My Screensaver Explained:
Some Brief Remarks on Kprobes

Jason Reeves
March 3, 2011
What are Kprobes?

- Tracing framework built into the kernel
  - “dtrace” for Linux

- Provides a snapshot of the kernel at a given address

- Variety of uses
  - Ex. Intrusion detection!

Diagram based on the figure in “Probing the Guts of Kprobes” (Mavinakayannahalli et al. ’06)
Kprobe Structures

```c
struct kprobe {
    struct hlist_node hlist;
    struct lit_head list;
    unsigned long nmissed;
    kprobeOpcode_t *addr;
    const char *symbol_name;
    unsigned int offset;
    kprobePreHandler_t pre_handler;
    kprobePostHandler_t post_handler;
    kprobeFaultHandler_t fault_handler;
    kprobeBreakHandler_t break_handler;
    kprobeOpcode_t opcode;
    struct archSpecificInsn ainsn;
    u32 flags;
}
```
Kprobe Structures

struct kprobe {
    struct hlist_node hlist;
    struct lit_head list;
    unsigned long nmissed;
    kprobe_opcode_t *addr;
    const char *symbol_name;
    unsigned int offset;
    kprobe_pre_handler_t pre_handler;
    kprobe_post_handler_t post_handler;
    kprobe_fault_handler_t fault_handler;
    kprobe_break_handler_t break_handler;
    kprobe_opcode_t opcode;
    struct arch_specificInsn ainsn;
    u32 flags;
};
Kprobe Structures

- **Alternate Probes**
  - jprobes (before a function call)
  - kretprobes (after a function call)

- **kprobe_table** (array of linked lists)
  - Used to lookup kprobes
    - Stores hlist of probe – table slot determined by hash

- **kprobe_insn_pages**
  - List of *executable* pages for stored instructions
  - Allocated on an on-demand basis
Kprobe API

- register/unregister_kprobe()
  - BYOK
- enable/disable_kprobe()
  - Uses text_poke() function to place breakpoint

- Once inside the kprobe…
  - Kernel API at your disposal (mostly)
  - Registers passed in as a parameter
  - Doing too much in a probe can be trouble…
How a Kprobe Works

add eax, 0x4
How a Kprobe Works

83 c0 04
How a Kprobe Works

-0xcc = breakpoint instruction (int3)
How a Kprobe Works

cc c0 04

TRAP!!!

do_int3() → notify_die() → atomic_notifier_call_chain() → __atomic_notifier_call_chain() → notifier_call_chain() → …
How a Kprobe Works

cc c0 04

- call arch-specific handler
- look up kprobe

Onwards and upwards!

do_int3() → … → kprobe_exceptions_nb
Can We Do Better?

• *Direct Jump probes*  
  *(Hiramatsu ’05)*  
  – Uses a jmp instruction in place of int3

• Where do we jump?  
  – Detour buffers and trampoline code

• Is it faster?  
  – Hiramatsu ‘05: 10x faster!  
  – Reeves ‘11: Not so much…
What Can We Do?

- int (*kprobe_pre_handler_t) (struct kprobe *, struct pt_regs *)
What Can We Do?

• `int (*kprobe_pre_handler_t)(struct kprobe *, struct pt_regs *)`

  This is a pointer to the current probe!

• The Kprobe itself isn’t all that interesting…but could we use it as a launching point?
  – Ramaswamy ‘09: Yes!
What Can We Do?

- int (*kprobe_pre_handler_t) (struct kprobe *, struct pt_regs *)

  This is a pointer to the current register values!

- This is a more direct route to mayhem...
  - What if we mess with a function’s arguments?
  - What if we change the instruction counter?
Closing Thoughts

• Kprobes are often used as a force for good (debugging, intrusion detection, etc.). How could they be used for evil?
Thank You!

• Questions?
• Comments?
• Concerns?
• Criticisms?
References

• LXR, Linux 2.6.32 kernel source  
  – http://lxr.linux.no/#linux+v2.6.32/

• “Probing the Guts of Kprobes”  
  – Ananth Mavinakayanahalli, Prasanna Panchamukhi, Jim Keniston, Anil Keshavamurthy, Masami Hiramatsu  
  – Proceedings of the Ottawa Linux Symposium, 2006
How a Kprobe Works

Diagram based on the figure in “Probing the Guts of Kprobes” (Mavinakayanahalli et al. ’06)