**Code Correctness: Call to System.gc() Development Mitigation SOP**

Code correctness vulnerabilities occur when an Object API is not used properly or as intended. Code correctness vulnerabilities can occur on a call to System.gc(). This method’s purpose is to run the garbage collector in an effort to rid of unused objects to make memory available. Use of this method typically indicates performance problems and a lack of reasoning as to why this method needed to be used in the first place, other than the fact that it was a magical solution that made the code run correctly.

**Defense Against Code Correctness: Call to System.gc()**

It is important to look for other answers to solve the problem that System.gc() seemed to solve in the first place. The problems generally lie in problems involving time and interaction between threads, processes, or the JVM and the operating system.

**Example**

public void handleJvm() {

System.gc();

long total = Runtime.getRuntime.totalMemory();

long free = Runtime.getRuntime.freeMemory();

…

}

**Explanation**

There seems to be no reason to be calling System.gc() This would hint at the fact that the developer had a problem that they were unable to debug to the point of calling System.gc() to blame the garbage collector. Another solution would need to be found as to why this was the solution used and how to fix it properly.

**References**

1. [HP Enterprise Security – Code Correctness: Call to System.gc()](https://vulncat.fortify.com/en/detail?id=desc.structural.java.code_correctness_call_to_system_gc#Java%2fJSP)
2. [OWASP – Code Correctness: Call to System.gc()](https://www.owasp.org/index.php/Code_Correctness:_Call_to_System.gc())
3. [Tutorials Point – System.gc()](http://www.tutorialspoint.com/java/lang/system_gc.htm)