arm

Practical fuzzing for C/C++ compilers

arm-public-2024-template - Copy

Oliver Stannard

2024-04-10

AI-generated image

Overview

- Random code generators
  - csmith, yarpgen
  - cctest
- Running fuzzers
  - Compiler option selection
  - Reducing & reporting bugs
  - Dealing with expected failures

What do I mean by "practical"?

- Aiming to find high-priority bugs
  - Miscompilation > crash
  - C/C++ > IR/MIR
- Targeting bug-prone parts of compiler
  - Calling convention
  - Stack layout
  - New architectures/features
- Differential testing
  - Avoid re-implementing expected compiler behaviour

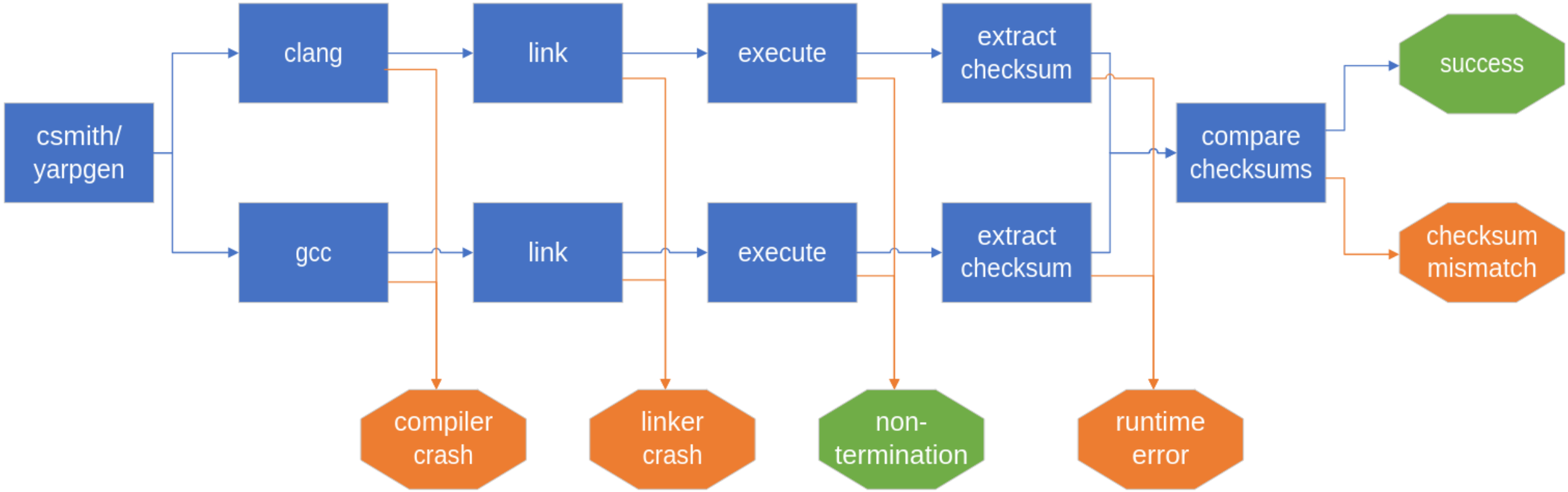
Random code generators

Csmith, yarpgen

- Open source tools to generate random C programs
- Generated programs:
  - Guaranteed free of UB
  - Not guaranteed to terminate (but most seeds will)
  - Prints CRC of global variables at end
  - Value of CRC not known by generators

- Csmith: more complex code
- Yarpgen: more structured loops

Csmith/yarpgen test flow

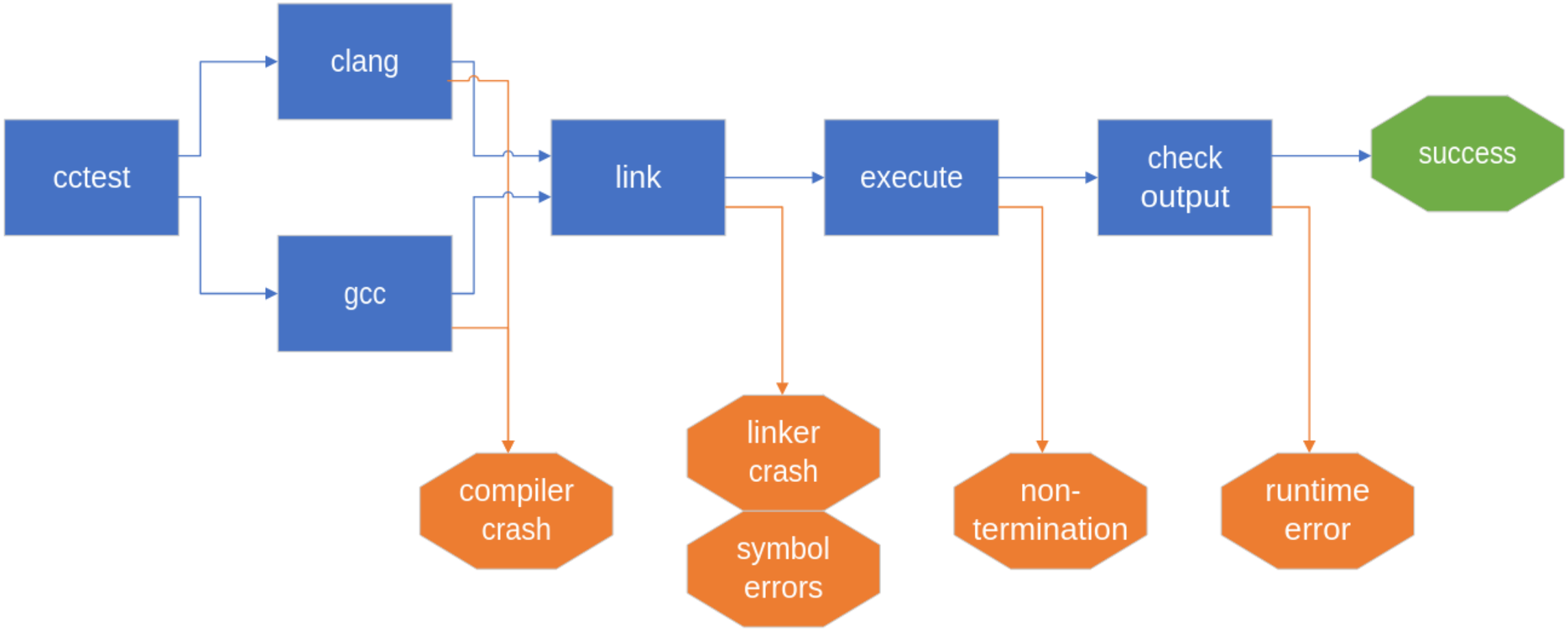


arm

cctest (calling convention test)

- Written by me, 2017-present
- Random C/C++ program generator to test calling convention
- Generates 2 source files and a header
- Function calls between files, with random argument/return types
- Assertions to test argument values
- Features:
  - Integer, float, pointer, complex types
  - Enums, structs, unions
  - Bitfields, including zero-size and over-size
  - Neon, MVE and SVE vectors
  - Packed/aligned attributes
  - Variadic functions
  - C++ exceptions, longjmp
  - Variable-size and over-aligned stack objects
  - CMSE security state transitions
  - Tail calls, indirect calls

cctest test flow



Running compiler fuzzers

Picking options to test

- Compiler
- Architecture
- FPU
- Endianness
- ISA
- ABI
- LTO
- Optimisation level
- Unaligned access
- PAC
- BTI
- MTE
- UBSan
- CFI
- Stack protector
- Auto var initialisation
- Used register zeroing
- Fast-math
- Position-independent code
- Debug info
- Frame pointer
- Execute-only
- Straight-line speculation
- Speculative load hardening
- Shadow call stack
- Code model

- Want to test as many combinations as possible
- Some combinations are invalid
- Different levels of compatibility:
  - Same implementation-defined behaviour
  - Can be linked together

Reducing and reporting failures

- Compiler/linker crashes:
  - Easy case, reduce with creduce, raise ticket
- Csmith miscompilations
  - Creduce will reduce to UB is not careful
  - Script checks with sanitisers, valgrind, static analysis
  - Works ~90% of the time
  - Otherwise, must reduce manually
  - Decide which compiler is buggy
- Cctest miscompilations
  - Architecture-specific code (e.g. vector intrinsics) makes using creduce hard
  - Assertions give line number
  - Manually reduce by deleting calls
  - Decide which compiler is buggy

Expected failures

- Different xfail strategy needed to normal test suites

- Compiler/linker crashes:
  - Match strings in stderr

- Miscompilations:
  - Do not run affected compiler options (or combination)
  - Do not generate affected code
  - Match runtime error message
  - Match pattern in generated code

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు