CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Security Testing

WS 2021/2022

Prof. Dr. Andreas Zeller
Leon Bettscheider
Marius Smytzek

Exercise 13 (10 Points)

Due: 13. February 2022

> 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 13-1: Delta Boeing (2.5 Points)

In this exercise, we provide you with a subclass of `Runner`, which we ingeniously named `YetAnotherMysteryRunner` ™.

This subclass has a `run` method that fails on certain inputs.

In [1]:
```
from fuzzingbook.Reducer import Runner

class YetAnotherMysteryRunner(Runner):
    def run(self, inp: str):
        this = chr(int(str((0x123 >> 3) + 1), 16))
        that = (3*3*3*3*3)%240
        if inp.count(this) == that:
            return (inp, Runner.FAIL)
        else:
            return (inp, Runner.PASS)
```

In `exercise_1.py`, implement the function `def delta_boeing() -> str:` at the marker

```
# TODO: Implement me
```

Your function `delta_boeing` should return the result of reducing `failing_input` (an instance of a failing input) with the `DeltaDebuggingReducer` class on the `YetAnotherMysteryRunner`.

Exercise 13-2: Quiz (2.5 Points)

In this exercise we will recap the chapter on *Reducing Failure-Inducing Inputs*.

Provide the BEST answers to the following questions in `exercise_2.py` by assigning to each variable `Q1, ..., Q5` 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: What is a goal of a Reducer, as presented in the lecture?

1. Taking a failure-inducing input and simplifying it while still reproducing the failure.
2. Taking a failure-inducing input and simplifying it while still keeping the same code coverage.
3. Taking a failure-inducing input and finding the minimal transformation that makes it valid.
4. Taking a valid input and finding the minimal transformation that makes it fail.

Q2: Why is it particularly useful to reduce an input during debugging?

1. To save memory.
2. To create new failing inputs.
3. To isolate the cause of a failure.
4. To remove failing parts of an input.

Q3: What is an important limitation of the Delta Debugging algorithm?

1. It cannot reduce inputs that are already minimal.
2. It cannot reduce inputs that have a hierarchical structure.
3. It might fail to reduce inputs due to a lack of knowledge of the syntactic constraints.
4. It might fail to reduce inputs that are part of a language with context-sensitive features.

Q4: What is the idea behind Grammar-Based Reduction?

1. Validate candidates generated during Delta Debugging using the grammar.
2. Work with subtrees of the derivation tree instead of substrings.
3. Generate new candidates using the grammar until one reproduces the failure.
4. Remove substrings of the input matching non-terminals of the grammar.

Q5: What is the Depth-Oriented Strategy for Grammar-Based Reduction?

1. Truncate the derivation tree at a given depth and close it with derivations of minimal depth.
2. Don’t consider subtrees that have a depth superior to a given limit.
3. Focus on smaller subtrees by first selecting leaves.
4. Focus on larger tree first by first selecting descendants that are higher in the tree.

Exercise 13-3: Feedback (5 Points)

This is the last exercise of the lecture. :-)

We ask you to provide feedback on the lecture in file `exercise_3.txt`.

We are happy to learn about your suggestions for improvements regarding the lecture, the exercises, the projects, and the presented material.

Your feedback will not be graded based on the expressed thoughts. Any relevant feedback (at least 150 words) will get full points.