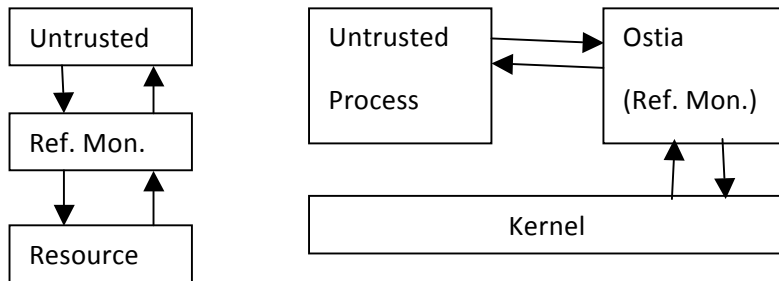


Reference Monitors



CFI – Control Flow Integrity – Complete mediation for IRMs

```
func(...) {
    ...
    check_perms(..);
    sensitive_ops();
}
```

Problems:

- Program could jump over check
- Program could modify own code (RO code)
- Program could generate code in data area (NX code)

Plugin 1
Plugin 2
Firefox

- SFI restricted memory and code access
- SFI is an “inline reference monitor” (IRM)
- Issue: complete mediation

Basic Block – is a sequence of instructions with no control transfer except possibly in last instruction and only the first instruction is target of other control transfers

```
e.g.  ld [$r0], $r0
      add 5, $r0, $r1
      cmp $r1, 7
      bne TGT
```

```
Example: void (func1(int x) {
          x = x+5;
          if(x<0) return -x;
          else return x;
        }
```

```

BB {
  prefetch -x12345678
  <preamble>
  add $r0, 5, $r0
  cmp $r0, 0
  mov L2, $r2
  cmp $r7 with prefetch 0x12345678
  bne ERROR
  blt [$r7+4]
}
BB {
  prefetch 0x12345678
  ret
  L2: mul $r0, -1, $r0
  ret
}

```

- Insert magic number at the start of each BB
- Check for magic instruction before each computed go to
- Remove all other occurrence of magic instruction
- Need non-executable stack

CFI

- Put checks in BB as operation they cover
- Force program to always jump to beginning of block

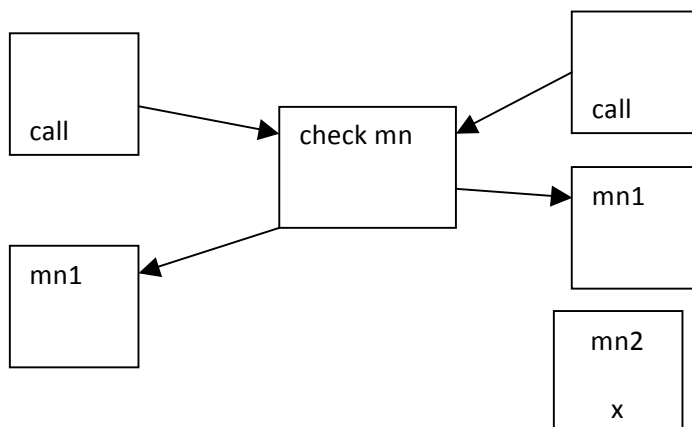
Issue: computed control transfer

Alternative:

- Align BB to a block
- A table of starting of block

How to insert CFI?

- Auto insert by program and verifier verify
- Manual insert



Before we execute code, check if we are execute code from stack
 Before we write, check if we write to code section