Control Flow Integrity: Clang/LLVM Implementation

System and Software Security Lab - FSU

Control Flow Integrity (CFI)

- Validate the program control flow against the pre-computed Control Flow Graph (CFG)
- Requires to validate: Indirect Control Flow Transfer
  - Indirect Call
    - C-Style Pointer
    - Virtual Function
  - Indirect Jump
    - Switch Statement
    - Goto Statement
  - Return Instruction

Address Taken

```c
#include <stdio.h>

typedef void (*vfnptr)();

void CallA() {}
void CallB() {}

int main(int argv, char **argc) {
  vfnptr fn = &CallA;
  if(argv == 1)
     fn = &CallB;
  fn();
  return 0;
}
```

Address-taken

Address-taken

Indirect Call

Address Taken and Type Match

```
#include <stdio.h>

typedef void (*vfnptr)();
typedef void (*ifnptr)(int);

void CallA() {}

void CallD(int a) {}

int main() {
  vfnptr fn1 = &CallA;
  ifnptr fn2 = &CallD;

  fn1();
  fn2();

  return 0;
}
```

```
class Hello {
private:
  vfnptr fp1 = &CallA;
  vfnptr fp2;
public:
  void (Hello::*x)();
  Hello(vfnptr f) { fp2 = f; }
  void ptofn() {}
  virtual void vFunc() {}
};
typedef struct ST {
  int a;
  ifnptr fp;
} st;
st st_arr[] = {{10, &CallD}, {20, &CallF}};
vfnptr gl = &CallE;
int main() {
  Hello *h = new Hello(&CallI);
  static vfnptr sfp[] = {&CallC, &CallE};
  st lc;
  lc.fp = &CallG;
  h->x = &Hello::ptofn;
  CallB();
  lc.fp(10);
  h->vFunc();
  return 0;
}
```

CFI Implementation

- Instrument reference monitor in every ICT
  - llvm/tools/clang/lib/CodeGen/CGCall.cpp
- Generate the CFG
  - LLVM Pass (Address Taken + Type Match)
- Instrument the CFG
  - Create a Global Constant Array with CFG from LLVM Pass