arm

Challenges in binary rewriting: enabling BOLT to optimize CFI-hardened binaries

Gergely Balint

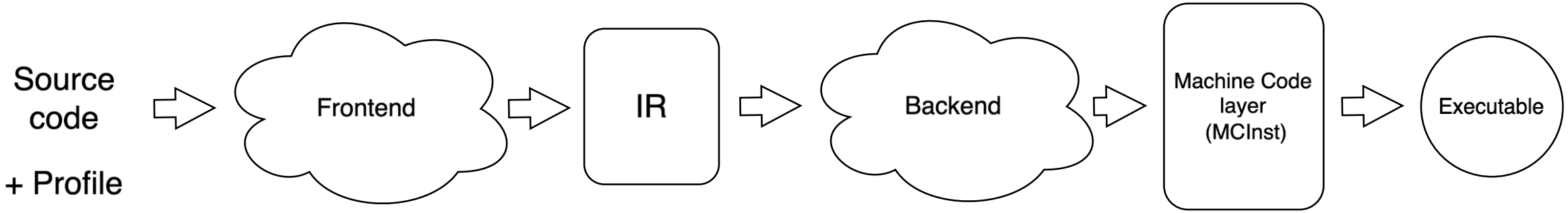
EuroLLVM26, Dublin

Introduction

- Goal: fast and secure software
- Profile Guided Optimizations
  - Goals: lower CPU frontend pressure (I$ miss, iTLB miss, branch miss)
  - PGO, CSPGO, BOLT, Propeller
- Can optimization tools break security features?

Optimization flow

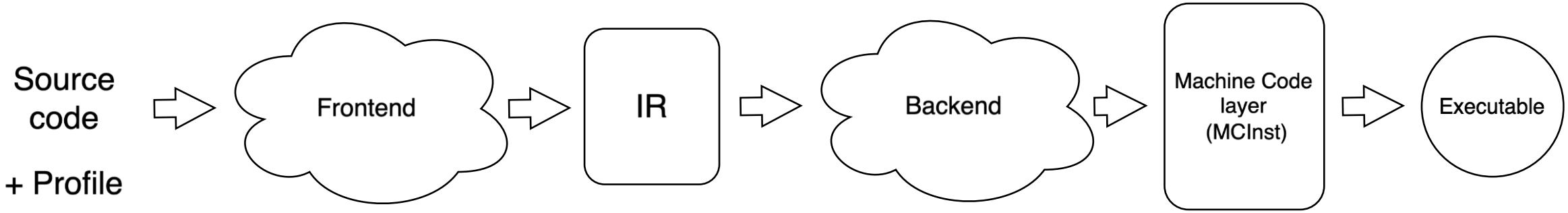
Compiler-based PGOs



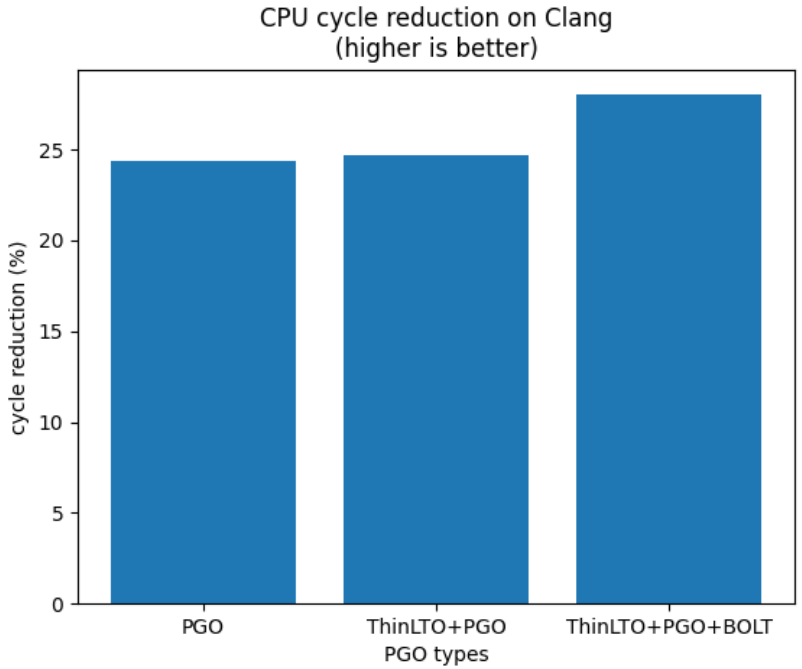
- Steps: compile binary, record profile, recompile
- Problem: profile correlation, out-of-sync profiles
- Upside: PGO doesn't affect features in LLVM, recompilation means *everything just works*

Optimization flow

Compiler-based PGOs

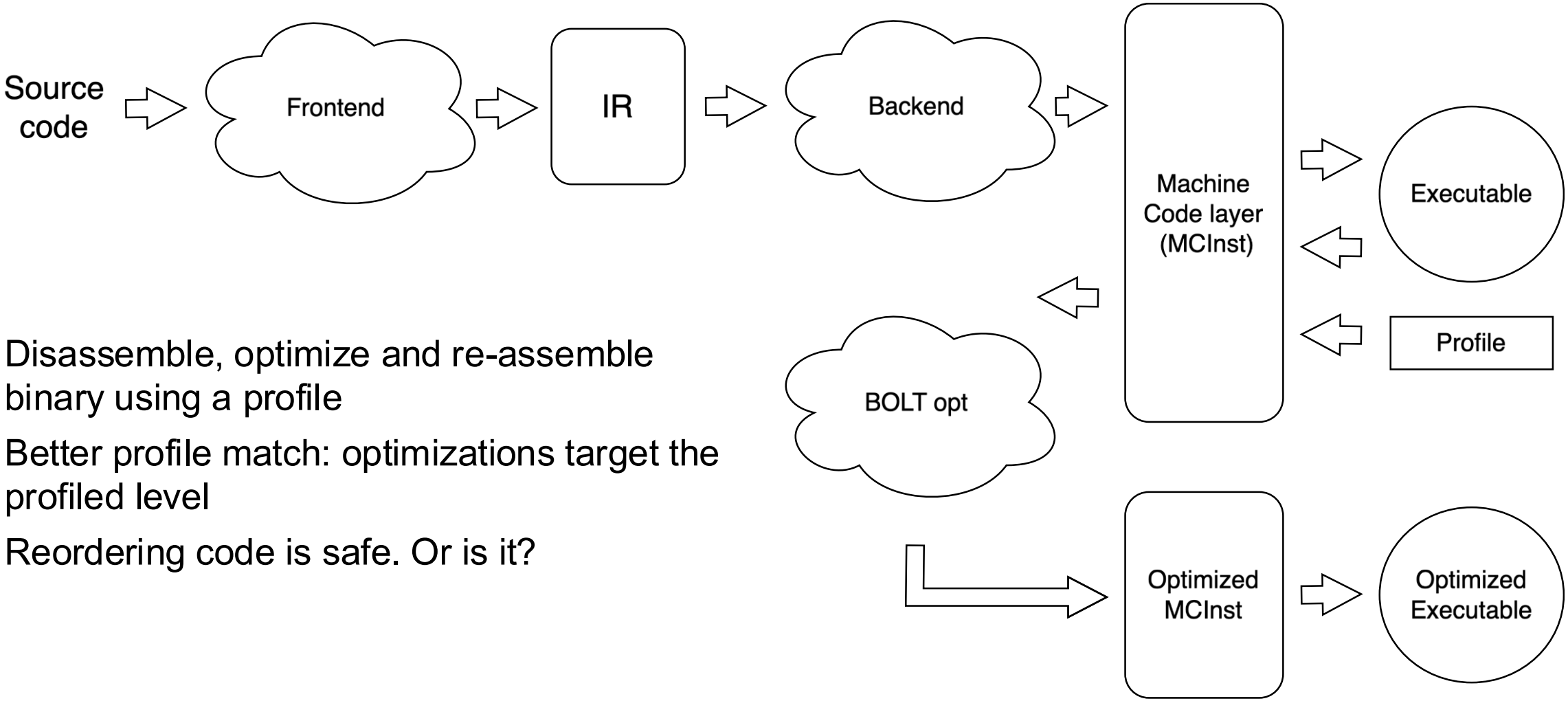


- Steps: compile binary, record profile, recompile
- Problem: profile correlation, out-of-sync profiles
- Upside: PGO doesn't affect features in LLVM, recompilation means *everything just works*



Optimization flow

BOLT

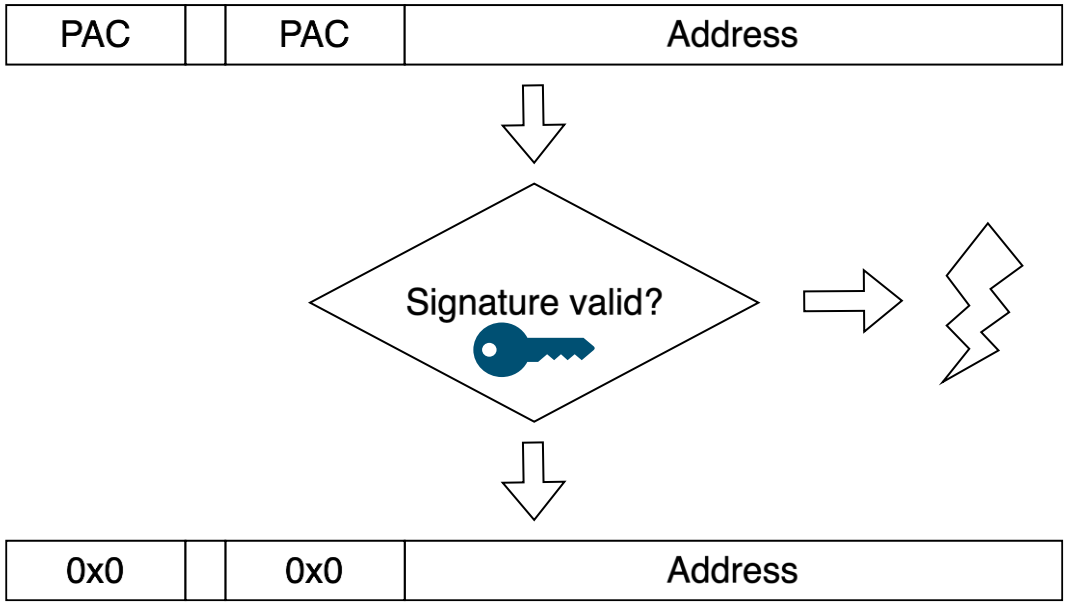


- Disassemble, optimize and re-assemble binary using a profile
- Better profile match: optimizations target the profiled level
- Reordering code is safe. Or is it?

Control Flow Integrity

Introduction

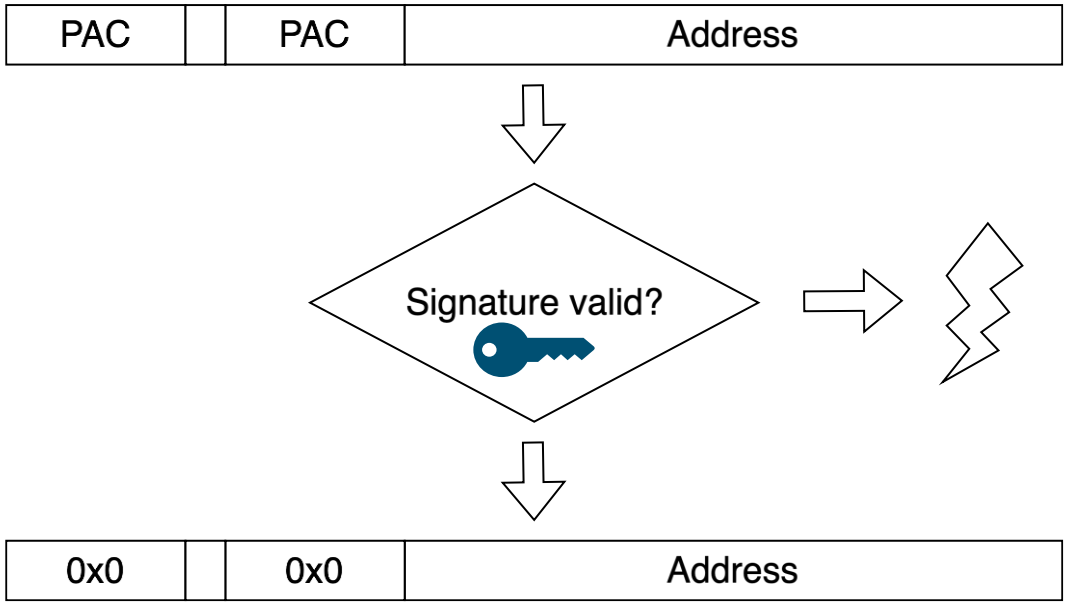
- Goal: make code-reuse attacks more difficult
- AArch64: Pointer Authentication, Branch Target Identification
- -mbranch-protection=standard

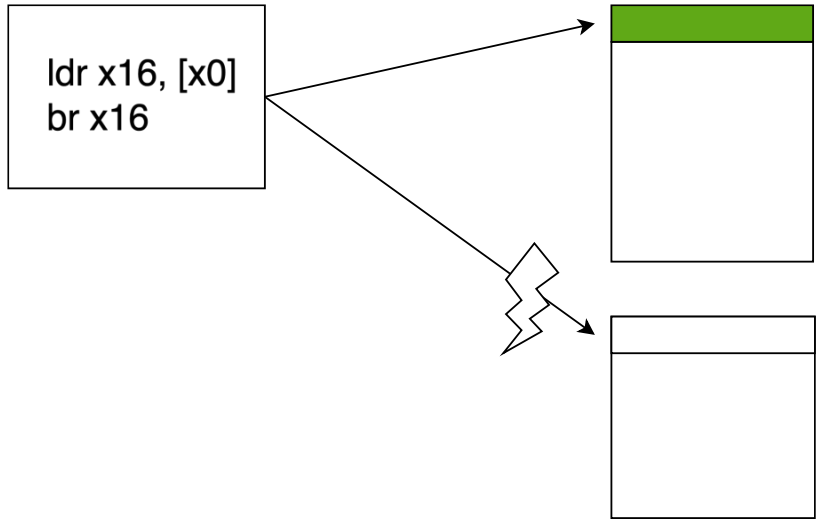


Control Flow Integrity

Introduction

- Goal: make code-reuse attacks more difficult
- AArch64: Pointer Authentication, Branch Target Identification
- -mbranch-protection=standard

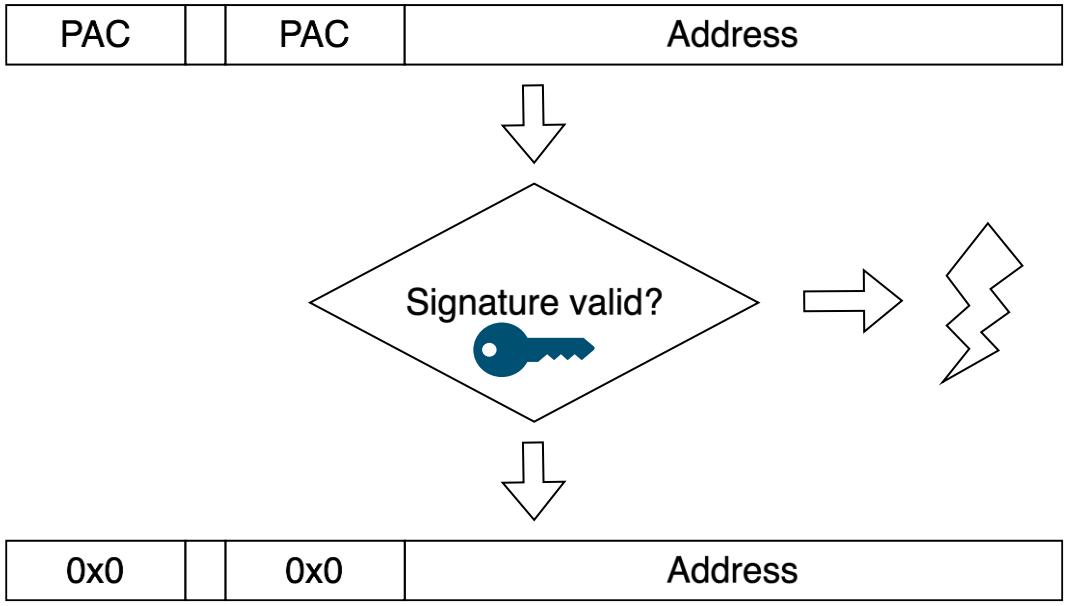


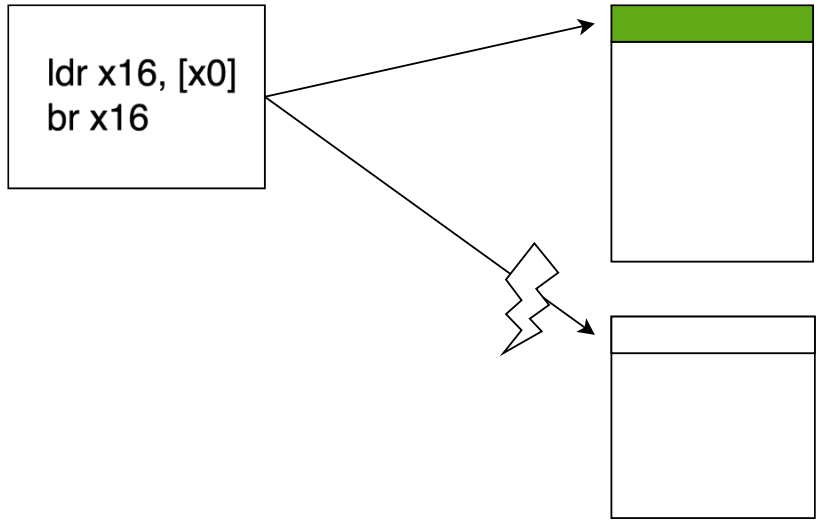


Control Flow Integrity

Introduction

- Goal: make code-reuse attacks more difficult
- AArch64: Pointer Authentication, Branch Target Identification
- -mbranch-protection=standard



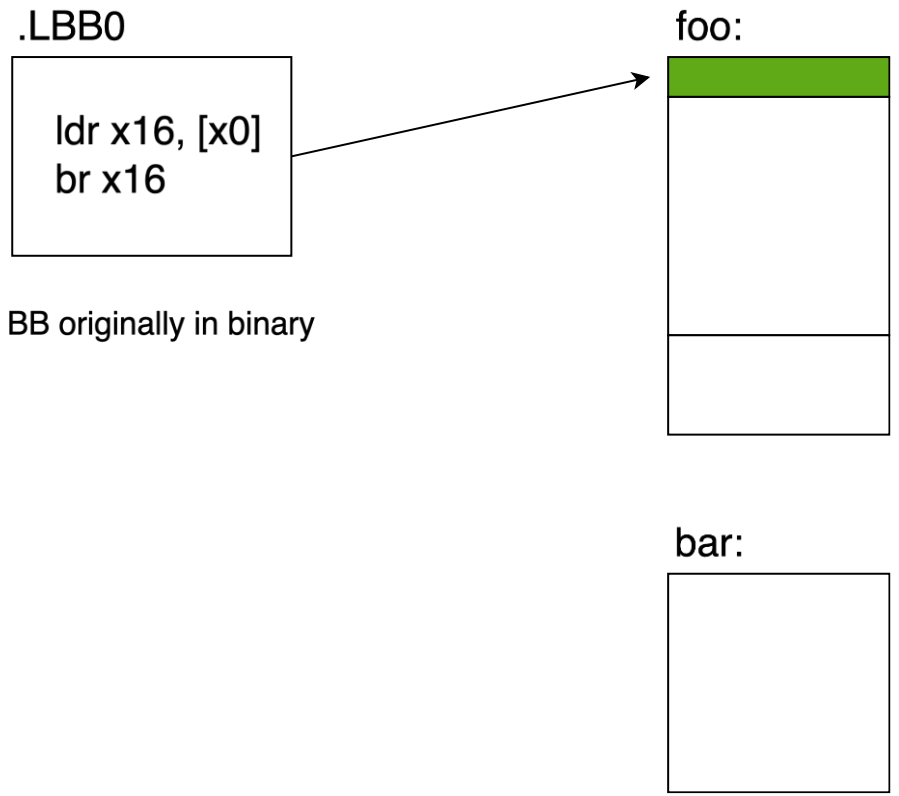


Android is a trademark of Google LLC

Control Flow Integrity

Branch Target Identification

- If BOLT inserts new indirect branches, it must also insert landing pads to their targets

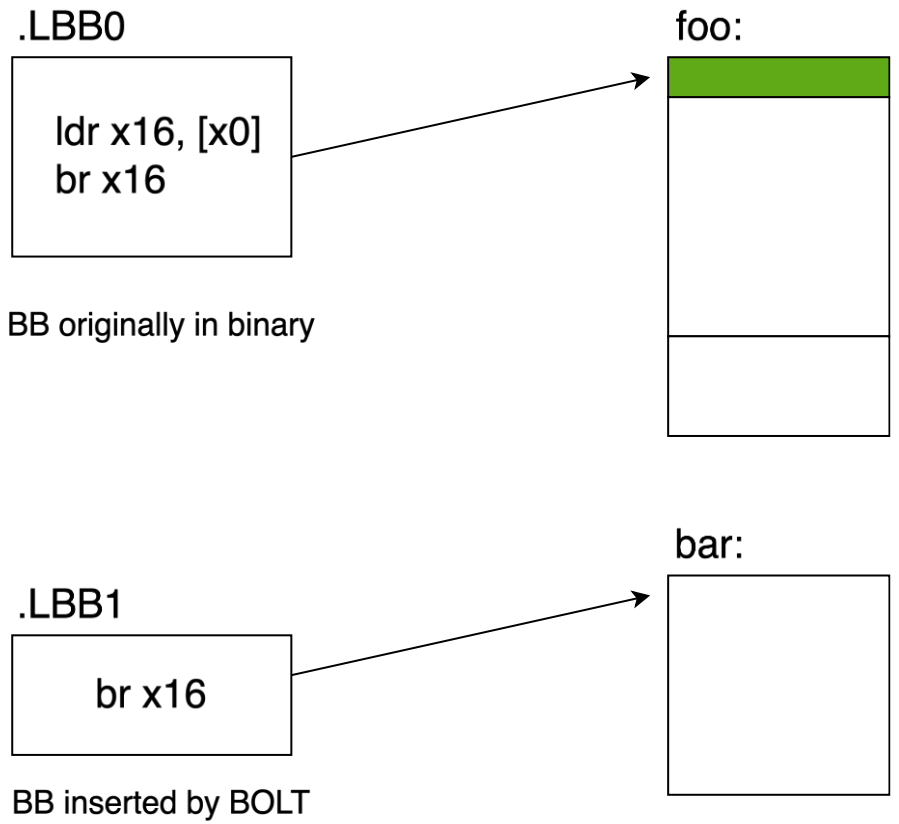


Control Flow Integrity

Branch Target Identification

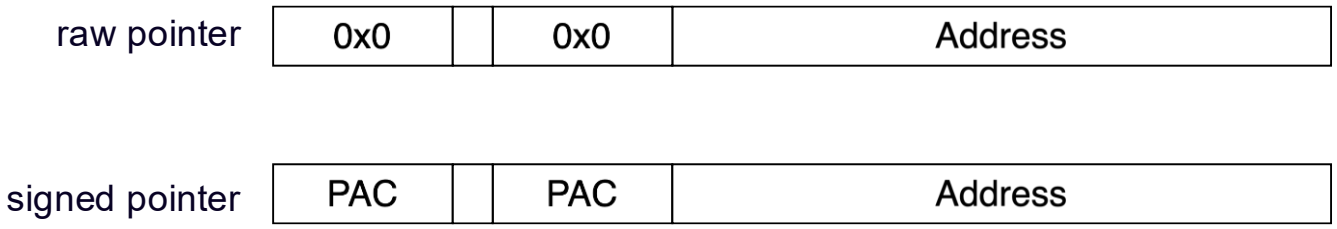
- If BOLT inserts new indirect branches, it must also insert landing pads to their targets
- Example: adding `LBB1` means BOLT must add a landing pad to `bar`

- BOLT-specific implementation issues:
  - What if BOLT cannot disassemble and optimize `bar`?
  - Needs to support code from all compilers and linkers
  - Can BOLT generate the landing pads in a pass?



Control Flow Integrity

Pointer Authentication



- pac-ret ABI requirements: vendor-specific DWARF Call Frame Instructions
- Their location depends on the code layout
- Solution: added two new BOLT passes
- PointerAuthCFIAnalyzer
  - Includes checks against incorrect codegen: correctly found issues when optimizing binaries build with an earlier Clang version
- PointerAuthCFIFixup
  - Generates correct CFIs for the output binary

Summary

- BOLT can now optimize binaries with AArch64 Control Flow Integrity features
- But: ISAs and LLVM are constantly evolving, BOLT needs support for new features
- Recompilation-based PGOs are not affected, but may provide less performance uplift
- BOLT users: check if BOLT supports every feature your binary uses
- Possible improvements to BOLT:
  - ABI extensions to standardize codegen patterns (linker-inserted veneers, jump-tables)
  - Provide more metadata to BOLT (e.g.: used architecture version, used compiler flags, precise Control Flow Graphs)