C++ static code analysis: "std::move" should only be used where moving can happen

2-3 minutes

When calling `std::move` on an object, we usually expect the resulting operation to be fast, using move semantic to rip data off the source object. If, despite the call to `std::move`, the source object ends up being copied, the code might be unexpectedly slow.

This can happen:

- When `std::move` is called on an object which does not provide a specific move constructor and will resort to copying when requested to move.
- When calling `std::move` with a const argument.
- When passing the result of `std::move` as a const reference argument. In this case, no object will be moved since it’s impossible to call the move constructor from within the function. `std::move` should only be used when the argument is passed by value or by r-value reference.

Noncompliant Code Example

```
struct MoveWillCopy{
  MoveWillCopy() = default;
  // This user-provided copy constructor prevents the automatic generation of a move constructor
  MoveWillCopy(NonMovable&) = default;
  Data d;
};

void f(MoveWillCopy m);
void f(std::string s);
void g(const std::string &s);

void test() {
  MoveWillCopy m;
  f(std::move(m)); // Noncompliant: std::move is useless on objects like m: Any attempt to move it will copy it

  const std::string constS="***";
  f(std::move(constS)); // Noncompliant: constS will not be moved

  std::string s="****";
  g(std::move(s)); // Noncompliant: s is cast back to const l-value reference. g cannot move from it


}
```

Compliant Solution

```
struct Movable{
  Movable() = default;
  // A move constructor is generated by default
  Data d;
};

void f(Movable m);
void f(std::string s);
void g(const std::string &s);

void test() {
  Movables m;
  f(std::move(m)); // Compliant: move constructor is available

  std::string s="****";
  f(std::move(s)); // Compliant:  move constructor is called

  g(s); // Compliant: no misleading std::move is used
}
```

See

- C++ Core Guidelines ES.56 - Write "std::move()" only when you need to explicitly move an object to another scope