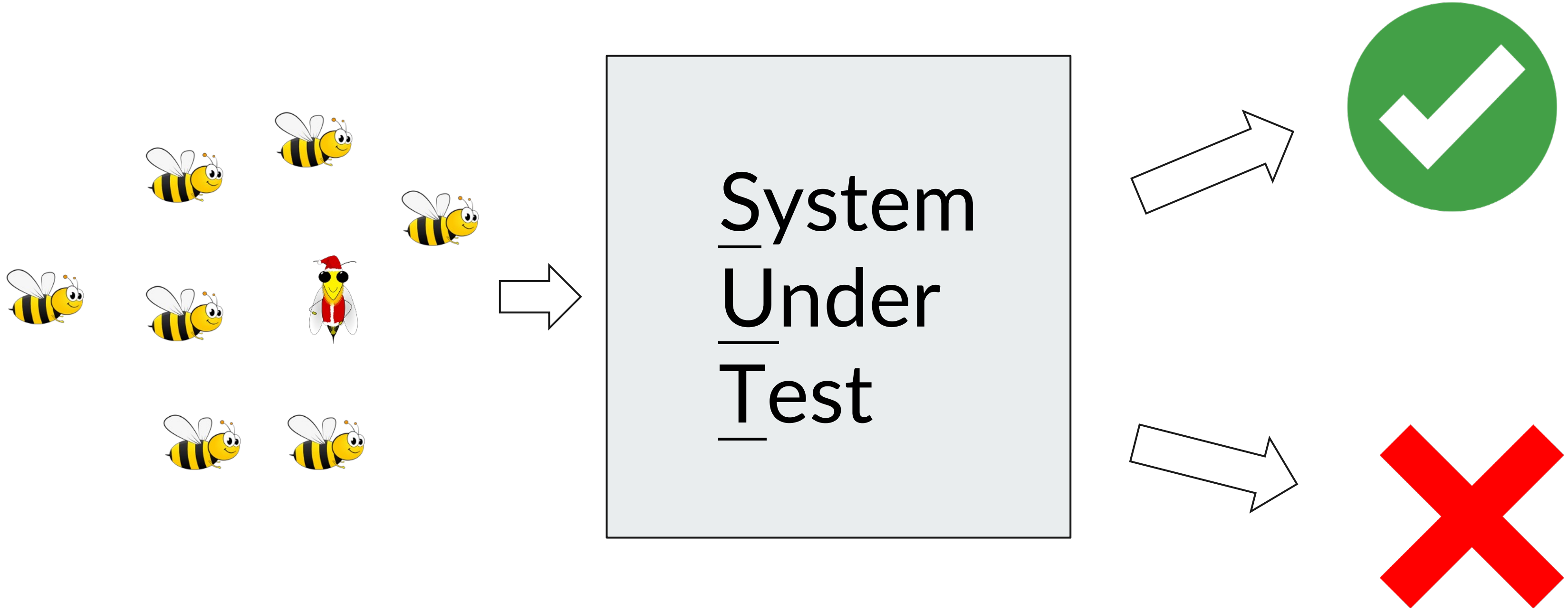
GrayC: Greybox Fuzzing of Compilers and Analysers for C

Karine Even-Mendoza[1,2], Arindam Sharma[1], Alastair F. Donaldson[1] and Cristian Cadar[1]

[1] Imperial College London

[2] King's College London

Fuzzing



Compiler Fuzzing Categories

intel / yarpgen

Csmith

GrayC

?

Greybox Fuzzing

- Successful for testing general software
  - Google: ~9k vulnerabilities and 28K bugs in 850 projects
- Not yet effective in compiler testing
  - Random, byte-level: High levels of invalidity
  - Tends to exercise the shallow (lexer, parser etc) parts of the compiler
- Attempts for static languages include keyword dictionaries, protobuf descriptions of language structure, regular expressions for common patterns
  - Still produce a high rate of invalid programs
  - Clang-Proto-Fuzzer: bugs are being fixed too slow (if at all)
  - No-fuss Compiler Fuzzing: code that crashes a C or C++ compiler, but that includes extensive undefined behaviour may well be ignored by developers.

GrayC

- Greybox fuzzing for testing compilers for C, a language with lots of UB

- Key idea: semantic-aware mutations
    - AST guidance
    - Modify individual programs or combine existing
    - A configurable level of aggressiveness
    - LibFuzzer: the underlying greybox fuzzing engine

Mutations

Mutators

- Duplicate Statement
- Delete Statement
- Expand Expressions
- Type Modifications

Recombiners

- Function Combination
- Function Body Replacement
- Code Fragment Addition

Recombiner Example

```
int dest_func(int x_dest, int y_dest){
   int b_dest = x_dest + y_dest;
   b_dest = b_dest + 5;
   return b_dest;
}
```

+

```
int source_func(int j_src, int k_src) {
   int m_src = j_src * k_src;
   return m_src;
}
```

```
int dest_func(int x_dest, int y_dest) {
   int j_src = x_dest;
   int k_src = y_dest;
   int m_src = j_src * k_src;
   int b_dest = x_dest + y_dest;
   b_dest = b_dest + 5;
   return b_dest;
}
```

Initialize variables corresponding to the src function to the args of dest function

Interleave statements from src function

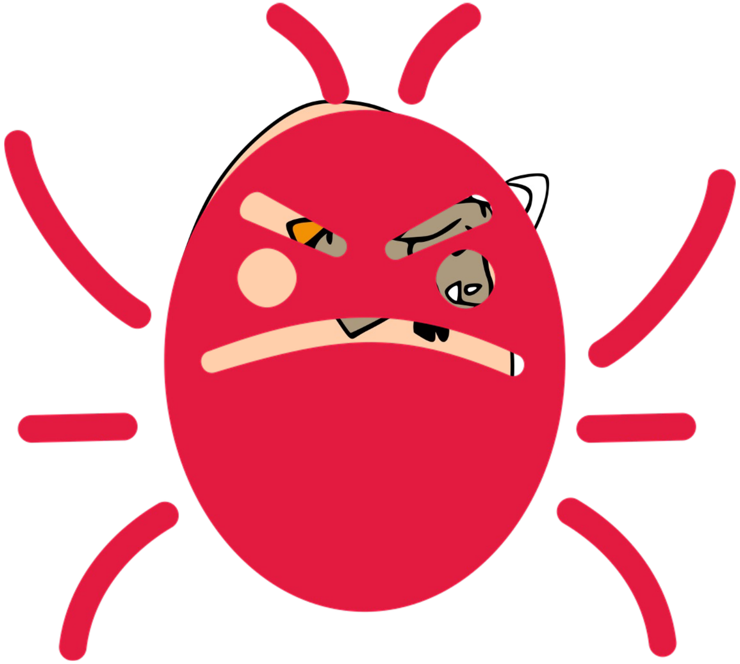
Randomly select return from src or dest

Evaluation in the Wild

	Front-end	Middle-/Back-end
GCC	2	10
LLVM	1	2
MSVC	3	0
Frama-C	2	10
TOTAL	8	22

Bug: GCC (Middle-End) 11, 12 (Bugzilla: #103813)

```
struct a d;
struct a {
   int b;
   int c[]
} main() { d.c[1]; } d.c[1]; }.c[1]; }
```



Controlled Experiments

1)	GrayC	Our tool
2)	GrayC-No-Cov-Guidance	Does coverage guidance matter?
3)	GrayC-Fragments-Fuzzing	Only code fragments injection, no coverage (similar to LangFuzz)
4)	Clang-Fuzzer	Greybox fuzzing with byte-level mutations
5)	Csmith	Generative, grammar-based fuzzing
6)	Grammarinator	Grammar-based fuzzing (ANTLR C grammar)
7)	PolyGlot	Language-agnostic AFL-based fuzzer, based on semantic error fixing
8)	RegExpMutator	LibFuzzer-based fuzzer based that uses regexp-based mutations

8 tools, 24h, 10 repetitions

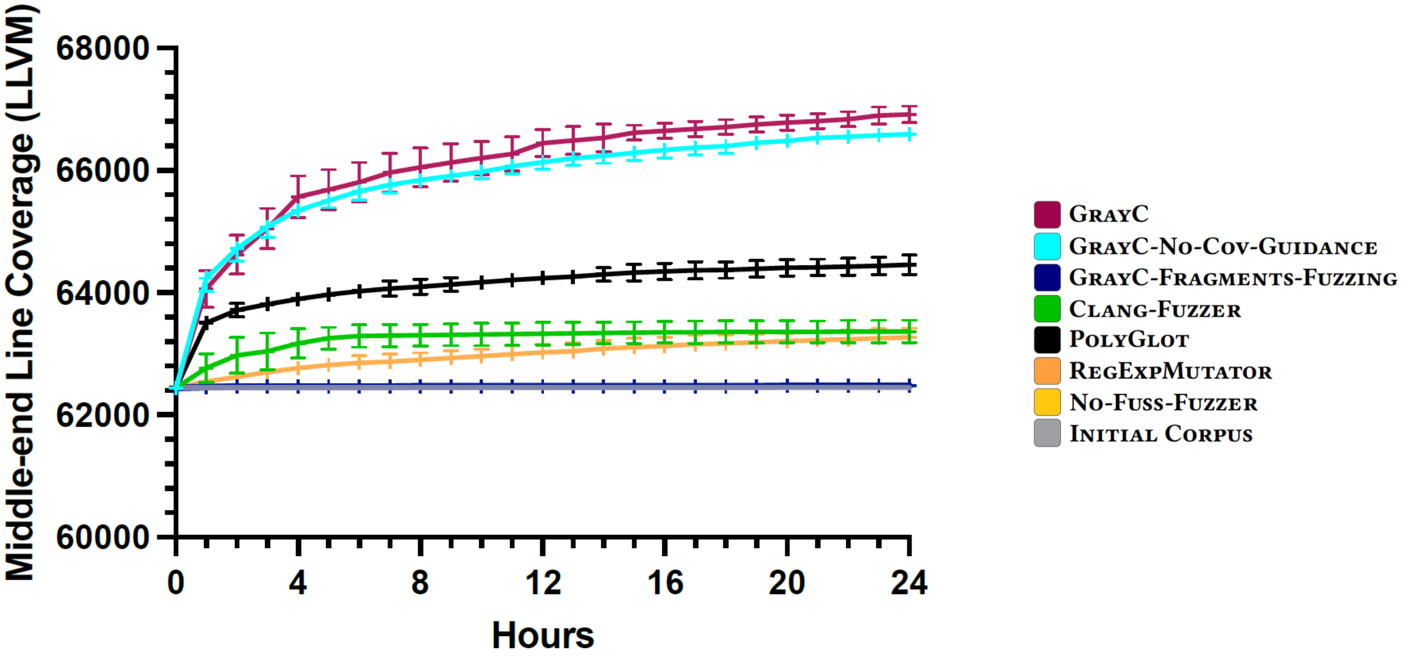
Throughput and Static Validity

GrayC is able to match the static validity rates of a generative fuzzer like Csmith

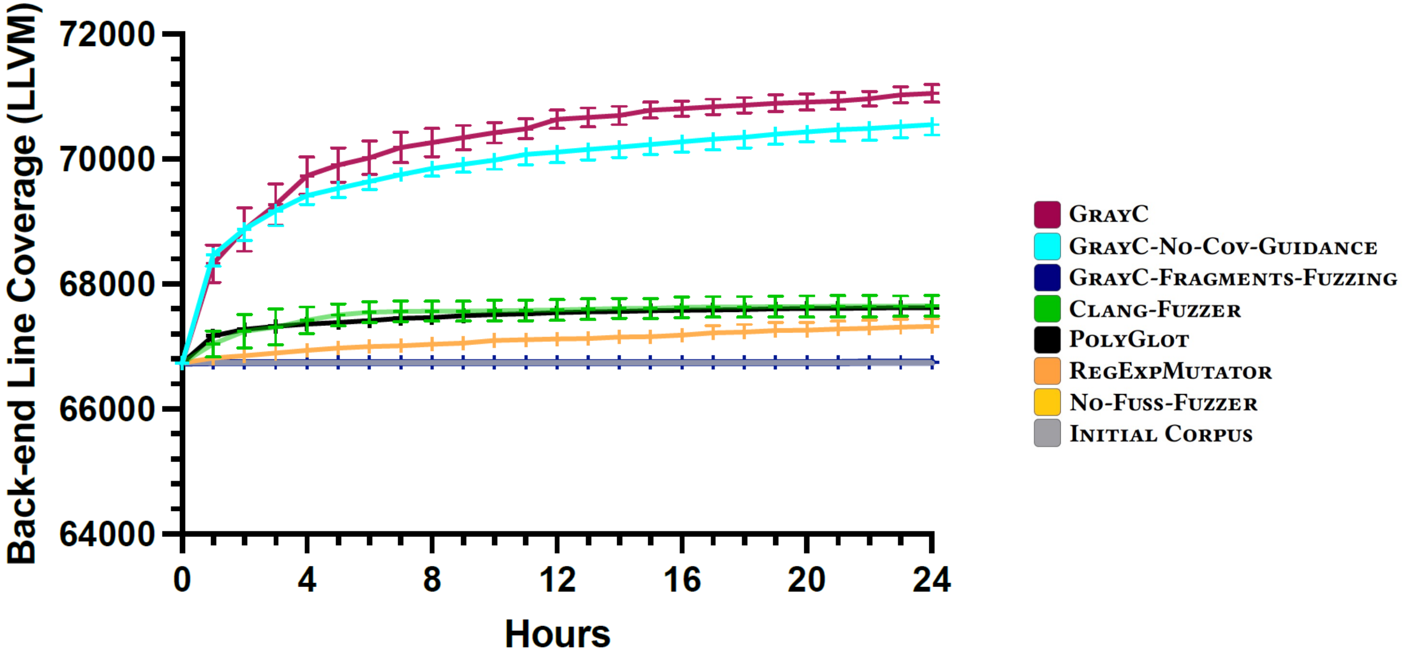
Vanilla greybox fuzzers and grammar-based fuzzers are primarily useful for solely testing the frontend

	Programs/h	Statically-valid (%)
CSMITH	1,144	99.96%
GRAYC	2,906	99.47%
GRAYC-FRAGMENTS-FUZZING	4,152	99.08%
. . .		
CLANG-FUZZER	1,183	1.55%
GRAMMARINATOR	5,391	0.0%

Compiler Middle-end Coverage



Compiler Backend Coverage



Bugs Found in 24h

Tool	Component		Fix Rate
	Middle	Front	
GrayC	6	-	100%
GrayC-No-Cov-Guidance	4	-	100%
RegExpMutator	2	1	67%
Clang-Fuzzer	1	4	60%
PolyGlot	-	1	0%

Testcase Contribution

We contributed **24** test cases to LLVM's test suite:
16 of them getting accepted
+
8 of them under review

Unit tests to improve code coverage

Adds a batch of C tests that have been found to cover several hundred lines of Clang/LLVM that are not covered by the unit and regression tests of the main LLVM project, nor by the test suite when run with the -O3 configuration.

The tests were originally generated using our fuzzer, and were then reduced using C-Reduce and some manual inspection. They have been checked for undefined behaviour-freedom using Frama-C and CompCert, and manually checked to eliminate implementation-defined behaviour.

Differential Revision: https://reviews.llvm.org/D118234

main
llvmorg-16.0.5 ... llvmorg-16.0.0-rc1

arindam-8 authored and **lenary** committed on Oct 11, 2022

Conclusion

Greybox compiler fuzzing for languages with extensive UB is feasible

Key idea is to use **AST-level** semantics- aware mutations

Significant gains in terms of bug-finding & coverage compare to prior work

GrayC found **30 bugs** (26 fixed), with 25 previously unknown (22 fixed)

We used GrayC to contribute 24 test cases (**16 accepted**) to the LLVM compiler