Uniswap V2 in Vyper

Marvin Chan

Background

Blockchains: a new computing paradigm

- Programmable computer that can be used by anyone but is not owned by anyone

Smart Contracts: code deployed on the blockchain

- Transforming finance and art in the form of decentralized finance(DeFi) and non-fungible tokens(NFTs)

Hacks

- Since anyone can interact with any smart contract, security vulnerabilities in contracts can be exploited and lead to financial loss
- $325 million hack in the Wormhole Protocol

Smart Contracts Landscape

Ethereum - Largest blockchain

Solidity

- Most popular language for smart contract develop (8,000+ public github repos)
- JavaScript-like Turing complete language
- Been under development since 2014
- Provides developers with immense flexibility, but flexibility comes with security flaws

Vyper

- Pythonic programming language that is non-Turing complete
- Designed around security, language simplicity and auditability
- First stable version released in 2020
- Only has 82 public github repos

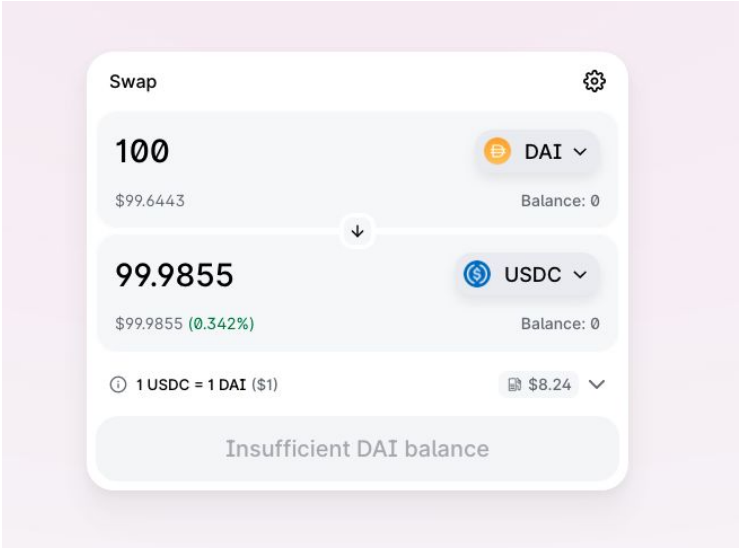
Project

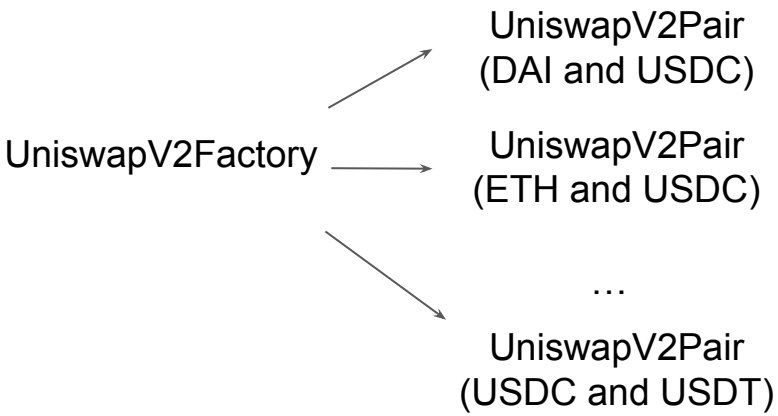
Audience: Beginner Smart Contract Developers

Determine how difficult it is to learn and develop a smart contract in Vyper by reimplementing the Uniswap V2 smart contract

Uniswap V2

- Smart contracts in Solidity that facilitates swapping between cryptocurrencies
- 3rd most widely used smart contract
- Uniswap accommodates all tokens by having a factory contract that creates a pair contract that allows exchange between 2 tokens





Obstacles converting from Solidity and Vyper

Vyper Differences

- No inheritance
- No infinite length arrays
- contract creates another contract
  - Solidity calls constructor
  - Vyper does not call __init__

UniswapV2Factory (Solidity)

```
contract UniswapV2Factory is IUniswapV2Factory {
    address public feeTo;
    address public feeToSetter;

    mapping(address => mapping(address => address)) public getPair;
    address[] public allPairs;

    event PairCreated(address indexed token0, address indexed token1, address pair, uint);

    constructor(address _feeToSetter) public {
        feeToSetter = _feeToSetter;
    }

    function allPairsLength() external view returns (uint) {
        return allPairs.length;
    }
```

UniswapV2Pair (Solidity)

```
constructor() public {
    factory = msg.sender;
}
```

UniswapV2Pair (Vyper)

```
# https://github.com/vyperlang/vyper/issues/2326
# __init__ is not run when created with create_forwarder_to()
@external
def __init__():
    self.factory = msg.sender
```