Санкт-Петербургский государственный университет

Кафедра системного программирования

Динамическая бинарная трансляция в RISC-V с помощью Instrew

Михайлов Илья Игоревич, группа 21.Б07-мм

Научный руководитель: Д.С. Косарев, ассистент кафедры системного программирования

Санкт-Петербург
2024

Введение

- Бинарные трансляторы являются инструментами для анализа кода, профиляции и эмуляции
- Важным классом которых являются динамические бинарные трансляторы (далее — ДБТ)
- В качестве промежуточного представления для ДБТ хорошо использовать LLVM IR
  - Тулчейн LLVM позволяет проводить качественные оптимизации
- Хотим ДБТ для RISC-V

Обзор

- Динамические бинарные трансляторы:
  - QEMU
  - Box64
  - Rosetta 2
  - Valgrind
- Динамические бинарные трансляторы с поднятием в LLVM IR:
  - McSema + Remill[1]
  - Instrew + Rellume
  - DBILL, HQEMU
- Выводы:
  - Instrew работает быстрее, чем QEMU и Valgrind, а так же имеет API для добавления инструментов бинарного анализа[2]
  - Из всех рассмотренных ДБТ с поднятием в LLVM IR Instrew + Rellume поддерживается контрибъютором и уже имеет guest-поддержку RISC-V

[1] https://se.math.spbu.ru/thesis/texts/Frolov_Timofej_Sergeevich_Spring_practice_2nd

[2] https://mediatum.ub.tum.de/doc/1614897/1614897.pdf

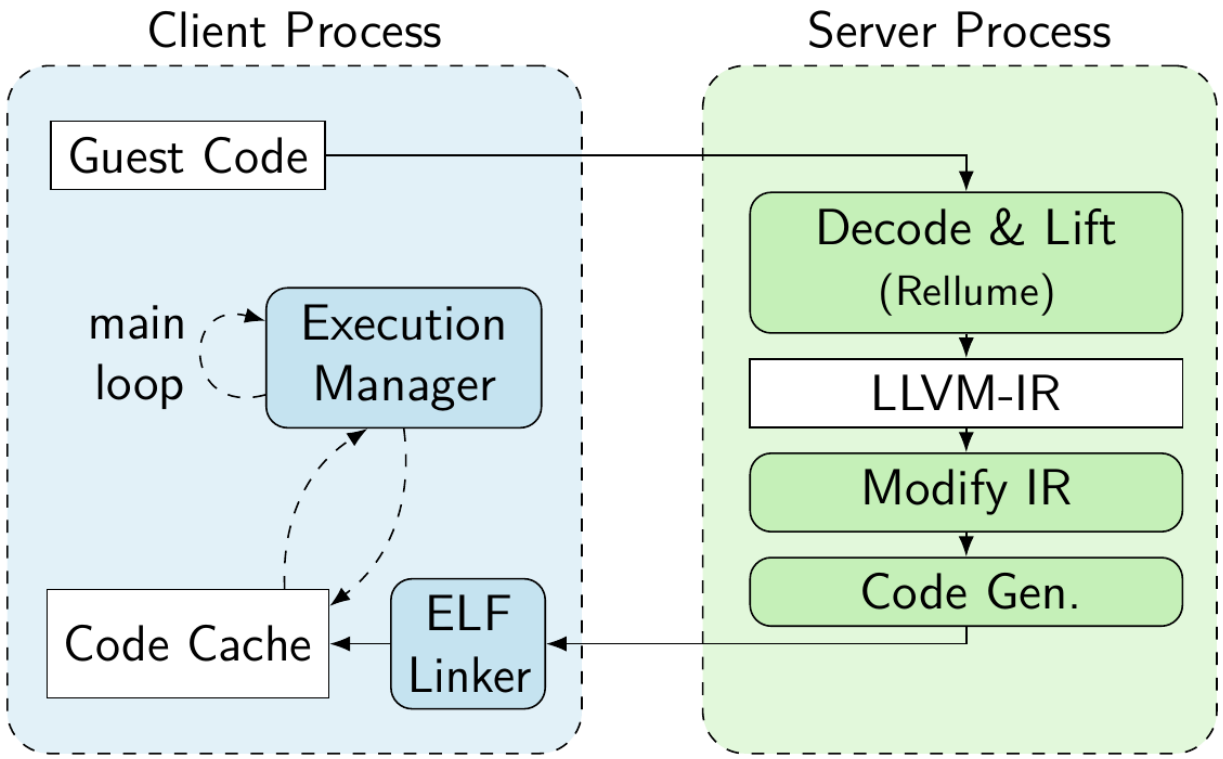
Постановка задачи

Цель: Добавить поддержку host-архитектуры RISC-V для ДБТ Instrew

Задачи:

1. Выполнить обзор архитектуры Instrew
2. Реализовать минимальную функциональность для запуска Instrew на RISC-V:
   - Скомпилировать, добавив процессорно-специфические патчи и проставив константы
   - Реализовать функции, связанные с RISC-V ABI
3. Протестировать на простых программах

Архитектура Instrew



[3]

- В данной практике была проведена работа над клиентом Instrew

[3] https://mediatum.ub.tum.de/doc/1614897/1614897.pdf

Реализация (1/2)

- Чтобы скомпилировать, необходимо:
  - Проставить константы
  - Сделать простой header на ассемблере для входа в программу, после execve
- Оказалось, что не все псевдоинструкции описаны в документации RISC-V[4]

[4] https://reviews.llvm.org/D49661

Особенности Instrew

- Флаги сборки клиента Instrew:
  - Клиент во время запуска выдаёт ошибку сегментации
  - Утилита pmap показывает, что к клиенту динамически линкуется ld-linux и выполнить релокации невозможно
  - Оказалось, что –static-pie флаг у gcc-13 на x86_64 и RISC-V отличается семантикой
- Динамические релокации:
  - Так как instrew-client собирается и линкуется без libc, ему нужно самому обработать свои релокации
  - Клиент загадочно падет на какой-то релокации
  - Оказалось, что над объявлением функции memset стоит атрибут weak и из-за этого нужно реализовывать другую релокацию
- Некоторый вывод:
  - Из-за специфических флагов сборки и линковки сложно диагностировать ошибки сегментации
  - Asan и gprof здесь не работают, Valgrind не помог
  - Остается только printf, gdb

Реализация (2/2)

- После успешного запуска клиента Instrew нужно:
  - Реализовать эмуляцию системных вызовов для RISC-V
  - Проставить джампы в PLT таблице
  - Реализовать некоторые релокации для запуска просто примера
- Оказалось, что:
  - Сервер запускает клиент Instrew через fork и fexecve
    - ★ Возникли сложности с диагностированием с помощью gdb
    - ★ follow-exec-mode не работал
    - ★ Отдельно запуская клиент в gdb, не подгружались символы
    - ★ Через некоторое время был найден флаг Instrew, который запускал клиент через execve
  - Диагностировать ошибки непросто:
    - ★ При неправильной подстановке опкода/адреса, нужно диагностировать память с помощью gdb
    - ★ Без RISC-V декодера не обойтись
- Небольшой итог:
  - Реализовывать эмуляцию системных вызовов и функции RISC-V ABI трудоёмко

Тестирование

- Удалось успешно протестировать:
  - Программах без libc, исполняющих системные вызовы
  - Факториале и фибоначчи, выводящих результат в exit code
  - Факториале с минимальным рантаймом, который выводит результат на экран
- Не удалось протестировать на:
  - Динамечески слинкованных программах и программах использующих libc
  - Программах, с флагами сборки -pie
  - Программах, где не конкретная релокация еще не реализована
- Вывод:
  - Работают статически слинкованные программы, с простым рантаймом и системными вызовами

Заключение

В результате работы были выполнены следующие задачи:

1. Выполнен обзор архитектуры Instrew
2. Реализованы процессорно-специфические патчи
3. Реализована эмуляция системных вызовов, PLT таблица и минимальный набор ELF релокаций
4. Instrew был протестирован на статически слинкованных программах, с простым рантаймом