(CS51000)
Software Engineering

Instructor: Lin Tan
TA: Yi Sun

November 7, 2019

Part I A (40%)

We grade this sub question on mc18 using automated scripts. A flat bonus of 10 points is given for submitting code that looks relevant. Additional points are awarded for passing the tests. There are 6 call graphs. Each call graph is tested with two sets of support and confidence. Each test is given two minutes. The values of each test run are shown below:

- Call graphs test1, test2 and test3: 1 point per test, totalling 6 points.
- Call graphs test4, test5 and test6: 4 points per test, totalling 24 points.

Part I B (10%)

In this sub question, I expect two reasons for false positives and ten instances of false positives. There should be comments for each false positive, or at least each pair.

Mark distribution:

- +5 points for discussion of false positive pairs.
- +2.5 points for each valid reason.

Some good reasons:

- The missing function in a pair is called in a wrapper function. This reason has many variants, e.g., called in one level up, or one level down.
- There is no relationship between two functions in a pair, such as a C library function. Use caution with this reason. There might be implicit relationships between a C library function and a user function. For example, one need to `memset` the parameters before calling a user function. I expect a case by case justification with this reason.
- Two functions in a pair are related, but performs different tasks that don't have to be called together. For example, different manipulations for the same data structure usually called together, but don't have to be always called together.

Unacceptable reasons:

- Low support and confidence. First, playing with support and confidence has equal pros and cons. It affects false positives as well as true positives. Second, this reason does not explain why a particular pair is a false positive.
- Likely invariant is heuristic, not sound evidence. Yes, this is why we are asking this sub question.
- Two similar reasons with different views/wordings.

Part I C (10%)

This sub question is open ended. From Part I B, we know that there are many false positives. One of the reasons is wrapper functions. If the analysis can search inside other functions to find the missing function in the pair, we can reduce many false positives. One approach is to pre-process the call graph to replace a function with its callees. This approach is natural, but instead of reducing false positives, it introduces many new false positives, too. Another approach is to generate potential bugs as in Part I A, but do a post-processing step that search for the missing functions a few levels up and down. This approach does not introduce new false positives but does not find new bugs either.

Both approaches are acceptable. Any other approach is also acceptable, as long as it is not trivial, and explained in the report. I expect three things:

- Source code submitted: (+3 points)
- Explanations of your implementation. I expect to see the details of your algorithm, instead of just saying "by expanding functions". (+4 points)
- Non-trivial experiments and examples (+3 points). Showing the output, diff with Part I A, and few numbers are not enough. I expect some experiments that concretely answer one question. As suggested by the project description, playing with different numbers of levels is a good experiment.

Part I D (20%)

This sub question is open ended. Mark distribution

- +10 points for submitted code
- +10 points for explaination of your approache. You may get partial credit based on the quality of your writing.

Part II A (10%)

#1 CID 10689 0.5pt

java/org/apache/catalina/ha/session/DeltaSession.java

True Positive. Other methods acquire the diffLock lock before accessing deltaRequest. It could be assigned null after the check occurs.

#2 CID 10654 1pt

java/org/apache/catalina/session/JDBCStore.java

False Positive.preparedRemoveSql could be null (from line 669). If it failes to be assigned, the exception will be caught and close(dbConnection) will be called. preparedLoadSql.close() will be called and a null dereference could take place, but it is covered by a try/catch statement (line 926-930).

#3 CID 10625 1pt

java/org/apache/catalina/ha/deploy/FarmWarDeployer.java

False Positive. If engine is null, at line 162 it will catch the null pointer exception and return, so it is not possible to reach line 174. This problem is actually related with #8. The faulty line is 151, which will always evaluate to false. (null instanceof object always returns false.) The condition should be changed to econtainer == null —— !(econtainer instanceof Engine).

#4 CID 10514 0.5pt

java/org/apache/catalina/tribes/group/ChannelCoordinator.java

False Positive. ChannelCoordinator uses ClusterSender to send messages and not the super class' sendMessage() method.

#5 CID 10507 0.5pt

java/org/apache/tomcat/util/buf/MessageBytes.java

False Positive.When strValue is null, type cannot be T STR. If you didn't notice this, simply answering as true positive and fixing by adding null check is also acceptable.

#6 CID 10453 0.5pt

java/org/apache/jk/common/ChannelUn.java

True Positive.wEnv is checked for null at line 79. In superclass JniHandler.initNative(), wEnv is dereferenced at line 87.

#7 CID 10435 0.5pt

java/org/apache/jasper/compiler/JDTCompiler.java

True Positive. InputStream is opened to check if it can be opened. It is never closed and rely on the garbage collector.

#8 CID 10396 0.5pt

java/org/apache/catalina/ha/deploy/FarmWarDeployer.java

False Positive.This is related to #3.

#9 CID 10395 0.5pt

java/org/apache/catalina/core/StandardWrapper.java

False Positive. Method simply returns a pointer to the object. This doesn't need to be synchronized.

#10 CID 10381 0.5pt

java/org/apache/naming/resources/VirtualDirContext.java

True Positive. File.listFiles() can return null (http://docs.oracle.com/javase/6/docs/api/java/ io/File.html#listFiles()). "Returns null if this abstract pathname does not denote a directory, or if an I/O error occurs."

#11 CID 10351 0.5pt

java/org/apache/catalina/ha/tcp/SimpleTcpCluster.java

True Positive.message is covered for null dereference by an if statement previously in setCluster(), so there must be some belief that it could be null. It is then dereferenced in message.getAddress().

#12 CID 10291 0.5pt

java/org/apache/catalina/ssi/SSIServletExternalResolver.java

False Positive. According to documentation (http://docs.oracle.com/javaee/6/api/javax/servlet/ ServletContext.html#getContextPath()), getContextPath() does not return null. Notice that the bug is warning normContext.getContextPath() can be null, not normContext.

#13 CID 10231 0.5pt

java/org/apache/jk/common/ChannelNioSocket.java

True Positive. ssc is never closed. Stating it as intentional or false positive because of Java's GC is also acceptable.

#14 CID 10176 0.5pt

java/org/apache/catalina/ha/session/SimpleTcpReplicationManager.java

False Positive. line 614 is protected by try/catch (line 570). If you didn't notice this, simply answering as true positive and fixing by adding null check is also acceptable.

#15 CID 10167 0.5pt

java/org/apache/catalina/ha/session/DeltaRequest.java

False Positive. this.actions is initialized as an LinkedList (line 65) and never set to null again. If you didn't notice this, simply answering as true positive and fixing by forwarding the null check is also acceptable.

#16 CID 10148 0.5pt

java/org/apache/jk/server/JkMain.java

True Positive. new FileInputStream passed to java.util.Properties.load(), which does not close it. Stating it as intentional or false positive because of gabage collection is also acceptable.

#17 CID 10102 0.5pt

java/org/apache/catalina/ssi/SSIProcessor.java

True Positive. parseCmd() can return null. (Although line 127 is in try/catch, this try/catch is not designed to catch every exception. So we think this is a true positive.)

#18 CID 10026 0.5pt

java/org/apache/jasper/compiler/JspDocumentParser.java

True Positive. checkPrefixes() dereferences attrs without null check. Line 294 implies attrs could be null, so attrs should be checked for null at line 274.

Part II B (10%)

Total is 10 points.

You should describe the warnings given by Coverity (1 point), its reason (2 points) and fix (2 points) for two bugs (total 10 pts). If no bugs are detected by Coverity, you should explain why (give at least 2 reasons, 5 pts per reason). For those cases in which Coverity only finds one bug, reasons of why not more bugs are also expected; otherwise, 1 point is deducted.