Automatic exploit generation

Maxime Bélair [1] Manh-Dung Nguyen [2] Emilien Fournier [3]
Tristan Benoit [4] Gabriel Sauger [5]

Subject by: Jules Villard -

[1]Orange Labs / IMT atlantique - maxime.belair@imt-atlantique.fr

[2]CEA LIST & Université Grenoble Alpes - manh-dung.nguyen@cea.fr

[3]ENSTA Bretagne / Lab-STICC - emilien.fournier@ensta-bretagne.org

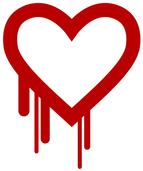
[4]LORIA - tristan.benoit@loria.fr

[5]LORIA - gabriel.sauger@loria.fr

Problem Overview

Context

- Bugs in devices
- Are they weaknesses ?



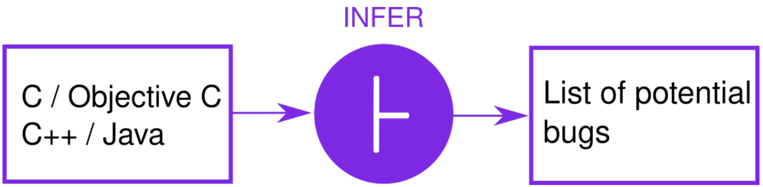
Formal challenge

Can we automatically turn static analysis reports into executable confirming the vulnerability of a program ?

Section example

Give an example of main.c with a bug We can show pictures or live performance. Ask the audience to detect the bug.

Infer tool



- Static analysis tool from Facebook
- **Capture** phase, then **Analysis** phase

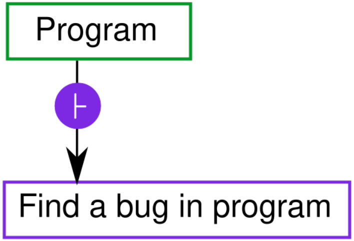
Infer tool example

Give an example of our use of Infer on main.c We can show pictures or live performance.

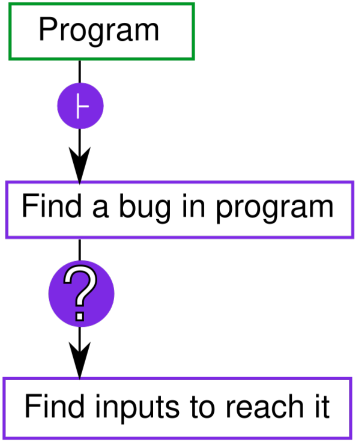
Practical approach

Program

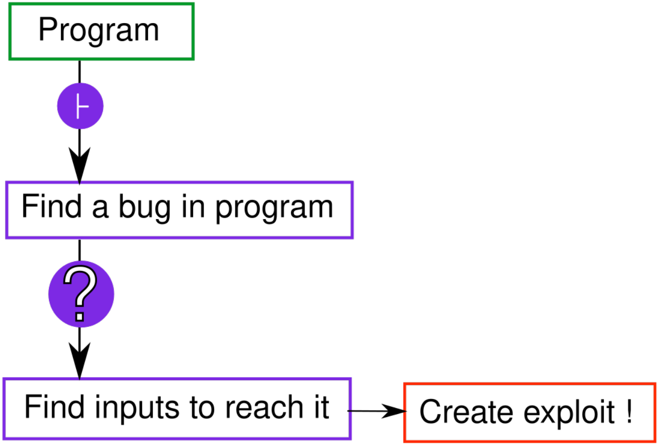
Practical approach



Practical approach



Practical approach



Practical approach

Practical challenge

Given the Infer information about bugs of a program A, create a program B that crashes A

Table of content

Proposed approaches

Approaches overview

Model checking

Present model checking solution with Divine

SMT solvers

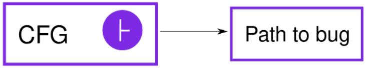
Present logic solvers

SMT Solvers

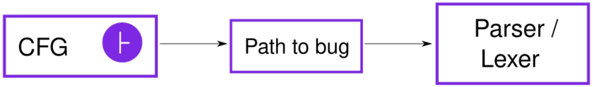
Compiler / Interpreter information

SMT Solver

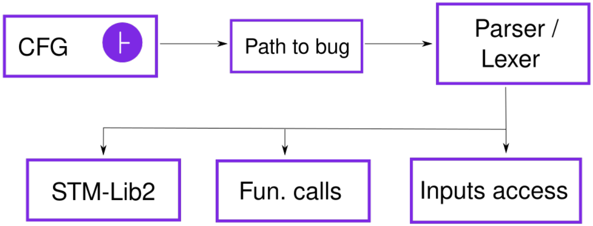
SMT Solver



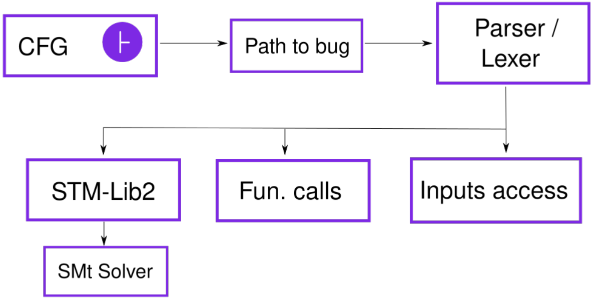
SMT Solver



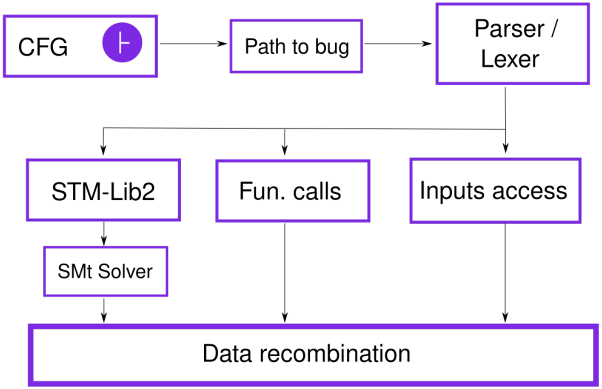
SMT Solver


CFG
Path to bug
Parser / Lexer
STM-Lib2
Fun. calls
Inputs access

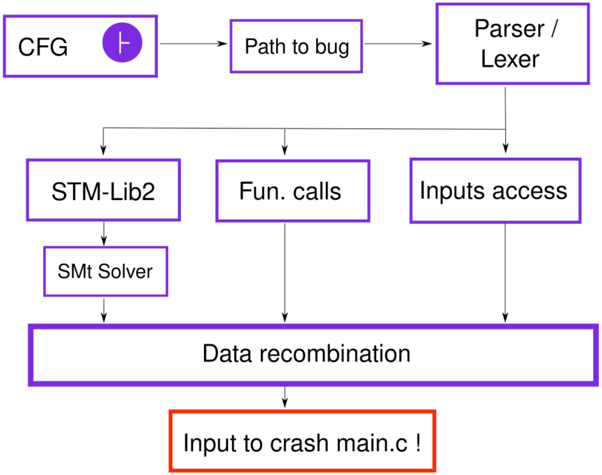
SMT Solver


CFG
Path to bug
Parser / Lexer
STM-Lib2
Fun. calls
Inputs access
SMt Solver

SMT Solver


CFG
Path to bug
Parser / Lexer
STM-Lib2
Fun. calls
Inputs access
SMt Solver
Data recombination

SMT Solver


CFG
⊦
Path to bug
Parser / Lexer
STM-Lib2
Fun. calls
Inputs access
SMt Solver
Data recombination
Input to crash main.c !

SMT results

Present the results we have and on which program. The performance review is NOT done here, but in Part 3/Result Comparison

Fuzzing technique

Present fuzzing background // add fuzzer logo

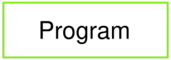
Coverage-guided Greybox fuzzing

Coverage-guided Greybox fuzzing

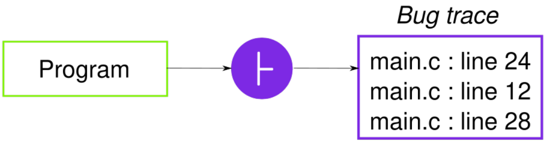
Motivations

Explain intuiton for our problem

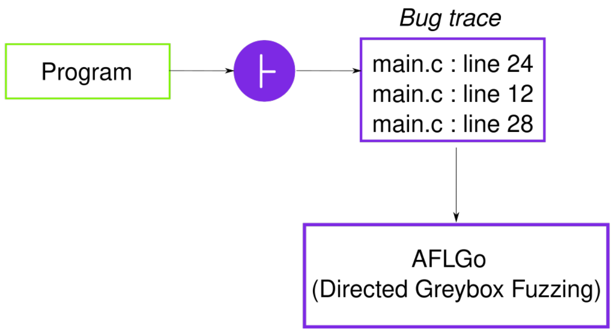
Fuzzing technique



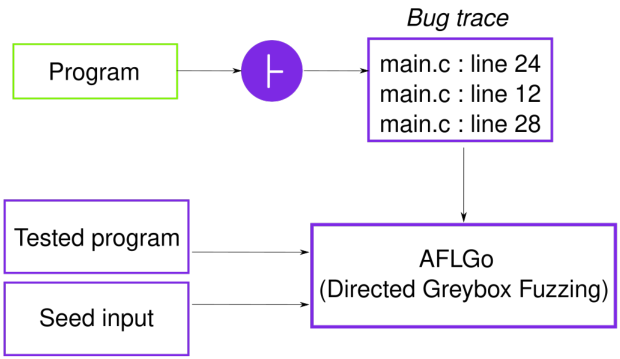
Fuzzing technique



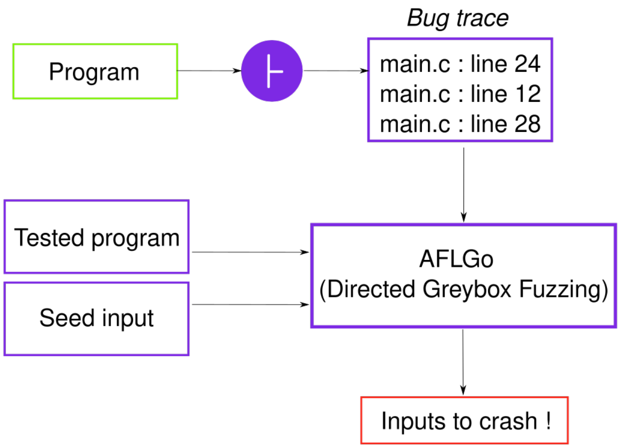
Fuzzing technique



Fuzzing technique



Fuzzing technique



Conclusions and perspectives

Results comparison

Show a table approaches / program comparing results (yes/no, running time, implementation complexity, computational complexity

Future work

Put eeeeeverything we think of. Ex:

- Create a fully automatic process
- **SMT approach**: Manage fonctions calls in main.c

Future Work

Add a graph of automatic exploits using expert models

Thank you Questions ?

See the title