86	TL	6.21.3	Page 55, 2nd paragraph	Treat a common case	After the paragraph add "If the unused variable is present due to anticipated development, it may be commented out now to reduce unnecessary compiler warnings."
DN	87	TL	6.27.1	Page 65, 1st paragraph	Clarify wrong	Change "... not wrong, but is unlikely to be right" to "... not contrary to the language standard, but is unlikely to be intended."
DN	88	TL	6.28.3	Page 68, 3rd paragraph	Clarify sentence	Replace the first sentence with "The presence of dead code is not in itself an error but its presence may be an indication that the programmer believed it to be necessary. This possibility may lead a code reviewer to question whether it should be present, or executed, or removed." Remove "also" from the next sentence. {or I don't understand this}
DN	89	Te	6.29.1	Page 69, 1st paragraph	Which switch	Change "... such as a switch statement," to "... such as a C-language switch statement," {It might be desirable to repeat the identification of the switch statement at several more places in 6.29.3 and 6.29.4 (or not)}
DN	90	TL	6.36.3	Page 80, 1st paragraph	May be language-dependent	Change "..., then the push and the pop will not be commensurable and the stack will be corrupted." to "..., then, depending upon the calling mechanism used by the language translator, the push and the pop may not be commensurable and, if so, the stack will be corrupted."

NB	No.	Category	Clause, Sub-clause	Paragraph, Figure, Table	Comment and rationale	Proposed new text
DN	91	TL	6.37.3	Page 82, 2nd paragraph	Clarify what is not true	Change "... not true in the general case." to "... not true when considering computer operations generally, especially when processing error conditions." {further on in the same paragraph- remove jargon} change '... attempting to "clean up" by closing ...' to '... attempting to recover resources by closing ...'