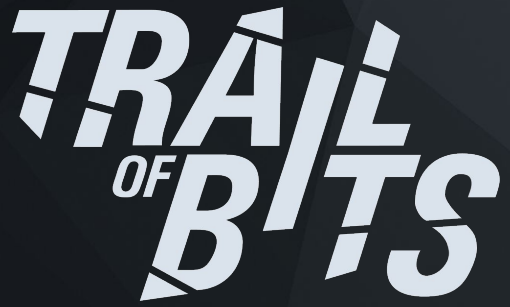

TRAIL
OF
BITS

Advanced DeFi Invariants

Upcoming workshops

Beginner

- Part 1: The Basics
- Part 2: Breaking ABDKMath (Week of Nov 21, 2022)

Intermediate

- Part 3: Breaking Uniswap I (Week of Nov 28, 2022)
- Part 4: Breaking Uniswap II (Week of Dec 5, 2022)

Advanced

- **Part 5: Advanced DeFi Invariants I (Week of Dec 12, 2022)**
- Part 6: Advanced DeFi Invariants II (Week of Dec 19, 2022)

Who am I?

Nat Chin, Security Engineer II

Who You Should Follow

- **Troy Sargent (@0xalpharush)**
- **Josselin Feist (@montyly)**
- **Anish Naik (@anishrnaik)**
- **Justin Jacob (@technovision99)**

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, ..

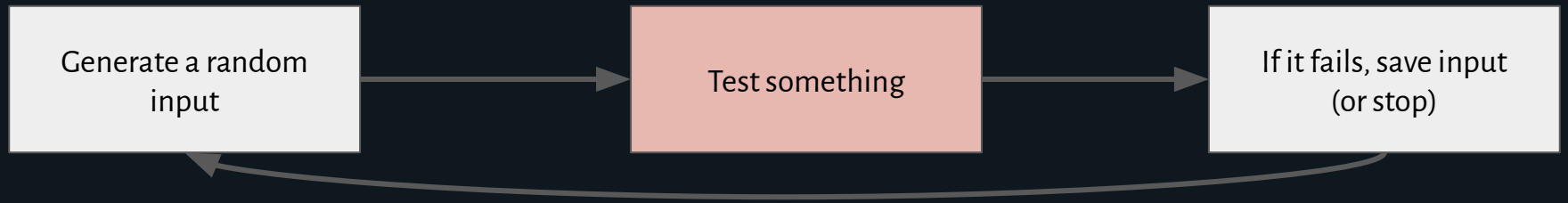
Agenda

- Recap: Fuzzing
- Introduction to Primitive
- Architecture
- How to find invariants
- Code Walkthrough
- Invariant Search
- Implementing basic invariants

So…how do I start fuzzing?

1. **Identify your invariants / system properties in English**
2. **Convert your properties to code**
3. **Run Echidna**
4. **FIND BUGS**

And… what is fuzzing?



Echidna vs Other Fuzzers

- **Echidna is more mature**
- **Allows testing of high gas assumptions**
- **Works with any compilation framework**
- **Supports various API's for testing**
- **Supports hevm/dapptool cheatcodes**

Tips on Identifying Invariants

- **Start with the smallest component first**
- **Analyze all preconditions and postconditions**
- **Determine safe bounds of inputs**
- **Identify inversely-related functions**
- **Focus on the happy and unhappy paths**

Useful Optimizations

- **Tests should have precondition, action, postcondition**
  - Pre-conditions: Scope the input space
  - Action: What we are testing
  - Post-conditions: The “truths” after the action

Coverage **is** your friend! Especially Echidna 2.0.4

```
34 |      |    // -------------------- Margin.sol ----------------------
35 | *r   |    function depositIncreasesBalance(uint256 risky, uint256 stable) public {
36 | *r   |            uint256 pre_deposit_bal_risky = margin.balanceRisky;
37 | *r   |            uint256 pre_deposit_bal_stable = margin.balanceStable;
38 | *r   |            Margin.deposit(margin, risky, stable);
39 |      |
40 | *    |            assert(margin.balanceRisky - pre_deposit_bal_risky == risky);
41 | *    |            assert(margin.balanceStable - pre_deposit_bal_stable == stable);
42 |      |    }
43 |      |    mapping (address => Margin.Data) margins;
44 | r    |    function withdrawDecreasesBalance(uint256 risky, uint256 stable) public {
45 | r    |            margins[address(this)] = margin;
46 | r    |            uint256 pre_deposit_bal_risky = margins[address(this)].balanceRisky;
47 | r    |            uint256 pre_deposit_bal_stable = margins[address(this)].balanceStable;
48 | r    |            Margin.withdraw(margins, risky, stable);
49 |      |
50 |      |            assert(pre_deposit_bal_risky - margins[address(this)].balanceRisky  == risky);
51 |      |            assert(pre_deposit_bal_stable - margins[address(this)].balanceStable == stable);
52 |      |    }
53 |      |
```

Disclaimer

- **This is complicated ✨**
- **Clone and follow along!**

Primitive

Primitive: What is it?

- **Replicating Market Maker**
- **Implements Black-Scholes interest options**

words, words, words

Uniswap vs Primitive

Uniswap

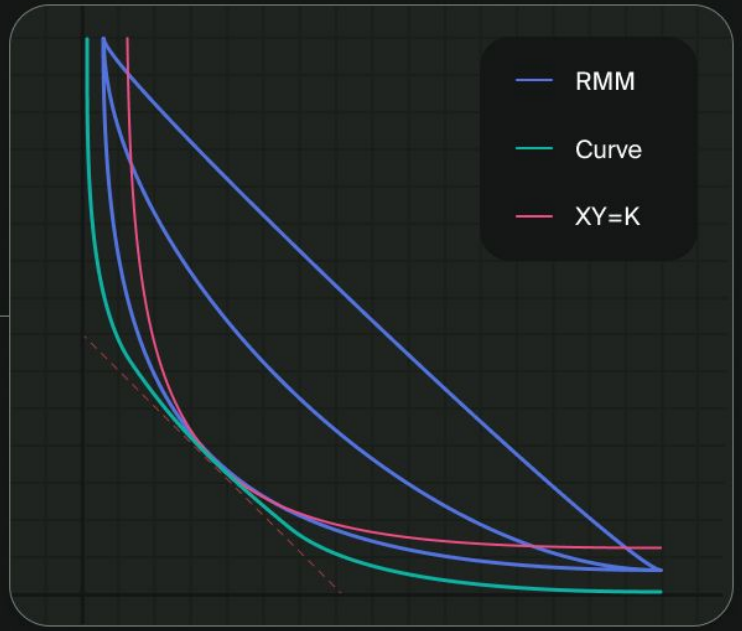
- Price changes on swap
- Pools don’t have a concept of time

Primitive

- Price changes on swap and over time
- At expiry, pool consists of an underlying token

Constant Function Market Makers

Trading Functions



Features

- Allows creation of pools against 2 tokens
- Relies on spot price – **no** oracles
- “Maturity” – specified point in time where curve changes
- “Strike Price” – how much the asset will be worth
- Price will converge to strike price at maturity

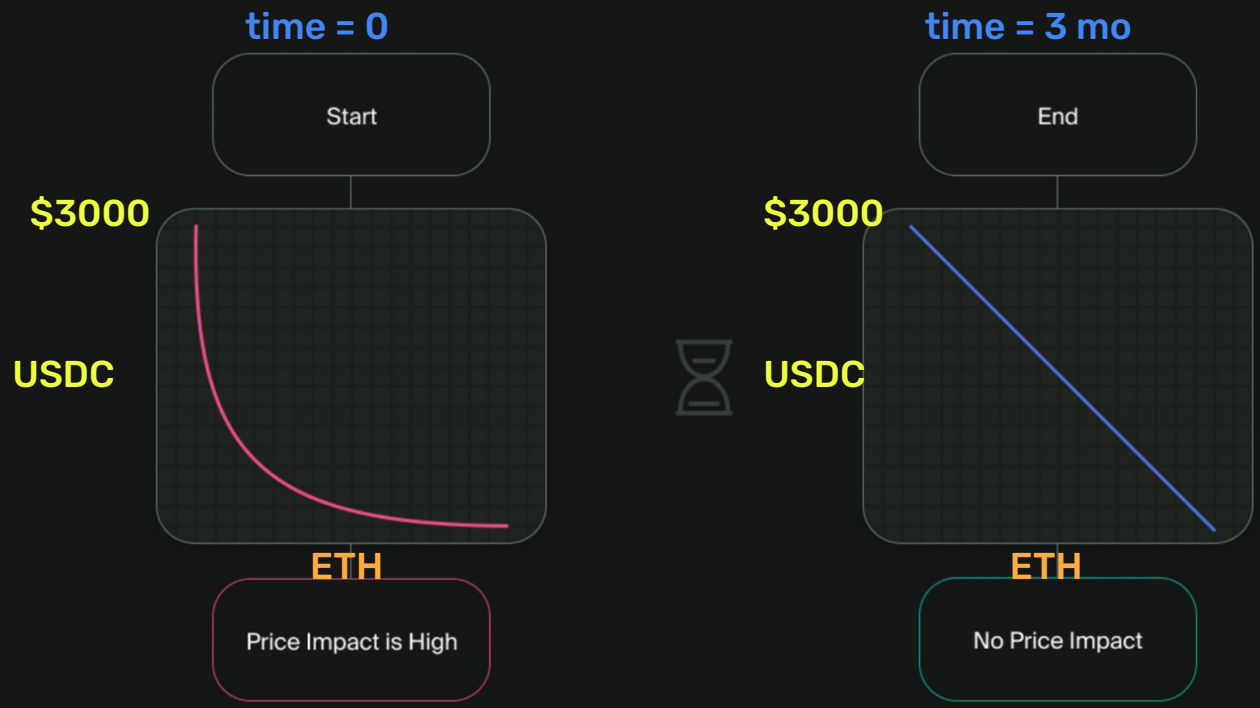
Black-Scholes Options

- **This is how the price changes over time**
- **European Option – right to exercise at expiration**
- **Pricing model depends on:**
  - Strike price (K)
  - Implied volatility (over a period of time)
  - Time to expiry (t)
  - Spot price of underlying (S)

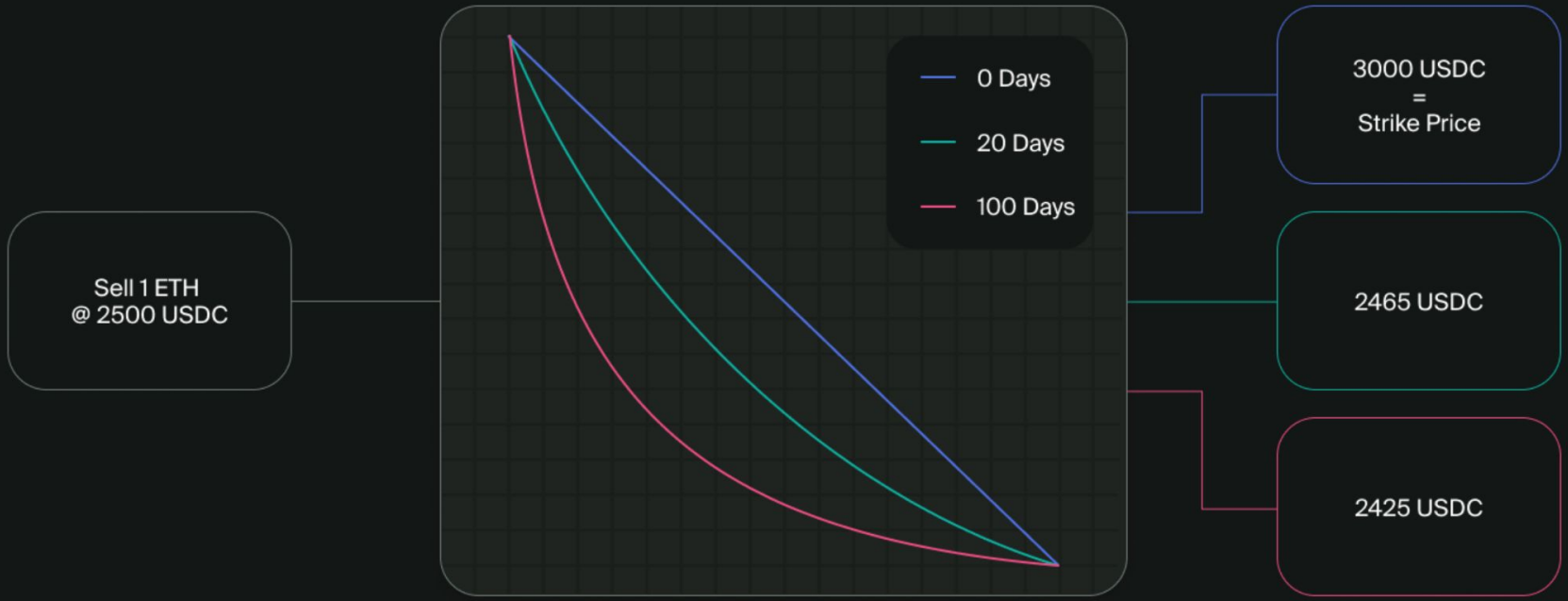
Concrete Example

- Pool consists of **USDC (underlying) - ETH (quote)***
- Strike price = 3000 USDC
- Maturity of **100 days**
- Implied volatility **150%**

Liquidity Compression - Concentration Over Time



Price Impact on Swaps - Over Time

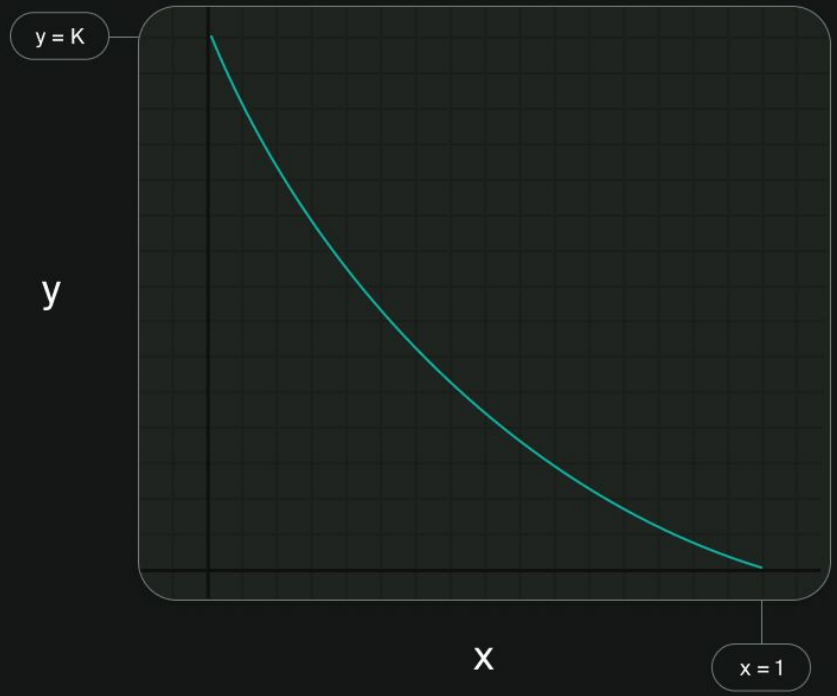


Primitive RMM Curve

K	Strike Price	x	Underlying Asset Reserve
σ	Implied Volatility	y	Quote Asset Reserve
τ	Time until Expiry	Φ	CDF
k	Invariant	Φ^{-1}	Inverse CDF

$$y - K\Phi(\Phi^{-1}(1 - x) - \sigma\sqrt{\tau}) = k$$

Trading Function



At maturity

- **Assets cannot be bought/sold**
- **Sells the second asset**
- **Pool prices at the strike price**

System Architecture

Architecture

- **Core system**
  - PrimitiveEngine – controls pools
  - PrimitiveFactory – creates engines
- **Manager: Periphery**

Primitive Engine

- Create a new pool (curve)
- Allocate and remove liquidity from a pool
- Swap between tokens
- Deposit and withdraw from system

Creating New Pool

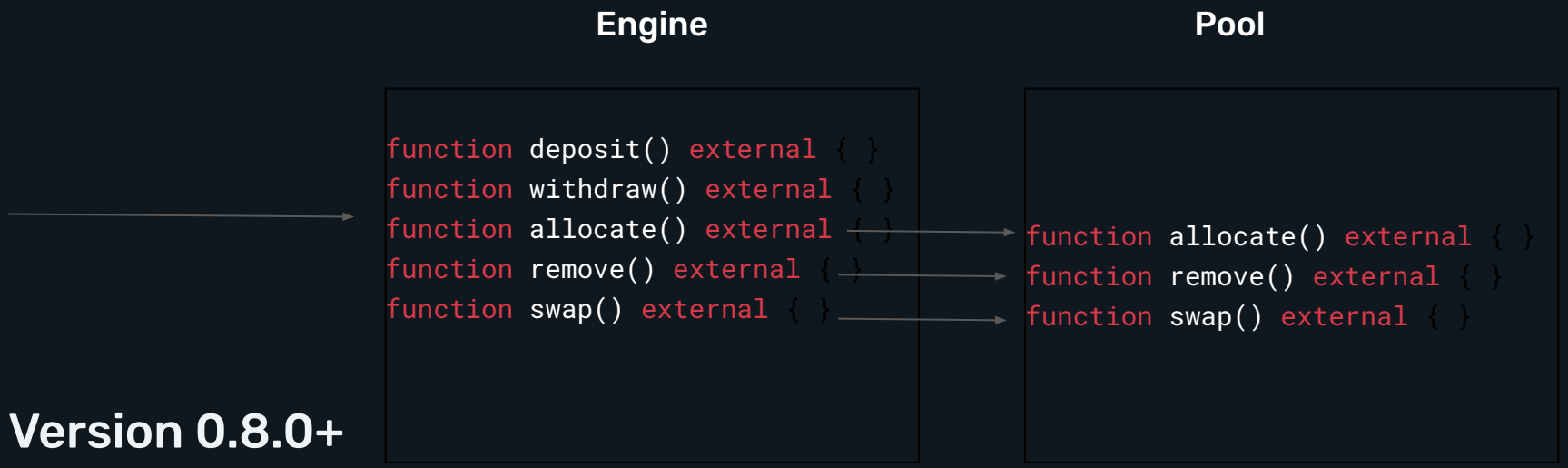
- Tokens – underlying and quote token
- Strike price – asset's worth at maturity
- Implied Volatility – how much price changes
- Maturity – expiration
- Gamma – trading fee percentage

Reference: Primitive Whitepaper

What can you do with a curve?

- **Allocate liquidity – supply tokens to curve**
- **Remove liquidity – remove tokens to curve**
- **Create liquidity position – deposit money to Engine**
- **Swap between tokens**

System Architecture



Version 0.8.0+

Splitting up Deposit and Allocate

- **Allows users to deposit into the system first**

Libraries

- ABDK Math
- Primitive Math*
- Margin balances
- Reserve balances
- SafeCast
- Transfers
- Unit conversions

Let’s look at code!

Lessons Learned

- **Use the coverage!**
- **Start with the libraries**
- **Don’t be afraid to mock**
- **Understand the system *first***

So on your own time……

- Test the `Units` library
- Test the `Margin` library
- Test the other libraries
- Look for E2E invariants* 😉