



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

- Бинарные трансляторы являются инструментами для анализа кода, профилиции 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

- Динамические бинарные трансляторы:
 - ▶ QEMU-user
 - ▶ Rosetta 2
 - ▶ Valgrind
 - ▶ Box64
- Динамические бинарные трансляторы с поднятием в LLVM IR:
 - ▶ Instrew + Rellume
 - ▶ BinRec, HQEMU (не обновлялись с 2022 и 2018 года соответственно)
 - ▶ DBILL, LnQ, Lasagne (есть только статьи, либо код закрыт)

- Выводы:

- ▶ Instrew производительнее, чем QEMU и Valgrind (основываясь на бенчмарках¹)
- ▶ Instrew "из коробки" имеет API для добавления инструментов бинарного анализа
- ▶ Из всех рассмотренных ДБТ с поднятием в LLVM IR Instrew + Rellume поддерживается мейнтейнером
- ▶ Instrew уже имеет guest-поддержку RISC-V

¹<https://mediatum.ub.tum.de/doc/1614897/1614897.pdf>

Постановка задачи

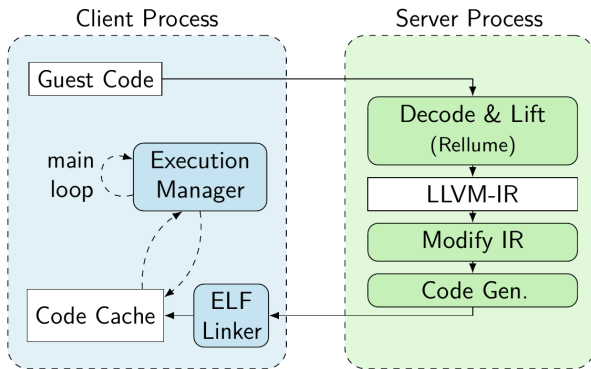
Цель: Добавить поддержку host-архитектуры RISC-V для ДБТ Instrew

Задачи:

- ❶ Выполнить обзор архитектуры Instrew
- ❷ Реализовать минимальную функциональность для запуска Instrew на RISC-V:
 - ▶ Скомпилировать, добавив процессорно-специфические патчи и проставив константы
 - ▶ Реализовать функции, связанные с RISC-V ABI
- ❸ Протестировать на простых программах

Архитектура Instrew

- Является клиент-серверной
- Работа была проведена над клиентом Instrew



2

² диаграмма была взята отсюда

Реализация (1/2)

- Чтобы скомпилировать, необходимо:
 - ▶ Проставить константы
 - ▶ Написать на ассемблере точку входа в программу
- Оказалось, что не все псевдоинструкции описаны в документации RISC-V³

³<https://reviews.llvm.org/D49661>

Особенности Instrew (1/2)

- Клиент Instrew довольно таки непросто диагностировать
- Флаги сборки клиента Instrew:
 - ▶ Клиент во время запуска выдаёт ошибку сегментации
 - ▶ Asan и gprof здесь не работают
 - ▶ После локализации ошибки, стало понятно, что к клиенту динамически линкуется ld-linux, чего быть не должно
 - ▶ Оказалось, что `-static-pie` флаг у gcc-13 на x86_64 и RISC-V отличается семантикой
- Динамические релокации и атрибуты:
 - ▶ 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

Особенности Instrew (2/2)

- Сервер запускает клиент Instrew через fork и fexecve
 - ▶ follow-exes-mode не работал
 - ▶ Отдельно запуская клиент в gdb, не подгружались символы
 - ▶ Через некоторое время был найден флаг Instrew, который запускал клиент через exesve
- Некоторый вывод:
 - ▶ Из-за специфических флагов сборки и линковки сложно диагностировать ошибки сегментации
 - ▶ Остается только gdb, Valgrind

- После успешного запуска клиента Instrew нужно:
 - ▶ Реализовать эмуляцию системных вызовов для RISC-V
 - ▶ Проставить трамплины в PLT таблице
 - ▶ Реализовать некоторые релокации для запуска просто примера
- Диагностировать ошибки непросто:
 - ▶ При неправильной подстановке опкода/адреса, нужно диагностировать память с помощью gdb
 - ▶ Без RISC-V декодера не обойтись

- 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:
 - ▶ Динамически линкованных программах и программах использующих libc
 - ▶ Программах, с флагами сборки -pie
 - ▶ Программах, где конкретная релокация еще не реализована
- Вывод:
 - ▶ Работают статически линкованные программы, с простым рантаймом и системными вызовами

В результате работы были выполнены следующие задачи:

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