

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Казанский (Приволжский) Федеральный Университет»**

**Институт вычислительной математики и информационных технологий
Кафедра системного анализа и информационных технологий**

Направление подготовки: 02.03.02 – Фундаментальная информатика и
информационные технологии

Профиль: Системный анализ и информационные технологии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**СИСТЕМА ФАЗЗИНГА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
НА ОСНОВЕ ЭВОЛЮЦИОННОГО ПОДХОДА**

Обучающийся 4 курса
группы 09-931

(Редькин В.С.)

Руководитель
ст. преподаватель

(Долгов Д.А.)

Заведующий кафедрой системного анализа
и информационных технологий
д-р техн. наук, профессор

(Латыпов Р.Х.)

Казань – 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
1 Генетические алгоритмы	4
2 Трассировка	5
2.1 Статическая инструментация	5
2.2 Динамическая инструментация	6
ЗАКЛЮЧЕНИЕ	7
СПИСОК ЛИТЕРАТУРЫ	8

ВВЕДЕНИЕ

Текст введения пишется вот здесь. Потом самое главное не забыть заменить этот образцовый текст перед демонстрацией научруку.

Абзац 2. Здесь начинается второй абзац введения. Потом самое главное не забыть заменить этот образцовый текст перед демонстрацией научруку.

1) вот так вот

2) будут выглядеть элементы списка

– а вот такими станут

– элементы маркированного списка. При этом стоит заметить, что текст по-прежнему выровнен по ширине и сохраняет отступы как в обычных абзацах

Цитата чего-нибудь выглядит вот так [1]

1. Генетические алгоритмы

2. Трассировка

Важным компонентом, значительно ускоряющим процесс фаззинга, является измерение покрытия кода программы при запуске очередного тестового примера. Существует несколько подходов для измерения покрытия, они будут рассмотрены далее.

2.1. Статическая инструментация

Статическая инструментация программы, полагающаяся на применение специальных библиотек и компиляторов, добавляющих в программу инструкции, на которые затем ориентируется фаззер для точного выяснения траектории выполнения программы.

Плюсом такого подхода является быстрота проведения фаззинга (например, в программе может быть искусственно выделена та или иная секция, подвергаемая тестированию в бесконечном цикле, за счёт чего отпадает необходимость в трате ресурсов на постоянный запуск новых процессов и загрузки библиотек).

Минус данного подхода состоит в необходимости наличия доступа к исходному коду программы и необходимости дополнительной работы, заключающейся в подключении специальных заголовочных файлов, выделении тестируемых участков программы, а также компиляции при помощи специальных инструментов.

Одним из фаззеров, использующих статическую инструментацию, является American fuzzy lop, или коротко afl [2]. Данная инструмент предоставляет большой набор подходов, позволяющих сделать фаззинг быстрее и эффективнее:

- afl-gcc – специальный компилятор, предназначенный для генерации исполняемых файлов с дополнительной инструментацией, используемой фаззером. Помимо прочего, afl-gcc может производить дополнительное мероприятия по ”укреплению” (hardening) исполняемых файлов, что

позволяет более эффективно обнаруживать ошибки в работе с памятью;

– afl-trim

2.2. Динамическая инструментация

Динамическая инструментация программы полагается на использование методов, схожих с таковыми, применяемыми в отладчиках - для сбора информации о траектории выполнения программы применяются точки останова, в которых записывается состояние регистра счётчика команд. В отличие от предыдущего подхода, в данном случае возможна работа с уже готовым исполняемым файлом, мы можем и не иметь исходного кода исследуемой программы.

Проблемой динамической инструментации является серьёзное влияние на скорость выполнения программы, вызванное необходимостью обрабатывать большое число прерываний и системных вызовов при общении между исследуемой программой и программой-трассировщиком, из-за чего время выполнения увеличивается пропорционально числу попадания указателя инструкций на точку останова.

Для снижения этого влияния могут применяться различные методы, например Coverage-guided tracing [1]. Данный подход предлагает вместо создающего серьёзную вычислительную нагрузку полного отслеживания траектории выполнения выявлять только факт посещения новых, ранее не обследованных участков программы. В данном случае мы исходим из предположения, что львиная доля тестовых примеров не вносит вклада в обнаружение новых участков программы, а вместо этого проходит по уже известным путям, и процент таких примеров по мере исследования программы увеличивается, а вероятность обнаружить непосещённый участок снижается.

ЗАКЛЮЧЕНИЕ

Текст нашего умного заключения будет написан вот тут.

СПИСОК ЛИТЕРАТУРЫ

1) Nagy S., Hicks M. Full-Speed Fuzzing: Reducing Fuzzing Overhead through Coverage-Guided Tracing // 2019 IEEE Symposium on Security and Privacy (SP). — 2019. — 787—802.

2) American fuzzy lop (2.52b). — <https://lcamtuf.coredump.cx/afl/> (дата обр. 27.02.2023).