Powerful and Programmable Kernel Debugging with drgn

KERNEL RECIPES 2022

Omar Sandoval

https://github.com/osandov/drgn

Meta

What Is drgn?

- “Programmable debugger”
- Wraps up target’s variables and types so that they can be used from Python
- Works on the live Linux kernel and kernel core dumps (and userspace programs)
- Provides a library of kernel-specific “helpers” for common data structures

Why drgn?

- I came up against some very tricky bugs
- Existing tools weren’t enough
  - GDB’s scripting interface and Linux kernel support were clunky
  - Crash wasn’t flexible enough
  - BPF, ftrace, printk don’t work for post-mortem debugging
- Designed to be usable as a library

Tutorial

Review

```
# Look up a global variable.
variable = prog["variable"]

# Operate on the variable.
variable.member + 1

# View helpers.
help(drgn.helpers.linux)

# Get a stack trace for a thread ID.
trace = prog.stack_trace(123)

# Get a stack frame.
frame = trace[1]

# Look up a local variable.
variable = frame["variable"]
```

Case Study

Case Study Background

- Got a bug report that container creation was failing with `ENOSPC`
- Using `strace` and `retsnoop`, found that this was coming from a limit on the number of IPC namespaces
- But we only had a handful of IPC namespaces

Advantages of Debugging With drgn

- Feels like programming!
- Familiar environment for both C and Python coders
- Scripts can be reused, shared

Implementation

- libdrgn: C library implementing core functionality
  - Core abstractions
  - DWARF debugging information parsing
  - Memory reading (`/proc/kcore`, core dumps, `/proc/<pid>/mem`)
  - Language emulation
- Python bindings for libdrgn
- Helpers: Python code using core drgn library to provide common functionality
- Command line interface

Limitations

- Racy for live targets
- Helpers need to be kept in sync with kernel changes
    - drgn has an extensive test suite run against many kernel versions
- Needs DWARF

DWARF-Less Debugging

- Kernel is almost self-describing thanks to BTF, ORC, and kallsyms
- With a bit more information, can use (most of) drgn without DWARF
  - Work in progress by Stephen Brennan
- Mainly: need to add all variables to BTF (~4MB -> ~6MB)

Beyond Debugging

- Originally envisioned as just an interactive debugger
- But designed as generic API for introspecting programs
- Enables many more use cases
    - Learning tool
    - Automation
    - Replacing in-kernel introspection (e.g., debugfs)
    - Userspace memory profiling?!

Future Work

- Always adding more helpers, tools
- Debug info discovery improvements (including DWARF-less debugging)
- Making more information accessible programmatically
- Feature parity on other architectures
- Better support for userspace and C++
- Tracing APIs (breakpoints, single stepping, etc. via `ptrace`, `gdbstub`)

Conclusion

- drgn makes it easy to debug large, complex programs like the Linux kernel
- Has powerful building blocks that can be used for other use cases
- Try it! File feature requests, bug reports, and pull requests at https://github.com/osandov/drgn

Conclusion

- drgn makes it easy to debug large, complex programs like the Linux kernel
- Has powerful building blocks that can be used for other use cases
- Try it! File feature requests, bug reports, and pull requests at https://github.com/osandov/drgn
- Questions?

Meta