

ТЕХНОЛОГИИ В ОБРАЗОВАНИИ УНИВЕРСИТЕТ

МИКРОЭЛЕКТРОНИКА
ИННОВАЦИИ
КАТАЛИТИЧЕСКИЕ
МАТЕРИАЛЫ

ДИЗАЙН
ЛЕКАРСТВ

НАУЧНАЯ
ЛАБОРАТОРИЯ
ГЕОХИМИЯ
ИНЖИНИРИНГ

ГЕОФИЗИКА
ГИБРИДНЫЕ
МАТЕРИАЛЫ

ЭНЕРГОСБЕРЕЖЕНИЕ

ВЫСОКИЕ
ЭНЕРГИИ

БИОТЕХНОЛОГИИ
МОДЕЛИРОВАНИЕ
НАНОТЕХНОЛОГИИ

СЕМИОТИКА

НАУКА

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

ЭЛЕМЕНТАРНЫЕ
ЧАСТИЦЫ
ГЕОЛОГИЯ

КВАНТОВЫЕ
ТЕХНОЛОГИИ
БИОЛОГИЯ

ТЕМНАЯ
МАТЕРИЯ
ФОТОНИКА

БИОМЕДИЦИНА
ПРИКЛАДНЫЕ
ИССЛЕДОВАНИЯ
РАЗВИТИЕ
АСТРОНОМИЯ
ГЛОБАЛЬНЫЕ ПРИОРИТЕТЫ

АСТРОФИЗИКА
БИОИНФОРМАТИКА

ЛАЗЕРНАЯ
ФИЗИКА

АРХЕОЛОГИЯ
ЭКОНОМИКА
ЗНАНИЙ

СОТРУДНИЧЕСТВО

АРКТИКА

КОГНИТИВНЫЕ ТЕХНОЛОГИИ

N* Новосибирский
государственный
университет
*НАСТОЯЩАЯ НАУКА



RISC-V как корень доверия в сфере безопасности и основа подготовки ИТ-инженеров ТОП-уровня.

к.т.н., заведующий лабораторией YADRO в НГУ
Власов Александр Александрович

Тумилевич В.А., Долгов Д.А.
Лаборатория YADRO, НГУ

* Проблема доверия к компилятору



Задача: реализовать bootstrap-цепочку для 64-битного RISC-V от минимального доверенного начального кода до компилятора TinyCC, в которой каждое звено является проверяемым и воспроизводимым.

¹ Атака Томпсона — K. Thompson, «Reflections on Trusting Trust», CACM, 1984, https://cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf

² Атаки на цепочку поставок программного обеспечения — A.-C. Bardaş, «Reflections on Trusting Trust After 40 Years», RoEduNet, 2025, <https://ieeexplore.ieee.org/document/11208472>

* Корень доверия (Root of Trust)

Закрытая платформа (x86, ARM и др.)

Встроенная прошивка (BIOS, UEFI)¹

Подсистемы управления (Intel ME, AMD PSP)²

Микрокод процессора

Доверяй

RISC-V

Открытая прошивка (OpenSBI, U-Boot)

Нет скрытых подсистем управления

Открытое ядро (RTL) без скрытого микрокода

Доверяй, но проверяй

Открытость — необходимое, но не достаточное условие: цепочку доверия нужно построить

¹ Базовая система ввода-вывода: прошивка, через которую процессор при запуске получает доступ к устройствам ввода-вывода, загружает следующую программу (загрузчик, ОС или иной код) и передает ей управление. Выполняется с наивысшими привилегиями, код поставляемых прошивок, как правило, закрыт.

² Отдельные процессоры внутри чипсета (Intel ME) или CPU (AMD PSP) с закрытой прошивкой, которые проверяют прошивку при загрузке, реализуют TPM и шифрование памяти, а также удаленное администрирование. Работают независимо от ОС, имеют доступ к памяти и сети и активны даже в выключенном состоянии компьютера.

* За основу взята цепочка live-bootstrap (x86)

Основа: live-bootstrap¹

Для архитектуры x86 (и частично для Risc-V) построена проверяемая bootstrap-цепочка: от минимальных исходных представлений до компилятора C и его самосборки.



Берём

идею цепи сборки, отсутствие «черных ящиков», сравнение исходных кодов с помощью контрольных сумм², идея смены управляющей программы при переходе на более высокий уровень

Реализуем для RISC-V заново

Стартовое ядро Simon-seed для поддержки UART, ramfs, hex0; в отличие от live-bootstrap для RISC-V, где цепочка идет внутри готового Linux, у нас все шаги выполняются на «голом железе», без доверенного ядра ОС

Часть шагов берём у live-bootstrap, а доверенную среду для RV64 строим сами

¹ Цепочка для x86 — live-bootstrap, <https://github.com/fosslinux/live-bootstrap>

² Воспроизводимые сборки — C. Lamb, S. Zacchioli, «Reproducible Builds: Increasing the Integrity of Software Supply Chains», IEEE Software, 2022, <https://doi.org/10.1109/MS.2021.3073045>

* Почему перенос с x86 на RISC-V нетривиален

Первые стадии зависят от платформы

Загрузчик, карта памяти, соглашения о вызовах, ранние форматы бинарников и доступ к устройствам нельзя перенести механически.

До полноценного компилятора C их нужно реализовать для RV64 вручную.

Компиляторы и запуск на железе

Первый TinyCC собираем через `pnut` вместо GNU Mes в `live-bootstrap`¹ из-за большого времени сборки ТСС и багов, связанных с собственным ядром Simon-seed, также, у `pnut` не было кодогенерации для RV64, ее мы написали сами. Для самосборки TinyCC первый ТСС собирает ядро Kronos с Linux ABI: загрузчик ELF, процессы, файловая система в памяти. Оно заменяет FiwiX (клон Linux 2.0), который в `live-bootstrap` существует только для x86.

Перенос на RISC-V — это не только компиляторы, но и собственная доверенная среда выполнения

¹ Bootstrap GCC для RISC-V — E. Zárraga, «Bootstrapping GCC in RISC-V», серия статей, 2022–2024, <https://ekaitz.elenq.tech/tag/bootstrapping-gcc-in-risc-v.html>

* Окружения Simon-seed и Kronos

Simon-seed¹

Собирается хостом из ассемблера RV64 — единственный код вне цепочки, содержит интерпретатор hex0, командную строку через UART, хранит файлы в оперативной памяти (ramfs), ведёт цепочку от hex0 до первого TinyCC

Kronos

Kronos поддерживает минимальное подмножество POSIX через системные вызовы Linux: работу с файлами (open, read, write, close, lseek, stat, mkdir, getcwd), управление памятью (brk, mmap) и время. Порождение процессов (fork, exec, wait), каналы, сигналы и командная оболочка не поддерживаются: последовательность сборки задается самим ядром.

tcc-pnut собирает Kronos



перезапуск системы
system reset



TinyCC собирает TinyCC в Kronos

Kronos добавляет ровно ту среду, которой не хватает TinyCC для обычной самокомпиляции

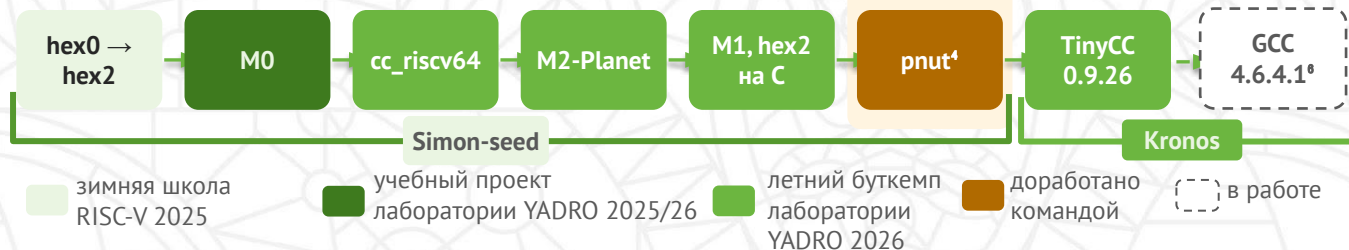
¹ Стартовое ядро для RISC-V — зимняя школа лаборатории YADRO НГУ, «ws25-bootstrap», 2025, <https://github.com/ylab-nsu/ws25-bootstrap>

* Bootstrap-цепочка: x86 и RISC-V

x86 · live-bootstrap¹



RISC-V 64 · наша цепочка²



Ступени до M2-Planet — из stage0-posix⁶, но в машинном коде RV64. Главное отличие: вместо GNU Mes промежуточный компилятор pnut с RV64-кодогенератором

¹ Bootstrap для x86 — S. Tyler, A. Štikonas и др., «live-bootstrap», <https://github.com/fosslinux/live-bootstrap>

² Bootstrap для RISC-V — «Aegis-V», <https://github.com/h1km4t1ll0/Aegis-V>

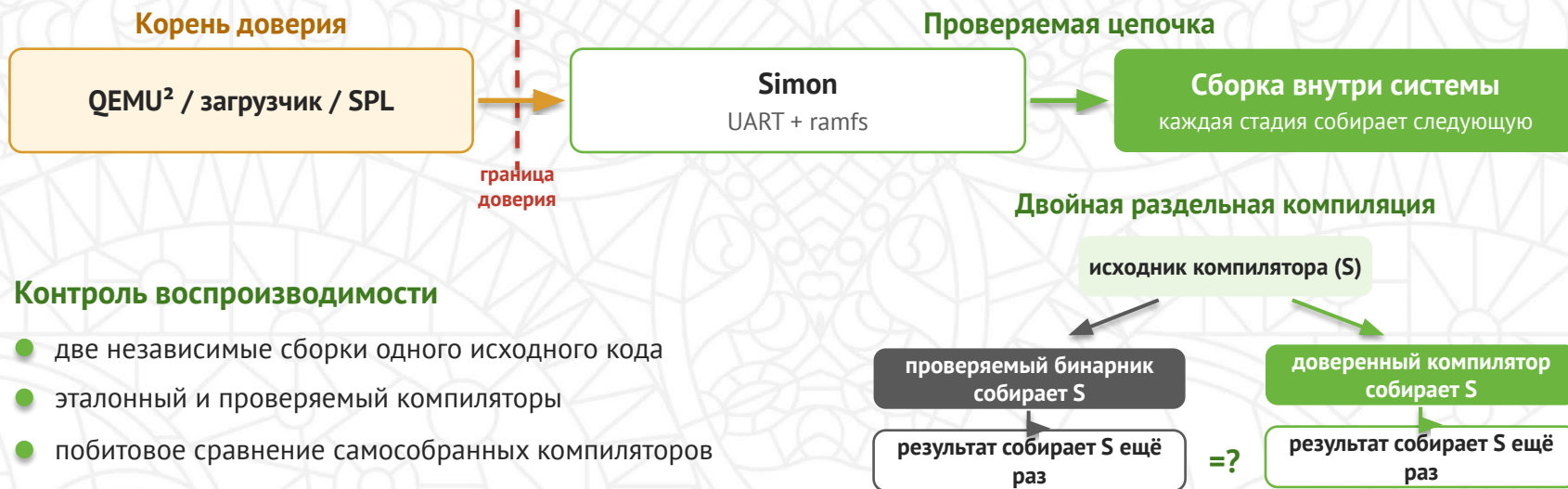
³ Путь к TinyCC через Scheme — J. Nieuwenhuizen, «GNU Mes», <https://gnu.org/software/mes>

⁴ Компилятор pnut — L. Huberdeau и др., «The Design of a Self-Compiling C Transpiler Targeting POSIX Shell», SLE, 2024, <https://doi.org/10.1145/3687997.3695639>

⁵ Ранние стадии для RISC-V — J. Orians и др., «stage0-posix-riscv64», <https://github.com/oriansj/stage0-posix-riscv64>

⁶ gcc 4.6.4.1 с RISC-V — E. Zarraga, «GCC 4.6.4 fork for bootstrapping, with added RISC-V support», 2022–2024, <https://codeberg.org/ekaitz-zarraga/gcc>

* Двойная раздельная компиляция (DDC¹)



Контроль воспроизводимости

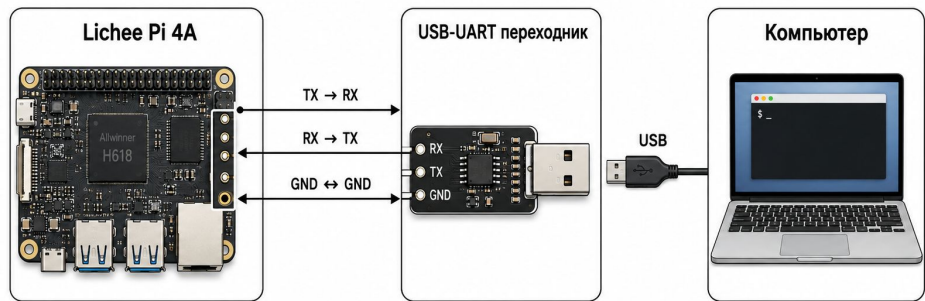
- две независимые сборки одного исходного кода
- эталонный и проверяемый компиляторы
- побитовое сравнение самособранных компиляторов

Бинарное несовпадение результатов компиляции свидетельствует об атаке Томпсона; бинарное совпадение же подтверждает ее отсутствие.

¹ Метод DDC – D. A. Wheeler, «Countering Trusting Trust through Diverse Double-Compiling», ACSAC, 2005, <https://arxiv.org/abs/1004.5548>

² Эмулятор QEMU – F. Bellard, «QEMU, a Fast and Portable Dynamic Translator», USENIX ATC, FREENIX Track, 2005

*Перенос на Lichee Pi 4A



Отладочное UART-подключение

Целевая платформа¹

TH1520, 4 ядра C910, RV64GCV
отдельный boot target и full-flash image
U-Boot SPL², DDR mapping и UART
общие Simon sources для QEMU и платы

Текущий статус

Boot target подготовлен, организован запуск и отладка через UART. Проведена двойная раздельная компиляция атакованного³ и чистого TinyCC.

На Lichee Pi подготовлено окружение и аппаратный запуск; самокомпиляция TinyCC уже подтверждена на QEMU и перенесена плату.

¹ Плата Lichee Pi 4A — Sipeed, «Lichee Pi 4A: документация», <https://wiki.sipeed.com>

² Загрузчик U-Boot — «Das U-Boot», <https://docs.u-boot.org>

³ Демонстрация атаки и DDC — лаборатория YADRO НГУ, «evil-tinycc», <https://github.com/yld-lab/evil-tinycc>

* Интеграция знаний в одной задаче

Знания

Кибербезопасность

Компиляторы

Операционная система

Цифровые платформы

Машинный код

Что осваивает студент

воспроизведение атаки Томпсона на компилятор

внутреннее устройство компиляторов

системные вызовы, память, процессы

понимание устройства архитектуры RISC-V

чтение и проверка программ на уровне байтов

Этап проекта

двойная раздельная компиляция

сс, pnut, TinyCC

ядро Kronos

среда Simon-seed

ранние стадии от hex0 до cc

Всё, что изучалось студентом **по отдельности**: ассемблеры, ОС, компиляторы, основы кибербезопасности, проект требует применить **сразу**. Таким образом отдельные знания студента применяются вместе в одной практической задаче.



Учебный план

образовательной программы бакалавриата ТОП ИТ

09.03.01 Информатика и вычислительная техника. Технологии индустриальной программной инженерии

1 курс

2 курс

3 курс

4 курс

1 семестр	2 семестр	3 семестр	4 семестр	5 семестр	6 семестр	7 семестр	8 семестр
- Основы матанализа и алгебры - Основы дискретной математики и математической логики - Цифровые платформы - Императивное программирование - Введение в операционные системы и инструменты разработчика - Моделирование данных - Основы Российской государственности - История России - Прикладная физическая культура - Иностранный язык для ТОП ИТ - ОУФ - Физическая культура	- Основы матанализа и алгебры - Основы дискретной математики и математической логики - Цифровые платформы - Алгоритмы и структуры данных - Введение в операционные системы и инструменты разработчика - История России - Прикладная физическая культура - Иностранный язык для ТОП ИТ - ОУФ - Физическая культура - Учебная практика (ознакомительная практика)	- Теория вероятностей и математическая статистика - Дифференциальные уравнения и ТФКП - Теория информации - Машинное обучение и глубокое обучение - Операционные системы - Объектно-ориентированное программирование - Разработка ПО для решения научных и прикладных задач (групповой проект) - Введение в искусственный интеллект - Иностранный язык - Прикладная физическая культура - Учебная практика (технологическая (проектно-технологическая) практика)	- Теория вероятностей и математическая статистика - Теория информации - Машинное обучение и глубокое обучение - Операционные системы - Объектно-ориентированное программирование - Разработка ПО для решения научных и прикладных задач (групповой проект) - Интегрированные информационные системы - Разработка ПО для решения научных и прикладных задач (групповой проект) - Деловой английский язык - Прикладная физическая культура - Учебная практика (технологическая (проектно-технологическая) практика)	- Вычислительная математика - Индустриальное программирование и управление разработкой ПО - Программная архитектура - Командная разработка индустриального ПО (групповой проект) - Базы данных - Блок дисциплин по выбору 1 продолжение (1 из 2) - Философия - Иностранный язык (факультатив) - Прикладная физическая культура - Производственная практика (технологическая (проектно-технологическая) практика)	- Индустриальное программирование и управление разработкой ПО - Программная архитектура - Командная разработка индустриального ПО (групповой проект) - Базы данных - Блок дисциплин по выбору 2 (1 из 2) - Иностранный язык (факультатив) - Правоведение - Прикладная физическая культура - Производственная практика (технологическая (проектно-технологическая) практика)	- Применение ИИ в разработке - Кибербезопасность интегрированных информационных систем - Управление ИТ в ИТ индустрии - Блок дисциплин по выбору 3 (1 из 2) - Иностранный язык - Технологическое и социальное предпринимательство - Экономика ИТ сферы - Производственная практика (научно-исследовательская работа)	- Управление проектами ИТ индустрии - Иностранный язык - Основы риторики - Безопасность жизнедеятельности - Производственная практика (преддипломная практика)

Обозначения

- соц-гум дисциплина
- фундаментальная дисциплина
- профильная дисциплина
- практика
- Блок дисциплин по выбору

Программа прошла экспертизу в Аналитическом центре при Правительстве РФ и утверждена Ученым советом ФИТ

ГИА, ВКР

* Заключение

Выводы

- RISC-V позволяет сделать корнем доверия законы физики, литограф и исходные коды программ. Открытая система команд позволяет выстроить проверяемую цепочку без закрытой прошивки, без доверенного ядра ОС и вычислительного устройства.
- Реализация бутстрапа хорошая задача для обучения студентов: построение bootstrap-цепочки от hex0 до компилятора TinyCC объединяет в одной задаче архитектуру, ОС, компиляторы и кибербезопасность.
- Метод двойной раздельной компиляции (DDC) проверен на компиляторе TinyCC. Сравнение чистого и зараженного компилятора с доверенным компилятором позволила идентифицировать атаку Томсона.
- Подход не привязан к конкретной реализации архитектуры RISC-V. Переход с QEMU на Lichee Pi 4A потребовал только собственной загрузки, разметки памяти и UART, остальная цепочка осталась без изменений.

В дальнейшие планы входит сборка GCC, двойная раздельная компиляция более сложных компиляторов и подготовка учебного стенда для лабораторных работ на выявление атаки Томпсона

Авторы благодарят компанию YADRO за поддержку проекта