Can a Fuzzer Match a Human?

Solidity Case Study

Bhargava Shastry
Ethereum Foundation

@ibags

bshastry

Fuzzer No Match for Human Tester, but...

- It can find security-critical bugs that a tester may have missed
  - Often elicits: “Oh, I hadn’t considered that!”
- Throw the kitchen sink at something
- Really useful for differential (A/B) testing

tl;dr:

- Threat model: Incorrect code generation
- Randomly generated valid Solidity programs test compiler
- Found 17 bugs using semantic fuzzing
- Continuous fuzzing for early bug discovery
- Virtually no Yul optimizer bugs post release in two years

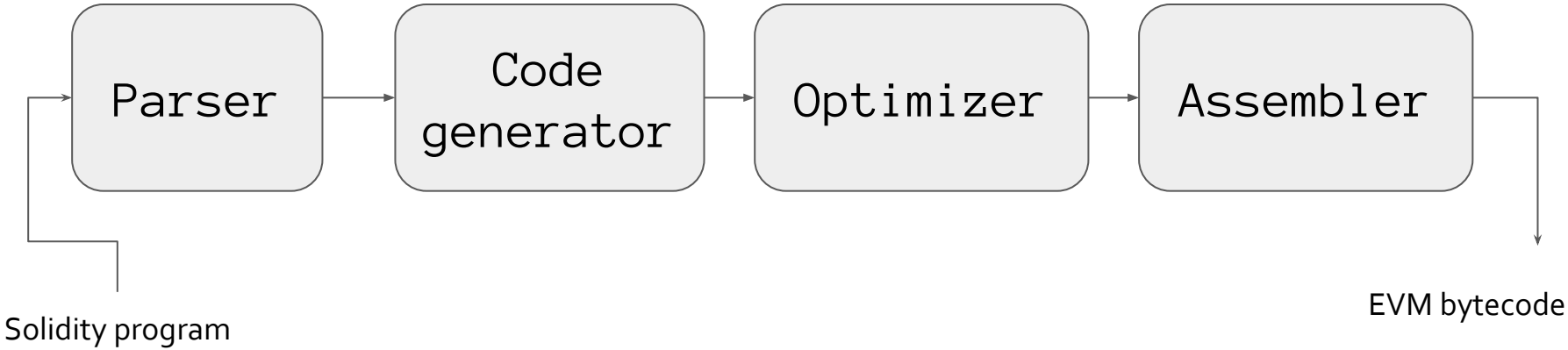
whoami

- Security engineer, Solidity team
- Semantic testing of Solidity compiler

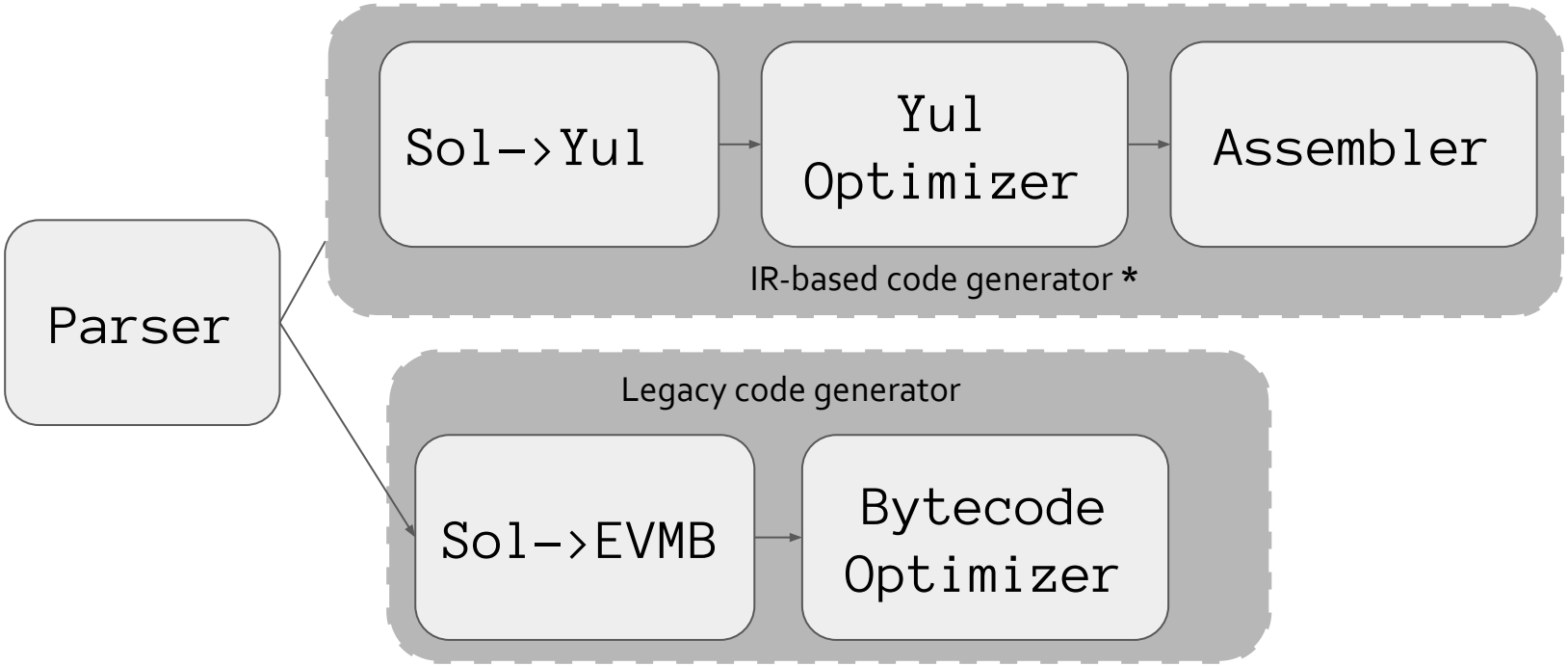
Find security-critical bugs in the compiler before it is shipped

Introduction

Compiler Overview



Code generators



* IR-based code generator also makes use of bytecode optimizer

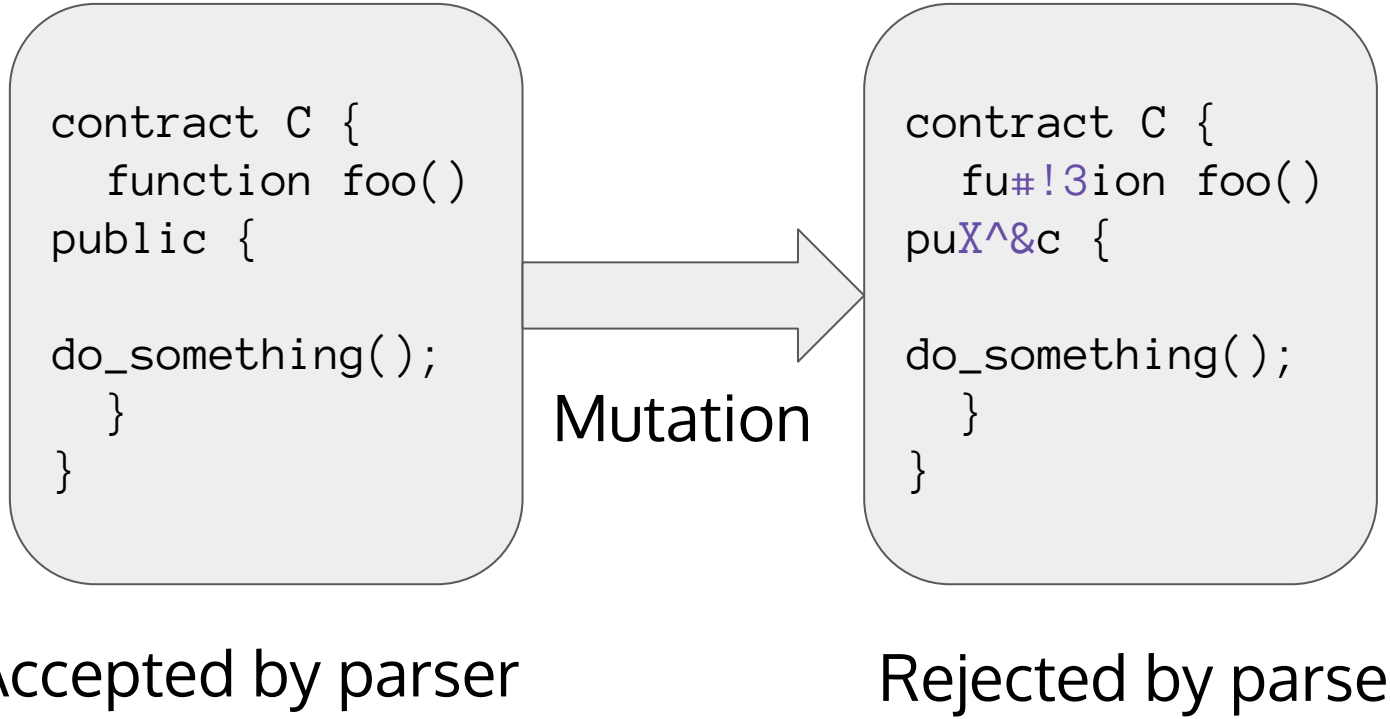
Threat model

- Compiler user (programmer) is not malicious
- Bugs introduced by the compiler itself
  - Optimizer(s)
  - Code generator(s)
  - Assembler
- Parser bugs are out of scope

Fuzz testing in a nutshell

```
while not ctrl + c

do

   input=gen_input()

   runProgram(input)

done
```


Limitation of random fuzzing



Fuzzing a compiler requires generating valid programs...

... generating a valid program requires structure awareness

Approach

Input Generation

- Input generation approached in two different ways
  - Grammar-based Solidity program generator written in C++ only
  - Protobuf based Yul program generator written using protobuf C++ binding

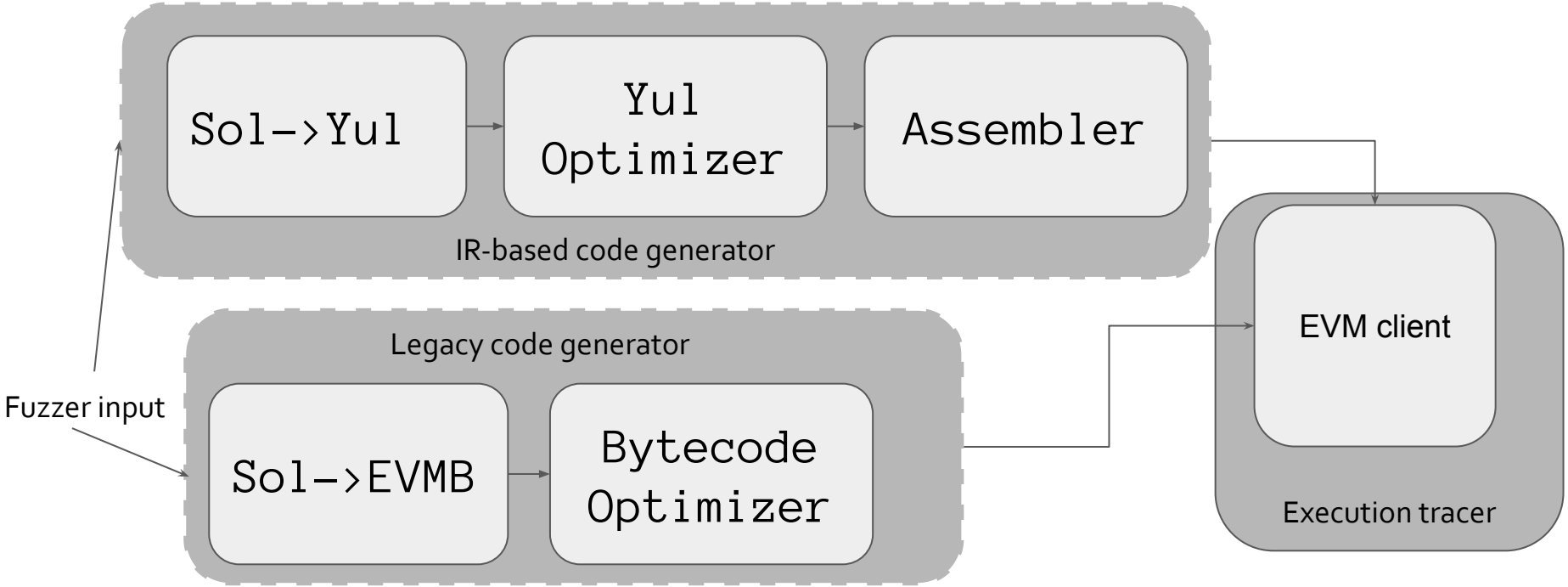
Differential Testing

- Always compare two entities in order to find bug in one of them
  - Optimized and unoptimised
  - Legacy and IR based code generators
- Execution Tracing approached in two different ways
  - EVM client based
  - Yul interpreter based

Grammar based Input Generation

- A full-fledged Solidity program generator written in C++
- Each fuzzer mutation is a randomly-generator program
- All programs are semantically valid

Finding bugs in the code generators



Yul Input Generation

Specification written in protobuf language

```
message Block {
  repeated Statement stmts;
}
...
message program {
  repeated Block blocks;
}
```

Full spec:
https://github.com/ethereum/solidity/blob/develop/test/tools/ossfuzz/yulProto.proto

Input generation

- Input generated and mutated by libprotobuf-mutator
- Each input is a tree

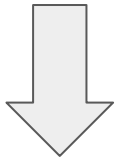
```
blocks { stmts { ifstmt { condition {
binaryOp { eq { op1: varref{id: 0} op2: 0}
} } } } }
```


Input conversion

- Converter is source-to-source translator
- Input: protobuf serialization format
- Output: yul program

Example

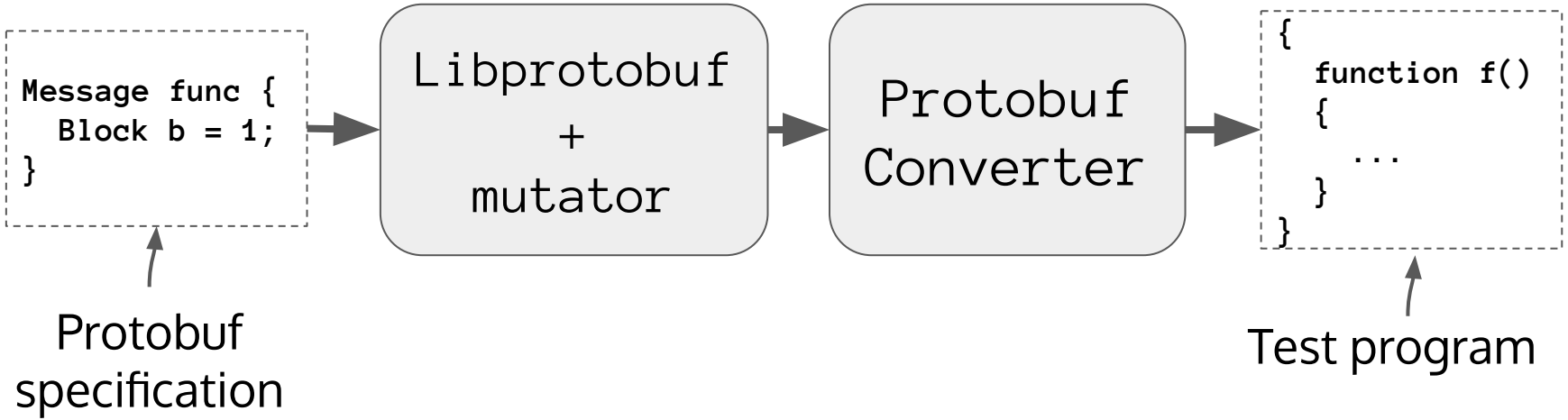
```
blocks { stmts { ifstmt { condition {
binaryOp { eq { op1: varref{id: 0} op2: 0}
} } } } }
```



Conversion

```
if x_0 == 0
```


Test program generation



Correctness testing requires encoding expectation somehow

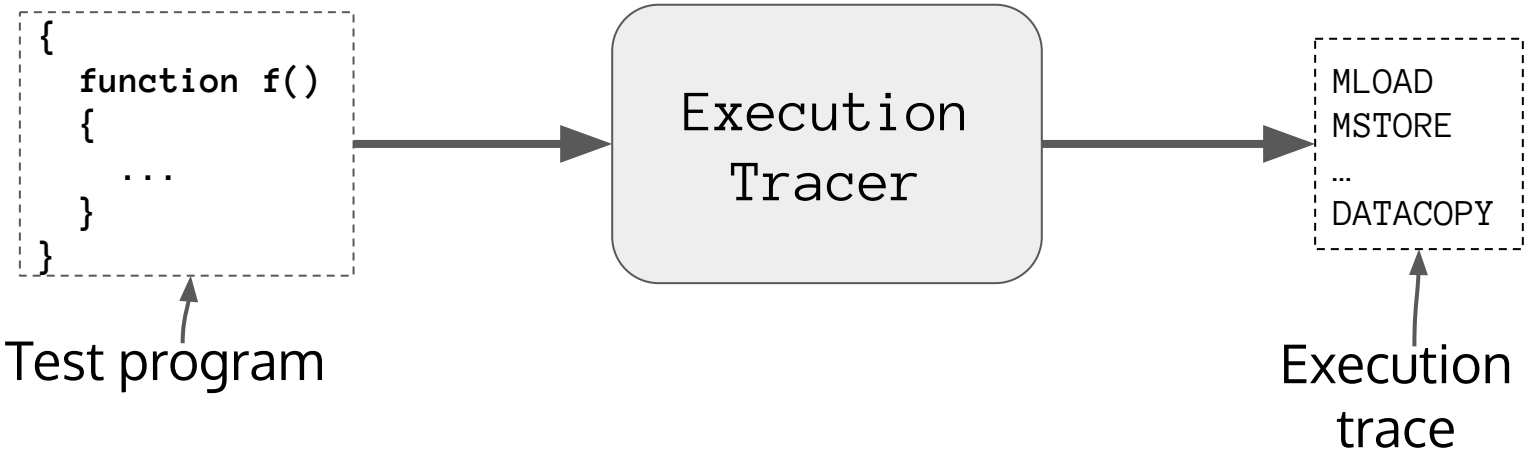
Differential testing

- Track side-effects of execution
- Run baseline and experiment programs
- Compare side-effects

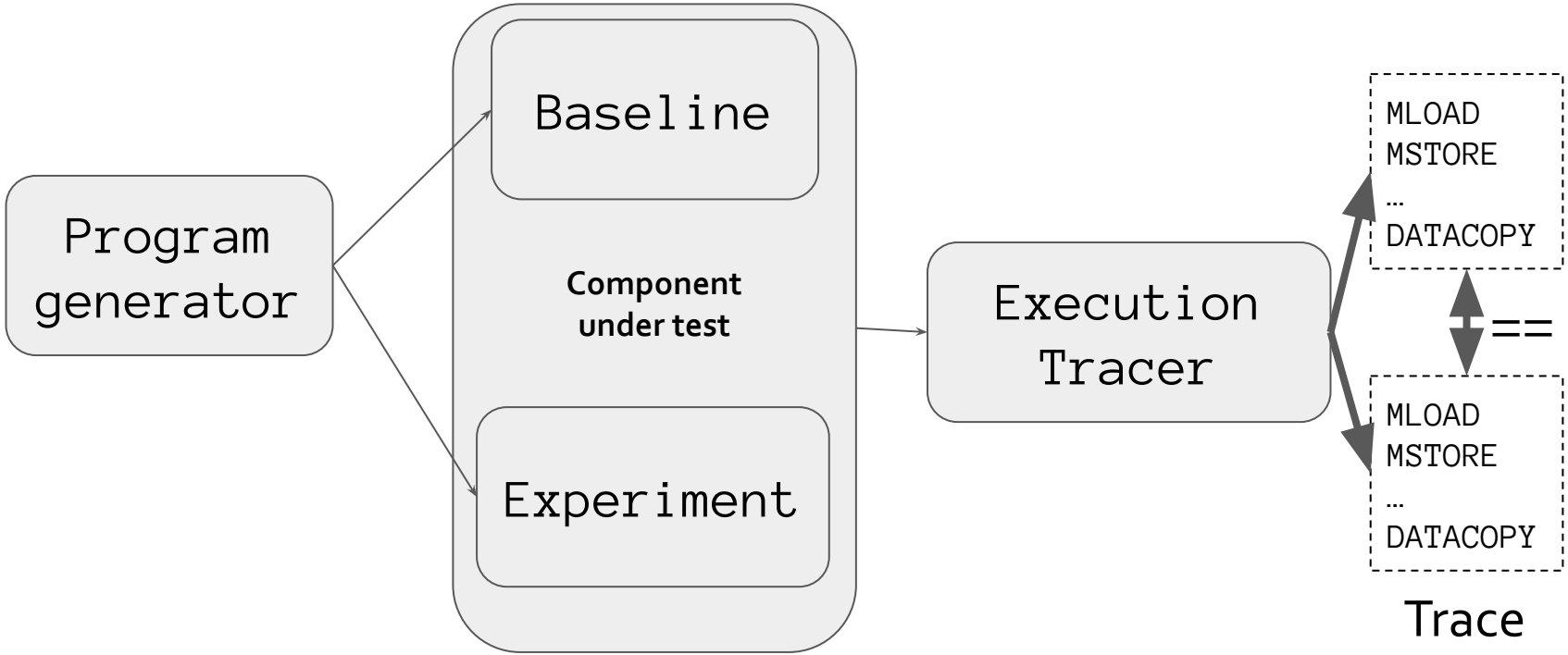
Execution Tracing

- Solidity programs drive EVM client (Evmone)
- Yul programs drive the Yul interpreter

Execution Tracing Overview

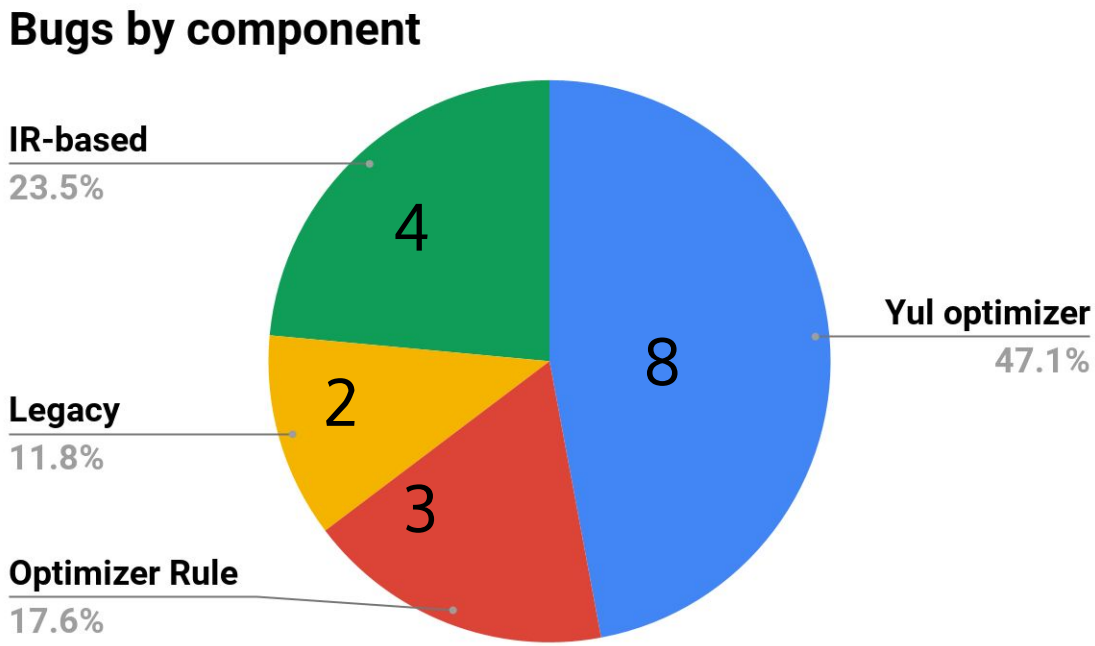


Fuzzing Setup

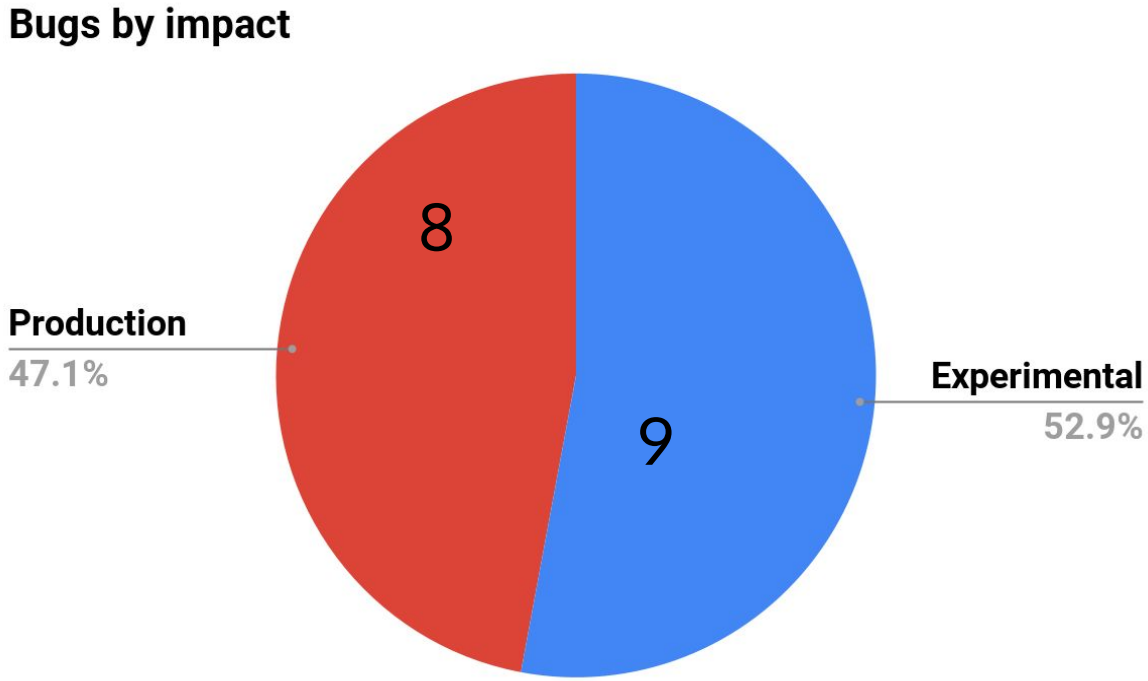


Results

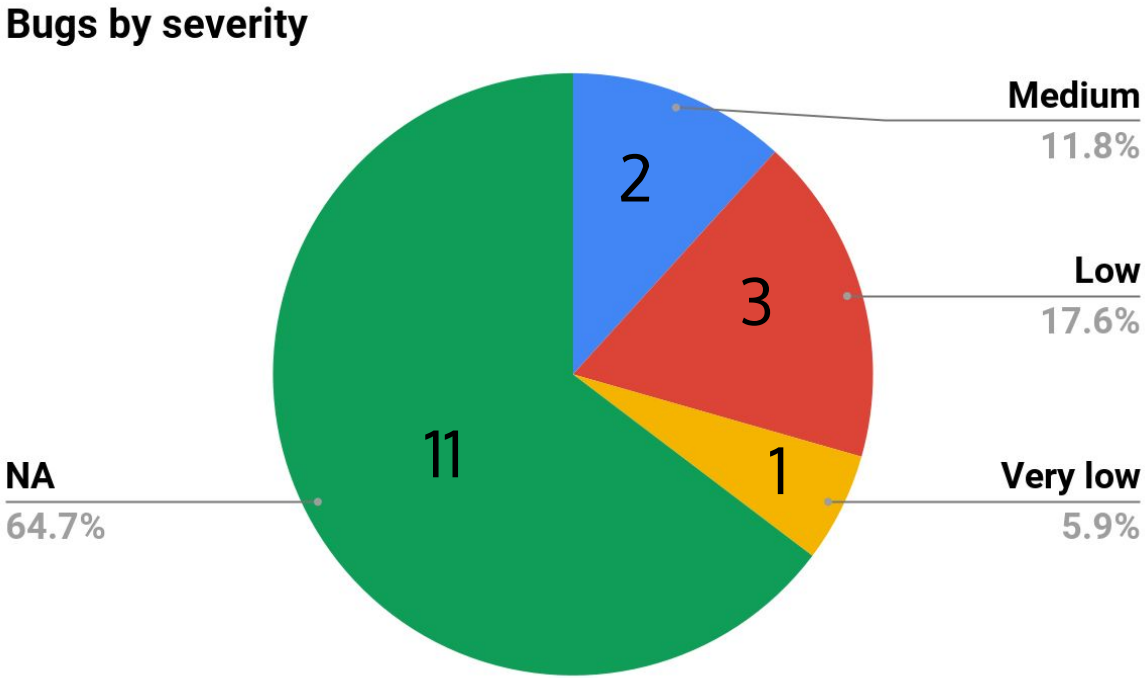
Bugs by component



Bugs by impact



Bugs by severity



Current Work

Two Bugs Required Human Assistance

- `returndatacopy(0, 1, 100)` inside a fallback function
  - HT @_hrkrshnn
- Storage corruption and empty push on bytes array
  - HT @ekpyron

Can Fuzzer Approach Humanness?

Heuristics + Randomness

- Pure randomness may be ill-suited sometimes
- Redundant memory store eliminator
  - Requires read location to be equal or not-equal to write location
  - Pure randomness will most likely not-equal than equal
  - Heuristic: Read from location that is already written to occasionally

Conclusion

Conclusion

- Continuous grammar-aware fuzzing for early bug discovery
- Useful for testing security-critical components of the Solidity Compiler
- Decent assurance
  - Evidence that it works
  - No formal guarantees though

Thank You!

ethereum/solidity.git

gitter.im/ethereum/solidity-dev