УДК 004.056

РЕАЛИЗАЦИЯ ПРОТОТИПА МЕТОДА ГЕНЕРАЦИИ ФАЗЗИНГ-ОБЕРТОК ДЛЯ JAVA-ПРИЛОЖЕНИЙ НА ОСНОВАНИИ АНАЛИЗА МОДУЛЬНЫХ ТЕСТОВ

Насонов А. Ю.[1]
Научный руководитель – канд. техн. наук, декан ФБИТ, Менщиков А. А. [1]
[1]Университет ИТМО
arteom.nasonov@yandex.ru

Работа выполнена в рамках темы НИР №3 «Реализация прототипа метода генерации фаззинг-оберток для Java-приложений на основании анализа модульных тестов».

Введение

Фаззинг является методом динамического анализа программ, заключающийся в подаче исследуемому программному компоненту неожиданных, возможно некорректных сгенерированных данных в попытке вызвать нежелательное поведение программы – падения, зависания, потребление большого количества памяти и другие [1]. Для передачи сгенерированных фаззером данных программе используют дополнительную программу, называемую фаззинг-оберткой [2]. Традиционно написание фаззинг-оберток происходит преимущественно вручную инженерами по фаззинг-тестированию, что занимает длительное время.

Основная часть

Для решения проблемы трудоёмкого написания фаззинг оберток в данной работе предлагается прототип метода автоматической генерации фаззинг-оберток для Java-приложений на основании анализа существующих модульных тестов. Предлагаемый метод основывается на:

1. статическом анализе кода модульных тестов для определения потенциально подходящих для мутаций объектов [3];
2. подтверждении или опровержении пригодности объектов для мутации на основании анализа статистических характеристик непродолжительного запуска фаззинга;
3. синтезе обертки, мутирующей наиболее выгодные для фаззинга объекты.

Реализация метода выполнена в виде Maven-плагина, состоящего из нескольких модулей, таких как модуль разбора исходного кода, модуль синтеза оберток, модуль запуска и модуль анализа.

Вопрос пригодности объектов для мутации предлагается решать следующим образом: составить набор данных, включающий артефакты выполнения непродолжительного фаззинга и продолжительного фаззинга для каждого объекта. Затем вручную разметить каждый объект как пригодный или непригодный для мутаций, основываясь на артефактах продолжительного запуска. Далее провести анализ артефактов краткосрочного прогона для выявления параметров, статистически значимо коррелирующих с итоговой эффективностью фаззинга.

Для проведения эксперимента в рамках данной работы были отобраны репозитории: парсер FastJSON [4] и парсер табличных данных Apache Commons CSV [5]. В этих репозиториях были определены 29 тестовых классов, на основе которых были автоматически сгенерированы и запущены 113 оберток. Все обертки успешно прошли этап компиляции, а 95% из них были корректно выполнены фаззером Jazzer [6].

Также одним из значимых результатов стало обнаружение нескольких программных недостатков в компоненте FastJSON. Для одной уязвимости автором данной работы было подготовленно исправление и принято разработчиками проекта [7].

Другие программные недостатки были описаны в системе контроля ошибок данного проекта и впоследствии исправлены разработчиками.

Выводы

В рамках данной работы был реализован прототип метода генерации фаззинг-оберток для Java-приложений на основании анализа модульных тестов. Подготовлена программная реализация предлагаемого метода, а также инфраструктура для выявления параметров оценки непродолжительного прогона фаззинг-тестирования. Помимо этого, была обнаружена уязвимость и ряд других программных недостатков в компоненте FastJSON, которые были позже исправлены.

Литература

1. Ерышов В. Г., Кабанец А. Г. ФАЗЗИНГ-ТЕСТИРОВАНИЕ. СОВРЕМЕННЫЕ СРЕДСТВА ФАЗЗИНГА //Обработка, передача и защита информации в компьютерных системах'22. – 2022. – С. 200-204.
2. Чан Ти Тхиен. Разработка нового метода автоматизированного тестирования программных библиотек : Дис. … канд. техн. наук: 2.3.5 / Чан Т. Т. – Москва, 2023. – 114 с.
3. Зыбин Р. С. и др. Технология Azov автоматизации массового создания тестов работоспособности //Труды Института системного программирования РАН. – 2008. – Т. 14. – №. 2. – С. 83-108.
4. Alibaba fastjson2 [Электронный ресурс] / Alibaba Group. – URL: https://github.com/alibaba/fastjson2 (дата обращения: 18.02.2026).
5. Apache Commons CSV [Электронный ресурс] / The Apache Software Foundation. – URL: https://github.com/apache/commons-csv (дата обращения: 18.02.2026).
6. Jazzer [Электронный ресурс] / Code Intelligence. – URL: https://github.com/CodeIntelligenceTesting/jazzer (дата обращения: 18.02.2026).
7. Pull request #3884 [Электронный ресурс] / Alibaba fastjson2. – URL: https://github.com/alibaba/fastjson2/pull/3884 (дата обращения: 18.02.2026).