ARR02-C. Explicitly specify array bounds, even if implicitly defined by an initializer

The C Standard allows an array variable to be declared both with a bound and with an initialization literal. The initialization literal also implies an array bound in the number of elements specified.

The size implied by an initialization literal is usually specified by the number of elements,

```
int array[] = {1, 2, 3}; /* 3-element array */
```

but it is also possible to use designators to initialize array elements in a noncontiguous fashion. Subclause 6.7.9, Example 12, of the C Standard [ISO/IEC 9899:2011] states:

> *Space can be "allocated" from both ends of an array by using a single designator:*
>
> ```
> int a[MAX] = {
>   1, 3, 5, 7, 9, [MAX-5] = 8, 6, 4, 2, 0
> };
> ```
>
> *In the above, if MAX is greater than ten, there will be some zero-valued elements in the middle; if it is less than ten, some of the values provided by the first five initializers will be overridden by the second five.*

The C Standard also dictates how array initialization is handled when the number of initialization elements does not equal the explicit array bound. Subclause 6.7.9, paragraphs 21 and 22, state:

> *If there are fewer initializers in a brace-enclosed list than there are elements or members of an aggregate, or fewer characters in a string literal used to initialize an array of known size than there are elements in the array, the remainder of the aggregate shall be initialized implicitly the same as objects that have static storage duration.*
> *If an array of unknown size is initialized, its size is determined by the largest indexed element with an explicit initializer. The array type is completed at the end of its initializer list.*

Although compilers can compute the size of an array on the basis of its initialization list, explicitly specifying the size of the array provides a redundancy check, ensuring that the array's size is correct. It also enables compilers to emit warnings if the array's size is less than the size implied by the initialization.

Note that this recommendation does not apply (in all cases) to character arrays initialized with string literals. See STR11-C. Do not specify the bound of a character array initialized with a string literal for more information.

Noncompliant Code Example (Incorrect Size)

This noncompliant code example initializes an array of integers using an initialization with too many elements for the array:

```
int a[3] = {1, 2, 3, 4};
```

The size of the array a is 3, although the size of the initialization is 4. The last element of the initialization (4) is ignored. Most compilers will diagnose this error.

Implementation Details

This noncompliant code example generates a warning in GCC. Microsoft Visual Studio generates a fatal diagnostic: `error C2078: too many initializers`.

Noncompliant Code Example (Implicit Size)

In this example, the compiler allocates an array of four integer elements and, because an array bound is not explicitly specified by the programmer, sets the array bound to 4. However, if the initializer changes, the array bound may also change, causing unexpected results.

```
int a[] = {1, 2, 3, 4};
```

Compliant Solution

This compliant solution explicitly specifies the array bound:

```
int a[4] = {1, 2, 3, 4};
```

Explicitly specifying the array bound, although it is implicitly defined by an initializer, allows a compiler or other static analysis tool to issue a diagnostic if these values do not agree.

Exceptions

ARR02-C-EX1: STR11-C. Do not specify the bound of a character array initialized with a string literal is a specific exception to this recommendation; it requires that the bound of a character array initialized with a string literal is unspecified.

Risk Assessment

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
ARR02-C	Medium	Unlikely	Low	**P6**	**L2**

Automated Detection

Tool	Version	Checker	Description
Astrée	19.04	**array-size-global**	Partially checked
Axivion Bauhaus Suite	6.9.0	**CertC-ARR02**	Fully implemented
CodeSonar	5.1p0	**LANG.STRUCT.DECL.FAM**	Declaration of flexible array member
Compass/ROSE			
ECLAIR	1.2	**CC2.ARR02**	Fully implemented
LDRA tool suite	9.7.1	**127 S** **397 S** **404 S**	Fully implemented
Parasoft C/C++test	10.4.2	**CERT_C-ARR02-a**	Explicitly specify array bounds in array declarations with initializers
Polyspace Bug Finder	R2019b	CERT C: Rec. ARR02-C	Checks for improper array initialization (rec, partially covered).
PRQA QA-C	9.5	**0688,3674,3684, 0678**	Fully implemented
PVS-Studio	6.23	**V798**	
RuleChecker	19.04	**array-size-global**	Partially checked
SonarQube C/C++ Plugin	3.11	**S834**	

Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Related Guidelines

Key here (explains table format and definitions)

Taxonomy	Taxonomy item	Relationship
CERT C	CTR02-CPP. Explicitly specify array bounds, even if implicitly defined by an initializer	Prior to 2018-01-12: CERT: Unspecified Relationship
CWE 2.11	CWE-665, Incorrect or incomplete initialization	Prior to 2018-01-12: CERT:
MISRA C: 2012	Rule 8.11 (advisory)	Prior to 2018-01-12: CERT: Unspecified Relationship
MISRA C: 2012	Rule 9.5 (required)	Prior to 2018-01-12: CERT: Unspecified Relationship

Bibliography

[ISO/IEC 9899:2011]	Subclause 6.7.9, "Initialization"