Smart Contracts

IN5420 Distributed Blockchain Technologies

Michael Eikeland

April 2018

Nick Szabo describes his idea of smart contracts in the paper "Formalizing and Securing Relationships on Public Networks". He argues that contracts in their current state are an artifact of the past, the paper era, and needs to be readjusted to the digital era. This paper sets the stage for just that. He points out some crucial principles of contract requirements:

- **Observability**: The ability for the principals to observe performance of the contract.
- **Verifiability**: The ability for a principal to prove to an adjudicator that the terms of a contract has either been breached or performed.
- **Privity**: Encapsulation and protection from third parties

It is suggested that the protocols that smart contracts run upon may be categorized as either self-enforcing, mediated or adjudicated. Further on he looks at research in order to find a way to meet the requirements. Central to all implementations seem cryptographic protocols.

In "Blockchains and Smart Contracts for the Internet of Things" the authors put Szabo's idea in the context of public blockchains. The contract is stored on the blockchain, and has it's own account containing the contract state. A contract is invoked by a user calling it by announcing a transaction (a signed message) to the network. Any participant can verify the outcome of the invocation by running it locally in a virtual machine. This does however require that the contract is deterministic in order for the network to reach consensus.

The authors of "Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab" held a course for developing smart contracts. The course specifically used Ethereum and its Serpent scripting language, but the errors made could to a large extent be generalized for all smart contract platforms. They found that smart contract development and software development have different requirements. Smart contracts deployed on cryptocurrency blockchains require incentive compatibility in order to prevent selfish behaviour and attacks. It is also dependent on using cryptographic primitives for secrets as any participant deterministically can see the outcome of an invocation.

The security of smart contracts is also pointed out as a weak point in "Making Smart Contracts Smarter" through analysis of smart contracts deployed on the Ethereum blockchain. More specifically, they argue that smart contract developers make wrong assumptions about the contracts distributed semantics. In Ethereum, the state of a contract depends on the ordering of transactions. Two invocations of the same contract, can lead to different outcomes. An adversary can use this information to attempt an attack for profit. Another example is when contracts are timestamp dependent. In the scenario where the smart contract uses the block hash to determine if someone gets a reward, a miner can manipulate other parts (i.e. the timestamp) of the block (which will lead to a different hash) in order to influence the outcome of the contract. In Ethereum contracts can call other contracts, but contracts also have a call-stack limit which means that an attacker can create a contract that will utilize this in order to only partly execute another contract, potentially achieving profits for the attacker.

The authors of this paper developed an analysis tool for Ethereum contracts. Out of the 19366 publicly deployed smart contracts, 8833 were flagged with security issues.