PJ2 - 模糊测试（Fuzzing）

软件质量保障与测试 2023-2024春

项目结构

该简易模糊测丝滑项目由一个样例执行入口 main.py 以及如下六个包组成：

fuzzer

该包下目前共有 3 个文件，具体体现为：

1. Fuzzer.py：该文件中的 Fuzzer 类，为所有 Fuzzers 的基类，Fuzzer 是处理输入生成以及 Schedule 调度的工具类
2. GreyBoxFuzzer.py：该文件中的 GreyBoxFuzzer 继承自 Fuzzer 类，其中编写了简易的输入选择，Mutation 调用以及覆盖率、Crashes统计的处理逻辑
3. PathGreyBoxFuzzer.py：该文件中的 PathGreyBoxFuzzer 继承自 GreyBoxFuzzer 类，目前没有编写逻辑，预期实现效果为 **根据 PathSchedule 的调度算法实现逻辑**

runner

该包下目前共有 2 个文件，具体体现为：

1. Runner.py：该文件中的 Runner 类，为所有 Runners 的基类，Runner 是将 input 放入目标程序进行执行的工具类
2. FunctionCoverageRunner.py：该文件中的 FunctionCoverageRunner 继承自 Runner 类，其中编写了简易的函数执行以及获取执行覆盖率的处理逻辑

schedule

该包下目前共有 2 个文件，具体体现为：

1. PowerSchedule.py：该文件中的 PowerSchedule 类，为所有 PowerSchedules 的基类，PowerSchedule 是进行 Seed 选择调度的工具类
2. PathSchedule.py：该文件中的 PathPowerSchedule 继承自 PowerSchedule 类，目前没有编写逻辑，预期实现效果为 **根据 inputs 经过的路径频率动态选择 Seed**

Samples

该包下的 Samples.py 中共有 4 个 Example Programs，你可以通过模糊测试这 4 个样例程序来测试 fuzzer 工具的有效性以及工作效率

corpus

该包下共有 4 个二进制文件，分别用作四个 Sample Program 的初始 Seed 输入

utils

该包下目前共有 3 个文件，具体体现为：

1. Coverage.py：该文件中的 Coverage 类是统计覆盖率信息的工具类
2. Seed.py：该文件中的 Seed 类，存储了每个 Seed 的具体信息
3. Mutator.py：给文件中的 Mutator 类是具体执行 Mutate 的工具类，目前没有编写具体的 Mutation 逻辑
4. ObjectUtils.py：该文件中包含 Dump 对象、Load 对象、计算对象 MD5 的工具函数

工程需求：

- 在 Mutator.py 中将未完成的变异逻辑补充完整，以达成 Fuzzing 的效果
- 完善 PathSchedule.py 以及 PathGreyBoxFuzzer.py，以完整实现 **根据 inputs 经过的路径频率动态选择 Seed**
- 添加更多可行的 Schedules 并放置于 schedule 包中
- 利用 ObjectUtils 对 Object 的操作实现将 Seed 持久化进入文件系统中的功能，防止内存占用过高

样例执行入口：

```
import os
import time

from fuzzer.PathGreyBoxFuzzer import PathGreyBoxFuzzer
from runner.FunctionCoverageRunner import FunctionCoverageRunner
from schedule.PathPowerSchedule import PathPowerSchedule
from samples.Samples import sample1, sample2, sample3, sample4
from utils.ObjectUtils import dump_object, load_object


class Result:
    def __init__(self, coverage, crashes, start_time, end_time):
        self.covered_line = coverage
        self.crashes = crashes
        self.start_time = start_time
        self.end_time = end_time

    def __str__(self):
        return "Covered Lines: " + str(self.covered_line) + ", Crashes Num: " +
str(self.crashes) + ", Start Time: " + str(self.start_time) + ", End Time: " +
str(self.end_time)



if __name__ == "__main__":
    # 构建相应程序的 Runner 对象
    f_runner = FunctionCoverageRunner(sample1)

    # 从本地语料库中读取 Seeds 并构建 Fuzzer
    seeds = load_object("corpus/corpus_1")
    grey_fuzzer = PathGreyBoxFuzzer(seeds=seeds, schedule=PathPowerSchedule(5),
is_print=True)
```

```
    # 记录开始时间
    start_time = time.time()

    # 使用 Runner 执行 Fuzzer 中的输入，并指定运行时间(s)
    grey_fuzzer.runs(f_runner, run_time=60)

    # 将 Coverage 与 Crash 的信息导出
    res = Result(grey_fuzzer.covered_line, set(grey_fuzzer.crash_map.values()),
start_time, time.time())
    dump_object("_result" + os.sep + "Sample-1.pkl", res)

    # 查看本次 fuzzing 的执行信息
    print(load_object("_result" + os.sep + "Sample-1.pkl"))
```