6.xx Violations of the Liskov Principle or the Contract Model [BLP]

6.xx.1 Description of application vulnerability

Object orientation typically allows polymorphic variables containing values of subclasses of the declared class of the variable. Methods of the declared class of a receiving object can be invoked and the caller has the right to expect that the semantics of the interface called upon are observed regardless of the precise nature of the value of the receiving object. Similarly the existence of accessed components of the declared class needs to be ensured. Instances of subclasses thus need to be both technically and logically specialized instances of the parent class. This is known as the Liskov Principle: an instance of a subclass is always an instance of the superclass as well if one ignores the added specializations. It implies that inheritance is used only if there is a logical "is-a"-relationship between the subclass and the superclass.

Moreover, preconditions of methods can at most be weakened and never strengthened as they are redefined for a subclass. Inversely, postconditions can at most be strengthened and never be weakened by such a redefinition. The caller of an interface needs to guarantee only the preconditions of the interface and is allowed to rely on its postconditions. The rules stated make sure of this property which is also known as the Contract Model.

Violations of the Liskov Principle or the Contract Model obviously can result in system malfunctions as additional preconditions of redefinitions or promised postconditions of interfaces are not met.

An alternative inheritance semantics is that of "has-a"-relationships, usually appearing in programs in languages with multi-inheritance, where the paradigm is sometimes referred to as a "mix-in". It is in stark conflict with the Liskow Principle: A polymorphic variable `motor` of class `engine` should not be able to hold a car, merely because the subclass `car` was created by a mix-in of the class `engine` to the class `vehicle`.

6.xx.2 Cross reference

CWE: << TBD >>
JSF AV Rules: 89, 91, 92
CERT C guidelines: <<TBD>>
Ada Quality and Style Guide: <<TBD>>

6.xx.3 Mechanism of failure

Failing to meet preconditions or to guarantee postconditions is bound to cause exceptions or system failures. The specific scenarios are manifold and range from complete loss of security and safety to faults that happen to be handled by the system.

Using visible inheritance to implement a “has-a”-relationships deteriorates class design and thereby may be the cause of consequential errors. There is no immediate failure mode, however.

6.xx.4 Applicable language characteristics

This vulnerability description is intended to be applicable to languages with the following characteristics:

- Languages that have polymorphic variables, particularly object-oriented languages.
- Languages that provide inheritance among classes.

6.xx.5 Avoiding the vulnerability or mitigating its effects

Software developers can avoid the vulnerability or mitigate its ill effects in the following ways:

- Be aware that methods have pre- and postconditions (specified in the language or not).
- Prohibit the strengthening of preconditions (specified or not) by redefinitions of methods.
- Prohibit the weakening of postconditions (specified or not) by redefinitions of methods.
- Prohibit the use of visible inheritance for “has-a” relationships.
- Use components of the respective class for “has-a”-relationships.

6.xx.6 Implications for standardization

In future standardization activities, the following items should be considered:

- Provide specified pre- and postconditions.