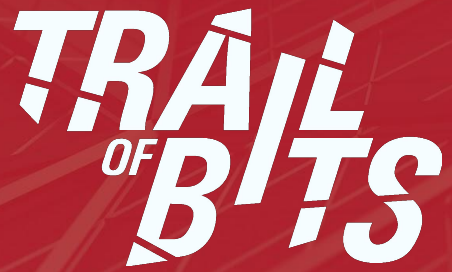

Property-based testing of smart contracts

Solidity testing

- **“Is our code correct?”**
- **Typically:**
  - Specify some sequence of interactions
  - Check the results
  - Run this every commit/release
  - “First, call `approve`, then `transferFrom`, then `balances` should…”

When doesn't this work?

- **Problem: we don't test most behavior**
- **"[...] `approve`, then `transferFrom`, then [...]"**
  - What about in the other order?
  - What about calling **`approve`** twice?
  - Writing tests for everything is hard

Can we write better tests?

- **Tests so far specify a single input**
  - f(3) == 1337
  - [1, 4, 2, 13].sort() == [1, 13, 2, 4]
- ***Property tests* cover any possible input**
  - ∀ x: leet(f(x)) >= leet(x)
  - ∀ l: l.sort() must be alphabetical

What's the difference?

- **Unit tests cover cases devs know about**
- **Property tests cover cases they don't**
  - Easy way to get test coverage up
  - Finds weeiiird edge cases
  - History of being unreasonably effective

OK, now do smart contracts

- **Unit tests specify a single transaction sequence**
  - If alice calls f, sends bob 2 ETH, then calls g, she'll have 3 ETH
- **Property tests cover any possible transaction sequence**
  - No matter what methods alice calls, bob can't lose money

How do we test like this?

- **The set of possible inputs is *giant***
- **Either we reason about some of them, or all of them**
  - Reason about some of them: Echidna
  - Reason about all of them: Manticore
- **One test, two ways to check**

Echidna

- **Given an ABI, generates random transactions**
  - function f(uint x, uint y) [...]
    - generate 10,000 pairs of uints
    - call f with all of them
    - check the property on each
- **No *guarantee*, better than one sequence**

Manticore

- **Implements a superset of EVM**
- **Regular EVM: values can be a number**
  - Examples: 0, 2 ** 256 -1, “hello world” (encoded)
- **Manticore: a number, or “all numbers such that…”**
  - Examples: “anything > 3, < 17”, “any prime number”, “literally anything”
  - Represented as constraints, so z3 can solve
- **Manticore : Geth :: Property testing : unit testing**

Manticore

- Execute with “any possible initial transaction”
- See if failure is ever possible
- If not, your property holds, congrats!
- If yes, use Z3 to solve constraints

What's the difference?

- **Manticore: complicated, but comprehensive**
  - Lots of effort to set up
  - Effectively formal verification
  - Super high assurance
- **Echidna: pretty easy, but it's random**
  - Runs more or less automatically
  - You could miss stuff though

How should I property test?

- Fun personal projects: don’t worry about it
- No existing tests: write some unit tests
- Existing unit tests: use Echidna
- Good test suite/*needs* correctness/$$$: use Manticore

Questions?

JP Smith

Security Engineer

jp@trailofbits.com, @japesinator

www.trailofbits.com