DRAFT International Standard

ISO/IEC DIS 24772-8

Programming languages — Avoiding vulnerabilities in programming languages —

Part 8: Vulnerability descriptions for the programming language Fortran

ICS: 35.060

ISO/IEC JTC **1**/SC **22**

Secretariat: **ANSI**

Voting begins on:
2025-12-22

Voting terminates on:
2026-03-16

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

This document is circulated as received from the committee secretariat.

Reference number
ISO/IEC DIS 24772-8:2025(en)

© ISO/IEC 2025



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2025

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
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword ..... **vii**

Introduction ..... **viii**

1 Scope **1**

2 Normative references **1**

3 Terms and definitions **1**

4 Language concepts **2**
4.1 General 2
4.2 Fortran standard concepts and terminology 3
4.3 Deleted and redundant features 3
4.4 Non-standard extensions 3
4.5 Conformance to the standard 4
4.6 Numeric model 4
4.7 Interoperability 4
4.8 Allocatable variables 4
4.9 Polymorphism 5
4.10 Parallelism 5
4.10.1 Images and coarrays 5
4.10.2 Locks 6
4.10.3 Teams 6
4.10.4 Segments 6
4.10.5 Atomic actions 7
4.10.6 Asynchronous variables 7
4.10.7 Volatile variables 7
4.10.8 Collective subroutines 8
4.10.9 Image failure 8
4.10.10 Do concurrent 8

5 General avoidance mechanisms for Fortran **8**

6 Specific analysis for Fortran **10**
6.1 General 10
6.2 Type system [IHN] 10
6.2.1 Applicability to language 10
6.2.2 Avoidance mechanisms for language users 11
6.3 Bit representation [STR] 11
6.3.1 Applicability to language 11
6.3.2 Avoidance mechanisms for language users 12
6.4 Floating-point arithmetic [PLF] 12
6.4.1 Applicability to language 12
6.4.2 Avoidance mechanisms for language users 13
6.5 Enumerator issues [CCB] 13
6.5.1 Applicability to language 13
6.5.2 Avoidance mechanisms for language users 13
6.6 Conversion errors [FLC] 14
6.6.1 Applicability to language 14
6.6.2 Avoidance mechanisms for language users 14
6.7 String termination [CJM] 15
6.8 Buffer boundary violation (Buffer overflow) [HCB] 15
6.8.1 Applicability to language 15
6.8.2 Avoidance mechanisms for language users 15
6.9 Unchecked array indexing [XYZ] 16
6.9.1 Applicability to language 16
6.9.2 Avoidance mechanisms for language users 16
6.10 Unchecked array copying [XYW] 17
6.10.1 Applicability to language 17

© ISO/IEC 2025 – All rights reserved

6.10.2 Avoidance mechanisms for language users 17
6.11 Pointer type conversions [HFC] 17
6.11.1 Applicability to language 17
6.11.2 Avoidance mechanisms for language users 17
6.12 Pointer arithmetic [RVG] 18
6.13 Null pointer dereference [XYH] 18
6.13.1 Applicability to language 18
6.13.2 Avoidance mechanisms for language users 18
6.14 Dangling reference to heap [XYK] 18
6.14.1 Applicability to language 18
6.14.2 Avoidance mechanisms for language users 18
6.15 Arithmetic wrap-around error [FIF] 19
6.15.1 Applicability to language 19
6.15.2 Avoidance mechanisms for language users 19
6.16 Using shift operations for multiplication and division [PIK] 19
6.16.1 Applicability to language 19
6.16.2 Avoidance mechanisms for language users 19
6.17 Choice of clear names [NAI] 19
6.17.1 Applicability to language 19
6.17.2 Avoidance mechanisms for language users 20
6.18 Dead store [WXQ] 20
6.18.1 Applicability to language 20
6.18.2 Avoidance mechanisms for language users 20
6.19 Unused variable [YZS] 20
6.19.1 Applicability to language 20
6.19.2 Avoidance mechanisms for language users 20
6.20 Identifier name reuse [YOW] 20
6.20.1 Applicability to language 20
6.20.2 Avoidance mechanisms for language users 21
6.21 Namespace issues [BJL] 21
6.21.1 Applicability to language 21
6.21.2 Avoidance mechanisms for language users 21
6.22 Missing initialization of variables [LAV] 21
6.22.1 Applicability to language 21
6.22.2 Avoidance mechanisms for language users 21
6.23 Operator precedence and associativity [JCW] 22
6.23.1 Applicability to language 22
6.23.2 Avoidance mechanisms for language users 22
6.24 Side-effects and order of evaluation [SAM] 22
6.24.1 Applicability to language 22
6.24.2 Avoidance mechanisms for language users 22
6.25 Likely incorrect expression [KOA] 23
6.25.1 Applicability to language 23
6.25.2 Avoidance mechanisms for language users 23
6.26 Dead and deactivated code [XYQ] 23
6.26.1 Applicability to language 23
6.26.2 Avoidance mechanisms for language users 23
6.27 Switch statements and static analysis [CLL] 23
6.27.1 Applicability to language 23
6.27.2 Avoidance mechanisms for language users 24
6.28 Demarcation of control flow [EOJ] 24
6.28.1 Applicability to language 24
6.28.2 Avoidance mechanisms for language users 24
6.29 Loop control variable abuse [TEX] 24
6.29.1 Applicability to language 24
6.29.2 Avoidance mechanisms for language users 25
6.30 Off-by-one error [XZH] 25
6.30.1 Applicability to language 25
6.30.2 Avoidance mechanisms for language users 25

© ISO/IEC 2025 – All rights reserved

6.31 Unstructured programming [EWD] 25
6.31.1 Applicability to language 25
6.31.2 Avoidance mechanisms for language users 26
6.32 Passing parameters and return values [CSJ] 26
6.32.1 Applicability to language 26
6.32.2 Avoidance mechanisms for language users 26
6.33 Dangling references to stack frames [DCM] 27
6.33.1 Applicability to language 27
6.33.2 Avoidance mechanisms for language users 27
6.34 Subprogram signature mismatch [OTR] 27
6.34.1 Applicability to language 27
6.34.2 Avoidance mechanisms for language users 27
6.35 Recursion [GDL] 28
6.35.1 Applicability to language 28
6.35.2 Avoidance mechanisms for language users 28
6.36 Ignored error status and unhandled exceptions [OYB] 28
6.36.1 Applicability to language 28
6.36.2 Avoidance mechanisms for language users 28
6.37 Type-breaking reinterpretation of data [AMV] 29
6.37.1 Applicability to language 29
6.37.2 Avoidance mechanisms for language users 29
6.38 Deep vs. shallow copying [YAN] 29
6.38.1 Applicability to language 29
6.38.2 Avoidance mechanisms for language users 29
6.39 Memory leaks and heap fragmentation [XYL] 29
6.39.1 Applicability to language 29
6.39.2 Avoidance mechanisms for language users 30
6.40 Templates and generics [SYM] 30
6.41 Inheritance [RIP] 30
6.41.1 Applicability to language 30
6.41.2 Avoidance mechanisms for language users 30
6.42 Violations of the Liskov substitution principle or the contract model [BLP] 30
6.42.1 Applicability to language 30
6.42.2 Avoidance mechanisms for language users 30
6.43 Redispatching [PPH] 31
6.43.1 Applicability to language 31
6.43.2 Avoidance mechanisms for language users 31
6.44 Polymorphic variables [BKK] 31
6.44.1 Applicability to language 31
6.44.2 Avoidance mechanisms for language users 31
6.45 Extra intrinsics [LRM] 31
6.45.1 Applicability to language 31
6.45.2 Avoidance mechanisms for language users 32
6.46 Argument passing to library functions [TRJ] 32
6.46.1 Applicability to language 32
6.46.2 Avoidance mechanisms for language users 32
6.47 Inter-language calling [DJS] 32
6.47.1 Applicability to language 32
6.47.2 Avoidance mechanisms for language users 33
6.48 Dynamically-linked code and self-modifying code [NYY] 33
6.49 Library signature [NSQ] 33
6.49.1 Applicability to language 33
6.49.2 Avoidance mechanisms for language users 33
6.50 Unanticipated exceptions from library routines [HJW] 33
6.50.1 Applicability to language 33
6.50.2 Avoidance mechanisms for language users 34
6.51 Pre-processor directives [NMP] 34
6.51.1 Applicability to language 34
6.51.2 Avoidance mechanisms for language users 34

© ISO/IEC 2025 – All rights reserved

6.52 Suppression of language-defined run-time checking [MXB] 34
6.52.1 Applicability to language 34
6.52.2 Avoidance mechanisms for language users 34
6.53 Provision of inherently unsafe operations [SKL] 35
6.53.1 Applicability to language 35
6.53.2 Avoidance mechanisms for language users 35
6.54 Obscure language features [BRS] 35
6.54.1 Applicability to language 35
6.54.2 Avoidance mechanisms for language users 36
6.55 Unspecified behaviour [BQF] 36
6.55.1 Applicability to language 36
6.55.2 Avoidance mechanisms for language users 36
6.56 Undefined behaviour [EWF] 36
6.56.1 Applicability to language 36
6.56.2 Avoidance mechanisms for language users 36
6.57 Implementation-defined behaviour [FAB] 37
6.57.1 Applicability to language 37
6.57.2 Avoidance mechanisms for language users 37
6.58 Deprecated language features [MEM] 37
6.58.1 Applicability to language 37
6.58.2 Avoidance mechanisms for language users 37
6.59 Concurrency – Activation [CGA] 37
6.59.1 Applicability to language 37
6.59.2 Avoidance mechanisms for language users 38
6.60 Concurrency – Directed termination [CGT] 38
6.60.1 Applicability to language 38
6.60.2 Avoidance mechanisms for language users 38
6.61 Concurrent data access [CGX] 38
6.61.1 Applicability to language 38
6.61.2 Avoidance mechanisms for language users 39
6.62 Concurrency – Premature termination [CGS] 39
6.62.1 Applicability to language 39
6.62.2 Avoidance mechanisms for language users 39
6.63 Protocol lock errors [CGM] 40
6.63.1 Applicability to language 40
6.63.2 Avoidance mechanisms for language users 40
6.64 Reliance on external format strings [SHL] 40
6.64.1 Applicability to language 40
6.64.2 Avoidance mechanisms for language users 40
6.65 Modifying constants [UJO] 40
6.65.1 Applicability to language 40
6.65.2 Avoidance mechanisms for language users 41

7 Language specific vulnerabilities for Fortran 41
7.1 General 41
7.2 Source form 41
7.2.1 Applicability to language 41
7.2.2 Avoidance mechanisms for language users 41
7.3 Unformatted files 41
7.3.1 Applicability to language 41
7.3.2 Avoidance mechanisms for language users 42

Bibliography 43

Index 45

© ISO/IEC 2025 – All rights reserved

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

© ISO/IEC 2025 – All rights reserved

Introduction

This Standard documents avoidance mechanisms for the programming language Fortran so that application developers considering Fortran or using Fortran will be better able to avoid the programming constructs that lead to vulnerabilities in software written in the Fortran language and their attendant consequences. This guidance can also be used by developers to select source code evaluation tools that can discover and eliminate some constructs that could lead to vulnerabilities in their software. It can also be used in comparison with companion standards including the language-independent standard, ISO/IEC 24772-1:2024*Programming languages — Avoiding vulnerabilities in programming language -- Part 1: Language-independent catalogue of vulnerabilities*, to select a programming language that provides the appropriate level of confidence that anticipated problems can be avoided.

It should be noted that this document is inherently incomplete. It is not possible to provide a complete list of programming language vulnerabilities because new weaknesses are discovered continually. Any such report can only describe those that have been found, characterized, and determined to have sufficient probability and consequence.

© ISO/IEC 2025 – All rights reserved

Programming languages — Avoiding vulnerabilities in programming languages —

Part 8:
Vulnerability descriptions for the programming language Fortran

1 Scope

This document itemizes programming language vulnerabilities to be avoided in the development of systems where assured behaviour is required for security, safety, mission-critical and business-critical software. In general, this guidance is applicable to the software developed, reviewed, or maintained for any application.

This document explains how the vulnerabilities described in the language-independent writeup, ISO/IEC 24772-1:2024 “Programming languages -- Avoiding vulnerabilities in programming languages -- Part 1: “Language-independent catalogue of vulnerabilities”, manifested in Fortran and documents mechanisms that can be used to avoid the vulnerabilities.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 24772-1:2024, *Programming languages — Avoiding vulnerabilities in programming languages — Part 1: Language-independent catalogue of vulnerabilities*

ISO/IEC 1539-1:2023, *Programming languages — Fortran — Part 1: Base language*

ISO/IEC/IEEE 60559:2020, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24772-1:2024, ISO/IEC 1539-1:2023 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
argument association
association between an effective argument and a dummy argument

3.2
assumed-shape array
dummy argument array whose shape is assumed from the corresponding actual argument

3.3
assumed-size array:
dummy argument array whose size is assumed from the corresponding actual argument

© ISO/IEC 2025 – All rights reserved

3.4
deleted feature
feature that existed in older versions of Fortran but has been removed from later versions of the standard

3.5
explicit interface
interface of a procedure that includes all the characteristics of the procedure and names for its dummy arguments

3.6
Image
one of a mutually cooperating set of instances of a Fortran program, each with its own execution state and set of data objects

3.7
implicit typing
archaic rule that declares a variable upon use according to the first letter of its name

3.8
kind type parameter
value that determines one of a set of processor-dependent data representation methods

3.9
Module
separate scope that contains definitions that can be accessed from other scopes

3.10
obsolescent feature
feature that is not recommended because better methods exist in the current standard

3.11
processor
combination of computing system and mechanism by which programs are transformed for use on that computing system

3.12
processor dependent
not completely specified in the Fortran standard, having one of a set of methods and semantics determined by the processor

3.13
pure procedure
procedure subject to constraints such that its execution has no side effects

3.14
type
named category of data characterized by a set of values, a syntax for denoting these values, and a set of operations that interpret and manipulate the values

4 Language concepts

4.1 General

Fortran is the oldest international standard programming language with the first Fortran processors appearing over fifty years ago. During half a century of computing, computing technology has changed immensely, and Fortran has evolved via several revisions of the standard. Also, during half a century of computing and in response to customer demand, some popular processors supported extensions. There remains a substantial body of Fortran code that is written to previous versions of the standard or with extensions to previous versions, and before modern techniques of software development came into widespread use. The process of revising the standard has been done carefully with a goal of protecting

© ISO/IEC 2025 – All rights reserved