Echidna Workshop Series: Part 4

Justin Jacob - Security Engineer I

Previous & Upcoming Workshops

Beginner

- Part 1: The Basics (Nov 16, 2022)
- Part 2: Breaking ABDK Math (Nov 22, 2022)

Intermediate

- Part 3: Breaking Uniswap I (Nov 30, 2022)
- **Part 4: Breaking Uniswap II (Today)**

Advanced

- Part 5: Breaking Primitive Finance I (Week of Dec 12, 2022)
- Part 6: Breaking Primitive Finance II (Week of Dec 19, 2022)

Who am I?

Justin Jacob, Security Engineer I

Who You Should Follow

- Troy Sargent (@0xalpharush)
- Josselin Feist (@montyly)
- Nat Chin (@0xicingdeath)
- Anish Naik (@anishrnaik)

Who are we?

Trail of Bits (@trailofbits)

- We help developers to build safer software
- R&D focused: we use the latest program analysis techniques
- Slither, Echidna, Tealer, Amarna, solc-select, ..

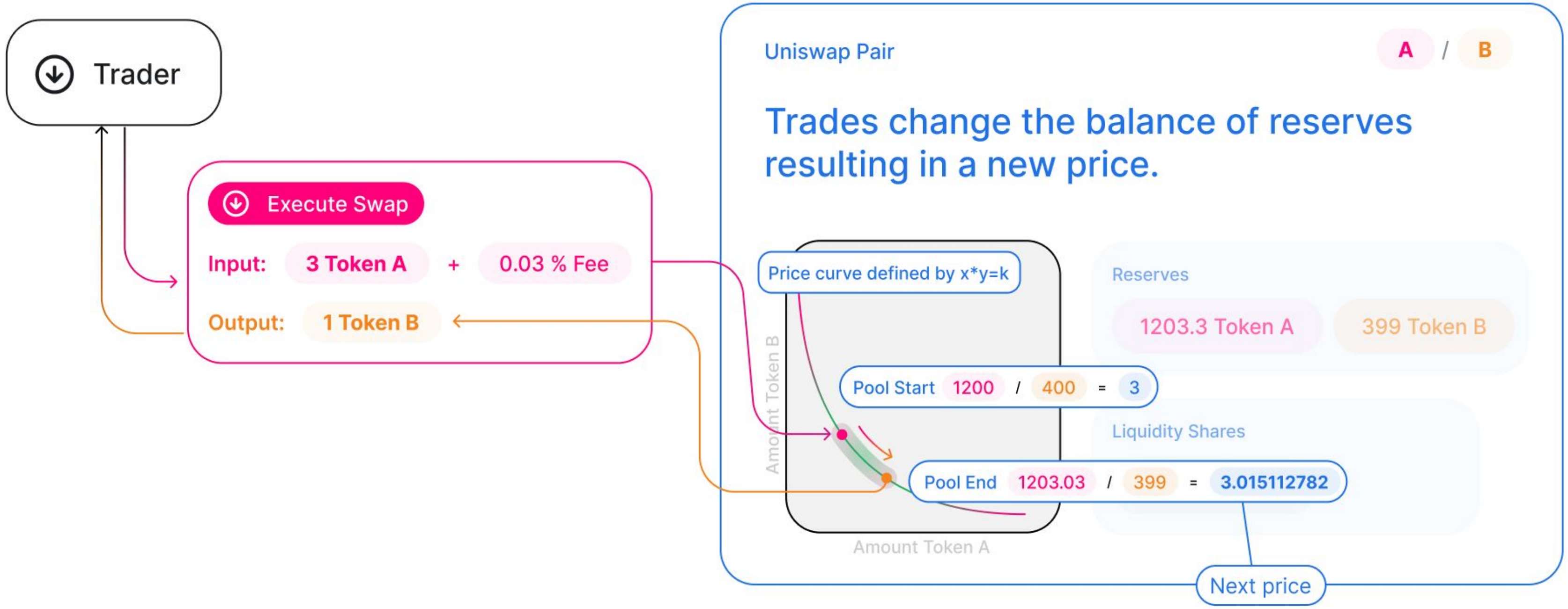
Recap -

- What is an AMM?
- Uniswap v2 Core

What is an AMM?

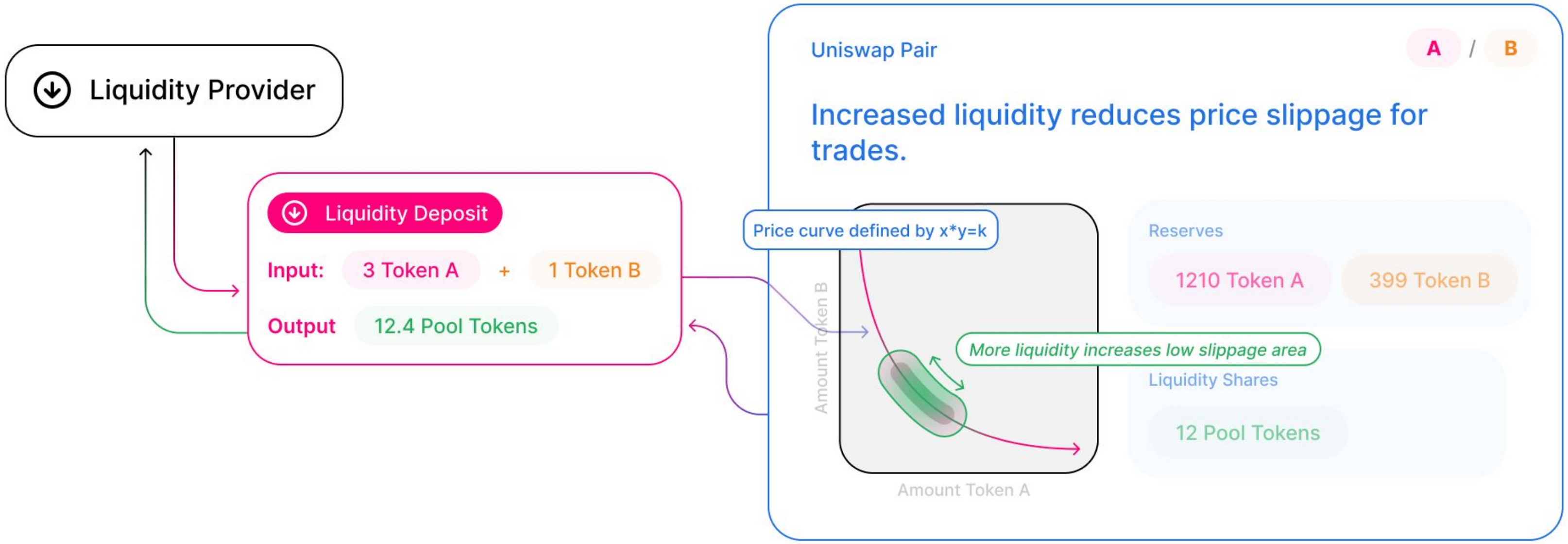
- **AMM Model:**
    - Exchange without orderbook
    - Pricing is based on pool’s liquidity formula
        - Simplest example: $xy = k$ (Uniswap!)
        - Price is calculated as ratio between two assets
    - Exchanges keep k (pool invariant) constant

Trades change the balance of reserves resulting in a new price.



Liquidity Providers (LPs)

- People provide ratio of tokens to the pool, get minted special LP token
- This LP token represents provided liquidity to the pool
- Initial LP Provider sets k => sets token price
- To get tokens back, must burn these LP tokens


Increased liquidity reduces price slippage for trades.

Core

- **Two contracts: factory and pairs**
- **Factory: creates pairs**
  - Creates unique pair contracts for each pool via CREATE2
  - Also has logic to turn on fees
- **Pairs:**
  - Represent liquidity pool, keep track of token balances
  - Also an ERC20 token
  - Contains the basic swapping logic

Today’s Agenda

- Testing Swaps
- Uniswap v2 Periphery
- More invariants…
- (simplified) End to end testing

Swap Invariants:

- **Transferring 0 tokens should net you 0 tokens out**
  - Otherwise bad things would happen

Swap Invariants:

- Swapping 0 tokens should net you 0 tokens out
  - Otherwise bad things would happen
- Swapping decreases and increases token balances appropriately

Swap Invariants:

- **Swapping 0 tokens should net you 0 tokens out**
  - Otherwise bad things would happen
- **Swapping decreases and increases token balances appropriately**
- **Swapping x of token A for y token of B and back should give you x of token A**
  - Called “Path Independence”
  - In practice not necessarily true b/c fees, rounding

Swap Invariants:

- Swapping 0 tokens should net you 0 tokens out
  - No such thing as free money :(
- Swapping decreases and increases token balances appropriately
- Swapping x of token A for y token of B and back should give you x of token A
  - Called “Path Independence”
  - In practice not necessarily true b/c fees, rounding
- Pool invariant stays constant during swaps
  - Otherwise tokens can be drained from the pool

Periphery

- Set of contracts that interact with Core
- Provide safety checks and helper functions

Periphery

- **Library: mainly contains helper functions**
    - Sort Tokens, calculate amounts in/out
    - Can calculate amounts in/out for chained swaps

Periphery

- **Library: mainly contains helper functions**
  - Sort Tokens, calculate amounts in/out
  - Can calculate amounts in/out for chained swaps
- **Router: “routes” trades**
  - Provides safety checks for minimal core contracts
  - In charge of transferring tokens properly
  - If you’ve done a swap/provided liquidity before, you went through the router

Swap Invariants:

- Swapping 0 tokens should net you 0 tokens out
  - No such thing as free money :(
- Swapping decreases and increases token balances appropriately
- Swapping x of token A for y token of B and back should give you x of token A
  - Called “Path Independence”
  - In practice not necessarily true b/c fees, rounding
- Pool invariant stays constant during swaps
  - Otherwise tokens can be drained from the pool

LP Invariants:

- **Providing liquidity increases invariant**
  - x*y =k => increasing x and y increases k!
- **LP tokens are minted either:**
  - Proportional to the pool share (if there is liquidity already)
  - Proportional to the sqrt of the token amounts (if creating a pool)

LP Invariants:

- **Providing liquidity increases invariant**
  - x*y =k => increasing x and y increases k!
- **LP tokens are minted either:**
  - Proportional to the pool share (if there is liquidity already)
  - Proportional to the sqrt of the token amounts (if creating a pool)
- **Providing and removing liquidity should give you starting amount (sans fees)**
  - Similar to “path independence” but for LPs
  - Account for rounding errors

LP Invariants (continued):

- Removing liquidity decreases invariant

LP Invariants (continued):

- Removing liquidity decreases invariant
- Removing liquidity decreases LP token balance

LP Invariants (continued):

- Removing liquidity decreases invariant
- Removing liquidity decreases LP token balance
- LP’s token balance should be monotonically increasing

Homework:

Test more invariants!

Write system properties!

Write your own tests and make PRs!

Thanks everyone for attending!

See you next week!