C++ static code analysis: Functions having rvalue reference arguments should "std::move" those arguments

2 minutes

Rvalue reference arguments allow to transfer ownership of objects.

When a function has an argument of type rvalue reference, it is expected to call `std::move` to take ownership of the corresponding parameter when it is called.

If it does not do it or if it does it conditionally, ownership is unclear and this might lead to bugs.

This rule does not apply when the argument is a forwarding reference.

Noncompliant Code Example

```
enum class Shape {
  empty, circle, square
};

class DrawingStore {
  std::vector<Shape> store;
public:
  void insertShape(Shape &&shape) {
    store.emplace_back(shape); // Noncompliant, call to std::move is expected
  }

  void insertIfCircle(Shape &&shape) {
      if (shape == Shape::circle) {
          store.emplace_back(std::move(shape)); // Noncompliant, std::move is not always called, ownership of shape is not clear
      }
  }
};
```

Compliant Solution

```
enum class Shape {
  empty, circle, square
};

class DrawingStoreOk {
  std::vector<Shape> store;
public:
  void insertShape(Shape &&shape) {
    store.emplace_back(std::move(shape));
  }

  void insertIfCircle(const Shape &shape) {
      if (shape == Shape::circle) {
          store.emplace_back(shape);
      }
  }
};
```

See

- C++ Core Guidelines F.18 - For “will-move-from” parameters, pass by X&& and std::move the parameter