[JEP 290](http://openjdk.java.net/jeps/290), which allows the filtering of incoming data when deserialising an object, and was initially targeted to Java 9, [has been backported to Java 6, 7, and 8](https://blogs.oracle.com/java-platform-group/entry/incoming_filter_serialization_data_a). The feature provides a mechanism to filter incoming data in an object input stream as it is being processed, and can help prevent deserialisation vulnerabilities like the one that [affected Apache Commons and other libraries](https://www.infoq.com/news/2015/11/commons-exploit) a while back.

Deserialisation of untrusted data is a commonly known risk listed by the [Open Web Application Security Project](https://www.owasp.org/index.php/Deserialization_of_untrusted_data) (OWASP) and the [CERT Oracle Coding Standard for Java](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=27492407) (particularly rules[SER12-J](https://www.securecoding.cert.org/confluence/x/ZwBzCg) and [SER13-J](https://www.securecoding.cert.org/confluence/x/tACzCw)), among others. Software developers are supposed to always check that the incoming data in an ObjectInputStream is valid, however, the existing tools within the JDK didn't always make this easy. JEP 290 changes this by providing a way to filter incoming data without having to extend ObjectInputStream. This is done through several mechanisms, depending on how involved developers need to be.

At a basic level, developers can configure the default [ObjectInputFilter](http://download.java.net/java/jdk9/docs/api/java/io/ObjectInputFilter.Config.html#getSerialFilter--) by editing the system property jdk.serialFilter, or the security property jdk.serialFilter inconf/security/java.properties. These properties accept one or more patterns that can be used to identify classes (using syntax similar to Ant's file patterns), or to set limits in the properties of the objects being deserialised:

// reject deserialisation of any class belonging to untrustedmodule,

// and of any array with more than 500 items in it

jdk.serialFilter=!untrustedmodule/.\*\*;maxarray=500

// white-list classes from package com.myorg.trusted,

// but not necessarily from its subpackages.

jdk.serialFilter=com.myorg.trusted.\*

If more flexibility is needed, developers can specify their own actions and checks implementing their own [ObjectInputFilter](http://download.java.net/java/jdk9/docs/api/java/io/ObjectInputFilter.html) and then applying it to an existing ObjectInputStream using[setObjectInputFilter](http://download.java.net/java/jdk9/docs/api/java/io/ObjectInputStream.html#setObjectInputFilter-java.io.ObjectInputFilter-). An ObjectInputFilter can use the information available at[ObjectInputFilter.FilterInfo](http://download.java.net/java/jdk9/docs/api/java/io/ObjectInputFilter.FilterInfo.html) to decide whether the object currently being deserialised can be accepted, needs to be rejected, or whether the filter doesn't have enough information to make a decision; in the latter case, the custom filter can leave the status as undecided and delegate the decision to another user-defined filter or to the default system one.

Finally, if developers wish to use their own mechanism for all deserialisations, a user-defined filter can be assigned as the system default using [ObjectInputFilter.Config.setSerialFilter](http://download.java.net/java/jdk9/docs/api/java/io/ObjectInputFilter.Config.html#setSerialFilter-java.io.ObjectInputFilter-).

As initially stated, developers won't need to wait until Java 9 to start applying filters to serialisation; JEP 290 is available in Java 8 update 121, Java 7 update 131, and Java 6 update 141.