CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

Security Testing

WS 2023/2024

Prof. Dr. Andreas Zeller
Leon Bettscheider
José Antonio Zamudio Amaya

Exercise 8 (10 Points + 2 Bonus)

Due: 7. January 2024 23:59

> The lecture is based on The Fuzzing Book, an *interactive textbook that allows you to try out code right in your web browser*.
> The Fuzzing Book code is additionally available as a Python pip package. To work on the exercises, please install the package locally:
>
> ```
> pip3 install fuzzingbook
> ```

Submit your solutions as a Zip file on your status page in the CMS.

We will provide you a structure to submit your solutions where each task has a dedicated file. You can add new files and scripts if you want, but you may not delete any provided ones. You can verify whether your submission is valid by executing `verify.py`:

```
python3 verify.py
```

The output provides an overview if a required file, variable, or function is missing and if a function pattern was altered. If you do not follow this structure or change it, we cannot evaluate your submission. A non evaluable exercise will result in 0 points, so make sure to verify your work before submitting it. Note that the script does not reveal if your solutions are correct.

Exercise 8-1: What is Lang and Grey and has to do with Grammars? (5 Points)

In this exercise you will conduct a small fuzzing experiment.

a. Setup Some Fuzzers (3 Points)

Please implement the following functions in **exercise_1a.py**:

- `get_random_fuzzer() -> RandomFuzzer`: This function should return a `RandomFuzzer` instance. You do not need to set any parameters.
- `get_grammar_fuzzer(grammar) -> GrammarFuzzer`: This function should return a `GrammarFuzzer` instance. You do need to provide the given `grammar` as the grammar for the fuzzer.
- `get_mutation_fuzzer(seeds) -> MutationFuzzer`: This function should return a `MutationFuzzer` instance. You do need to set the `seed` parameter to the provided `seeds`.
- `get_greybox_fuzzer(seeds) -> GreyboxFuzzer`: This function should return a `GreyboxFuzzer` instance. You do need to set the `seeds` parameter to the provided `seeds`. Use `Mutator()` as the mutator and `PowerSchedule()` as the schedule.
- `get_lang_fuzzer(seeds, grammar) -> LangFuzzer`: This function should return a `LangFuzzer` instance. You do need to set the `seeds` parameter to the provided `seeds`.

Use `FragmentMutator()` as the mutator, `EarleyParser()` with the given `grammar` as the parser for the mutator, and `PowerSchedule()` as the schedule.

- `get_greybox_grammar_fuzzer(seeds, grammar) -> GreyboxGrammarFuzzer`: This function should return a `GreyboxGrammarFuzzer` instance. You do need to set the `seeds` parameter to the provided `seeds`. Use `Mutator()` as the byte mutator, `RegionMutator()` as the tree mutator, `AFLSmartSchedule()` as the schedule, and `EarleyParser()` with the given `grammar` as the parser for the tree mutator and the schedule.

b. The Experiment (2 Points)

In this exercise you should run the experiment. To accomplish this execute `exercise_1b.py`. You need to install `beautifulsoup4` with `pip` This script may take a while so be patient. The experiment generates two files. **plots.png** will contain a plot of the population coverage the fuzzers achieve and **results_1b.py** contains the data points for the plots. Please include both of these files in your submission.

c. Christmas? (0 Points)

This is a special Christmas exercise. You do not believe us? Take a look at the seeds in the **html** directory. These seeds contain some thrown together Christmas songs. If you want you can try to figure out which songs we used for this.

Exercise 8-2: Quiz (5 Points)

In this exercise we will recap different aspects of greybox fuzzing blackbox fuzzing.

Provide the BEST answers to the following questions in **exercise_2.py** by assigning to each variable `Q1, ..., Q10` the values `1` to `4`.

For instance, if you think the first answer is the BEST answer to Q1, set `Q1=1` in `exercise_2.py`.

There is only **one BEST answer** to each question.

Questions

Q1: In the context of fuzzing, what is a mutation?

1. A modification applied to a program’s source code.
2. A modification applied to a program’s binary.
3. A modification applied to a program’s input.
4. A modification applied to a program’s unit test suite.

Q2: What is the main strategy used by the AFL fuzzer to guide test generation?

1. Favor inputs that have discovered new paths in the program.
2. Favor inputs that have a large number of mutations.
3. Favor inputs that have a small number of mutations.
4. Favor inputs that have never been mutated.

Q3: What is the main difference between Greybox Fuzzing and basic mutation fuzzing?

1. Greybox fuzzing replaces mutations by control-flow graph analysis.
2. Greybox fuzzing creates new mutators during the fuzzing process.
3. Greybox fuzzing removes less useful mutators during the fuzzing process.
4. Greybox fuzzing leverages program feedback such as code coverage.

Q4: What action is performed by the Power Schedules in the AFL fuzzer?

1. Optimize power consumption.
2. Prioritize the inputs in the fuzzer corpus.
3. Prioritize programs to be fuzzed.
4. Prioritize the fuzzer mutators.

Q5: How does the seed population of a greybox fuzzer like AFL change over time?

1. Each seed is used exactly once and then removed from the population.
2. The population is fixed at the beginning of the fuzzing and never changes.
3. The population is expanded with the inputs that exhibit interesting behavior.
4. The population is progressively replaced with randomly generated inputs.

Q6: What is the goal of Directed Greybox Fuzzing?

1. To direct the fuzzing process toward unexplored code locations.
2. To direct the fuzzing process toward smaller inputs.
3. To direct the fuzzing process toward targeted code locations.
4. To direct the fuzzing process toward smaller functions.

Q7: Which of the following made the original LangFuzz fuzzer so powerful?

1. A set of initial seeds with high likelihood of causing failure.
2. A set of initial seeds with high likelihood of being valid.
3. A set of initial seeds with high variability.
4. A set of initial seeds with very low variability.

Q8: Which of the following is an important limitation of the Fragment-Based Mutations?

1. The seeds must be parsed successfully.
2. The seeds must match a single fragment.
3. The grammar must be unambiguous.
4. The fragments must be parsed by the grammar.

Q9: What is the main idea behind Region-Based Fuzzing?

1. Identify regions of the input that cannot be parsed.
2. Try to associate grammar symbols with regions of the input.
3. Distribute mutations over several regions of the input.
4. Focus mutations on a single region of the input.

Q10: What is the idea of Validity-Based Power Schedules?

1. Consider seeds covering more code as having a higher degree of validity.
2. Assign more energy to seeds with a higher degree of validity.
3. Assign more energy to the initial seeds.
4. Assign more energy to seeds that reuse parts of the initial seeds.

Exercise 8-3: A small Christmas Present (2 Bonus)

As it is this special time of the year, you will receive *2* bonus points when submitting this exercise sheet.