
**Programming languages — Guidance
to avoiding vulnerabilities in
programming languages —**

**Part 1:
Language-independent guidance**

*Langages de programmation — Conduite pour éviter les
vulnérabilités dans les langages de programmation —*

Partie 1: Conduite indépendante du langage

Document Preview

ISO/IEC TR 24772-1:2019

<https://standards.iteh.ai/catalog/standards/iso/2160a28c-6e50-48b8-a921-bfd9b239bfd0/iso-iec-tr-24772-1-2019>



Reference number
ISO/IEC TR 24772-1:2019(E)

© ISO/IEC 2019

iTeh Standards
(<https://standards.iteh.ai>)
Document Preview

ISO/IEC TR 24772-1:2019

<https://standards.iteh.ai/catalog/standards/iso/2160a28c-6e50-48b8-a921-bfd9b239bfd0/iso-iec-tr-24772-1-2019>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2019

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	xv
Introduction	xvi
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
3.1 Terms related to communication	1
3.2 Terms related to execution model	2
3.3 Properties	4
3.4 Safety	4
3.5 Vulnerabilities	4
4 Applying this document	5
5 Vulnerability issues and general avoidance mechanisms	7
5.1 Predictable execution	7
5.2 Sources of unpredictability in language specification	8
5.2.1 Incomplete or evolving specification	8
5.2.2 Undefined behaviour	8
5.2.3 Unspecified behaviour	8
5.2.4 Implementation-defined behaviour	8
5.2.5 Difficult features	8
5.2.6 Inadequate language support	8
5.3 Sources of unpredictability in language usage	9
5.3.1 Porting and interoperation	9
5.3.2 Compiler selection and usage	9
5.4 Top avoidance mechanisms	9
6 Programming language vulnerabilities	11
6.1 General	11
6.2 Type system [IHN]	11
6.2.1 Description of application vulnerability	11
6.2.2 Cross reference	11
6.2.3 Mechanism of failure	11
6.2.4 Applicable language characteristics	13
6.2.5 Avoiding the vulnerability or mitigating its effects	13
6.2.6 Implications for language design and evolution	14
6.3 Bit representations [STR]	14
6.3.1 Description of application vulnerability	14
6.3.2 Cross reference	14
6.3.3 Mechanism of failure	14
6.3.4 Applicable language characteristics	15
6.3.5 Avoiding the vulnerability or mitigating its effects	15
6.3.6 Implications for language design and evolution	15
6.4 Floating-point arithmetic [PLF]	15
6.4.1 Description of application vulnerability	15
6.4.2 Cross reference	16
6.4.3 Mechanism of failure	16
6.4.4 Applicable language characteristics	17
6.4.5 Avoiding the vulnerability or mitigating its effects	17
6.4.6 Implications for language design and evolution	17
6.5 Enumerator issues [CCB]	18
6.5.1 Description of application vulnerability	18
6.5.2 Cross reference	18
6.5.3 Mechanism of failure	18
6.5.4 Applicable language characteristics	19

6.5.5	Avoiding the vulnerability or mitigating its effects	19
6.5.6	Implications for language design and evolution	19
6.6	Conversion errors [FLC]	19
6.6.1	Description of application vulnerability	19
6.6.2	Cross reference	20
6.6.3	Mechanism of failure	20
6.6.4	Applicable language characteristics	20
6.6.5	Avoiding the vulnerability or mitigating its effects	21
6.6.6	Implications for language design and evolution	21
6.7	String termination [CJM]	21
6.7.1	Description of application vulnerability	21
6.7.2	Cross reference	21
6.7.3	Mechanism of failure	22
6.7.4	Applicable language characteristics	22
6.7.5	Avoiding the vulnerability or mitigating its effects	22
6.7.6	Implications for language design and evolution	22
6.8	Buffer boundary violation (buffer overflow) [HCB]	22
6.8.1	Description of application vulnerability	22
6.8.2	Cross reference	22
6.8.3	Mechanism of failure	23
6.8.4	Applicable language characteristics	23
6.8.5	Avoiding the vulnerability or mitigating its effects	24
6.8.6	Implications for language design and evolution	24
6.9	Unchecked array indexing [XYZ]	24
6.9.1	Description of application vulnerability	24
6.9.2	Cross reference	25
6.9.3	Mechanism of failure	25
6.9.4	Applicable language characteristics	25
6.9.5	Avoiding the vulnerability or mitigating its effects	26
6.9.6	Implications for language designers	26
6.10	Unchecked array copying [XYW]	26
6.10.1	Description of application vulnerability	26
6.10.2	Cross reference	26
6.10.3	Mechanism of failure	26
6.10.4	Applicable language characteristics	27
6.10.5	Avoiding the vulnerability or mitigating its effects	27
6.10.6	Implications for language design and evolution	27
6.11	Pointer type conversions [HFC]	27
6.11.1	Description of application vulnerability	27
6.11.2	Cross reference	28
6.11.3	Mechanism of failure	28
6.11.4	Applicable language characteristics	28
6.11.5	Avoiding the vulnerability or mitigating its effects	28
6.11.6	Implications for language design and evolution	29
6.12	Pointer arithmetic [RVG]	29
6.12.1	Description of application vulnerability	29
6.12.2	Cross reference	29
6.12.3	Mechanism of failure	29
6.12.4	Applicable language characteristics	29
6.12.5	Avoiding the vulnerability or mitigating its effects	29
6.12.6	Implications for language design and evolution	29
6.13	Null pointer dereference [XYH]	29
6.13.1	Description of application vulnerability	29
6.13.2	Cross reference	30
6.13.3	Mechanism of failure	30
6.13.4	Applicable language characteristics	30
6.13.5	Avoiding the vulnerability or mitigating its effects	30
6.13.6	Implications for language design and evolution	30

6.14	Dangling reference to heap [XYK]	30
6.14.1	Description of application vulnerability	30
6.14.2	Cross reference	31
6.14.3	Mechanism of failure	31
6.14.4	Applicable language characteristics	31
6.14.5	Avoiding the vulnerability or mitigating its effects	32
6.14.6	Implications for language design and evolution	32
6.15	Arithmetic wrap-around error [FIF]	32
6.15.1	Description of application vulnerability	32
6.15.2	Cross reference	32
6.15.3	Mechanism of failure	33
6.15.4	Applicable language characteristics	33
6.15.5	Avoiding the vulnerability or mitigating its effects	33
6.15.6	Implications for language design and evolution	33
6.16	Using shift operations for multiplication and division [PIK]	34
6.16.1	Description of application vulnerability	34
6.16.2	Cross reference	34
6.16.3	Mechanism of failure	34
6.16.4	Applicable language characteristics	34
6.16.5	Avoiding the vulnerability or mitigating its effects	34
6.16.6	Implications for language design and evolution	34
6.17	Choice of clear names [NAI]	35
6.17.1	Description of application vulnerability	35
6.17.2	Cross reference	35
6.17.3	Mechanism of Failure	35
6.17.4	Applicable language characteristics	36
6.17.5	Avoiding the vulnerability or mitigating its effects	36
6.17.6	Implications for language design and evolution	36
6.18	Dead store [WXQ]	36
6.18.1	Description of application vulnerability	36
6.18.2	Cross reference	36
6.18.3	Mechanism of failure	37
6.18.4	Applicable language characteristics	37
6.18.5	Avoiding the vulnerability or mitigating its effects	37
6.18.6	Implications for language design and evolution	37
6.19	Unused variable [YZS]	38
6.19.1	Description of application vulnerability	38
6.19.2	Cross reference	38
6.19.3	Mechanism of failure	38
6.19.4	Applicable language characteristics	38
6.19.5	Avoiding the vulnerability or mitigating its effects	38
6.19.6	Implications for language design and evolution	38
6.20	Identifier name reuse [YOW]	38
6.20.1	Description of application vulnerability	38
6.20.2	Cross reference	39
6.20.3	Mechanism of failure	39
6.20.4	Applicable language characteristics	40
6.20.5	Avoiding the vulnerability or mitigating its effects	40
6.20.6	Implications for language design and evolution	40
6.21	Namespace issues [BJL]	40
6.21.1	Description of application vulnerability	40
6.21.2	Cross-references	41
6.21.3	Mechanism of failure	41
6.21.4	Applicable language characteristics	41
6.21.5	Avoiding the vulnerability or mitigating its effects	41
6.21.6	Implications for language design and evolution	42
6.22	Initialization of variables [LAV]	42
6.22.1	Description of application vulnerability	42

6.22.2	Cross reference.....	42
6.22.3	Mechanism of failure.....	42
6.22.4	Applicable language characteristics.....	43
6.22.5	Avoiding the vulnerability or mitigating its effects.....	43
6.22.6	Implications for language design and evolution.....	44
6.23	Operator precedence and associativity [JCW].....	44
6.23.1	Description of application vulnerability.....	44
6.23.2	Cross reference.....	44
6.23.3	Mechanism of failure.....	44
6.23.4	Applicable language characteristics.....	45
6.23.5	Avoiding the vulnerability or mitigating its effects.....	45
6.23.6	Implications for language design and evolution.....	45
6.24	Side-effects and order of evaluation of operands [SAM].....	45
6.24.1	Description of application vulnerability.....	45
6.24.2	Cross reference.....	45
6.24.3	Mechanism of failure.....	46
6.24.4	Applicable language characteristics.....	46
6.24.5	Avoiding the vulnerability or mitigating its effects.....	46
6.24.6	Implications for language design and evolution.....	46
6.25	Likely incorrect expression [KOA].....	47
6.25.1	Description of application vulnerability.....	47
6.25.2	Cross reference.....	47
6.25.3	Mechanism of failure.....	47
6.25.4	Applicable language characteristics.....	48
6.25.5	Avoiding the vulnerability or mitigating its effects.....	48
6.25.6	Implications for language design and evolution.....	48
6.26	Dead and deactivated code [XYQ].....	48
6.26.1	Description of application vulnerability.....	48
6.26.2	Cross reference.....	48
6.26.3	Mechanism of failure.....	49
6.26.4	Applicable language characteristics.....	50
6.26.5	Avoiding the vulnerability or mitigating its effects.....	50
6.26.6	Implications for language design and evolution.....	50
6.27	Switch statements and static analysis [CLL].....	50
6.27.1	Description of application vulnerability.....	50
6.27.2	Cross reference.....	50
6.27.3	Mechanism of failure.....	51
6.27.4	Applicable language characteristics.....	51
6.27.5	Avoiding the vulnerability or mitigating its effects.....	51
6.27.6	Implications for language design and evolution.....	51
6.28	Demarcation of control flow [EOJ].....	52
6.28.1	Description of application vulnerability.....	52
6.28.2	Cross reference.....	52
6.28.3	Mechanism of failure.....	52
6.28.4	Applicable language characteristics.....	52
6.28.5	Avoiding the vulnerability or mitigating its effects.....	52
6.28.6	Implications for language design and evolution.....	53
6.29	Loop control variables [TEX].....	53
6.29.1	Description of application vulnerability.....	53
6.29.2	Cross reference.....	53
6.29.3	Mechanism of failure.....	53
6.29.4	Applicable language characteristics.....	53
6.29.5	Avoiding the vulnerability or mitigating its effects.....	53
6.29.6	Implications for language design and evolution.....	54
6.30	Off-by-one error [XZH].....	54
6.30.1	Description of application vulnerability.....	54
6.30.2	Cross reference.....	54
6.30.3	Mechanism of failure.....	54

6.30.4	Applicable language characteristics.....	55
6.30.5	Avoiding the vulnerability or mitigating its effects.....	55
6.30.6	Implications for language design and evolution	55
6.31	Unstructured programming [EWD]	55
6.31.1	Description of application vulnerability.....	55
6.31.2	Cross reference.....	55
6.31.3	Mechanism of failure.....	56
6.31.4	Applicable language characteristics.....	56
6.31.5	Avoiding the vulnerability or mitigating its effects.....	56
6.31.6	Implications for language design and evolution	56
6.32	Passing parameters and return values [CSJ]	56
6.32.1	Description of application vulnerability.....	56
6.32.2	Cross reference.....	57
6.32.3	Mechanism of failure.....	57
6.32.4	Applicable language characteristics.....	58
6.32.5	Avoiding the vulnerability or mitigating its effects.....	58
6.32.6	Implications for language design and evolution	58
6.33	Dangling references to stack frames [DCM].....	58
6.33.1	Description of application vulnerability.....	58
6.33.2	Cross reference.....	59
6.33.3	Mechanism of failure.....	59
6.33.4	Applicable language characteristics.....	60
6.33.5	Avoiding the vulnerability or mitigating its effects.....	60
6.33.6	Implications for language design and evolution	60
6.34	Subprogram signature mismatch [OTR].....	60
6.34.1	Description of application vulnerability.....	60
6.34.2	Cross reference.....	60
6.34.3	Mechanism of failure.....	61
6.34.4	Applicable language characteristics.....	61
6.34.5	Avoiding the vulnerability or mitigating its effects.....	61
6.34.6	Implications for language design and evolution	61
6.35	Recursion [GDL].....	62
6.35.1	Description of application vulnerability.....	62
6.35.2	Cross reference.....	62
6.35.3	Mechanism of failure.....	62
6.35.4	Applicable language characteristics.....	62
6.35.5	Avoiding the vulnerability or mitigating its effects.....	62
6.35.6	Implications for language design and evolution	63
6.36	Ignored error status and unhandled exceptions [OYB]	63
6.36.1	Description of application vulnerability.....	63
6.36.2	Cross reference.....	63
6.36.3	Mechanism of failure.....	63
6.36.4	Applicable language characteristics.....	64
6.36.5	Avoiding the vulnerability or mitigating its effects.....	64
6.36.6	Implications for language design and evolution	65
6.37	Type-breaking reinterpretation of data [AMV].....	65
6.37.1	Description of application vulnerability.....	65
6.37.2	Cross reference.....	65
6.37.3	Mechanism of failure.....	65
6.37.4	Applicable language characteristics.....	66
6.37.5	Avoiding the vulnerability or mitigating its effects.....	66
6.37.6	Implications for language design and evolution	66
6.38	Deep vs. shallow copying [YAN]	67
6.38.1	Description of application vulnerability.....	67
6.38.2	Cross reference.....	67
6.38.3	Mechanism of failure.....	67
6.38.4	Applicable language characteristics.....	67
6.38.5	Avoiding the vulnerability or mitigating its effects.....	67