Open Discussion on Solidity Fuzzing

Bhargava Shastry

@ibags

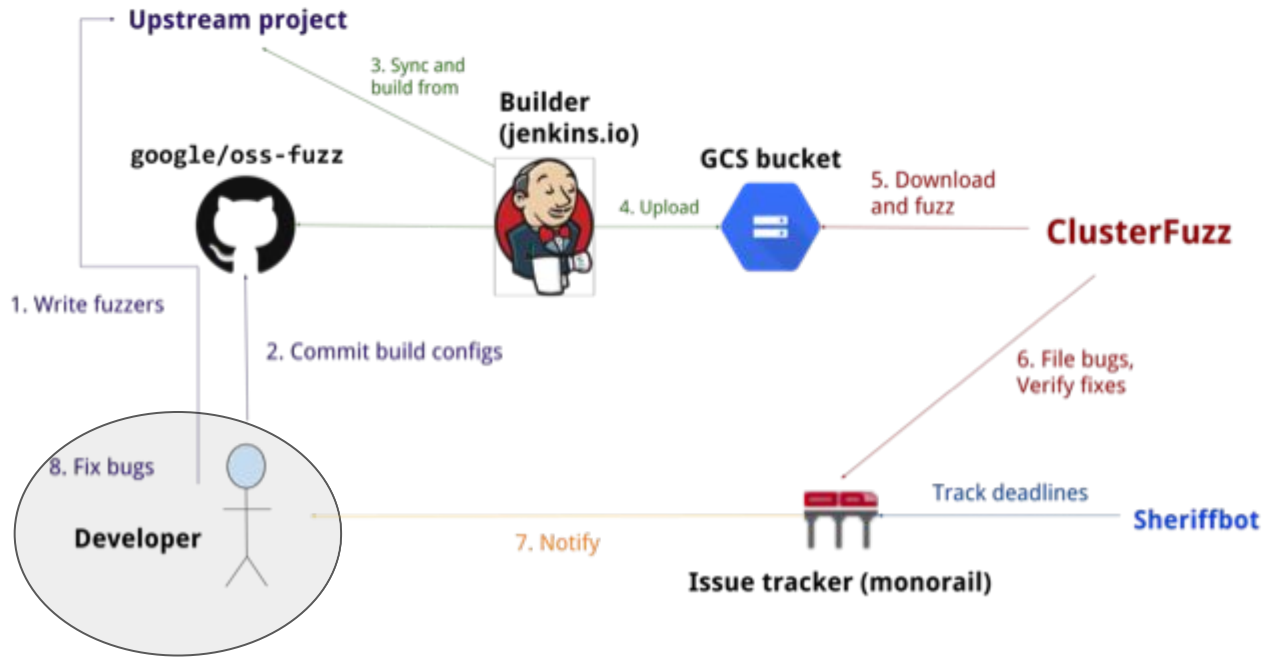
whoami

- Security Engineer at Ethereum Foundation
- Solidity team member
- Helping test the Solidity compiler

tl;dr State of Solidity Testing

- Unit tests
  - EXPECT(add(4,2), 6)
- Regression tests
  - EXPECT(0**uint8(uint8(2) ** uint8(8)), 1)
- Fuzz tests
  - add(adasdsad, $%@&)

Continuous Fuzzing



Bug Classes

- Benign: Compiler throws exception and aborts
  - Still bad but you know, not dangerous
- Malicious: Compiler generates incorrect code

Example: Code Generation Bug

```
contract C {
 function f() public pure returns (uint8) {
    return uint8(0) ** uint8(uint8(2)**uint8(8));
 }
} // 0 ^ (uint8(2^8))
```

Uint8 overflow basics

<u>`uint8`</u>

$$\begin{tikzcd}
 & 0 \\
 & 1 \\
 & \ldots \\
(2^8 - 1) = & 255 \arrow[uu, bend right=90, shift right=8]
\end{tikzcd}$$

Correct exponentiation (> 0.4.24)

```
0 ^ uint8(2 ^ 8)
=
0 ^ uint8(256)
=
0 ^ 0
=
1
```

Incorrect exponentiation (<=0.4.24)

```
0 ^ uint8(2 ^ 8)
=
0 ^ uint8(256)
=
0 ^ 256
=
0
```

Bug Summary

```
        "name": "ExpExponentCleanup",
        "summary": "Using the ** operator with an
exponent of type shorter than 256 bits can result
in unexpected values."
        "severity": "medium/high"
```

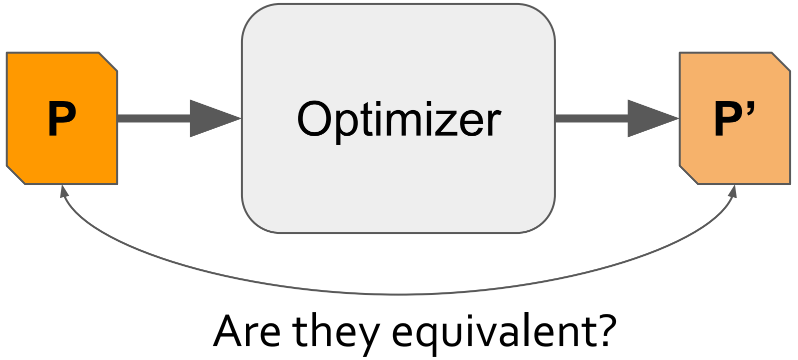
Patch: Clean up exponent

```diff
- else if (_type == Type::Category::Integer && (_op ==
  Token::Div || _op == Token::Mod))
+ else if (_type == Type::Category::Integer && (_op ==
  Token::Div || _op == Token::Mod || _op == Token::Exp))
```

How to discover such bugs automatically?

Proposed Solution

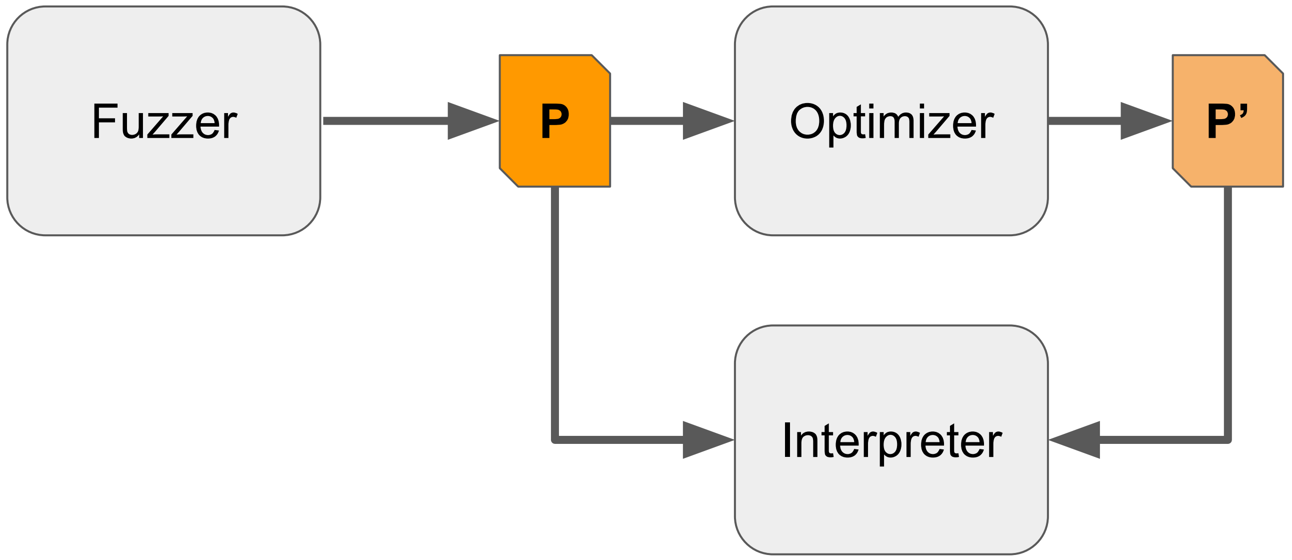
- Differential Testing
- Problem setting: Are there bugs introduced by optimizer?



Problem: Testing Equivalence

- Testing equivalence is hard
- Two solutions
    - Fuzz + Interpret
    - Rely on test generator that preserves equivalence across transformations

Fuzz + Interpret



assert(Trace_P == Trace_P')

Questions?

Source:
github.com/ethereum/solidity.git