drgn Writing to Memory and Breakpoints

LINUX PLUMBERS CONFERENCE 2023

Omar Sandoval

https://github.com/osandov/drgn
Agenda

• Quick introduction to drgn
• Memory writing and breakpoint basics
• Why in production?
• Brainstorming: mechanism and API
Introduction to drgn

- “Programmable debugger” in Python
- Building blocks: objects, types, stack traces
- Kernel-specific “helpers”
- Complex interactive sessions and scripts
Memory Writing and Breakpoints

- drgn is currently read-only
- Users have been asking for read-write features: overwriting memory and setting breakpoints
- Makes sense for development workflows (e.g. in QEMU over gdbstub)
Memory Writing API Proposal

# Write bytes to an address.
prog.write(address, bytes)

# Set the value of an object in memory.
user = find_user(prog, 0)
user.locked_vm.counter.write_(0)
Breakpoint API Proposal

# Set a breakpoint.
prog.set_breakpoint(address)
prog.set_breakpoint("function_name")
prog.set_breakpoint("file_name.c:lineno")

while True:
    # Wait for a thread to hit a breakpoint.
    event = prog.get_thread_event()

    # Get some information from the event
    stack_trace = event.thread.stack_trace()
    print(stack_trace)
    print(stack_trace[1]["local_variable"])

    # Resume the thread.
    event.thread.resume()
Why In Production?

- Quick-and-dirty mitigation before a livepatch or kernel update
- Fix reference count bugs, accounting over/underflows, invalid states, etc.
- Example: 981a37bab5e5 ("btrfs: properly enable async discard when switching from RO->RW")
Brainstorming

- Memory writing ideas
  - Bring back /dev/kmem
  - Custom kernel module
  - KGDB
- Breakpoint ideas
  - KGDB
  - BPF?
    - (Might need watchdog that kicks threads that have been stuck too long)
- Access control ideas
  - CAP_SYS_ADMIN and/or CAP_SYS_MODULE
  - Keyring