Санкт-Петербургский государственный университет
Кафедра системного программирования

Гибридная генерация модульных тестов

Виктор Алексеевич Карасев, 20.Б11-мм

Научный руководитель: М.П.Костицын, инженер-исследователь кафедры системного программирования

Санкт-Петербург
2022

Генерация тестов

- «Черный ящик»
- «Белый ящик»
- «Серый ящик»

Сравнение

Символьное исполнение:

+ Объем покрытия
+ Небольшое количество тестов

Фаззинг:

Сравнение

Символьное исполнение:

- \+ Объем покрытия
- \+ Небольшое количество тестов
- – Скорость
- – Внешние вызовы, взрыв путей

Фаззинг:

Сравнение

Символьное исполнение:

- \+ Объем покрытия
- \+ Небольшое количество тестов
- – Скорость
- – Внешние вызовы, взрыв путей

Фаззинг:

- \+ Скорость
- \+ Внешние вызовы, взрыв путей

Сравнение

Символьное исполнение:

+ Объем покрытия
+ Небольшое количество тестов
– Скорость
– Внешние вызовы, взрыв путей

Фаззинг:

+ Скорость
+ Внешние вызовы, взрыв путей
– Объем покрытия
– Большое количество тестов

Сравнение

Символьное исполнение:

- \+ Объем покрытия
- \+ Небольшое количество тестов
- – Скорость
- – Внешние вызовы, взрыв путей

Фаззинг:

- \+ Скорость
- \+ Внешние вызовы, взрыв путей
- – Объем покрытия
- – Большое количество тестов

Серый ящик:

- \+ Скорость
- \+ Объем покрытия
- \+ Количество тестов

Генерация тестов в .NET

Фаззеры:

- FsCheck
- SharpFuzz

Генерация тестов в .NET

Фаззеры:

- FsCheck
- SharpFuzz

Символьное исполнение:

- V#
- Pex

Генерация тестов в .NET

Фаззеры:

- FsCheck
- SharpFuzz

Символьное исполнение:

- V#
- Pex

Гибридные техники:

- V#

Генерация тестов в .NET

Фаззеры:

- FsCheck
- SharpFuzz

Символьное исполнение:

- V#
- Pex

Гибридные техники:

- V#
- Небезопасно и неэффективно

Цель и задачи

Целью учебной практики является создание системы взаимодействия фаззинга и символьного исполнения

Задачи:

- Выполнить обзор средств изоляции исполняемого кода;
- Разработать архитектуру системы взаимодействия фаззера и символьного исполнеия;
- Реализовать созданную модель взаимодействия в символьной виртуальной машине $\mathrm{V\#}$;
- Провести тестирование полученного решения.

Обзор средств изоляции

- **Harmony** – «патчинг» библиотек во время исполнения

Обзор средств изоляции

- **Harmony** – «патчинг» библиотек во время исполнения
- **Code Access Security** – правила доступа

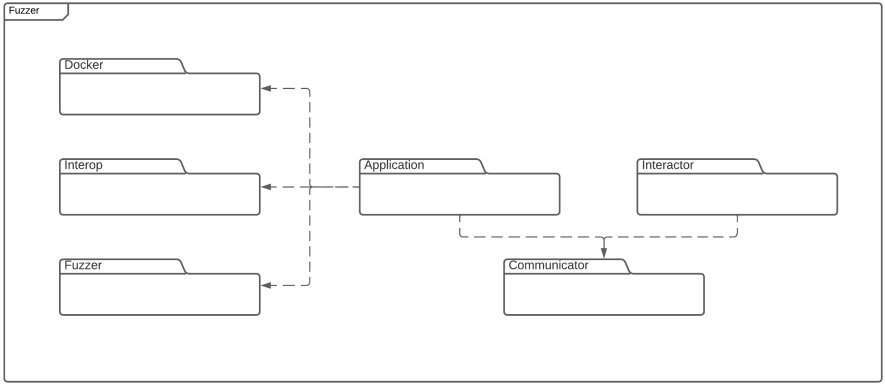
Обзор средств изоляции

- **Harmony** – «патчинг» библиотек во время исполнения
- **Code Access Security** – правила доступа
- **Unbreakable** – белый список библиотек

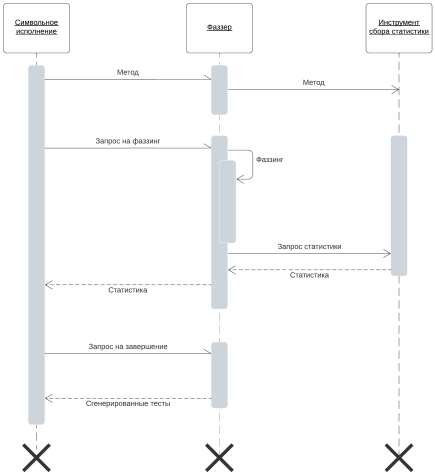
Обзор средств изоляции

- **Harmony** – «патчинг» библиотек во время исполнения
- **Code Access Security** – правила доступа
- **Unbreakable** – белый список библиотек
- **Docker** – контейнеризация

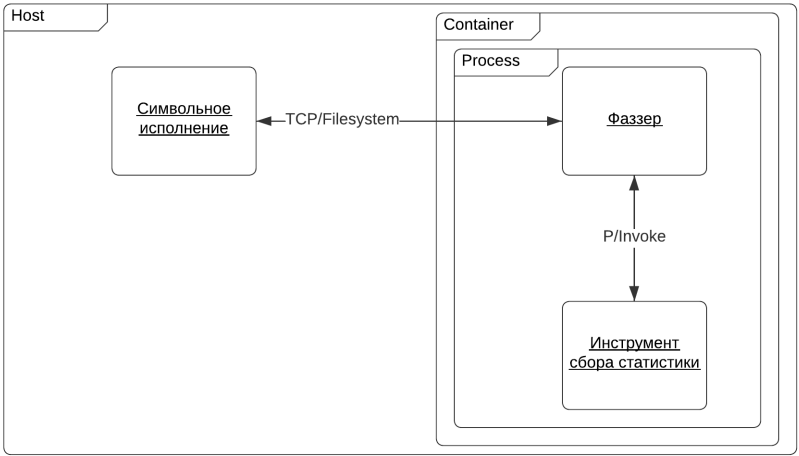
Архитектура



Архитектура



Особенности реализация



Тестирование

Название теста	Покрытие «Symbolic» (%)	Покрытие «Hybrid» (%)
OnlyExternalCall	0	100
ExternalCallOneBranch	67	100
ExternalCallManyBranches	60	100
ExternalCallReadWrite	0	100
NativeRegexp	0	100
ForLoop	100	100
NestedForLoop	100	100
MutualRec	100	100
Fibonacci	0	100
Cosmos	23	26
Lifetimes.Utils	12	19

Результаты

- Выполнен обзор следующих средств изоляции кода: Harmony, Code Access Security, Unbreakable, Docker;
- Разработана модель системы взаимодействия фаззера и символьного исполнения;
- Разработана архитектура системы взаимодействия фаззера и символьного исполнения фаззера для платформы .NET, содержащая две основные компоненты: Fuzzer.Interactor и Fuzzer.Application ;
- Созданная модель реализована в рамках проекта V#;
- Проведено тестирование полученного решения.

Ссылка на репозиторий: https://github.com/KarasssDev/VSharp/tree/Fuzzer-v5