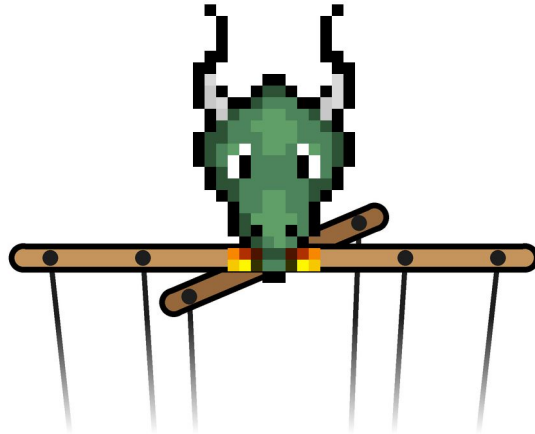


Breakpoints and Execution Control in drgn

Omar Sandoval



Background

- drgn is read-only
 - Except “kmodify”, which allows calling arbitrary functions and writing to memory
- Breakpoints are the most-requested feature
- Execution control is stopping, resuming, single-stepping, etc.
- Most functionality is currently prototyped

Goals

- Nice, programmable API for scripting and interactive use
 - This is the hardest part
- Backends
 - For kernel development: GDB Remote Serial Protocol (KGDB and QEMU gdbstub)
 - For userspace: ptrace
 - For production kernels: kprobes via custom module

API

```
events = collections.deque()
prog.add_event_listener(lambda prog, event: events.append(event))

with prog.attach():
    prog.set_breakpoint(prog.symbol("vfs_unlink").address)
    while not events:
        prog.poll()
    event = events.popleft()
    if not isinstance(event, drgn.ThreadStopEvent):
        continue
    thread = event.thread
    print(thread.stop_reason)
    print(thread.stack_trace())
    thread.cont()
```

Use Cases

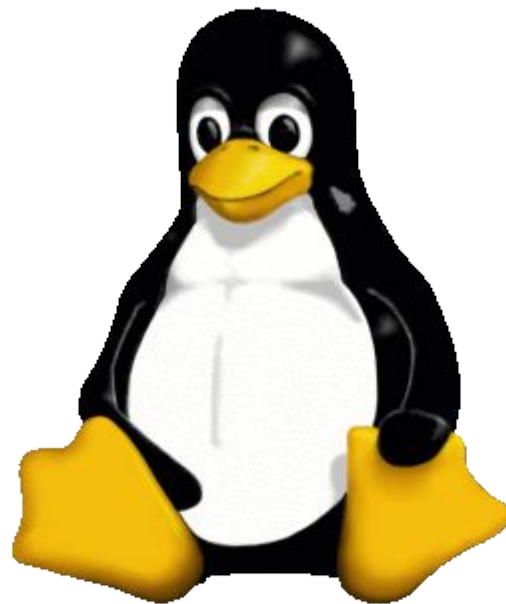
Reproducing Race Conditions

- Use breakpoints to reproduce suspected interleaving
- Proof-of-concept works for some races
 - [920f893f735e \("posix-cpu-timers: Prevent UAF caused by non-leader exec\(\) race"\)](#)
 - [58a8108bc73d \("perf: Fix use-after-free when perf mmap\(\) revival races with the last munmap\(\)"\)](#)
- But no backend works in every context

Backend Limitations

KGDB

- All-or-nothing control
- Must be initiated from target (e.g., sysrq-g)
- Protocol implementation is very minimal



QEMU gdbstub

- “Threads” in protocol are CPUs
- No non-stop mode



ptrace

- SIGCHLD and waitpid(2) are terrible interfaces for this
 - drgn is a library, so can't clobber signal handler or reap unrelated processes

Kprobes

- Need custom out-of-tree module
 - Could extend kmodify?
- Kprobes can't sleep, so need trampoline

Discussion