**Code Correctness: Erroneous Class Compare Development Mitigation SOP**

Code correctness vulnerabilities occur when an Object API is not used properly or as intended. Code correctness vulnerabilities can occur from erroneous class comparisons because it can lead to unexpected behaviors or allow an attacker to inject a malicious class. An attacker could deliberately duplicate class names in order to cause a program to execute malicious code.

**Defense Against Code Correctness: Erroneous Class Compare**

A class-equivalence comparison should be used to identify the type of an object.

**Examples**

**General Example**

if(inputReader.getClass().getName().equals(“TrustedName”)) {

inputReader.getInput();

…

}

**Explanation**

The code above trusts the input from the inputReader object based on its class name. If an attacker is able to supply an implementation of inputReader that executes malicious commands, this code will be unable to differentiate the benign and malicious versions of the object.

**Recommendation**

The code below is rewritten to use a class-equivalency comparison to determine whether the inputReader object has the expected type:

if(inputReader.getClass() == TrustedClass) {

input = inputReader.getInput();

…

}

**Example**

if(propertyDescription.getClass().getName().equals(FILENET\_STRING\_PROPERTY\_CLASSNAME))) {…}

**Explanation**

This example from our project shows the exact problem as the general example above. The input from propertyDescription is trusted, which allows for an attacker to inject a malicious class.

**Resources**

1. [HP Enterprise Security – Code Correctness: Erroneous Class Compare](https://vulncat.fortify.com/en/detail?id=desc.dataflow.java.code_correctness_erroneous_class_compare#Java%2fJSP)