Optimization reverse-engineering w/ Cutter

Hadrien Grasland

2021-03-31

Background

- Compiler optimizers can sometimes act strangely…
  - Seemingly vectorizable code isn’t auto-vectorized
  - Increasing optimization level slows program down
- …and they aren’t good at explaining themselves
  - -fopt-info & friends are both verbose and cryptic
- Program disassembly is often the main source of info

Classic tools

- Compiler Explorer (https://godbolt.org/) is a nice online tool
  - Allows comparing output of many compilers
  - *Some* assembly-source correlation, nice x86 docs tooltips
  - BUT ill-suited to larger programs with dependencies
- Typical local tools are... less nice
  - objdump's output gets huge on larger programs
  - perf annotate points to hot functions, that can still be large

Cutter (https://cutter.re/)

- Free & open-source GUI reverse-engineering tool
- Mainly does disassembly, decompilation and debugging :
  - Decomposes assembly code into a control flow graph
  - Allows stepwise execution & breakpoints
  - Can translate simple assembly patterns into C code
- Based on Rizin (radare2 fork) and Ghidra (decompiler)

Usage example

- A radio-astronomy correlator mostly sums A.conj(B) products
- First draft >2x slower with -O3 than -Ofast using std::complex...
  - ...while the difference was small with thrust::complex
  - Code author wanted me to explain this difference
- Using godbolt on that code would require much adaptations
  - Everything tucked into main() : compute, timing, I/O…
  - Calls both thrust and an exotic in-house library

Demo

Conclusion

- Cutter is a nice tool for analyzing larger programs
  - Good complement to other tools like godbolt & perf
- Of course, it is not flawless:
  - Debugger is immature (slow and buggy)
  - More focused on logic-heavy code than compute-heavy one
  - Project's AppImages work well, but recompilation is hard

Thanks for your attention!