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

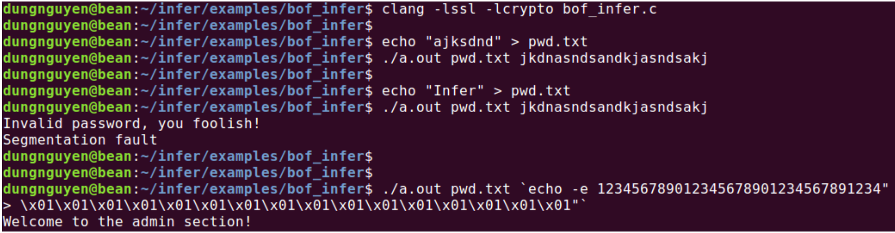
Context

- Bugs in devices
- Are they weaknesses ?

Formal challenge

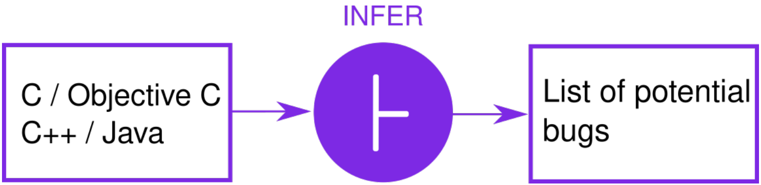
Can we automatically turn static analysis reports into executable confirming the vulnerability of a program ?

Program bug example



```
dungnguyen@bean:~/infer/examples/bof_infer$ clang -lssl -lcrypto bof_infer.c
dungnguyen@bean:~/infer/examples/bof_infer$
dungnguyen@bean:~/infer/examples/bof_infer$ echo "ajksdnd" > pwd.txt
dungnguyen@bean:~/infer/examples/bof_infer$ ./a.out pwd.txt jkdnasndsandkjasndsakj
dungnguyen@bean:~/infer/examples/bof_infer$
dungnguyen@bean:~/infer/examples/bof_infer$ echo "Infer" > pwd.txt
dungnguyen@bean:~/infer/examples/bof_infer$ ./a.out pwd.txt jkdnasndsandkjasndsakj
Invalid password, you foolish!
Segmentation fault
dungnguyen@bean:~/infer/examples/bof_infer$
dungnguyen@bean:~/infer/examples/bof_infer$
dungnguyen@bean:~/infer/examples/bof_infer$ ./a.out pwd.txt `echo -e 1234567890123456789012345678912334"
> \x01\x01\x01\x01\x01\x01\x01\x01\x01\x01\x01\x01\x01\x01\x01\x01"`
Welcome to the admin section!
```

Infer tool



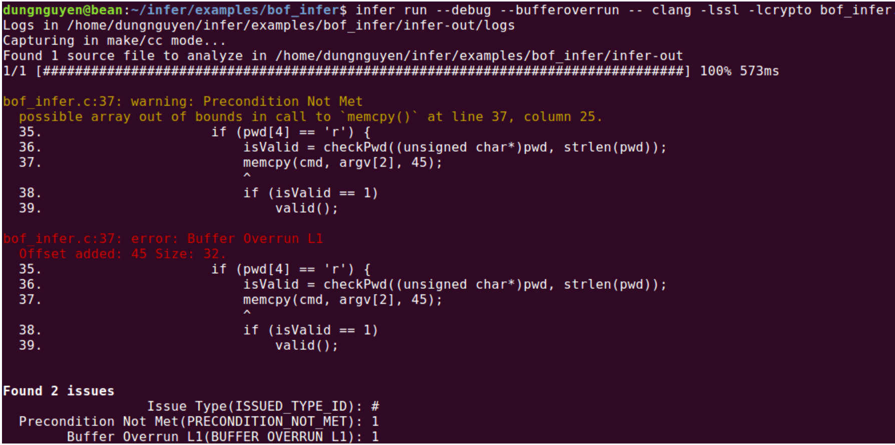
C / Objective C
C++ / Java

INFER

List of potential bugs

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

Infer tool example

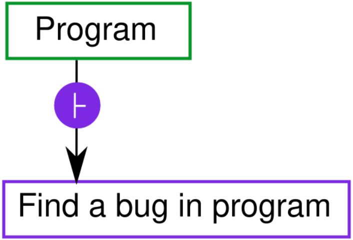


```
dungnguyen@bean:~/infer/examples/bof_infer$ infer run --debug --bufferoverrun -- clang -lssl -lcrypto bof_infer
Logs in /home/dungnguyen/infer/examples/bof_infer/infer-out/logs
Capturing in make/cc mode...
Found 1 source file to analyze in /home/dungnguyen/infer/examples/bof_infer/infer-out
1/1 [################################################################################] 100% 573ms

bof_infer.c:37: warning: Precondition Not Met
  possible array out of bounds in call to `memcpy()` at line 37, column 25.
  35.                         if (pwd[4] == 'r') {
  36.                             isValid = checkPwd((unsigned char*)pwd, strlen(pwd));
  37.                             memcpy(cmd, argv[2], 45);
                                ^
  38.                             if (isValid == 1)
  39.                                 valid();

bof_infer.c:37: error: Buffer Overrun L1
  Offset added: 45 Size: 32.
  35.                         if (pwd[4] == 'r') {
  36.                             isValid = checkPwd((unsigned char*)pwd, strlen(pwd));
  37.                             memcpy(cmd, argv[2], 45);
                                ^
  38.                             if (isValid == 1)
  39.                                 valid();


Found 2 issues
                  Issue Type(ISSUED_TYPE_ID): #
  Precondition Not Met(PRECONDITION_NOT_MET): 1
        Buffer Overrun L1(BUFFER_OVERRUN_L1): 1
```

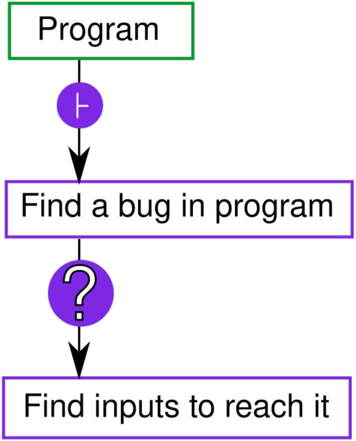
Practical approach

Program

Practical approach



Practical approach



Program

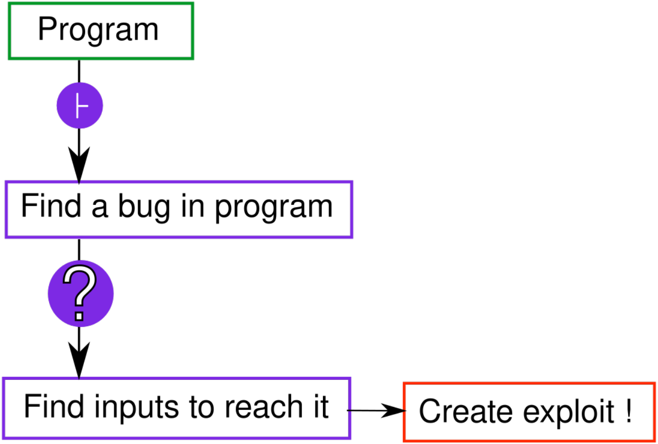
⊢

Find a bug in program

?

Find inputs to reach it

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

1 Problem overview
- Context
- Infer tool
- Practical approach

2 Proposed approaches
- Model checking
- SMT solvers
- Fuzzing technique

3 Conclusions and perspectives
- Results comparison
- Future Work

Proposed approaches

Model checking

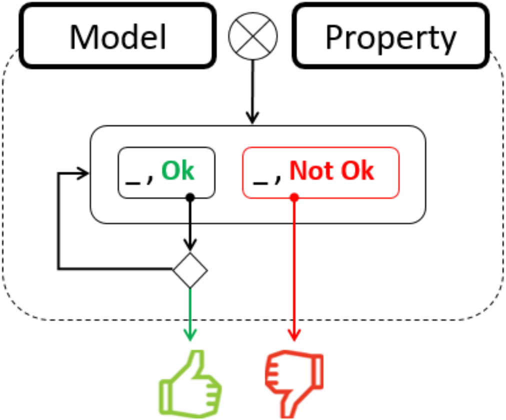
Model Checking

- Intuitive
- Automated
- Provides counter-example

× State-space explosion

What is it ?

- Fixed-point algorithm
- Plenty of algorithmic variations


Model
Property
_, Ok
_, Not Ok

SMT solvers

Present logic solvers

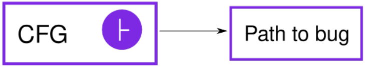
SMT Solvers

Compiler / Interpreter information

SMT Solver

CFG

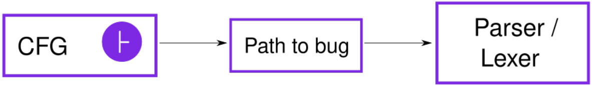
SMT Solver



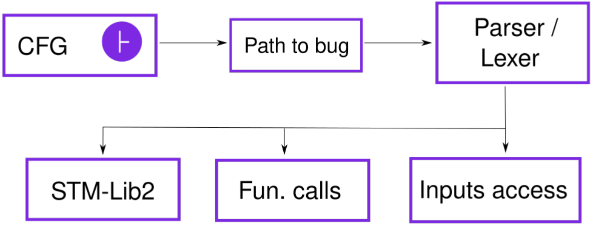
CFG ⊢

Path to bug

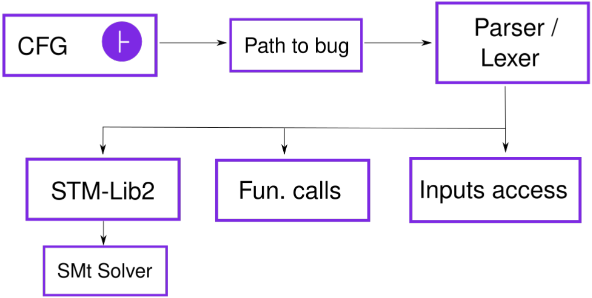
SMT Solver



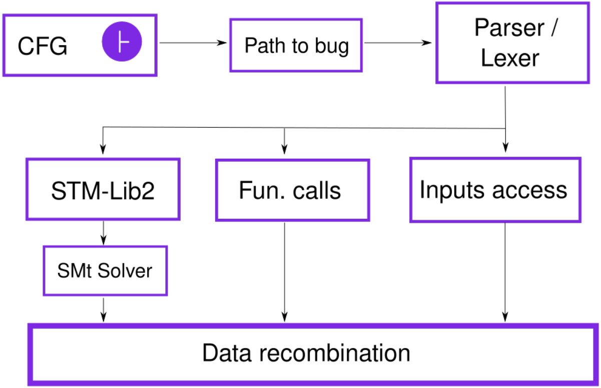
SMT Solver


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

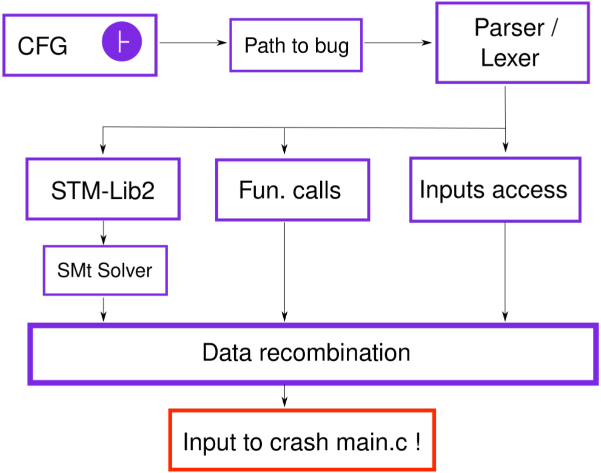
SMT Solver


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

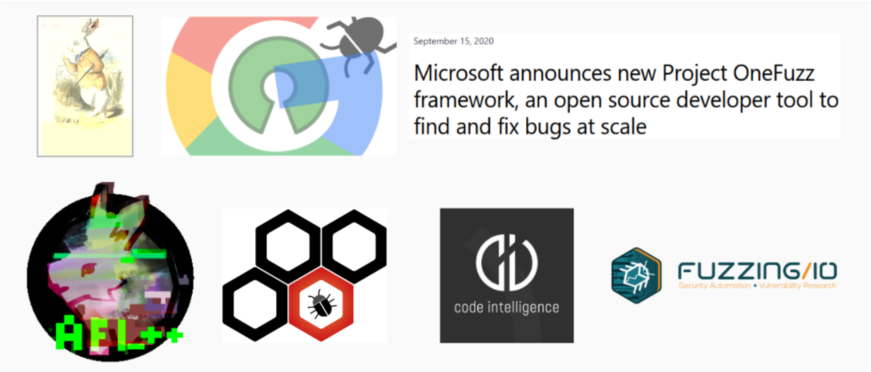
SMT Solver


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

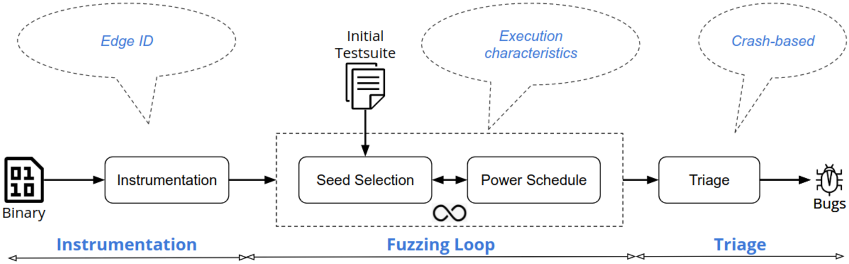
Fuzzing technique



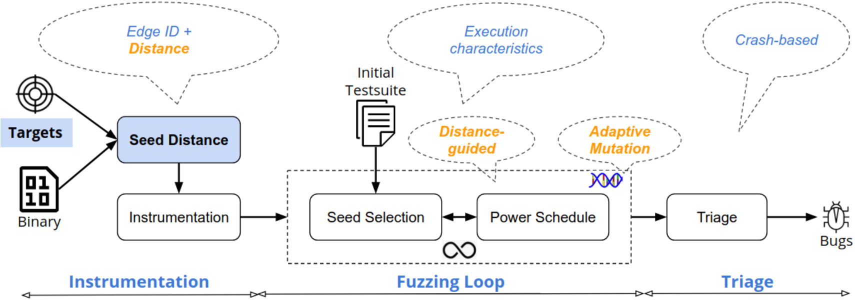
September 15, 2020

Microsoft announces new Project OneFuzz framework, an open source developer tool to find and fix bugs at scale

Coverage-guided Greybox fuzzing



Direct Greybox fuzzing

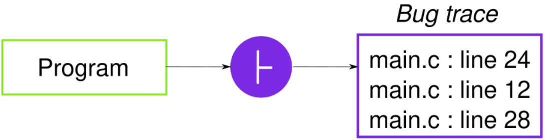


Motivations

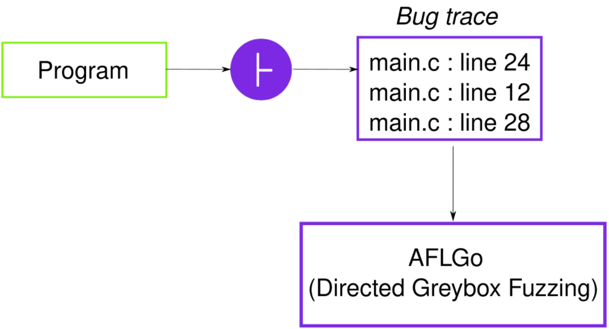
Explain intuiton for our problem

Fuzzing technique

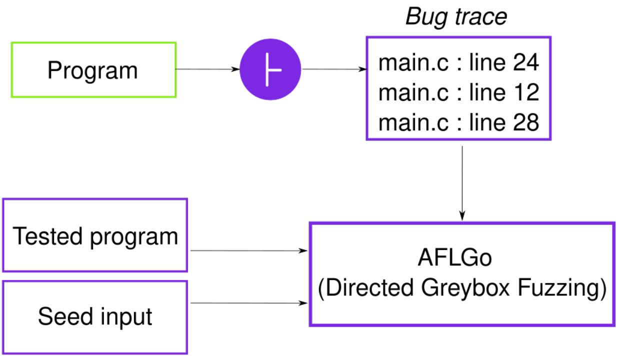
Fuzzing technique



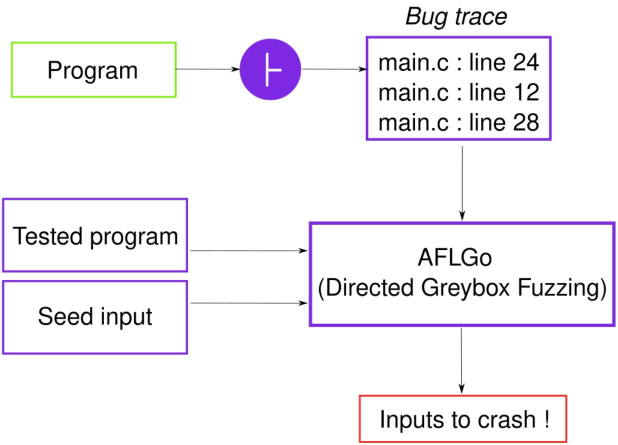
Fuzzing technique



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