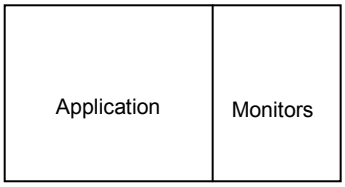
Lecture Notes: March 22, 2007

- **Inline Reference Monitors**

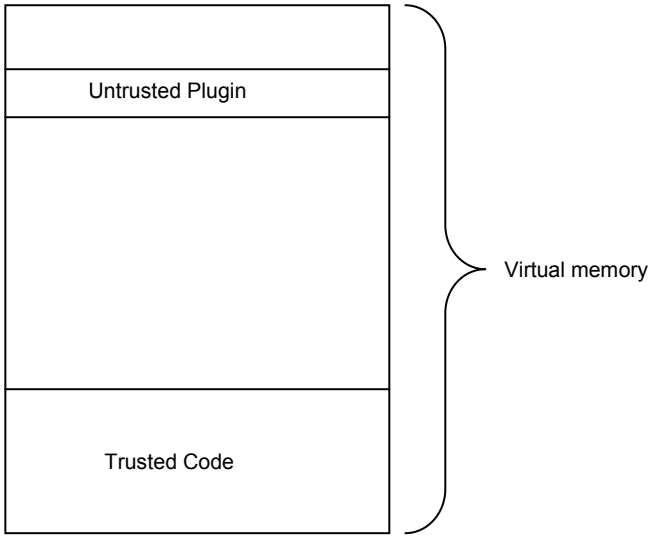


- Move monitor into the application
- Share address space
- Must protect integrity of the monitor

Advantages:
- Lower overhead
- Can monitor more internal state & actions of applications
- Strictly more powerful than external monitor

Challenges:
- Monitor integrity
- How to do it?
- Complete mediation

- **Software Fault Isolation**



Scenario:
- Untrusted plugin to trusted code
- Lots of RPC calls
- Goals: Restrict untrusted code to its own memory

Rewrite memory references:

Original	Rewrite
ld % r0, % r5	mov % r31, % r5 and % r31, % r31, mask cmp % r31, segid jne ABORT ld % r0, % r5

Here, segid refers to segment ID.
For the above approach, an attack might be possible if we directly jump to the 'ld' instruction.

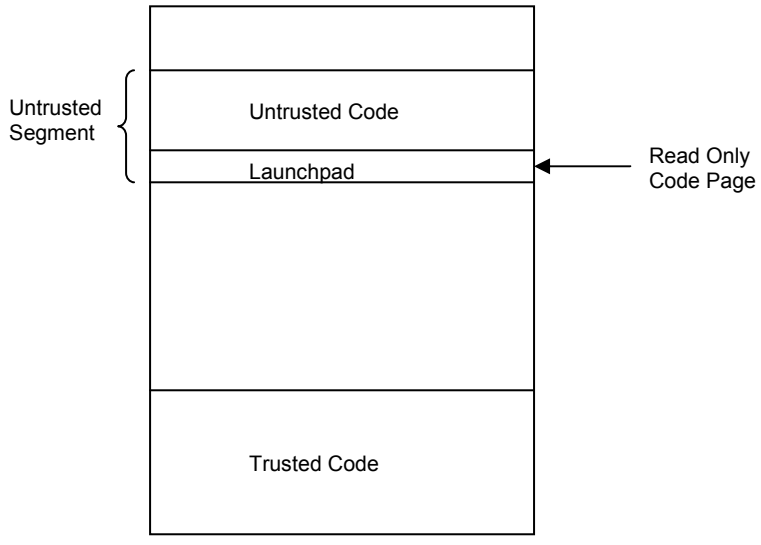
Solution:

- We need a dedicated register % r30
- All loads/stores use % r30
- All moves to % r30 followed by checks
- NO Signals

Checking	Forcing
mov % r30, % r5 mov % r31, % r5 and % r31, % r31, mask cmp % r31, segid jne ABORT ld % r0, % r5	mov % r30, % r5 and % r30, % r30, ~segmask or % r30,% r30, segid ld % r0, % r30

jmp % r5
|
|
V
mov % r30, % r5
and % r30,% r30, ~mask
or % r30, % r30, segid
jmp % r30

- **<u>Software Fault Isolation (SFI) Remote Procedure Call (RPC):</u>**
  - Untrusted module can jump only into its own segment



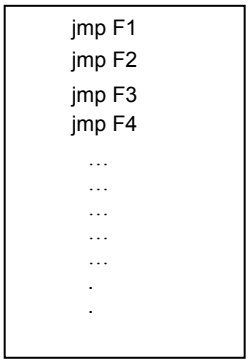
- Only escape is via Launchpad.
- Code in Launchpad jumps to trusted code.

In case we don't do this, a following attack is possible:

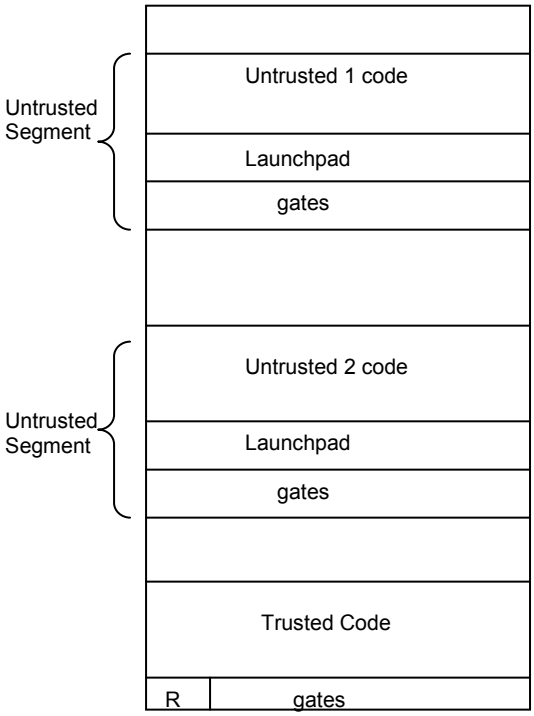
- We could jump into the middle of the trusted code.

Goal: Untrusted code can only jump to specified locations.

- **Launchpad:**

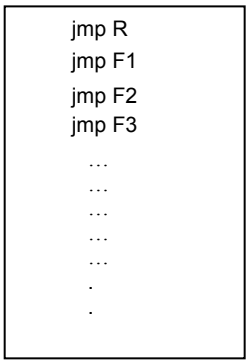


- Untrusted code can safely jump anywhere in launchpad

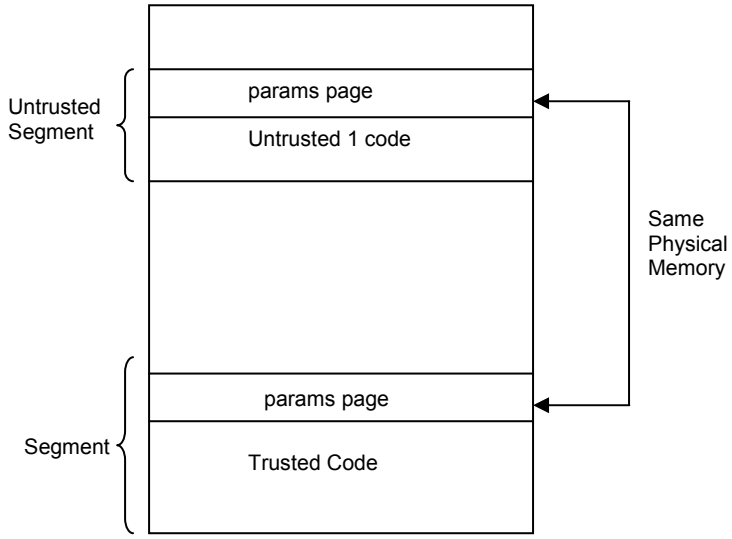


Untrusted 1 code ----> launchpad -----> gates -----> reset registers ------> untrusted 2 code

How do we know where to return in case of function calls?
We store the value of return address along with the gates.
Also, now the first entry in case of launchpad would be
jmp R (in the launchpad diagram shown above)
The remaining entries in the launchpad table remain the same.



When arguments are considered, the memory can be shown as follows:



Both the params pages share the same physical address.

- **Difference between SFI RPC and OS RPC:**

SFI RPC	OS RPC
- copy args (once)	- copy args - caller to kernel - kernel to callee
- 2 extra jumps	- save caller state
- dedicated registers	- load caller state
	- OS overhead

Performance measures:
OS RPC: 200 μs
SFI RPC: ~1 μs
Function call: 0.1 μs