**Code Correctness: Erroneous String Compare**

Strings should be compared with the equals() method, not == or !=.

**Explanation**

This program uses == or != to compare two strings for equality, which compares two objects for equality, not their values. Chances are good that the two references will never be equal.

**Example**

The following branch will never be taken.

if (args[0] == STRING\_CONSTANT) {

logger.info("miracle");

}

The == and != operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the String class. Once a string is interned, all uses of that string will use the same object and equality operators will behave as expected. All string literals and string-valued constants are interned automatically. Other strings can be interned manually by calling String.intern(), which will return a canonical instance of the current string, creating one if necessary.

Recommendations

Use equals() to compare strings.

**Example**

The code in Example 1 could be rewritten in the following way:

if (STRING\_CONSTANT.equals(args[0])) {

logger.info("could happen");

}

**Example**

if (args[0] == STRING\_CONSTANT) {

doWork(args[0]);

} else if (STRING\_CONSTANT.equals(args[0])) {

doWork(args[0]);

}

In many circumstances this small performance gain comes at the cost of duplicated or convoluted code. Because String.equals() performs a == comparison before doing any deeper evaluation, the performance benefits of using == instead of String.equals() are limited to the overhead of performing a method call and are likely to be negligible in most environments.

**Reference**

* http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/code\_correctness\_erroneous\_string\_compare.html