Inline Reference Monitors

Jared Verdi - 3/22/07

Previous Approaches until now:

Application	Monitor
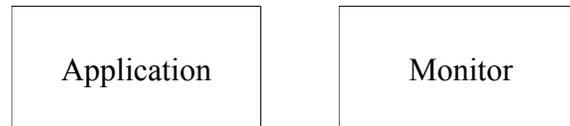
Inline Reference Monitor:
Move monitor into the application. Share address space
Must protect the integrity of the monitor

Application   Monitor

<u>Advantages</u>
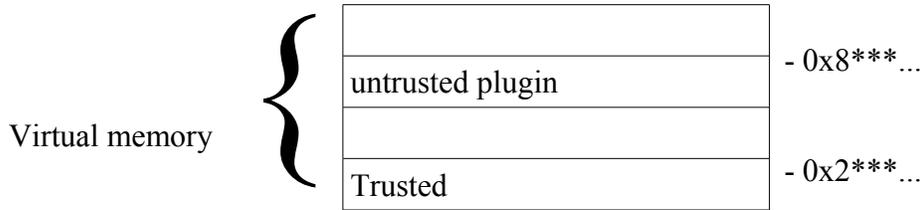
    - lower overhead
    - can monitor more internal state & actions of the application

<u>Challenges</u>

    - How?
    - monitor integrity
    - complete mediation

Software Fault Isolation – later: control flow integrity

Fault Domains

		- 0x8***...
	untrusted plugin	
	Trusted	- 0x2***...

Virtual memory {

Scenario

- untrusted plugin to trusted code
- lots of RPC calls (cross domain)

Goal

- restrict untrusted code to its own memory + RPC

Rewrite Memory References

original	rewrite

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

Attack

- We could jump straight to load instruction!

Prevention

- Dedicate a register for loads
- Invariant: this register will only contain safe values

    - Dedicate register %r30
    - All loads / stores use %r30
    - All moves to %r30 are followed by a check
    - NO SIGNALS

check	forcing

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

Fixing jump instruction

jmp %r5   mov %r30, %r5
                                 and %r30, %r30, mask
                                 or %r30, %r30, segid
                                 jmp %r30

SFI RPC
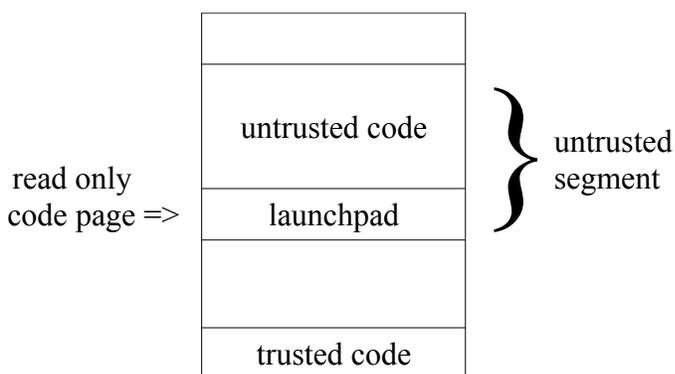
-untrusted module can only jump into
  its own segment

-only escape is via launchpad

-code in launchpad jumps to trusted
  code

Goal: untrusted code can only jump to
  specified locations.

Attacks

    - could jump into the middle of a trusted function

<u>Launchpad</u>

jmp F1
jmp F2
jmp F3
...

Untrusted code can
safely jump anywhere
in launchpad

<u>Mutually Untrusting Code Segments</u>

untrusted code 1
launchpad
R   gates
untrusted code 2
launchpad
R   gates
trusted
launchpad
R   gates

} untrusted
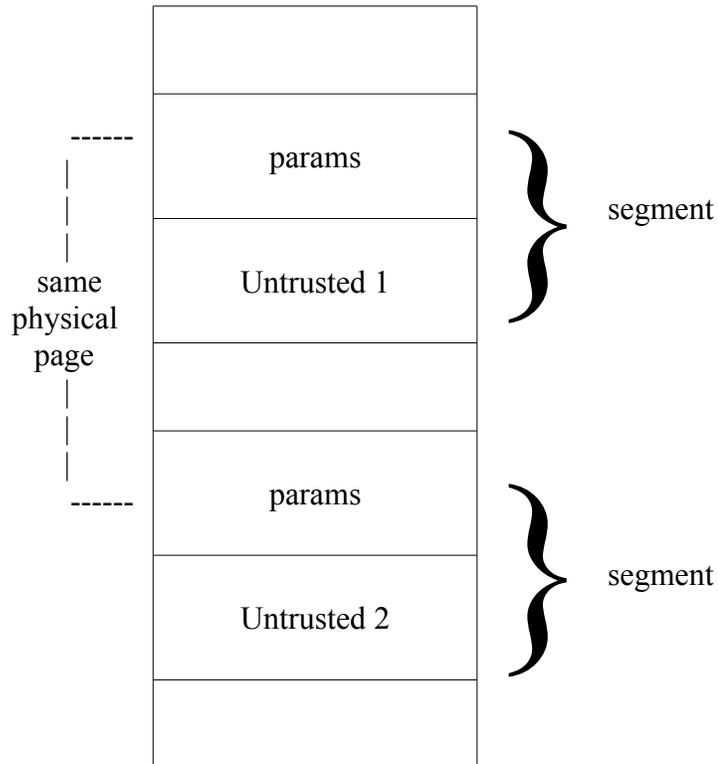segment

} untrusted
segment

Flow in Manually Untrusting Segments:

untrusted 1 =>
launchpad =>
gates =>
reset regs =>
untrusted 2

launchpad – how you get out of your
                        segment

gates – how you get into another segment

Handling parameters

```
          ┌──────────────┐
          │              │
  ------   ├──────────────┤  ⎫
    |      │    params    │  ⎬
    |      ├──────────────┤   ⎬ segment
    |      │              │  ⎬
  same     │  Untrusted 1 │  ⎭
physical   ├──────────────┤
  page     │              │
    |      ├──────────────┤
    |      │    params    │  ⎫
  ------   ├──────────────┤  ⎬
           │              │   ⎬ segment
           │  Untrusted 2 │  ⎬
           ├──────────────┤  ⎭
           │              │
           └──────────────┘
```

Overhead Analysis

SFI RPC	OS RPC

-copy args
-2 extra jumps
-update dedicated regs 2x
	-copy args
	    -caller to kernel
	    -kernel to callee
	-save caller state
	-load callee state
	-OS overhead

OS RPC: ~200 microseconds

SFI RPC: ~1 microseconds

Func call: ~0.1 microseconds