CTR53-CPP. Use valid iterator ranges

When iterating over elements of a container, the iterators used must iterate over a valid range. An iterator range is a pair of iterators that refer to the first and past-the-end elements of the range respectively.

A valid iterator range has all of the following characteristics:

- Both iterators refer into the same container.
- The iterator representing the start of the range precedes the iterator representing the end of the range.
- The iterators are not invalidated, in conformance with CTR51-CPP. Use valid references, pointers, and iterators to reference elements of a container.

An empty iterator range (where the two iterators are valid and equivalent) is considered to be valid.

Using a range of two iterators that are invalidated or do not refer into the same container results in undefined behavior.

Noncompliant Code Example

In this noncompliant example, the two iterators that delimit the range point into the same container, but the first iterator does not precede the second. On each iteration of its internal loop, `std::for_each()` compares the first iterator (after incrementing it) with the second for equality; as long as they are not equal, it will continue to increment the first iterator. Incrementing the iterator representing the past-the-end element of the range results in undefined behavior.

```
#include <algorithm>
#include <iostream>
#include <vector>

void f(const std::vector<int> &c) {
  std::for_each(c.end(), c.begin(), [](int i) { std::cout << i; });
}
```

Invalid iterator ranges can also result from comparison functions that return true for equal values. See CTR57-CPP. Provide a valid ordering predicate for more information about comparators.

Compliant Solution

In this compliant solution, the iterator values passed to `std::for_each()` are passed in the proper order.

```
#include <algorithm>
#include <iostream>
#include <vector>

void f(const std::vector<int> &c) {
  std::for_each(c.begin(), c.end(), [](int i) { std::cout << i; });
}
```

Noncompliant Code Example

In this noncompliant code example, iterators from different containers are passed for the same iterator range. Although many STL implementations will compile this code and the program may behave as the developer expects, there is no requirement that an STL implementation treat a default-initialized iterator as a synonym for the iterator returned by `end()`.

```
#include <algorithm>
#include <iostream>
#include <vector>

void f(const std::vector<int> &c) {
  std::vector<int>::const_iterator e;
  std::for_each(c.begin(), e, [](int i) { std::cout << i; });
}
```

Compliant Solution

In this compliant solution, the proper iterator generated by a call to `end()` is passed.

```
#include <algorithm>
#include <iostream>
#include <vector>

void f(const std::vector<int> &c) {
  std::for_each(c.begin(), c.end(), [](int i) { std::cout << i; });
}
```

Risk Assessment

Using an invalid iterator range is similar to allowing a buffer overflow, which can lead to an attacker running arbitrary code.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
CTR53-CPP	High	Probable	High	P6	L2

Automated Detection

Tool	Version	Checker	Description
Parasoft C/C++test	10.4.2	**CERT_CPP-CTR53-a** **CERT_CPP-CTR53-b**	Do not use an iterator range that isn't really a range Do not compare iterators from different containers
PRQA QA-C++	4.4	**3802**	
PVS-Studio	6.23	**V539**, **V662**, **V789**	

Related Vulnerabilities

In *Fun with erase()*, Chris Rohlf discusses the exploit potential of a program that calls `vector::erase()` with invalid iterator ranges [Rohlf 2009].

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

Related Guidelines

SEI CERT C++ Coding Standard	CTR51-CPP. Use valid references, pointers, and iterators to reference elements of a container CTR57-CPP. Provide a valid ordering predicate

Bibliography

[ISO/IEC 14882-2014]	Clause 24, "Iterators Library" Subclause 25.3, "Mutating Sequence Operations"
[Meyers 2001]	Item 32, "Follow Remove-Like Algorithms with `erase` If You Really Want to Remove Something"