# CVE-2018-10237

## Description

Unbounded memory allocation in Google Guava 11.0 through 24.1 allows remote attackers to conduct denial of service attacks against servers that depend on this library and deserialize attacker-provided data, because the `AtomicDoubleArray` class (when serialized with Java serialization) and the `CompoundOrdering` class (when serialized with GWT serialization) perform eager allocation without appropriate checks on what a client has sent and whether the data size is reasonable.

## Additional Information

During deserialization, two Guava classes accept a caller-specified size parameter and eagerly allocate an array of that size:

- `AtomicDoubleArray` (when serialized with Java serialization)
- `CompoundOrdering` (when serialized with GWT serialization)

If a server deserializes instances sent by an attacker, the attacker can quickly force the server to allocate all its memory, without even sending the promised number of elements. Note that most servers that accept serialized data will deserialize objects of these types as long as they are on the classpath, even if they are not used by the server. (It is possible to set up an allow- or denylist for Java serialization, but few service owners do. GWT serialization does operate with an allowlist by default, but it is usually a large, automatically generated list that often includes the problem class.)

Guava 24.1.1 and Guava 25.0 eliminate the eager allocation of the arrays. This fixes the vulnerability. (Users who need to maintain compatibility with Java 7 can use the `-android` flavor, described on our the Guava project page.)

Note that it will still be possible for an attacker to send an `AtomicDoubleArray` or `CompoundOrdering` with a large number of items. However, this problem is endemic to serialization. (For example, it's present in `ArrayList`.) Service owners who are concerned about this problem should set a limit on the size of the object graph that their servers will accept. (For Java serialization, see JEP 290, which also permits including and excluding particular classes, useful for defense in depth and as a mitigation if you can't immediately upgrade your version of Guava. For GWT-RPC, consider migrating to another RPC system, as it is deprecated. Aside from migration, we don't know the best practices for GWT-RPC users for addressing the endemic problem.)

Final note for users of old versions of Guava: Guava previously had a batch of similar problems, which were fixed in Guava 19.0.

## Metadata

- Vulnerability Type: Other (Unbounded memory allocation)
- Affected Product Code Base: introduced in 11.0; last present in 24.1; fixed in 24.1.1 and 25.0
- Affected Component: code that depends on Guava and uses Java serialization or GWT-RPC
- Attack Type: Remote
- Impact: Denial of Service
- Attack Vectors: To be affected, a server running Guava must deserialize data sent by an attacker (either Java serialization or GWT-RPC).
- Discoverer: Apostolos Giannakidis

CVE Entry