Pact: Smart Contracts You Can Take To The Bank

by Doug Beardsley

Outline

1. Blockchain Background
2. Problems with Smart Contracts
3. How Pact Solves Them

Background

What are Blockchains?

- Tamper proof append-only digital storage
- Same data stored on many computers
- Everyone agrees exactly (consensus)
- Don't need to trust a central source of truth
- The network is the source of truth

Smart Contracts

- Not smart
- Not contracts
- Just computer programs
- Stored and run in a blockchain

Smart Contract Programming is Different

- Every operation runs on thousands of different computers
- Same inputs must always give exactly the same outputs

  Otherwise how will you establish consensus?

- Stakes are high with lots of digital money on the line
- Must pay transaction fees to interact with blockchain programs
- Code is immortalized forever


Correctness more
important than usual

The Problem


Bugs

Bugs

- 25+% of contracts tested had exception vulnerabilities (a specific type of bug)

  https://eprint.iacr.org/2016/633.pdf

- And that's just one category of bugs


Increases risk and
slows adoption

Problems with Smart Contract Languages

- Based on mainstream languages
- Code stored in the blockchain is not human readable
- Hard to verify
- ...

Rubixi: Mainstream paradigm strikes

Contract Source Code </>

```
contract Rubixi {

        //Declare variables for storage critical to contract
        uint private balance = 0;
        uint private collectedFees = 0;
        uint private feePercent = 10;
        uint private pyramidMultiplier = 300;
        uint private payoutOrder = 0;

        address private creator;

        //Sets creator
        function DynamicPyramid() {
                creator = msg.sender;
        }
```

- Actual smart contract
- Contract got renamed, but not the constructor
- Result: anyone can become owner of the contract

The OO paradigm and the associated DRY violation directly caused Rubixi bug

Mainstream Languages Not Sufficient

- **Rubixi problem would never be perceived as significant in OO languages**

  Two minute fix and then move on

- **Don't guarantee determinism**

  (same inputs = same outputs)

- **Don't provide enough safety**

  Null pointers

  Unprincipled type conversions

  etc...

Unreadable code in the blockchain

- Authors can publish the code and associate it with the contract but...
- Only ~1% of contracts have available source code

  https://arxiv.org/pdf/1802.06038.pdf

Hard to Verify

- Fallback functions complicate control flow
- Values can be mutated
- Turing Complete

How Pact Addresses These Problems

Didn't default to flawed paradigm just because it's popular

Purpose-built with smart contracts in mind

- Code in the blockchain is human readable
- Smart contracts can be updated/fixed
- Designed for storing data
- Built-in auth
- Can't put encryption keys in code
- NOT Turing Complete
- Values are immutable
- Open source, written in Haskell by people familiar with cutting-edge programming languages.

Hello World

```
(define-keyset 'hello-admin (read-keyset 'hello-keyset))

(module hello 'hello-admin
        "Pact hello-world with database example"

        (defschema hello-schema
          "VALUE stores greeting recipient"
          value:string)

        (deftable hellos:{hello-schema})

        (defun hello (value)
          "Store VALUE to say hello with."
          (write hellos "hello" { 'value: value }))

        (defun greet ()
          "Say hello to stored value."
          (with-read hellos "hello" { "value" := value }
                     (format "Hello, {}!" [value])))
        )

(create-table hellos)

(hello "world") ;; store "hello"
(greet)         ;; say hello!
```

Formal Verification

Formal Verification

- Proves things about your program for all possible inputs
- Impossible to actually test them all
- Significantly benefits from not being Turing Complete

Formal Verification Example

```
(defun abs:integer (x:integer)
  ("Returns the absolute value of an integer"
             (properties
              [(>= result 0)
               (not (table-read 'accounts))
               (not (table-write 'accounts))])))
  (if (< x 0)
      (- x)
    x))
```

Recap

Recap

- **Humans mess up**

  Always operating at the edge of manageable complexity

- **Stakes are higher with smart contracts, correctness more important**

- **Need systems that make common mistakes impossible**

- **Pact: Purpose built, human readable, verifiable**

More Info

- Docs: http://kadena.io/pact
- REPL: http://kadena.io/try-pact
- https://github.com/kadena-io/pact
- We're hiring!
- doug@kadena.io