ISO/IEC JTC 1/SC 22/OWGV N0054

Contribution from Stephen Michell, Revisions of "Vulnerabilities Issues from TR15942"

Revises [N0048]

12 December 2006

- **SM 004 Arrays**

  Sect 5.1

Unpredictable behaviour can exist when accessing the elements of an array without first checking the bounds of the array and ensuring that the access is within the bounds. For statically defined arrays in language systems that enforce correct specification of the elemnts to be accessed this is provided by the language systems. For dynamicaly specified arrays in language systems that enforce legal access, out-of-bounds access can result in exceptional behaviour. For access in languages that do no no language-defined checks, out of range access can result in undefined behaviour and additional checks such as static analysis tools or human review is required.

Another vulnerability can occur when other techniques such as direct pointers that rely upon knowledge of the representation provided by the implementation are used. Such techniques are problematic for tools and for human review and should be avoided.

-

SM 005 Objects with variant structure

Sect 5.1

Most programming languages have ways to permit a contiguous set of storage locations to be viewed in different ways at different times within the application. The most common application-visible way to accomplish this is the union (C/C++) or variant record (Ada, Pascal), and equivalences in Fortran.

The risk in such data types is that data is at risk of being viewed using a different set of rules and representation than those that created the data, possibly leading to exceptional behaviour or undefined behaviour. There is also the possibility that objects of such types could be used as private channels to move information through an application.

Applications should always ensure that objetcs written under one view are not read using another view. Language systems that prevent such behaviours require no additional analysis. Language systems that permit this behaviour need to show that all reads on such objects respect the type information created when the object was last written. Where such behaviours are localized, this may be done with analysis tools or review: where such possible behaviours are not localized, more extensive reasoning and reviews are required, such as the possible the addition of extra information to the structure to retain knowledge of its current type.

SM 007 Variable Sized Objects

Sect 5.1

Variable size objects are created at runtime and usually use dynamic memory (see xx.x) or static buffers that exceed all expected uses of the object . They are often used to commuincate with the environment or “foreign” portions of the system to exchange information. There is a significant risk that such objects can have their size exceeded, lack mechanisims to ensure that the size allocated is honored by the unobserved environment, or can be used to exchange private data. Language systems that strictly bound all such data will create exceptional processing if such bounds are exceeded. For systems that do not strictly bound such objects, application-definied bonds controls are required. Analysis tools or human review will be required to verify the safety of these mechanisms.

Facet : Dynamic storage techniques