**SUMMARY**

A concurrency bug in Eclipse 3.5 Platform Debug.

**DETAILS**

Some details can also be found at: https://bugs.eclipse.org/bugs/show\_bug.cgi?id=287834

This bug is due to a data race.

Accessing a hashmap from more than one thread can corrupt its internal structure and it can go into an infinite loop.

|  |  |
| --- | --- |
| Thread1 (JavaLineBreakpoint.java) | Thread2 (JavaLineBreakpoint.java) |
| protected void clearCachedExpressionFor(JDIDebugTarget target) {  removeCachedThreads(fCompiledExpressions, target);  // clean up cached projects for stack frames  Set frames= fProjectsByFrame.keySet();  List framesToRemove= new ArrayList();  Iterator iter= frames.iterator();  JDIStackFrame frame;  while (iter.hasNext()) {  frame= (JDIStackFrame)iter.next();  if (frame.getDebugTarget() == target) {  framesToRemove.add(frame);  }  }  iter= framesToRemove.iterator();  while (iter.hasNext()) {  fProjectsByFrame.remove(iter.next());  }    } | protected void clearCachedExpressionFor(JDIDebugTarget target) {  removeCachedThreads(fCompiledExpressions, target);  // clean up cached projects for stack frames  Set frames= fProjectsByFrame.keySet();  List framesToRemove= new ArrayList();  Iterator iter= frames.iterator();  JDIStackFrame frame;  while (iter.hasNext()) {  frame= (JDIStackFrame)iter.next();  if (frame.getDebugTarget() == target) {  framesToRemove.add(frame);  }  }  iter= framesToRemove.iterator();  while (iter.hasNext()) {  fProjectsByFrame.remove(iter.next());  }    } |
| 0-lock | 0-lock |