allvm - Binary Decompilation

Sandeep Dasgupta
University of Illinois Urbana Champaign

March 25, 2016

Outline

Goal & Motivation

Possible Directions

Decompile `Machine Code` $\longrightarrow$ `LLVM IR`

Our Approach

mcsema

Demo

Research Goal

- Research Goal
    - Obtain “richer” LLVM IR than native machine code.
- Motivation
    - Absence of source-code
    - What-you-see-is-not-what-you-execute
    - End-user security enforcement
    - Platform aware optimizations

Outline

Goal & Motivation

Possible Directions

Decompile `Machine Code` $\longrightarrow$ `LLVM IR`

Our Approach

mcsema

Demo

Decompile `Machine Code` → `LLVM IR`

- Challenge: Quality
    - Reconstructing code and control flow - Much researched.
    - Variable recovery
    - Function & ABI rules recovery

“Annotated” `Machine Code` $\rightarrow$ `LLVM IR`

- Challenge: Annotations must be “minimal” & sufficient.

Ship `LLVM IR`

- Challenges: Adoption, risks to intellectual property

Outline

Goal & Motivation

Possible Directions

Decompile `Machine Code` → `LLVM IR`

Our Approach

mcsema

Demo

Variable & Function parameter recovery

- Benefit
    - Enables many fundamental analysis (Dependence, Pointer analysis)
    - Functional IR
- State of the art
    - Grammatech
        - value set analysis (VSA) & structure aggregate identification.
    - Second Write
        - Heuristics for function parameter detection
        - Scalable VSA
    - TIE
        - Type Recovery

Outline

Goal & Motivation

Possible Directions

Decompile `Machine Code` $\longrightarrow$ `LLVM IR`

Our Approach

mcsema

Demo

Our Approach

- Choose an existing decompilation framework.
- Experimentation with various variable and type recovery strategies
- Use the knowledge to infer “minimal annotation”

Outline

Goal & Motivation

Possible Directions

Decompile `Machine Code` $\longrightarrow$ `LLVM IR`

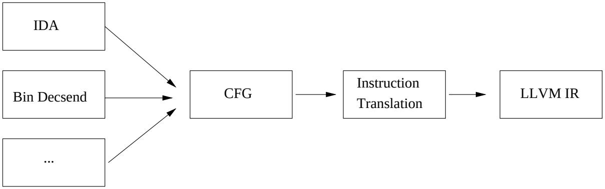
Our Approach

mcsema

Demo

Choosing mcsema

- Functional `LLVM IR`
- Separation of modules: CFG recovery and CFG $\rightarrow$ `LLVM IR`
- Actively supported and open sourced



Support & Limitations

- What Works
  - Integer Instructions
  - FPU and SSE registers
  - Callbacks, External Call, Jump tables
- In Progress
  - FPU and SSE Instructions: Not fully supported
  - Exceptions
  - Better Optimizations

Outline

Goal & Motivation

Possible Directions

Decompile `Machine Code` $\longrightarrow$ `LLVM IR`

Our Approach

mcsema

Demo

mcsema: Demo