Applying static analysis on real world projects

Informatics 115 – Fall 2022
Professor Iftekhar Ahmed
Homework 4
Assigned: Tuesday, November 1st, 2022
Due: Thursday, November 17th, 2022 at 11:59PM

You need Linux/Unix environment to do this assignment.

This assignment is about running a static analysis tool one real world projects and check to see if the they are useful or may be totally useless. For this purpose we will use Defects4J (https://github.com/rjust/defects4j). Defects4J is a collection of reproducible bugs and a supporting infrastructure containing 835 bugs from the following open-source projects:

Identifier	Project name	Number of bugs	Active bug ids	Deprecated bug ids (*)
Chart	jfreechart	26	1-26	None
Cli	commons-cli	39	1-5,7-40	6
Closure	closure-compiler	174	1-62,64-92,94-176	63,93
Codec	commons-codec	18	1-18	None
Collections	commons-collections	4	25-28	1-24
Compress	commons-compress	47	1-47	None
Csv	commons-csv	16	1-16	None
Gson	gson	18	1-18	None
JacksonCore	jackson-core	26	1-26	None
JacksonDatabind	jackson-databind	112	1-112	None
JacksonXml	jackson-dataformat-xml	6	1-6	None
Jsoup	jsoup	93	1-93	None
JxPath	commons-jxpath	22	1-22	None
Lang	commons-lang	64	1,3-65	2
Math	commons-math	106	1-106	None
Mockito	mockito	38	1-38	None

Follow the steps to install Defects4J (Taken from https://github.com/rjust/defects4j):
1. Read the instructions on https://github.com/rjust/defects4j page very carefully.
2. Complete the setup by following direction (https://github.com/rjust/defects4j#steps-to-set-up-defects4j)

One common mistake is while exporting the path. Find your defects4j folder, then do the export PATH step as described in the instructions (use 'pwd' command to determine the path of defects4j after cd'ing into it). Make sure there are no whitespaces in the PATH.

You might need to do some trouble shooting such as installing clank or installing wget etc.

Example commands for using Defects4J (Taken from https://github.com/rjust/defects4j):
1. Get information for a specific project (commons lang):
    defects4j info -p Lang

Note: the project name used here is the identifier from the defects4j website.
2. Get information for a specific bug (commons lang, bug 1):
    defects4j info -p Lang -b 1

The output, along with other information, would contain something like the following

```
---------------------------------------------------------------------------
List of modified sources:
 - org.apache.commons.lang3.math.NumberUtils
---------------------------------------------------------------------------
```

This is giving you information regarding the file containing the bug. For example this output is referring to the file /tmp/lang_1_buggy/src/main/java/org/apache/commons/lang3/math/NumberUtils.java

In some cases, you might get multiple file names. In that cases pick any one file for analysis and make sure to mention that in the summary.txt file.

Usage details is provided in https://github.com/rjust/defects4j#using-defects4j

defects4j query command can be also used for this purpose. (https://github.com/rjust/defects4j#export-project-specific-metadata)

3. Checkout a buggy source code version (commons lang, bug 1, buggy version):
    defects4j checkout -p Lang -v 1b -w /tmp/lang_1_buggy

Note: The path "/tmp/lang_1_buggy" can be anything of your choice.

Note: If you are trying to check out a fixed version, use **f** instead of **b**
Example: *defects4j checkout -p Lang -v 1f -w /tmp/lang_1_fixed*

4. Change to the working directory, compile sources and tests, and run tests:
    cd /tmp/lang_1_buggy
    defects4j compile
    defects4j test

Note: Some students might get error when they try to compile some buggy versions. Since for this assignment you don't need to compile, ignore this error.

Assignment details:

1. Everyone is assigned a specific bug and a project which you will find in the Bug assignment.csv file. You are not fixing any bugs because defects4j has the fixed version of that bug.
2. Checkout (*defects4j checkout -p ....)* the buggy source code version and fixed source code version of the project and bug assigned to you. **If you are unable to checkout the buggy and fixed versions due to errors, make sure that you can successfully execute all steps for the example bug and example project (Lang, bug id-1). Then send an email to me for assigning a new bug with the proof that you have successfully completed all the steps for the example bug and example project. Proof includes output files generated by running PMD on both buggy and fixed version of source code. You need to get the new bug assignment within one week of releasing the assignment. So make sure to check.**
3. Identify the buggy and fixed source code files and lines. Though defects4j doesn't report exact line numbers, you can infer the faulty lines by comparing buggy and fixed versions of the source code.
4. Download and install PMD (https://pmd.github.io/)
5. Run **PMD** on buggy version.

Example command: ./run.sh pmd -d */tmp/lang_1_buggy* -f csv -rulesets java-basic > bug21report.csv
There are multiple rulesets provided by PMD. Use the java-basic. If you use some other ruleset, make sure to mention it in the summary.txt file.

6. Provide an output file with only the lines that report some warnings for the buggy file.
   a. Name the file using the structure Projectname_bugnumber_buggy. For example, if you are working on Lang and bug 1 then the file name will be: Lang_1_buggy.csv. And the contains of these file should be the warnings related to NumberUtils.java as shown in an early example in this assignment.
7. Run **PMD** on fixed version.
8. Provide an output file with only the lines that report some warnings for the buggy file.
   a. Name the file using the structure Projectname_bugnumber_fixed For example, if you are working on Lang and bug 1 then the file name will be: Lang_1_fixed.csv. And the contains of these file should be the warnings related to NumberUtils.java as shown in an early example in this assignment.
9. Summarize the information for the specific bug assigned to you and write them in Summary.txt
   a. Comparing the fixed version with buggy version (i.e. compare the buggy version of the source code file with fixed version of the source code file) and identify the differences.
   b. Comparing the **PMD** outputs (which statements are flagged by PMD in both version, only in fixed version, only in buggy version, did the tool flag the actual buggy lines etc.) Though defects4j doesn't report exact line numbers, you can infer the faulty lines by comparing buggy and fixed versions of the source code.
10. The summarization categories listed above are not the only ways to think about a bug. **What is your intuition regarding the likelihood of finding the fault using information such as source code, a static analysis tool warnings and multiple test cases**. Write the intuition in Summary.txt with a separate header *Intuition*.

Note: We will be grading based on the successful completion of steps. In addition to that, for the written summary, we will be checking for the attempt of inference, discussing different observations, rather than simply copy pasting the output from the PMD outputs.

Upload Instructions:

Turn in one zip file that contains the following files:
1. Output from PMD named: **Projectname_bugnumber_fixed.csv**
2. Output from PMD named: **Projectname_bugnumber_buggy.csv**
3. Summary.txt

Useful Information:

1. In case you need an older version of JDK, below are some examples.

https://jdk.java.net/java-se-ri/7

https://download.java.net/openjdk/jdk7u75/ri/openjdk-7u75-b13-linux-x64-18_dec_2014.tar.gz

https://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html

Applying static analysis on real world projects

2. I am assuming that you are trying to do this on a server such as Openlab. If you need to change the java version for your local usage in Openlab. Here is an example of how to do it, using java 1.7 as an example. After you have downloaded the proper JDK version, copy the JDK folder to a path inside your home directory and execute the following commands to set your PATH java to 1.7. For step 2 you have to change the path according to your username.

i. bash

ii. export JAVA_HOME="/home/iftekha/jdk1.7.0_80/"

iii. export PATH="$JAVA_HOME/bin:$PATH"

iv. java -version

This should show the java version of 1.7. If not, check if java is executable and update it by the following commands:

v. cd /home/iftekha/jdk1.7.0_80/bin

vi. chmod a+x java

Remember this will not change your Openlab profile to java 1.7 permanently. This will work only for the current shell. So you have to follow steps 1 through 4 every time you login/create a new shell.

3. If you are using Mac and running into issues such as “Couldn't find wget to download dependencies. Please install wget and re-run this script.” Following commands can help:

i. ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)" < /dev/null 2> /dev/null

ii. brew install wget