The Chartered Institute for IT

Enabling the information society

Update on the Language Vulnerabilities Report and the Fortran Annex

David Muxworthy d.muxworthy @ bcs.org.uk 27 September 2012

TR 24772 – Intended Audience

4.2 Intended Audience

The intended audience for this Technical Report are those who are concerned with <u>assuring the predictable execution</u> of the software of their system; that is, those who ... need to avoid language constructs that could cause the software to execute in a manner other than intended.

Developers of applications … could use this Technical Report to ensure that their development practices address the issues presented by the chosen programming languages, for example by subsetting or providing coding guidelines.

Progress on the main body of the TR

2005 SC22 approved the project by 10 votes to 2 (Netherlands and UK)

:

:

2010 First edition of the Technical Report formally published (without language-specific annexes) as TR24772:2010

2012 Proposed second edition has been balloted by SC22 member bodies and WG23 are reviewing the comments. There are language-specific annexes for Ada, C, Python, Ruby, SPARK, PHP.

The draft is available free of charge from the WG23 website:

http://grouper.ieee.org/groups/plv/DocLog/400-499/400-419/22-WG23-N-0410/n0410.pdf

Structure of the TR – main clauses

1 Scope (4 lines)

2 Normative references (6 lines)

3 Terms and definitions (4½ pages)

4 Basic Concepts (2½ pages)

5 Vulnerability issues (3 pages)

General concepts e.g undefined or unspecified behaviour

6 **Programming Language Vulnerabilities (85 pages)**

Language-independent description of 57 particular vulnerabilities

7 Application Vulnerabilities (27 pages)

Selected application problems, drawn from experience

8 New Vulnerabilities (10 pages)

Vulnerabilities under consideration for future incorporation

Differences 2010 to 2012

Section 5 (Vulnerability Issues) rewritten

Section 6 (Programming Language Vulnerabilities) expanded from 53 to 57 items with some dropped, some merged and some new

The template for each vulnerability description changed

Some language-specific annexes added

Sample Language Vulnerabilities

6.3	Type Systems
6.4	Bit Representations
6.5	Floating-point Arithmetic
6.10	Unchecked Array Indexing
6.11	Unchecked Array Copying
6.17	Using Shift Operations for Multiplication and Division
6.26	Side-effects and Order of Evaluation
6.27	Likely Incorrect Expression
6.31	Loop Control Variables
6.37	Recursion
6.49	Unanticipated Exceptions from Library Routines
6.53	Obscure Language Features
6.54	Unspecified Behaviour [sic – British spelling!]
6.55	Undefined Behaviour

Structure of the TR – Annexes

A Vulnerability Taxonomy and List (4½ pages)

B Language Specific Vulnerability Template (1½ pages)

C Vulnerability descriptions for the language Ada (30 pages)

D Vulnerability descriptions for the language C (38 pages)

E Vulnerability descriptions for the language Python (39 pages)

F Vulnerability descriptions for the language Ruby (17 pages)

G Vulnerability descriptions for the language SPARK (10 pages)

H Vulnerability descriptions for the language PHP (30 pages)

Bibliography (3 pages)

Format of each Language-Specific Annex

For each language:

– Identification of standards

– General terminology and concepts

For each vulnerability within each language:

– Status, history and bibliography

– Applicability to language

(or statement that vulnerability is not applicable)

– Implications for standardization

(These headings have changed from 2010 to 2012)

Progress with the Fortran Annex

2006	Dan Nagle (J3 chairman) joins WG23 and acts as liaison to WG5
2007-2009	WG5 members are encouraged to follow developments in WG23
2009	Dan Nagle produces a skeleton Fortran annex of 51 items with a few items fleshed out
2011	WG5 formally agrees to produce a Fortran annex
2012 (April)	First draft annex, N1915, circulated for comment and discussion at WG5 meeting
2012 (August)	Second draft annex, N1929, circulated for comment with view to forwarding to WG23
2012 (September)	Decided much more work is needed

Excerpt from minutes of September 2012 WG23 meeting: "WG 5 plans to have an annex for summer 2013."

Fortran Annex: Current Status

The vote on N1929 was:

2	Yes
1	Yes with comments
7	No
1	Abstain

The comments included:

- The timescales for forwarding to WG23 are wrong as the next edition of the TR will not be produced for several months
- Numerous minor technical and editorial corrections are needed
- Some more important technical points were made
- Oracle are against the TR in principle
- Since many vulnerabilities arise from retention of F66 and F77 features, there is too little on pre-1990 history

Fortran.26 Side-effects and Order of Evaluation

Fortran.26.1 Applicability to Fortran

Fortran functions are permitted to have side effects. Within some expressions, the order of invocation of functions is not specified. The standard explicitly requires that evaluating any part of an expression does not change the value of any other part of the expression, but there is no requirement for this to be diagnosed by the compiler.

Further, the Fortran standard allows a processor to ignore any part of an expression that is not needed to compute the value of the expression. Processors vary as to how aggressively they take advantage of this permission.

Fortran.26.2 Guidance to Fortran Users

Fortran developers can avoid the vulnerability or mitigate its ill effects in the following ways:

- Replace any function with a side effect by a subroutine so that its place in the sequence of computation is certain.
- Assign function values to temporary variables and use the temporary variables in the original expression.
- Declare a function as pure whenever possible.

Fortran.27 Likely Incorrect Expression

Fortran.27.1 Applicability to Fortran

While Fortran is not as susceptible to this issue as some languages (largely because assignment = is not an operator), nevertheless, some situations exist where a single character, present or absent, could change the meaning of an expression. For example, assignment could be confused with pointer assignment when the name on the left-hand side has the pointer attribute and the name on the right-hand side has the target attribute.

Some compilers allow a dyadic operator immediately preceding a unary operator, which should be avoided. However, this can be detected by using compiler options to detect violations of the standard.

Fortran is not susceptible to the “dangling else" version of this problem because each construct has a unique end-of-construct statement.

Fortran.27.2 Guidance to Fortran Users

Fortran developers can avoid the vulnerability or mitigate its ill effects in the following ways:

- Use an automatic tool to simplify expressions.
- Check for assignment versus pointer assignment carefully when assigning to names having the pointer attribute.
- Use dummy argument intents to assist the compiler's ability to detect such occurrences.

Fortran.31 Loop Control Variables

Fortran.31.1 Applicability to Fortran

A Fortran enumerated do loop has the trip increment and trip count established when the do statement is executed. These do not change during the execution of the loop.

The program is prohibited from changing the value of an iteration variable during execution of the loop. The compiler is usually able to detect violation of this rule, but there are situations where this is difficult or requires use of a processor option; for example, an iteration variable might be changed by a procedure that is referenced within the loop.

Fortran.31.2 Guidance to Fortran Users

Fortran developers can avoid the vulnerability or mitigate its ill effects in the following ways:

- Ensure that the value of the iteration variable is not changed other than by the loop control mechanism during the execution of a do loop.

Fortran.55 Undefined Behaviour

Fortran.55.1 Applicability to Fortran

A Fortran processor is unconstrained unless the program uses forms and relations specified by the Fortran standard, and gives them the meaning described therein.

The behaviour of non-standard code can change between processors.

A processor is permitted to provide additional intrinsic procedures. One of these might be invoked instead of an intended external procedure with the same name.

Fortran.55.2 Guidance to Fortran Users

Fortran developers can avoid the vulnerability or mitigate its ill effects in the following ways:

- Use processor options to detect and report use of non-standard features.
- Obtain diagnostics from more than one source, for example, use code checking tools.
- Specify the external attribute for all external procedures invoked.

References

WG23 website:
http://grouper.ieee.org/groups/plv/

WG5 website:
http://www.nag.co.uk/sc22wg5/

To join the email group developing the Fortran annex, contact
Dan Nagle <dannagle @ verizon.net>