6.32.1 Applicability to language

The vulnerability specified in ISO/IEC 24772-1:2019 clause 6.32 applies to Fortran, but is mitigated to some extent.

Module procedures, intrinsic procedures, and internal procedures have explicit interfaces. An external procedure has an explicit interface only when one is provided by a procedure declaration or interface body. Such an interface body could be generated automatically using a software tool. Explicit interfaces allow processors to check the attributes, including type, kind, and rank, of arguments and result variables of functions.

Fortran does not specify the argument-passing mechanism, but rather specifies the rules of *argument association*. These rules are generally implemented either by reference or by copy. More restrictive rules apply to coarrays and to arrays with the `contiguous` attribute. Rules for procedures declared to have a C binding follow the rules of C. Copying can be limited by the programmer specifying `intent(in)`, `intent(out)`, or `value`.

Incorrect choice of parameter passing mechanism is therefore minimized, provided the intent specifications for the arguments are provided and correct. Moreover, the vulnerability of passing an incorrect address of a data structure is limited by the requirement that targets of pointers always have the correct type, kind, and rank.

On the other hand, a vulnerability arises if the programmer relies on a particular parameter mechanism while the compiler chooses a different one. This is particularly the case when aliasing is present.

Aliasing cannot occur for arguments declared with the `value` attribute. Aliasing does not accord with the Fortran standard, but its detection is unlikely unless runtime checks are available and are employed. Aliasing effects inside procedures can depend on the argument-passing mechanism chosen by the compiler.

The vulnerability of an uninitialized result value or `intent(out)` argument exists, when it is not assigned a value in the subprogram.

6.32.2 Avoidance mechanisms for language users

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

….

- If available, use runtime checks against aliasing, at least during development;
- Ensure that the result of a function is assigned;