Effective and Efficient API Misuse Detection via Exception Propagation and Search-based Testing (Artifact Evaluation)

Maria Kechagia[*]
University College London
London, United Kingdom
m.kechagia@ucl.ac.uk

Xavier Devroey
Delft University of Technology
Delft, Netherlands
x.d.m.devroey@tudelft.nl

Annibale Panichella
Delft University of Technology
Delft, Netherlands
a.panichella@tudelft.nl

Georgios Gousios
Delft University of Technology
Delft, Netherlands
g.gousios@tudelft.nl

Arie van Deursen
Delft University of Technology
Delft, Netherlands
arie.vandeursen@tudelft.nl

1 Set-up

Catcher[1] is in the attached Virtual Machine (VM). To use the VM and run Catcher follow the next steps.

1. Download the Oracle VirtualBox.[2]

[*]This work was largely completed at the Delft University of Technology.

[1]Tool of the accepted paper at ISSTA'19 with the title "Effective and Efficient API Misuse Detection via Exception Propagation and Search-based Testing."

[2]https://www.virtualbox.org/

2. Import the virtual machine `Catcher.ova`.[3]

3. Select Catcher on the left of the Oracle VM VirtualBox Manager and press Start (green arrow on top).

4. The VM is an Ubuntu VirtualBox. Use as username: guest and password: guest.

2 Virtual Machine Contents

Log-in on the provided Ubuntu VM and navigate the Documents. There, you will find the Catcher (along with its dependencies) and the examined applications for replication. Note that Catcher uses Soot[4] for the static exception propagation analysis and extends EvoSuite[5] for the search-based testing. Catcher depends on Soot and EvoSuite, which are both provided with the virtual machine.

3 Running a Simple Example

Catcher works in two steps: first, it applies exception propagation on the project under test, and second it synthesizes stack traces and runs the provided extended version of EvoSuite to generate the test cases. In the following, we explain how to run Catcher on one project used during our evaluation: `xwiki-commons-text-10.6`.

3.1 Static Exception Propagation

For applying the exception propagation for one project, open a terminal and go to `Documents/data/apitestgen` on the VM and run the static exception propagation. To do so from the terminal, use the following commands:

```
cd
cd Documents/data/apitestgen
bash evaluation/run-erec.sh
```

[3]If you have previously imported and deleted `Catcher.ova`, do not forget to delete the associated hard-drive in File → Virtual Media Manager. Otherwise VirtualBox will generate an error during the import process.

[4]https://github.com/Sable/soot

[5]http://www.evosuite.org/

3.2 Search-based Software Testing

For the search-based software testing part to work, we transform the data from Soot so that they can be processed by EvoSuite by synthesizing stack traces from the paths identified by the static exception propagation. From `Documents/data/-apitestgen`, execute the following command:

```
bash evaluation/run-synthesizer.sh
```

Then, for generating the test cases, execute the following commands from `Do-cuments/data/apitestgen/evaluation`:

```
cd
cd Documents/data/apitestgen/evaluation
bash run-sbst.sh
```

It will start EvoSuite with the following configuration:

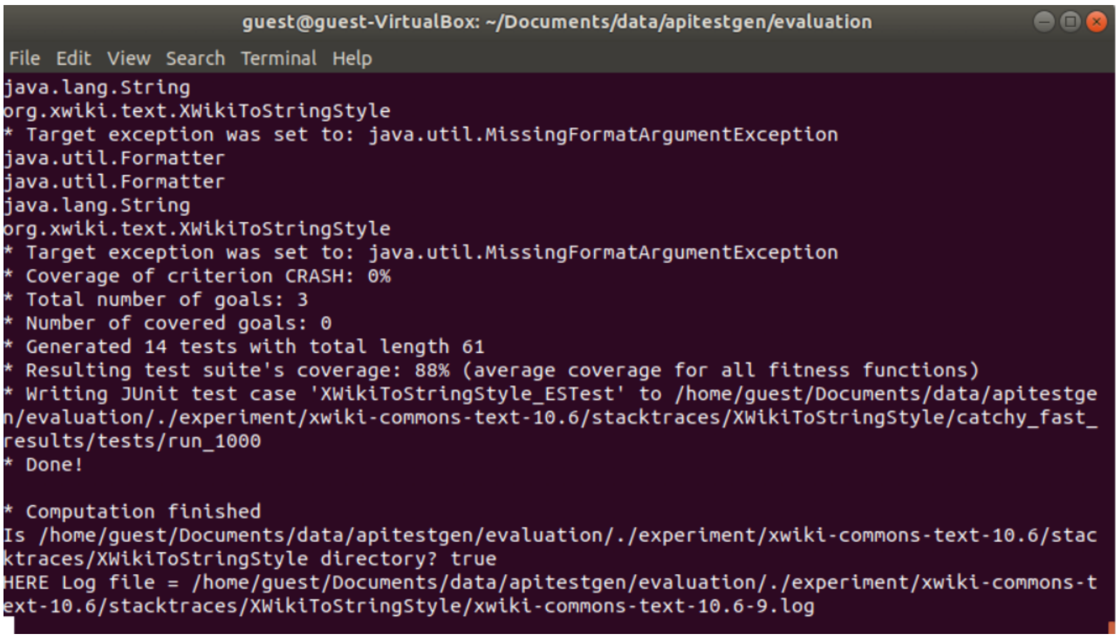
1. 1000 as random seed;
2. 1 run only;
3. 1 thread;
4. and `catchy_fast` as the name of the output folder where the results are generated.

Please note that it takes time to run this part of the analysis. So, if you don't want to wait a full execution of EvoSuite on the example project, when the `Done` message appears (a bit more than 5 minutes after starting EvoSuite) on the screen (see Figure 1), you can stop the evaluation (using `Ctrl+C`). The results (`.csv` files and tests) are generated in `Documents/data/apitestgen/evaluation/experi-ment/xwiki-commons-text-10.6/stacktraces/XWikiToStringStyle/catchy_-fast_results`.

3.3 Test Re-execution

To copy and paste the commands from the following tutorial, please, enable the copy-and-paste feature on the VM: select Catcher on the VirtualBox Manager, go to Settings and select Advanced, and, then, choose Bidirectional for both the Shared Clipboard and Drag-n-Drop fields.

The tests generated by our example may be compiled and run using the following commands (adapted from http://www.evosuite.org/documentation/tutorial-part-1/):



```
guest@guest-VirtualBox: ~/Documents/data/apitestgen/evaluation
File  Edit  View  Search  Terminal  Help
java.lang.String
org.xwiki.text.XWikiToStringStyle
* Target exception was set to: java.util.MissingFormatArgumentException
java.util.Formatter
java.util.Formatter
java.lang.String
org.xwiki.text.XWikiToStringStyle
* Target exception was set to: java.util.MissingFormatArgumentException
* Coverage of criterion CRASH: 0%
* Total number of goals: 3
* Number of covered goals: 0
* Generated 14 tests with total length 61
* Resulting test suite's coverage: 88% (average coverage for all fitness functions)
* Writing JUnit test case 'XWikiToStringStyle_ESTest' to /home/guest/Documents/data/apitestge
n/evaluation/./experiment/xwiki-commons-text-10.6/stacktraces/XWikiToStringStyle/catchy_fast_
results/tests/run_1000
* Done!

* Computation finished
Is /home/guest/Documents/data/apitestgen/evaluation/./experiment/xwiki-commons-text-10.6/stac
ktraces/XWikiToStringStyle directory? true
HERE Log file = /home/guest/Documents/data/apitestgen/evaluation/./experiment/xwiki-commons-t
ext-10.6/stacktraces/XWikiToStringStyle/xwiki-commons-text-10.6-9.log
```

Figure 1: First results finished.

```
cd ~/Documents/data/apitestgen/evaluation
```

1. Get EvoSuite standalone, JUnit, and Hamcrest dependencies.

   ```
   wget https://github.com/EvoSuite/evosuite/releases/download/
   v1.0.6/evosuite-standalone-runtime-1.0.6.jar -O
   tools/evosuite/evosuite-standalone-runtime-1.0.6.jar

   wget https://search.maven.org/remotecontent?filepath=
   junit/junit/4.12/junit-4.12.jar -O tools/junit-4.12.jar

   wget https://search.maven.org/remotecontent?filepath=
   org/hamcrest/hamcrest/2.1/hamcrest-2.1.jar -O
   tools/hamcrest-2.1.jar
   ```

2. Export the class path for compilation and execution of the tests.

   ```
   export CLASSPATH=subjects/xwiki-commons-text-10.6/
   jars/commons-lang3-3.7.jar:subjects/xwiki-commons-text-10.6/
   jars/xwiki-commons-stability-10.6.jar:subjects/
   xwiki-commons-text-10.6/jars/xwiki-commons-text-10.6.jar:
   ```

```
subjects/xwiki-commons-text-10.6/xwiki-commons-text-10.6.jar:
tools/evosuite/evosuite-standalone-runtime-1.0.6.jar:
tools/junit-4.12.jar:tools/hamcrest-2.1.jar
```

3. Compile the tests.

   ```
   javac experiment/xwiki-commons-text-10.6/stacktraces/
   XWikiToStringStyle/catchy_fast_results/
   tests/run_1000/org/xwiki/text/*.java
   ```

4. Execute the tests (since EvoSuite catches stack traces, those are not visible by default, the tests need to be manually updated to see the stack trace. See below.)

   ```
   java -cp $CLASSPATH:experiment/xwiki-commons-text-10.6/
   stacktraces/XWikiToStringStyle/catchy_fast_results/
   tests/run_1000/ org.junit.runner.JUnitCore
   org.xwiki.text.XWikiToStringStyle_ESTest
   ```

5. Please, note that the tests DO NOT print a stack trace as-is. EvoSuite places instructions triggering a runtime exception in a `try-catch` block with a, by default, empty `catch`. To see the stack traces, one could edit the generated tests to print the stack traces.

   - Update the tests to print the stack traces by adding the instruction `e.printStackTrace();` in the catch blocks.
   - In nano, use `CTRL+X` to exit and save the modifications made to the tests.

     ```
     nano experiment/xwiki-commons-text-10.6/stacktraces/
     XWikiToStringStyle/catchy_fast_results/tests/run_1000/
     org/xwiki/text/XWikiToStringStyle_ESTest.java
     ```

   - Recompile and re-execute the tests.

     ```
     javac experiment/xwiki-commons-text-10.6/stacktraces/
     XWikiToStringStyle/catchy_fast_results/tests/
     run_1000/org/xwiki/text/*.java

     java -cp $CLASSPATH:experiment/xwiki-commons-text-10.6/
     stacktraces/XWikiToStringStyle/catchy_fast_results/
     tests/run_1000/ org.junit.runner.JUnitCore
     org.xwiki.text.XWikiToStringStyle_ESTest
     ```

4 Full Evaluation

The subjects used for the evaluation are available in `Documents/data/apitest-gen/evaluation/subjects.zip` and the full results (generated test cases) are available in `Documents/data/apitestgen/evaluation/experiment_all.zip`.

To run Catcher on a particular project, you need to unzip `subjects.zip` located in `Documents/data/apitestgen/evaluation`, copy the project you wish to check and put it into `Documents/data/apitestgen/evaluation/subjects`. Then, you update the paths of the scripts (`run-erec.sh`, `run-synthesizer.sh`, `run-sbst.sh`) located in `Documents/data/apitestgen/evaluation` as well as the paths in the above commands, by replacing the paths related to XWiki with those of another project of your choice.

To run the full evaluation, unzip the `subjects.zip` in `evaluation/subjects` and run the following command: `bash evaluation/runall.sh`

Finally, to processes the generated results (`coverageResults.csv`), please unzip `Documents/data/apitestgen/evaluation/experiment_all.zip` and give the following commands:

1. Navigate to the `experiment_all` folder:

   ```
   cd ~/Documents/data/apitestgen/evaluation/experiment_all
   ```

2. From there, give the command to get a list with all the triggered logs (`files.csv`):

   ```
   bash ../../../../postprocessing/walk.sh
   ```