

# Improving the Security of Garbage Collectors in the Java Virtual Machine

Zlepšování bezpečnosti garbage collectorů v Java Virtual Machine

**AUTHOR**  
**SUPERVISOR**  
**CONSULTANTS**

Ing. Patrik Čerbák  
Ing. David Kozák, Ph.D.  
Francisco Ferrari Bihurriet  
Martin Balao Alonso  
*and others*

**SanitizeGC:** a proof-of-concept extension of the G1 garbage collector that turns silent **use-after-free** vulnerabilities into immediate, debuggable crashes.

## Why does a memory-safe language need this?

Java programs themselves do not need to manage memory by hand, this is done automatically by the **Java Virtual Machine** (JVM) for them. However, the HotSpot JVM itself is written in C++, a memory-unsafe language. A potential memory bug in its garbage collector can quietly corrupt the data it works with, causing many problems.

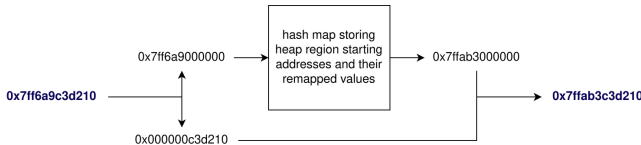
### The Garbage-First (G1) collector

The default JVM garbage collector. It splits its pre-allocated heap into equal **heap regions**. Empty ones wait in a free region list, G1 takes them from there for new objects and returns them when the objects are evacuated.

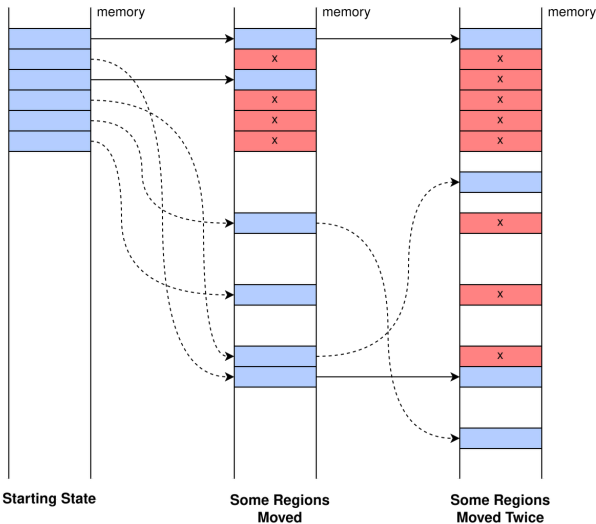
## Moving regions, locking what stays behind

The SanitizeGC works like this:

- 1) Move:** Whenever G1 takes a region from the free list, it gets a freshly reserved memory chunk.
- 2) Lock:** The old memory of the region is uncommitted and the protection of it is set to *PROT\_NONE*. That way, the virtual addresses are inaccessible, but the physical pages are freed.
- 3) Remap:** There are G1 structures that compute and work with offsets from the original heap. An address mapper was created to translate addresses between the original and moved regions at 30+ locations in the JVM code.
- 4) Catch:** A dereference of a dangling pointer to the locked memory will cause an OS fault that the JVM catches and prints a stack trace.



**Address mapper.** Low bits (24 here) are masked off to get the region start, which is looked up in a hash map (one per each direction). The offset is then added back, creating the equivalent original or moved address.



**Continuous region relocation.** Heap regions are blue. Locked memory is red. Every reuse of a region gives it a new address, the virtual addresses of the old places get locked.

## The results

Two **bug seeding** scenarios (introduction of bugs on purpose) were created to show that SanitizeGC works as intended. In both cases, the bugs were caught immediately and not caught at all with SanitizeGC turned off.

However, the added security comes at the cost of **4.5x slower** execution on average (which is why this should be used as a development tool).