Reflections on Blockchain Security

Jan Gorzny, Blockchain Researcher

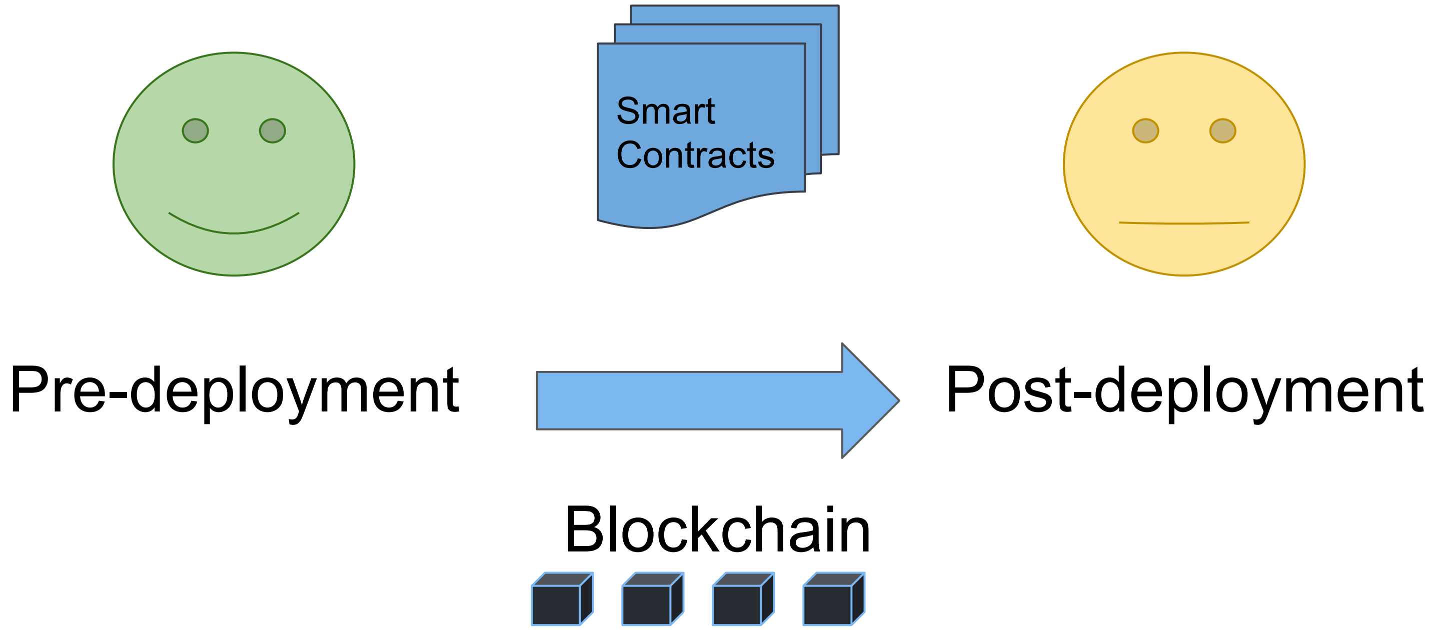
Quantstamp™

Smart Contracts

```
function withdraw() {
    if (balances[msg.sender] > 0
        && bankBalance > 0){
        msg.sender.call.value(balances[msg.sender])
        bankBalance -= balances[msg.sender];
        balances[msg.sender] = 0;
    }
}
```

Immutable code deployed on a blockchain; public

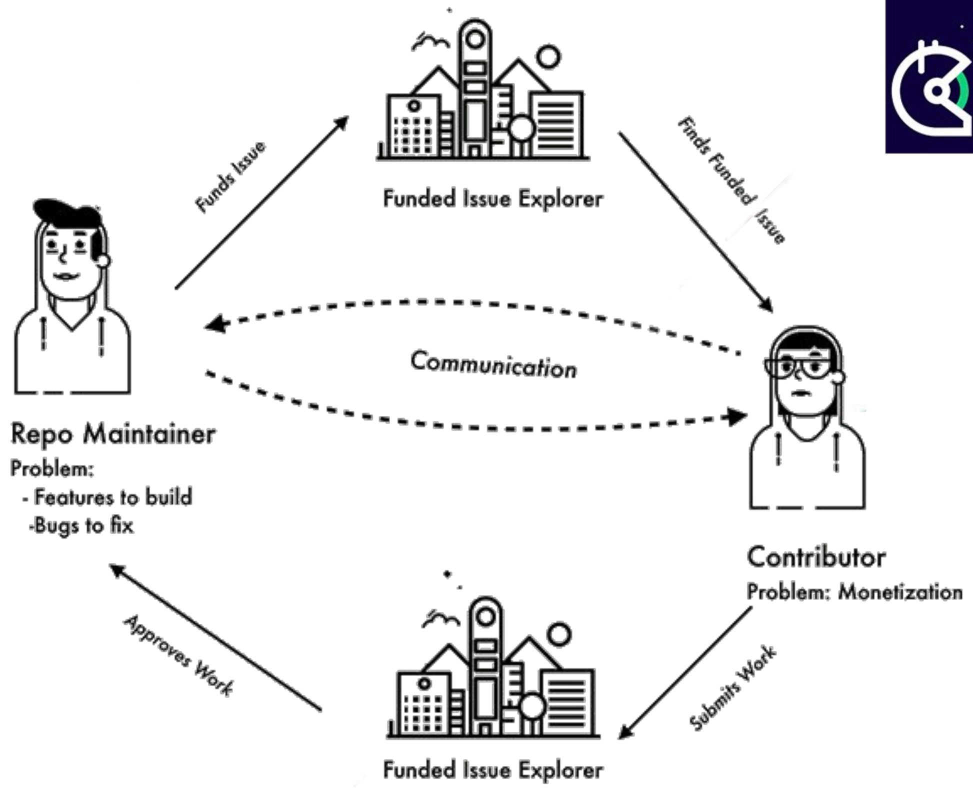
Introduction



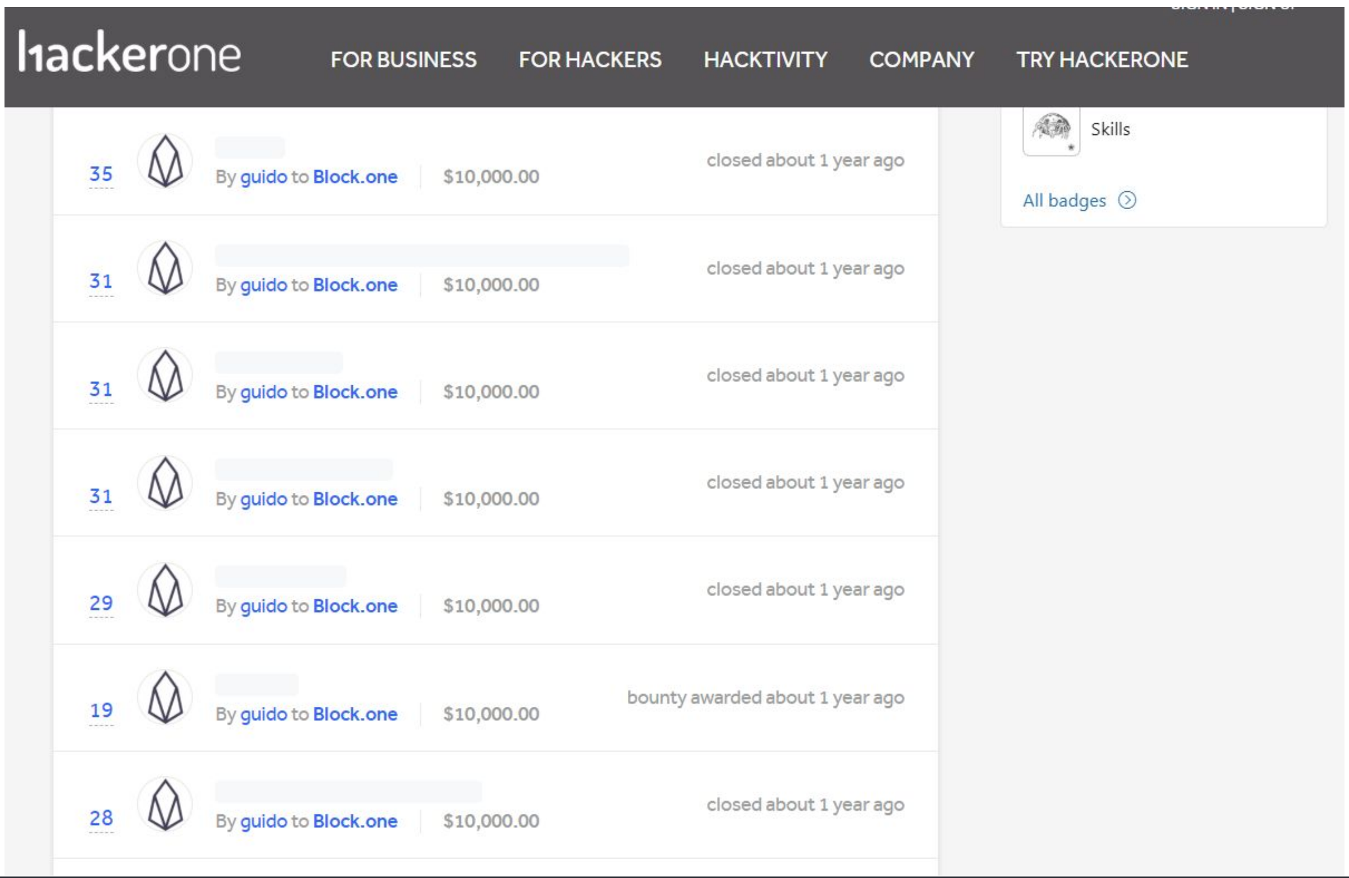
Pre-Deployment

- Design Choices
- Specifications
- Bug Bounties
- Analyses & Formal Verifications
- Audit

Bug Bounties



Bug Bounty Wins



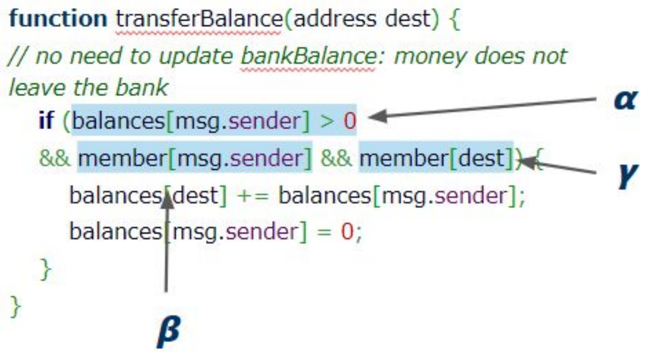
hackerone FOR BUSINESS FOR HACKERS HACKTIVITY COMPANY TRY HACKERONE

35	By guido to Block.one	$10,000.00	closed about 1 year ago
31	By guido to Block.one	$10,000.00	closed about 1 year ago
31	By guido to Block.one	$10,000.00	closed about 1 year ago
31	By guido to Block.one	$10,000.00	closed about 1 year ago
29	By guido to Block.one	$10,000.00	closed about 1 year ago
19	By guido to Block.one	$10,000.00	bounty awarded about 1 year ago
28	By guido to Block.one	$10,000.00	closed about 1 year ago

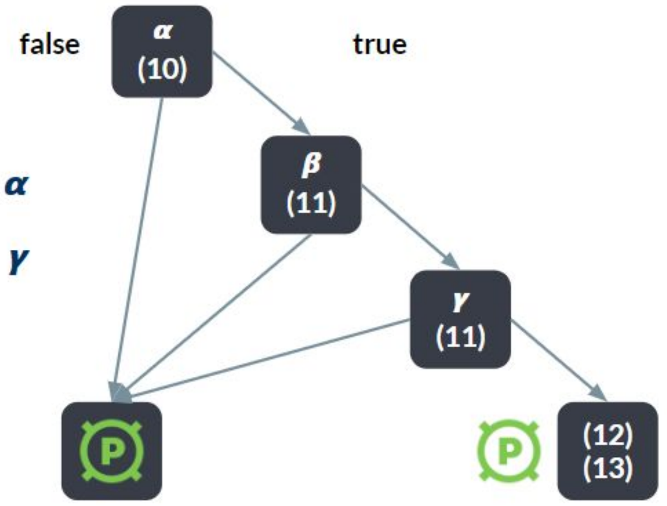
Skills

All badges

(Static) Analysis

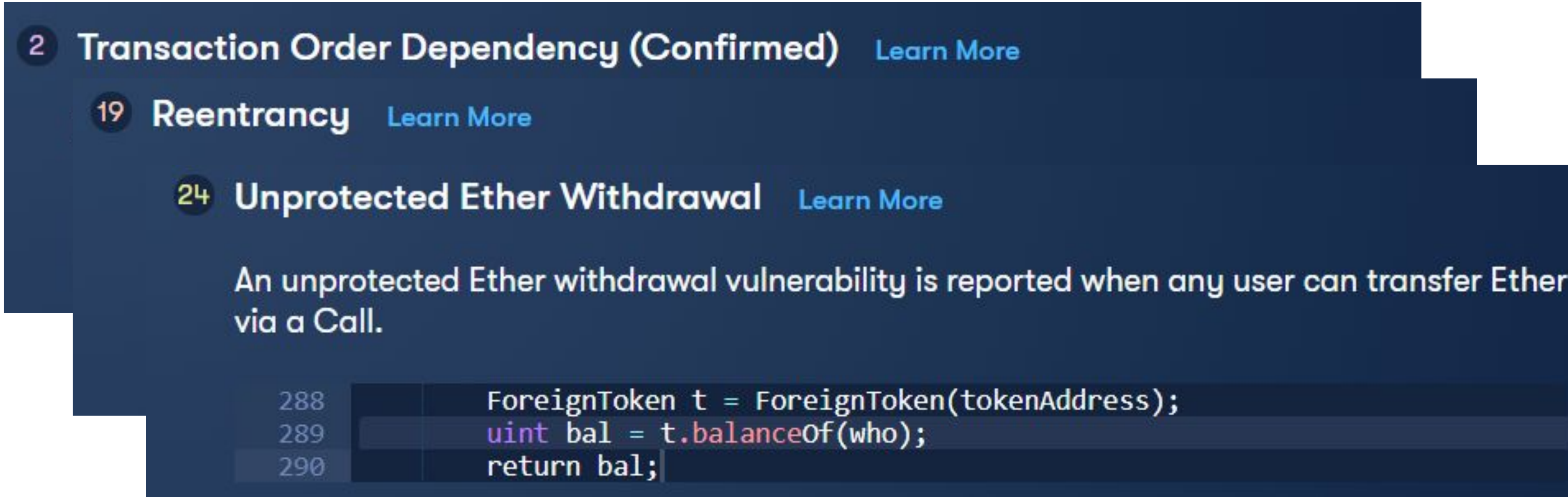


```
function transferBalance(address dest) {
// no need to update bankBalance: money does not
leave the bank
    if (balances[msg.sender] > 0
    && member[msg.sender] && member[dest]) {
        balances[dest] += balances[msg.sender];
        balances[msg.sender] = 0;
    }
}
```



Quantstamp Security Network V2
Smart contract
Quantstamp Security Network User Interface
Network of decentralized protocol nodes
Stored on the Ethereum blockchain

(Static) Analysis Wins



2 Transaction Order Dependency (Confirmed) Learn More

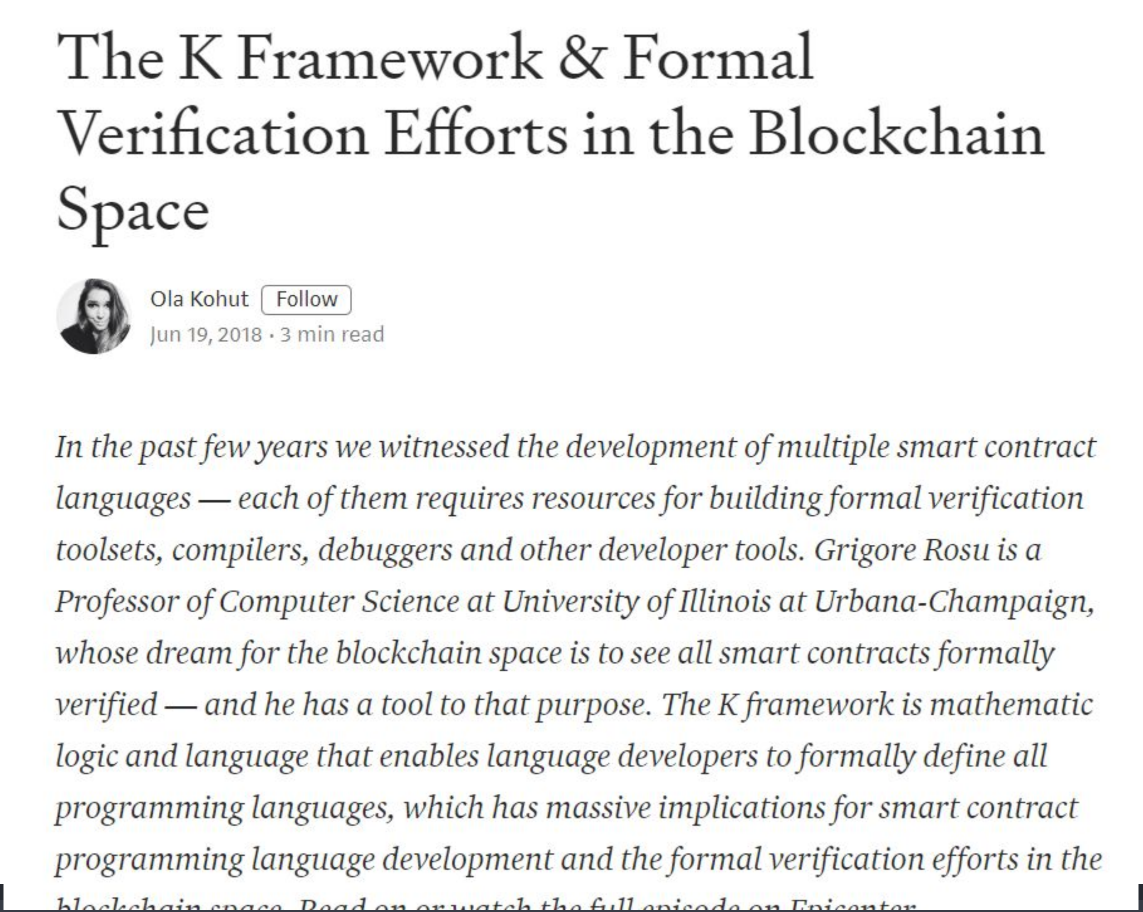
19 Reentrancy Learn More

24 Unprotected Ether Withdrawal Learn More

An unprotected Ether withdrawal vulnerability is reported when any user can transfer Ether
via a Call.

```
        ForeignToken t = ForeignToken(tokenAddress);
        uint bal = t.balanceOf(who);
        return bal;
```

Formalizations



The K Framework & Formal Verification Efforts in the Blockchain Space

Ola Kohut Follow

Jun 19, 2018 · 3 min read

In the past few years we witnessed the development of multiple smart contract languages — each of them requires resources for building formal verification toolsets, compilers, debuggers and other developer tools. Grigore Rosu is a Professor of Computer Science at University of Illinois at Urbana-Champaign, whose dream for the blockchain space is to see all smart contracts formally verified — and he has a tool to that purpose. The K framework is mathematic logic and language that enables language developers to formally define all programming languages, which has massive implications for smart contract programming language development and the formal verification efforts in the blockchain space. Read on or watch the full episode on Epicenter.

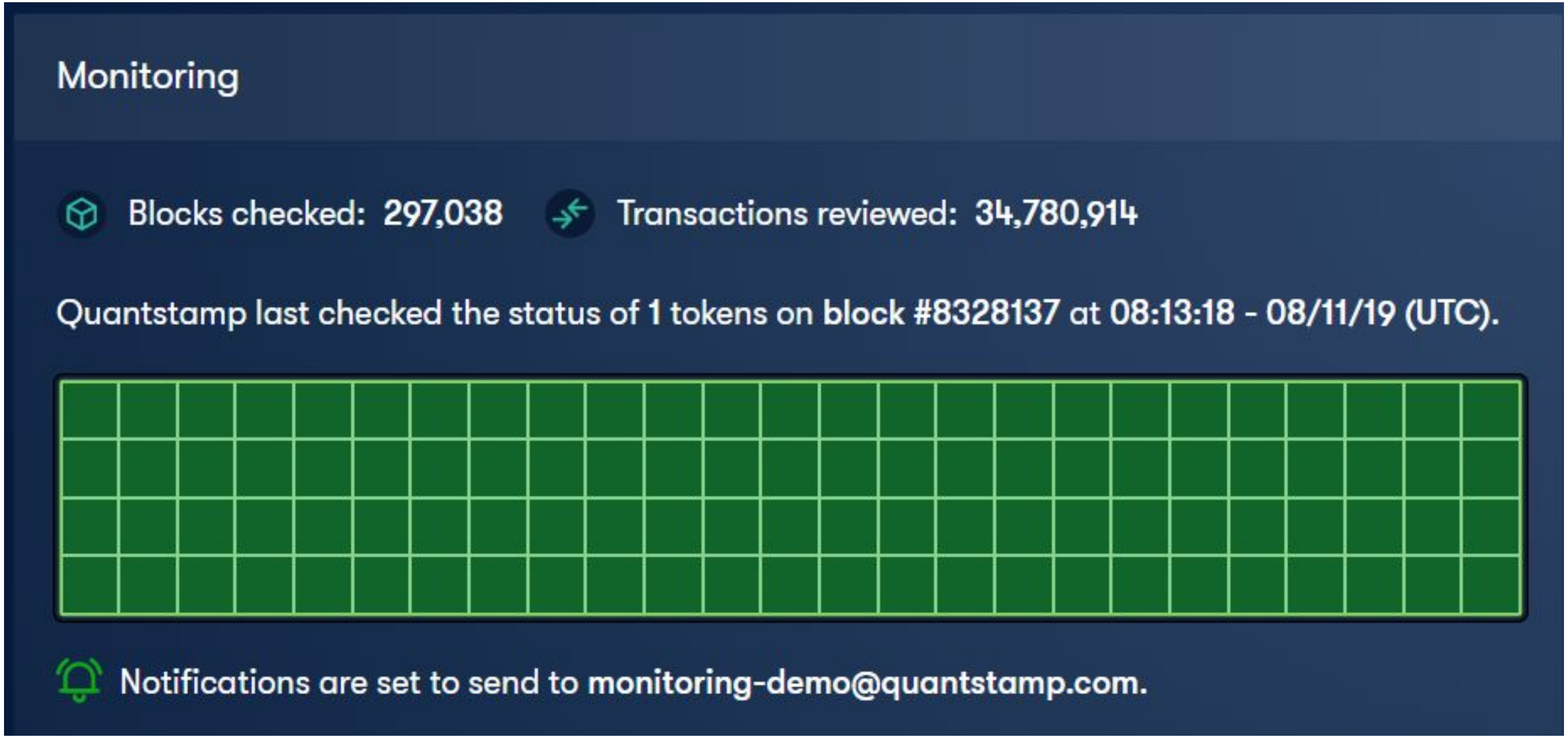
https://medium.com/epicenterpodcast/the-k-framework-formal-verification-efforts-in-the-blockchain-space-1651b789e5d4

Post-Deployment

Recall: smart contracts are *immutable*

Upgradeable contracts?

Monitoring



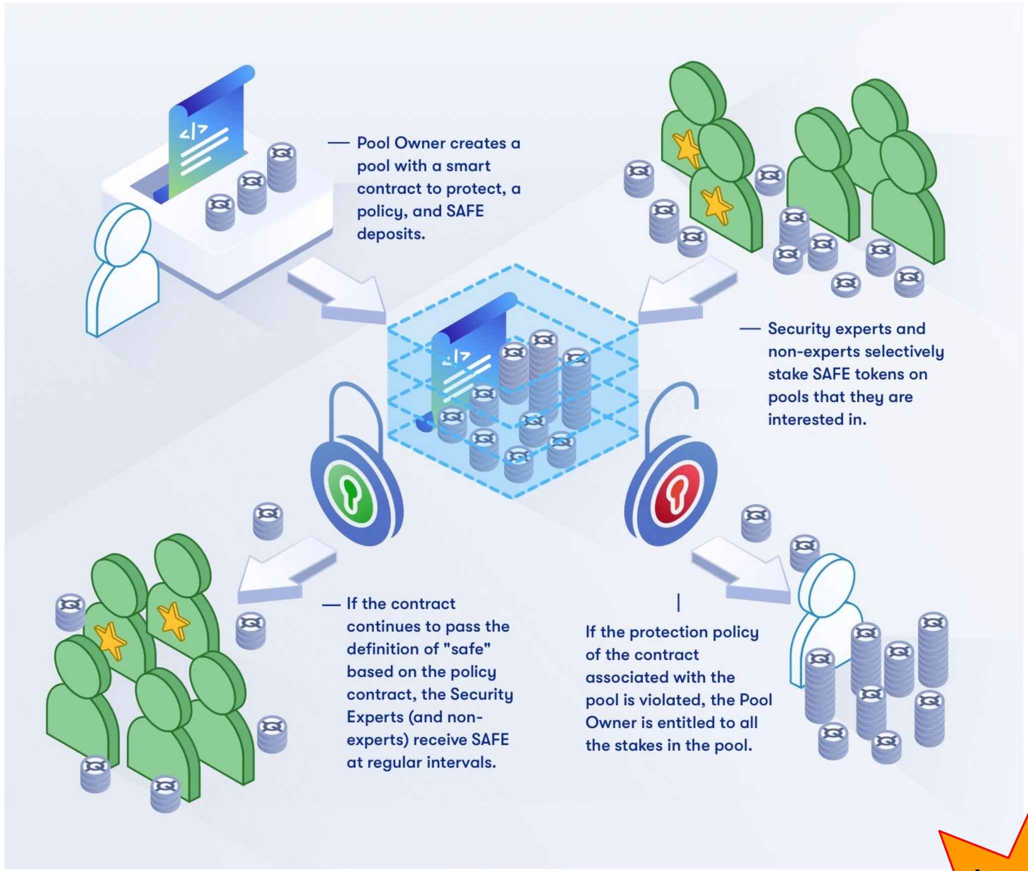
Monitoring

Blocks checked: 297,038 Transactions reviewed: 34,780,914

Quantstamp last checked the status of 1 tokens on block #8328137 at 08:13:18 - 08/11/19 (UTC).

Notifications are set to send to monitoring-demo@quantstamp.com.

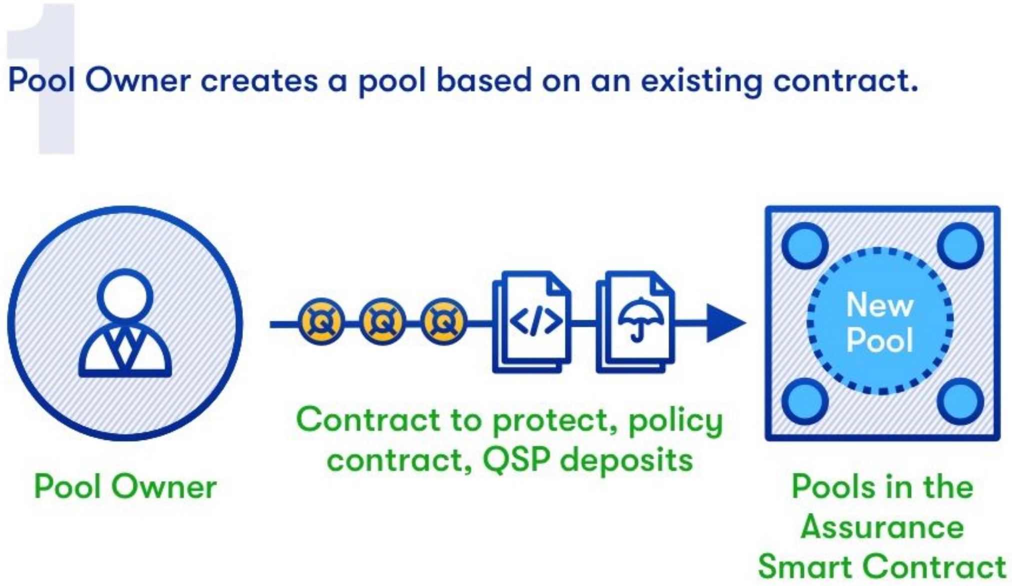
Assurance


Pool Owner creates a pool with a smart contract to protect, a policy, and SAFE deposits.
Security experts and non-experts selectively stake SAFE tokens on pools that they are interested in.
If the contract continues to pass the definition of "safe" based on the policy contract, the Security Experts (and non-experts) receive SAFE at regular intervals.
If the protection policy of the contract associated with the pool is violated, the Pool Owner is entitled to all the stakes in the pool.

In Beta!

https://assurance.quantstamp.com/

Assurance Walkthrough


1
Pool Owner creates a pool based on an existing contract.
Pool Owner
Contract to protect, policy contract, QSP deposits
New Pool
Pools in the Assurance Smart Contract

In Beta!

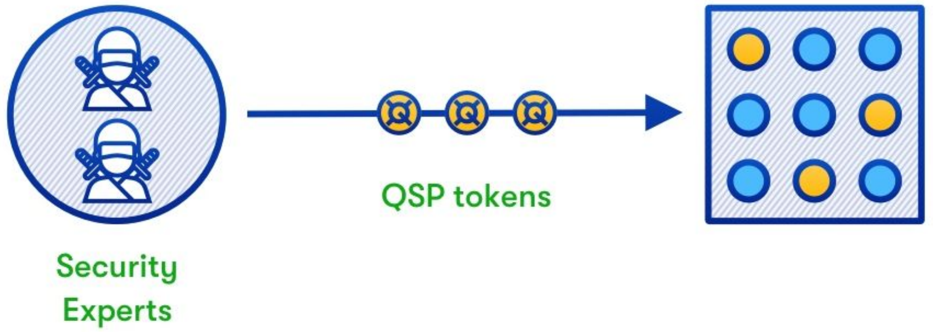
Assurance Walkthrough

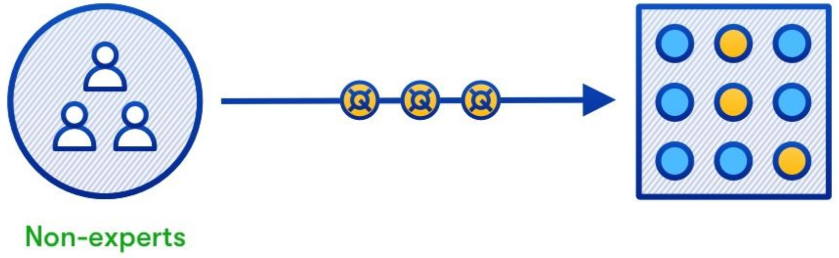
2

Security expert Assurance Providers (defined by the CCR) selectively stake QSP tokens on pools they are interested in.

3

Non-expert users also stake QSP tokens in existing pools.



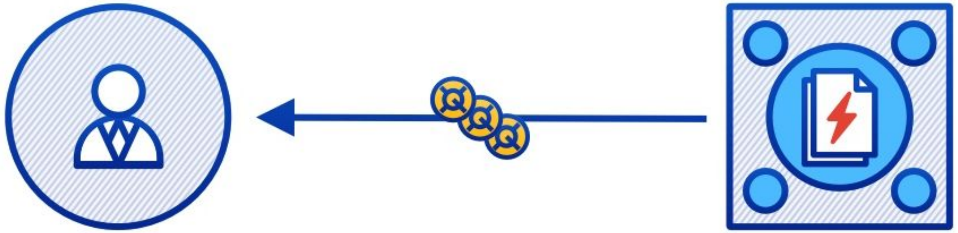


In Beta!

Assurance Walkthrough

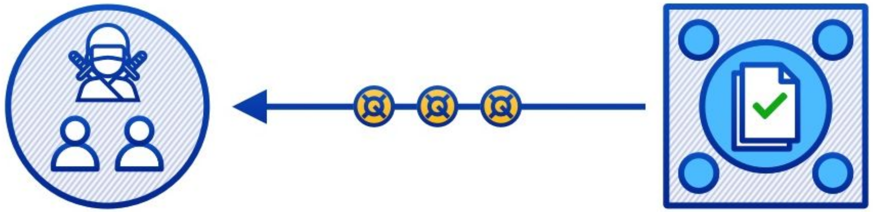
4

If the protection policy of the contract associated with the pool is violated, the Pool Owner is entitled to all the stakes in the pool.



5

If the contract continues to pass the definition of "safe" based on the policy contract, the Security Experts (and non-experts) receive QSP at regular intervals.



In Beta!

Example Assurance Policy

```
pragma solidity 0.4.24;
interface IPolicy {
   function isViolated(address contractAddress) external view
returns(bool);
}
import "../test/CandidateToken.sol";

/// @title TotalSupplyNotExceededPolicy - the policy is violated if too many
coins are minted
contract TotalSupplyNotExceededPolicy is IPolicy {
   uint256 public maximumSupply;

   constructor(uint256 max) public {  maximumSupply =  max;   }

   function isViolated(address contractAddress) external view returns(bool) {
        CandidateToken candidateToken = CandidateToken(contractAddress);
        if (candidateToken.totalSupply() > maximumSupply) { return true;
        } else { return false; }
   }
}
```

https://assurance.quantstamp.com/

In Beta!

Future Concerns (Ethereum)

- Layer 2 Solutions
  - Standards please!

- New solutions to old problems (e.g., randomness)

- Eth 2.0? eWASM?

Future Concerns

- STARKS

- Other blockchains and their concerns?

Thank you!

Questions?

Quantstamp™

jan@quantstamp.com