New Leaf Term Project Report

Chapter 5: Fault Injection

Luke McGuire, Chris Taylor, Kasper Dugaw
College of Charleston
Fall 2020

Introduction

In this chapter of our report we will discuss the design and implementation of 5 faults into the moodle source code, describe the development of a script that will automate the fault injection process, and analyze the results of the testing framework we have developed over the course of this project when these faults have been injected into the moodle source code.

Designing Faults

When designing the faults to test our framework we decided to inject one fault into each of the methods we had written test cases for. The faults are defined in the ./project/faults directory where modified lines are indicated by comments in the format "// FAULT:" and the file that the fault definition should replace is specified on line 1 with a comment in the format "#Destination: ./path/from/moodle". The details for the 5 faults we have defined will be provided below.

fault1.php

./moodle/lib/html2text/Html2Text.php

Line 144 changed so that the class improperly handles html with
 elements.

"'/<(br)[^>]*>[ ]*/i'," to "'/<(brk)[^>]*>[ ]*/i',"

This fault causes the test case with the following id to fail: 8

fault2.php

./moodle/lib/maxmind/GeoIp2/Util.php

Line 23 changed so that the loop executes 1 time less then necessary.

“for ($i = 0; $i < \strlen($ipBytes) && $curPrefix > 0; $i++) {“

to

“for ($i = 0; $i < \strlen($ipBytes)-1 && $curPrefix > 0; $i++) {“

This fault causes the test case with the following id to fail: 12

fault3.php

./moodle/lib/htmlpurifier/HTMLPurifier/UnitConverter.php

Line 194 changed so that negative numbers are improperly handled.

“$n = ltrim($n, '0+-');” to “$n = ltrim($n, '0+');”

This fault causes the test case with the following id to fail: 19

fault4-5.php

./moodle/lib/evalmath/evalmath.class.php

4

Line 436 changed so that expressions with subtraction throw an error.

“} elseif (in_array($token, array('+', '-', '*', '/', '^', '>', '<', '==', '<=', '>='), true)) {“

to

“} elseif (in_array($token, array('+', '+', '*', '/', '^', '>', '<', '==', '<=', '>='), true)) {“

This fault causes the test case with the following id to fail: 2

5

Line 586 changed so that mod is calculated incorrectly.

“return $op1 % $op2;” to “return $op2 % $op1;”

This fault causes the test cases with the following ids to fail: 22, 23, 24

Injecting Faults

To automate the injection of faults into the moodle source code two routines, or actions, were added to the moodleMod script: inject and reset. The inject routine finds all fault definitions in ../project/faults/ and copies them to their destinations in the moodle source code, which are specified in a comment in the first line of each fault definition file. The reset routine cleans the local moodle repository, resetting it to the original state. The routines are executed by running the moodleMod script from the TestAutomation directory as follows:

./scripts/moodleMod.sh inject

./scripts/moodleMod.sh reset

In addition to managing faults, the moodleMod script also implements two other actions to manage the cloned moodle repository: clone and delete. The clone action will clone the appropriate version of moodle from the official moodle repository into ./project/moodle and the delete action will delete the repository.

Testing the Framework

To test our framework we performed the following procedure:

1. Clone the New-Leaf team repository
2. Move into ../New-Leaf/TestAutomation/
3. Clone moodle by executing: ./scripts/moodleMod.sh clone
4. Execute: ./scripts/runAllTests.py
5. Record results without faults
6. Inject faults by executing: ./scripts/moodleMod.sh inject
7. Execute: ./scripts/runAllTests.py
8. Record results with faults
9. Remove faults by executing: ./scripts/moodleMod.sh reset

Initial Results (without fault injection):

15	maxmind_GeoI...	cidr	2acd:ffff::aabb ...	2acd:ffff::aa00 ...	2acd:ffff::aa00 ...	Pass
16	htmlpurifier_U...	getSigFigs	0001.0123	5	5	Pass
17	htmlpurifier_U...	getSigFigs	1.1	2	2	Pass
18	htmlpurifier_U...	getSigFigs	12.304	5	5	Pass
19	htmlpurifier_U...	getSigFigs	-10.01	4	4	Pass
20	htmlpurifier_U...	getSigFigs	000.000	0	0	Pass
21	evalmath	mod	7,7	0	0	Pass
22	evalmath	mod	7,6	1	1	Pass
23	evalmath	mod	100,50	0	0	Pass
24	evalmath	mod	50,60	50	50	Pass
25	evalmath	mod	1,1	0	0	Pass

Summary: 25 / 25 tests passed.

Final Results (with fault injection):

15	maxmind_GeoI...	cidr	2acd:ffff::aabb ...	2acd:ffff::aa00 ...	2acd:ffff::aa00 ...	Pass
16	htmlpurifier_U...	getSigFigs	0001.0123	5	5	Pass
17	htmlpurifier_U...	getSigFigs	1.1	2	2	Pass
18	htmlpurifier_U...	getSigFigs	12.304	5	5	Pass
19	htmlpurifier_U...	getSigFigs	-10.01	4	5	Fail
20	htmlpurifier_U...	getSigFigs	000.000	0	0	Pass
21	evalmath	mod	7,7	0	0	Pass
22	evalmath	mod	7,6	1	6	Fail
23	evalmath	mod	100,50	0	50	Fail
24	evalmath	mod	50,60	50	10	Fail
25	evalmath	mod	1,1	0	0	Pass

Summary: 18 / 25 tests passed.

As expected, all tests passed prior to fault injection, and after fault injection the expected test cases failed.