Smart Contract Security for Developers

A Guide to Auditing and Best Practices

Chapter 1:
Introduction to Smart Contract Security

Ruhul Amin

1.1. What is a Smart Contract?

A smart contract is a self-executing contract with the terms of the agreement directly written into code. It runs on a blockchain, allowing for decentralized, trustless execution without the need for intermediaries. Smart contracts leverage blockchain's inherent properties, such as immutability, transparency, and consensus, to automate and enforce contract terms in a secure and transparent manner.

1.1.1 Key Components of a Smart Contract

1. Code
2. The Contract's State
3. Gas (Transaction Fees)

Code:

Smart contracts are typically written in programming languages specific to the blockchain platform they are deployed on. The most commonly used language is **Solidity**, which is designed for Ethereum-based smart contracts. Other blockchain platforms may use different languages, such as **Vyper** (Ethereum), **Rust** (for Solana), or **Michelson** (for Tezos).

The Contract's State:

Smart contracts are typically written in programming languages specific to the blockchain platform they are deployed on. The most commonly used language is **Solidity**, which is designed for Ethereum-based smart contracts. Other blockchain platforms may use different languages, such as **Vyper** (Ethereum), **Rust** (for Solana), or **Michelson** (for Tezos).

Gas (Transaction Fees)

For a smart contract to execute its functions on the blockchain, the transaction must be processed, which often requires computational resources. In the case of Ethereum, these resources are measured in **gas**, a unit of computational work. Gas fees are paid in the blockchain's native token (ETH in Ethereum) and serve as an incentive for miners (or validators in Proof of Stake) to process transactions and execute the contract's logic.

1.1.2 How Smart Contracts Work

1. Deployment
2. Execution
3. Self-execution
4. Security and Immutability

Deployment

A smart contract is deployed to the blockchain by an account (usually the developer's or the organization's wallet). This deployment is a transaction on the blockchain, and once confirmed, the contract is stored in a specific address on the blockchain.

Execution

Once deployed, the smart contract can be interacted with by sending transactions to its address. These transactions are processed based on the contract's logic and may result in a change in state (e.g., transferring tokens or assets).

Self-execution

Smart contracts execute automatically based on pre-defined conditions. If the conditions are met (e.g., a specific time or value), the contract performs the actions without any further intervention from external parties. For instance, a simple smart contract could automatically release a payment when goods are delivered and tracked via a trusted oracle.

Security and Immutability

Once deployed, the contract's code and the data within it cannot be modified (unless specified through upgradable contract patterns like proxy contracts). This immutability, while enhancing security, also places a responsibility on developers to ensure that the contract code is correct and secure before deployment.

1.1.3 Smart Contracts in Action

1. Example 1: Token Contract
2. Example 2: Non-Fungible Tokens (ERC-721)
3. Example 3: Decentralized Finance (DeFi) Protocol

Example 1: Token Contract

One of the most common examples of a smart contract is the **ERC-20 token contract** on Ethereum. It governs the creation, transfer, and management of fungible tokens. When a user sends tokens from one wallet to another, the smart contract automatically updates the balances on the blockchain, ensuring that the transfer is transparent and irreversible.

Example 2: Non-Fungible Tokens (ERC-721)

The **ERC-721** token contract standard governs the creation and management of non-fungible tokens **(NFTs)** on Ethereum. Unlike ERC-20 tokens, each ERC-721 token is **unique** and represents ownership of a specific digital or physical asset. These assets could include digital art, real estate, collectibles, or gaming items.

Example 3: Decentralized Finance (DeFi) Protocol

In DeFi protocols, smart contracts automate lending, borrowing, and trading without the need for intermediaries like banks. For example, a smart contract could lock a collateral asset and issue a loan in the form of another asset, such as stablecoins, based on certain terms (e.g., collateral value).

Want to read full book or contribute ?

https://bit.ly/smartcontract-security-for-developers

https://github.com/ruhulamin1398/Smart-Contract-Security-for-Developers-A-Guide-to-Auditing-and-Best-Practices/

Click Here To Visit ⟶