BranchCorr: Detecting Incompatible Branch Behavior by Enforcing Branch Correlation Integrity

Long Cheng\textsuperscript{1}, Ebuka Okpala\textsuperscript{1}, Song Liao\textsuperscript{1}, Danfeng (Daphne) Yao\textsuperscript{2}
\textsuperscript{1}Clemson University, \textsuperscript{2}Virginia Tech

**Motivation**

![BranchCorr Workflow Diagram]

**Branch Correlation Integrity (BCI)**

BCI validates whether a conditional branch execution (branch taken or not-taken) is consistent with the outcomes of other correlated conditional branches.

![BCI Validation Diagram]

**Incompatible Branch Behavior**

![Incompatible Branch Behavior Diagram]

**Challenges**

- Multiple variables and operands
- Indirect correlation
- Condition changes in branch
- Path-sensitive

**BranchCorr Workflow**

- Coarse-grained Branch Correlation Analysis
- Symbolic Execution
- Program Slicing
- SMT Solver
- PT Tracing

**Learning Phase**

- Program dependency analysis
- Symbolic execution
- Backward slicing
- Satisfiability modulo theories (SMT) solver

**Detection Phase**

- Branch status table
- Correlated branch
- Expected value
- Outcome
- Same: yes
- Different: no
- Alarm

**On-going Work**

- Use LLVM-based branch correlation analysis tool to identify data-dependent branches
- Use angr to infer correlated branches
- Use Intel Processor Trace (PT) tool to trace the branch's taken-or-nor-taken (TNT) information for online detection
- Cross-app Cross-platform branch correlation in IoT