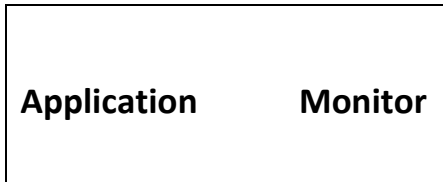


## Inline Reference Monitor



-Move monitor into app

-share address space

-must protect integrity of monitor

-advantages

-lower overhead

-can monitor more internal state & actions of application

-strictly more powerful than external monitor

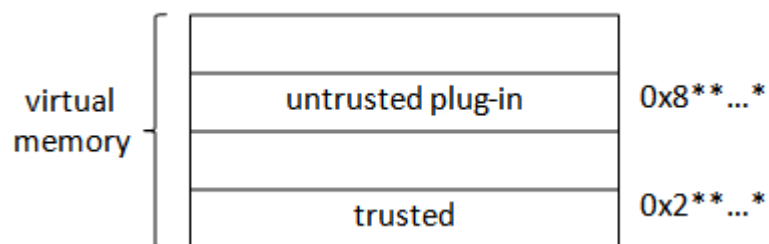
-challenges

-monitor integrity

-how to do it

-complete mediation

## SFI (later CFI)



Scenario:

-untrusted plug-in to trusted code

-lots of RPC calls

-GOAL: restrict untrusted code to its own memory + RPC

## Rewrite memory references

### Original

ld %r0, %r5

Attack: jump straight to ld

### Rewrite

mov %r30, %r5

mov %r31, %r30

and %r31, %r31, mask

cmp %r31, segid

jne ABORT

ld %r0, %r30

-dedicate register %r30 (%r28 %r29)

-all loads/stores use %r30

-all moves to %r30 followed by check

-no signals

### Enforcing

mov %r30, %r5

and %r30, %r30, Nsegmask

or %r30, %r30, segid

ld %r0, %r30

jmp %r5



mov %r30, %r5

and %r30, %r30, mask

or %r30, %r30, segid

jmp %r30

## SFI RPC

-untrusted module can only jump into its own segment

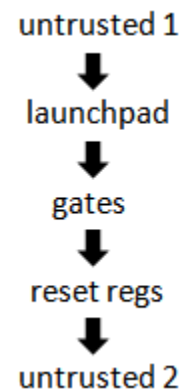
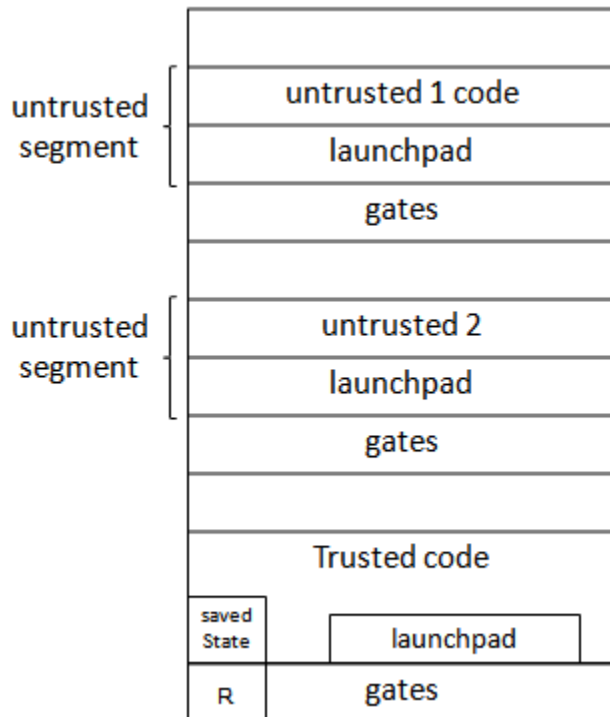
-only escape is via launchpad

-code in launchpad jumps to trusted code

-untrusted code can only jump to specified locations

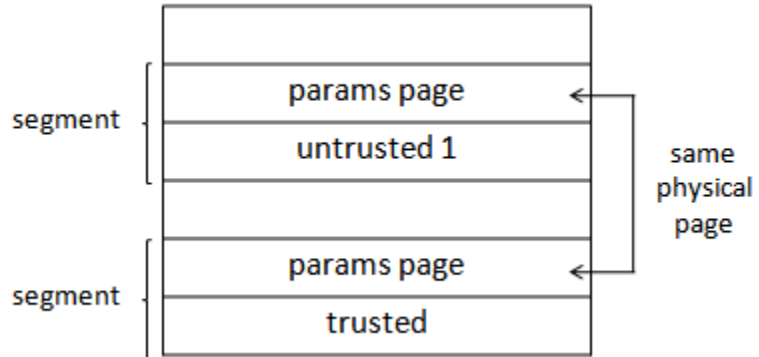
## Attacks

could jumps into middle of trusted function



```
jmp F1
jmp F2
jmp F3
.
.
.
```

untrusted code can safely  
jump anywhere in launchpad



### SFI RPC

- copy args
- 2 extra jumps
- dedicated regs

### OS RPC

- copy args
- caller to kernel
- kernel to callee
- save caller state
- load callee state
- OS overhead

-----  
OS RPC: 200  $\mu$ s

SFI RPC:  $\sim$ 1  $\mu$ s

Funcall: .1 $\mu$ s