

Making Garbage Collection Dependable through a Run-Time Monitor



Chia-Tien Dan Lo

Department of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249
danlo@cs.utsa.edu

Why Study This?

- ✦ GC maintains heap memory
- ✦ What if GC is under DoS attack!
- ✦ It is easy to keep GC busy
- ✦ More memory, more energy and more GC efforts
- ✦ To achieve low power, high performance

Vulnerabilities in GC

✦ Two Scenarios

- Fast Death: A frequent overburdening of the GC that brings the system to a stand still
- Slow Death: A request of additional memory that is not severe enough to throw an “out of memory” error



Example of Heap Memory Attack

- ◆ Byte code instrumentation
 - The class `java.lang.String` in JDK library is modified
- ◆ Re-archive the library
- ◆ Run an application that invokes the modified method



Illustrations of Heap Under Attack

Benchmark	Tower of Hanoi			PhotoAlbum			JBenchmark2		
Type of Attacks	normal	slow	fast	normal	slow	fast	normal	slow	fast
# of allocated objects	40077	44240	44240	4031	4511	4513	63201	71929	5964
# of freed objects	22643	43110	43677	2568	3996	3990	62444	71107	5522
# of garbage collection invocations	6	28	164	8	10	46	47	102	217
Execution time (seconds)	2.05	2.31	2.48	0.5	0.58	-	22.06	23.26	-
Final status	Pass	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Fail

Worse Case Scenarios

- ✦ System is still running with 4.67 times more GC and 13% more execution time.
- ✦ System is dead with 27 times more GC and 21% more execution time.



Anomalous Behavior Detection

- ✦ Run-time metrics inference
 - Allocation rate, number of GC invocation, object life spans, GC pause times, call-site analysis, memory leaks, etc.
 - Minimized the cost of metrics collection through GC profiling
 - Correlate these metrics to memory attacks

Correlation between Metrics and Causes

Metrics	“Fast Death”	“Slow Death”		Faults	
		Generational GC Attack	Power Drain or Real-time Attack	GC Faults	Programming Errors
High allocation rate	X			X	X
Large object size	X			X	
Large number of GC invocations	X	X	X	X	X
Long object life spans		X	X	X	X
Memory leaks				X	
Long GC pause time	X	X	X	X	
Specific call site	X	X	X		
Long execution time	X	X	X	X	

Remarks

- ◆ When the first Cabir worm wriggled into mobile phones in December, 2004, resulted in power drain and DoS, the memory security issues in mobile computing were unveiled.
- ◆ A run-time monitoring system provides an option to defy memory attacks by carefully supervising the program behaviors and identifying abnormal ones.