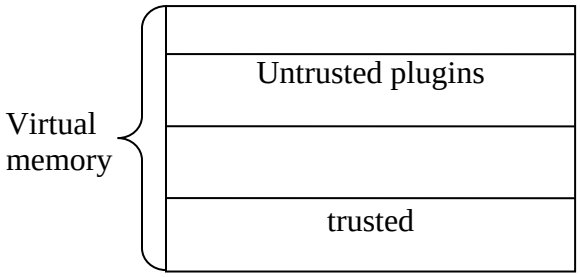
CSE 509 Class Note for 3/22

Inline Reference Monitor

- Move monitor into App
  - Share address space
  - Must protect integrity of monitor
  - Advantages
    - Lower overhead
    - Can monitor more internal state and actions of application
    - Strictly move powerful than external monitor
  - challenges
    - monitor integrity
    - how to do it
    - complete mediation

SFI



Scenario

- untrusted plugin to trusted code
- lots of RPC calls
- Goal : restrict untrusted code to its own memory + RPC

Rewrite memory references

Original	rewrite (checking)
	Mov %r30, %r5
Ld %r0, %r5	mov %r31, %r30
	And %r31, %r31, mask
	Cmp %r31, segid
	Jne ABORT
	Ld %r0 , %r30

Attack – jump straight to ld

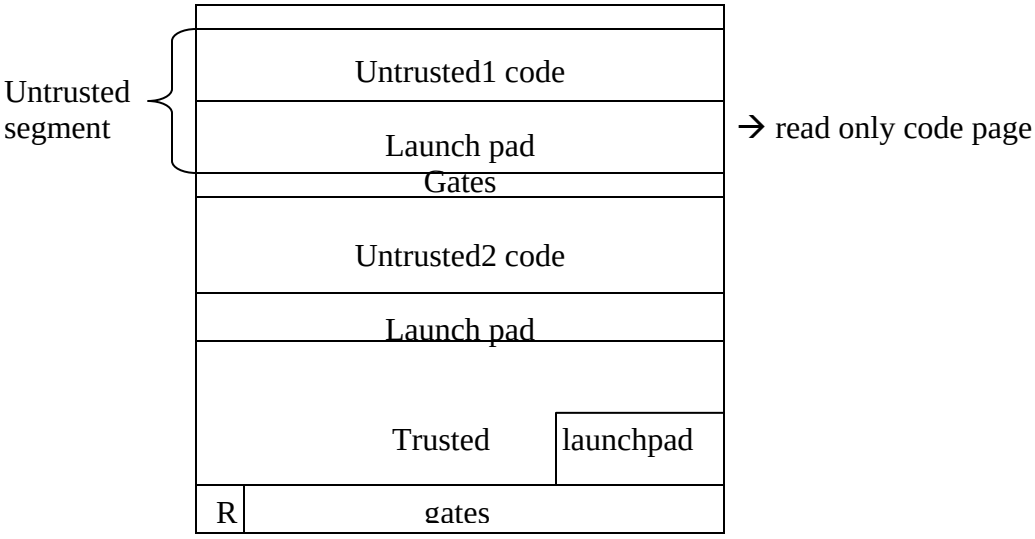
- Dedicate register : %r30
- All loads/stores use %r30
- All moves to %r30 followed by check
- No SIGNALS

Another version of rewrite (forcing)

```
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



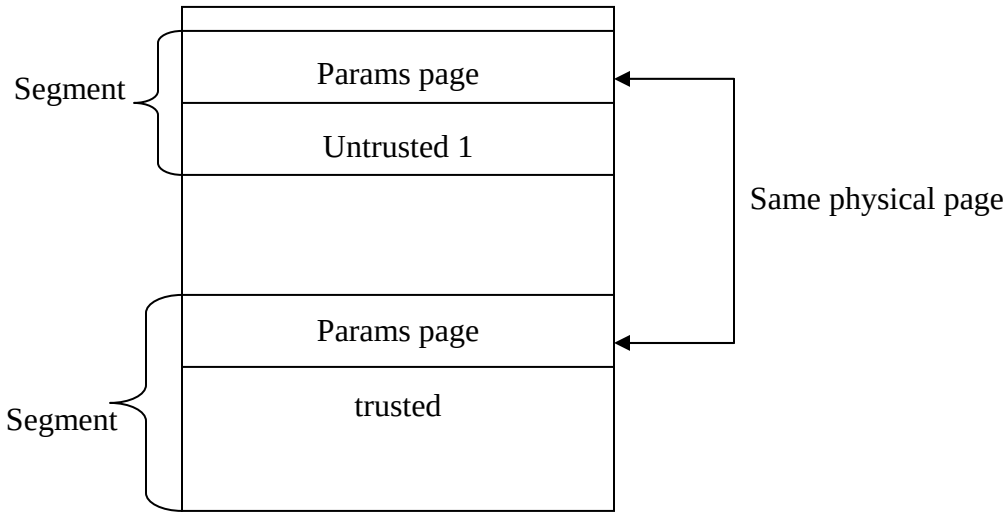
- Only escape is via launchpad
- Code in launchpad jumps to trusted code attacks
- Could jump into middle of trusted function
- Goal : untrusted code can only jump to specified locations

Launchpad

```
Jmp     R
Jmp     F2
Jmp     F3
……
```

Untrusted code can safely jump anywhere in launchpad

Untrusted1 → launchpad → gates → reset register → untrusted2



SFI RPC	OS RPC
- copy args	- copy args : from caller to kernel Kernel to callee
- 2 jumps	- save caller state
- dedicated registers	- load caller state
	- OS overhead

OS RPC : 200 micro second
SFI RPC : ~~ 1 micro second