HexCFI: Context Sensitive Dynamic Control-Flow Integrity

Priyam Biswas       Nathan Burow       Mathias Payer

Motivation

- Despite hardening with LLVM-CFI, Chrome is still vulnerable to control-flow hijacks
- Attacks target indirect control flow transfers that are computed at runtime
- LLVM-CFI statically computes allowed target sets for indirect control-flow transfers and is over-approximate

Goals

- Compute an optimum target set per indirect call site
- An optimum target set is the smallest set such that the program can still execute correctly
- This will provide the strongest possible CFI security policy
- Promote call sites with one target to direct calls

Evaluation

Motivating Example

```c
void foo() {}  
void bar() {}  
void fun() {}  

int main() {  
  void(*fnptr)();  
  int a = 2;  
  if(a % 2 == 0)  
    fnptr = &bar;  
  else  
    fnptr = &foo;  
  fnptr();  
  return 0;  }
```

Target Sets

- LLVM_CFI { foo(), bar(), fun() }
- HexCFI { bar() }

HexCFI Architecture

- Analysis Phase
  - Instrumentation logs targets for each indirect control-flow transfer
  - Test Suite / fuzzing used to observe all valid execution paths
- Enforcement Phase
  - Computes target set per callsite from Target Log
  - Instruments indirect call sites to enforce valid target set