A Survey on Price Oracle Attacks

rkm0959

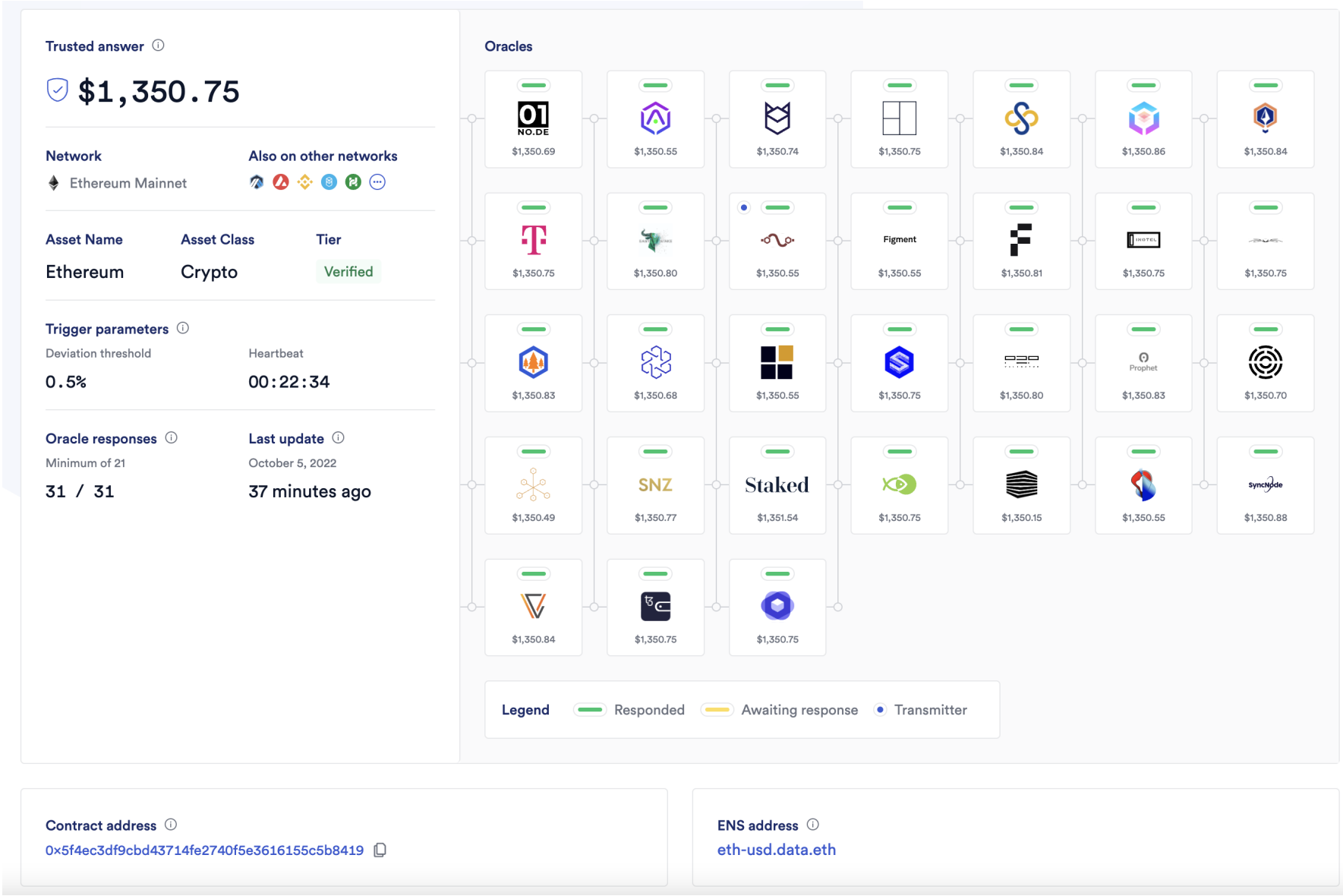
Security Researcher @ HAECHI LABS

November 8th

Outline

Table of Contents

Price Oracle



Price Oracle

Most assets on DeFi has a *price*. The mechanism for determining the price of an asset is named a *price oracle*. Chainlink, shown above, is an example.

Price Oracles are used in many DeFi applications, namely

- Lending Markets
- Leveraged Liquidity Provider
- Algorithmic Stablecoins
- Options, Derivatives

Vulnerable Price Oracles are Dangerous

If a lending market uses a weak price oracle, there are two known attacks

- Undercollateralized Loans: an attacker may trick the system into lending large value of assets under small deposit of assets
- Liquidation Attacks: an attacker may trick the system into allowing liquidation of other positions that are supposed to be healthy

We will now focus on the various types of vulnerable price oracles.

Table of Contents

Trusting the Spot Price of a DEX

Spot price on a DEX can be manipulated easily

- Buy a token to inflate the price
- Call the attack target contract
- Sell the bought token to send the price back to normal

It's clear that the attacker only needs to pay trading fees to the LP.
Initial capital may not be needed much if accompanied with flashloan/swap.

Trusting the Spot Price of a DEX

```
uint160 sqrtPrice = TickMath.getSqrtRatioAtTick(currentTick());
uint256 price = FullMath.mulDiv(uint256(sqrtPrice).mul(uint256(sqrtPrice)), PRECISION, 2**(96 * 2));

(uint256 pool0, uint256 pool1) = getTotalAmounts();

uint256 deposit0PricedInToken1 = deposit0.mul(price).div(PRECISION);
shares = deposit1.add(deposit0PricedInToken1);

if (deposit0 > 0) {
  token0.safeTransferFrom(msg.sender, address(this), deposit0);
}
if (deposit1 > 0) {
  token1.safeTransferFrom(msg.sender, address(this), deposit1);
}
```

Visor Finance used the current tick of UniswapV3 pools for price oracle.

TWAP Oracles

Usually, to prevent such attacks, time-weighted average price is used.
A price accumulator $a_t = \sum_{i=1}^{t} p_i$ is kept track of in the contract state, and the average price in the time period $[t_1, t_2]$ will be computed as

$$\frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

UniswapV2 has this accumulator, which other DApps can utilize for TWAP.

This **does not** mean TWAP's are invincible. More on this later.

Pricing LP Tokens

Some protocols use LP tokens as collateral, so we need to price them.

Consider a UniswapV2 pair over two tokens A, B with

- LP token total supply T
- reserves of the two tokens (L_A, L_B)
- fair price of the two tokens (P_A, P_B)

Pricing LP Tokens

How would we

- decide on the fair price P_A, P_B
- combine them to get the price of the LP token?

We'd also need to decide whether the price would be vs A/B or USD.

Pricing LP Tokens: Warp Finance Hack

In the case of Warp Finance, they tried to price the LP token in USD.

- Fair price P_A, P_B is via $A - USDC$, $B - USDC$ TWAP in UniswapV2
- Combining the prices is done by calculating

$$\text{LP Fair Price} = \frac{L_A P_A + L_B P_B}{T}$$

This is bad, as swaps in A/B LP pool doesn't affect P_A, P_B at all - so large swaps will inflate the LP Fair Price. This was used to hack about 8M USD.

Pricing LP Tokens: Warp Finance Hack

For example, assume a USDC-USDT stablecoin UniswapV2 pool.
In this case, $L = L_A = L_B$ with $P_A = P_B = 1$.

By performing large swaps, the LP price can be

$$\frac{Lx + L/x}{T} = \frac{2L}{T} \cdot \frac{1}{2}\left(x + \frac{1}{x}\right)$$

where $x + 1/x \geq 2$ for any $x > 0$.

Pricing LP Tokens: Deus DAO Hack

Here, the two tokens in the LP were both stables.

- Fair price $P_A = P_B = 1$ as they are stables.
- Combining the prices is done by calculating

$$\text{LP Fair Price} = \frac{L_A P_A + L_B P_B}{T}$$

However, the **balance** was used instead of the **reserve**!!
Here, a flashswap to decrease L_A, L_B was used to deflate the LP price, making it possible to liquidate borrowers for a 3M USD profit.

Pricing LP Tokens: Inverse Finance Hack

This one was also very similar to Warp Finance Hack, but for Curve.

- Fair price P_A, P_B, P_C was from Chainlink Oracles.
- Combining the prices is done by calculating

$$\text{LP Fair Price} = \frac{L_A P_A + L_B P_B + L_C P_C}{T}$$

```
function latestAnswer() public view returns (uint256) {
    uint256 crvPoolBtcVal = WBTC.balanceOf(address(CRV3CRYPTO)) * uint256(BTCFeed.latestAnswer()) * 1e2;
    uint256 crvPoolWethVal = WETH.balanceOf(address(CRV3CRYPTO)) * uint256(ETHFeed.latestAnswer()) / 1e8;
    uint256 crvPoolUsdtVal = USDT.balanceOf(address(CRV3CRYPTO)) * uint256(USDTFeed.latestAnswer()) * 1e4;

    uint256 crvLPTokenPrice = (crvPoolBtcVal + crvPoolWethVal + crvPoolUsdtVal) * 1e18 / crv3CryptoLPToken.totalSupply();

    return (crvLPTokenPrice * vault.pricePerShare()) / 1e18;
}
```

Inverse Finance was hacked with around 5.8M USD in losses.

Pricing LP Tokens: Alpha Homora V2

To compute the fair price, Alpha Homora takes an alternate approach.

- Fair price P_A, P_B is from oracle aggregators
- Combining the prices is done by calculating

$$\text{LP Fair Price} = 2 \cdot \frac{\sqrt{P_A P_B} \cdot \sqrt{L_A L_B}}{T}$$

which is nice, as P_A, P_B are hard to manipulate, and $L_A L_B$ are hard to manipulate as well. This is currently used in Alpha Homora V2.

Liquidity Rules Everything: Rari Capital

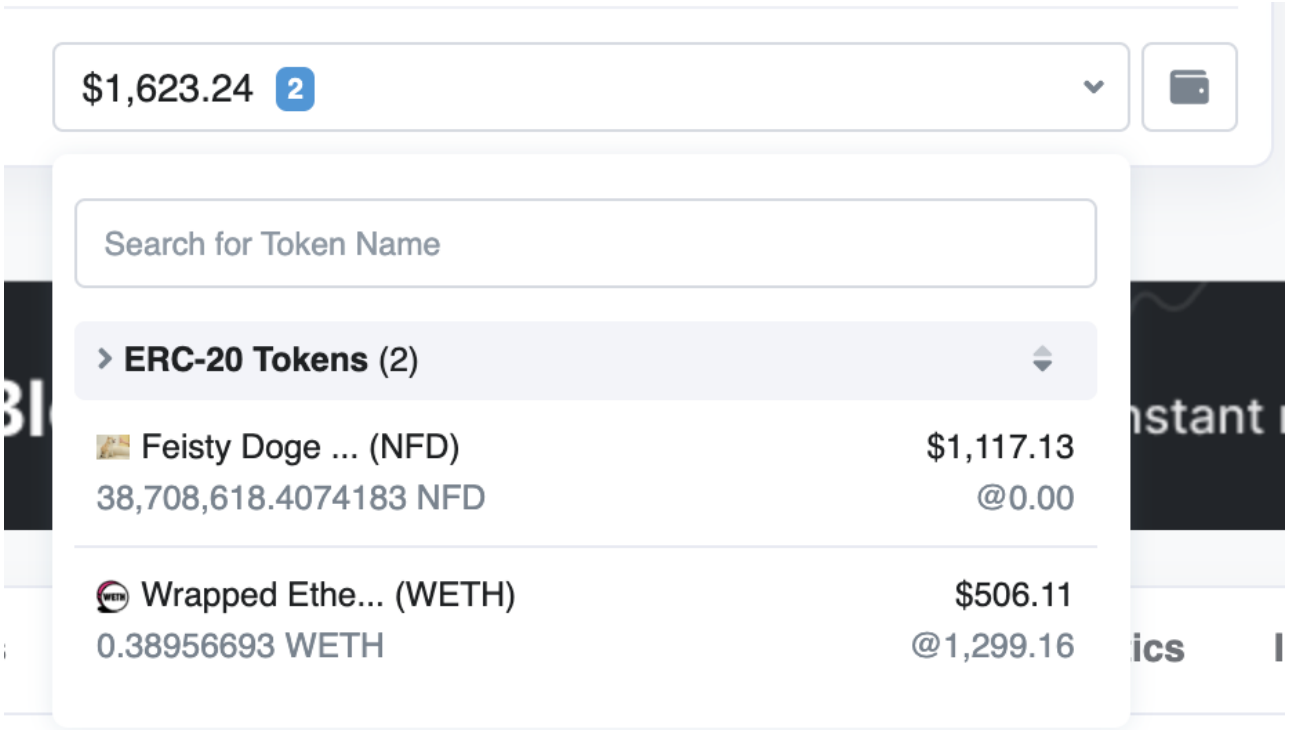
The following were reported by Hacxyk via ImmuneFi this April.

There was a verified Rari Fuse pool (i.e. lending market) which allowed DOG as collateral, with around 4M USD worth of tokens up for borrowing.

They used the native UniswapV3 price oracle, which isn't a bad thing.

Except...

Liquidity Rules Everything: Rari Capital



Snapshot at Oct 12. It was similar in April as well. Oops.

Crypto World is Crazy: The Case of LUNA

Sometimes the crypto world goes so berserk that even the fundamental assumptions get dunked on. Such examples can be found from LUNA.

We saw

- **Massive** de-pegging of a Top 5 TVL Stablecoin
- **Massive** price drop of a Top 10 Cryptocurrency

Case 1: No, UST is not 1 USD

Some protocols fix the price of a stablecoin to be 1 USD.

Kava Network did this for UST - they fixed the oracle price for UST at 1 USD - and they paid the price. Attackers could buy cheap UST and add it as collateral, then mint USDX, another stablecoin, against it then dump.

USDX price. The impact can be easily seen.

Case 2: No, LUNA is not at least 0.1 USD

Chainlink price oracles have a min/max cap for the reported price.

For LUNA, such minimum cap was 0.1 USD.

If you saw the markets, it's not hard to guess what happened.

Venus Protocol and Blizz Finance lost roughly 13.5M and 8.3M USD.

Recent Topics: Curve LP Token Price Manipulation

The following vuln was reported by ChainSecurity and presented at Devcon.

Curve LP Token's virtual price function reads the current balances.

This is a problem, as you can

- add a ton of liquidity, then remove the liquidity
- as one of the tokens are sent back, call other functions

Tokens that have callbacks (like native ETH or ERC-777's) were vulnerable.
This leads to LP token price inflation, as more tokens are in the contract.

Table of Contents

TWAP Price

As we discussed before, the TWAP of a token is computed by

$$\frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

where $a_t = \sum_{i=1}^{t} p_i$ is the price accumulator.

We'll discuss the methods to manipulate TWAP from 2022/445 (IACR)

Assumptions

We'll consider the following scenario.

- UniswapV2 market $A - USDC$ with reserves (R_A, R_{USD})
- True price of A is p USDC, with infinite liquidity at CEX
- No 0.3% fees, so the $xy = k$ invariant is in place
- Arbitrage profits go straight to the block miner or validator

Note that UniswapV2 takes the end of the block for price oracle snapshot.

Manipulation of a Single Block

Assume we want to inflate the price of A to $(1+\epsilon)p$.

We want to swap δ_{USD} USDC to make

$$(R_{USD}+\delta_{USD})/(R_A-\delta_A) = (1+\epsilon)p$$
$$(R_{USD}+\delta_{USD})(R_A-\delta_A) = R_A R_{USD}$$

which gives the cost

$$\delta_{USD} - p\delta_A = \left((1+\epsilon)^{1/2} + (1+\epsilon)^{-1/2} - 2\right) R_{USD}$$

Manipulation of a TWAP

Assume that we need a N-block TWAP manip to $(1+\epsilon)p$.

There are two ways.

- Manipulate the price to $(1+\epsilon)p$ for N times
- Manipulate the price to $(1+N\epsilon)p$ for exactly once

Manipulation of a TWAP

We can compare the two cases with N, ϵ.

$$\frac{\text{Manipulate Once}}{\text{Manipulate All}} = \frac{(1+N\epsilon)^{1/2} + (1+N\epsilon)^{-1/2} - 2}{\left((1+\epsilon)^{1/2} + (1+\epsilon)^{-1/2} - 2\right) \cdot N}$$

Note that this goes to 0 as N increases, so manipulating once is easier.

Controlling Two Blocks in a Row

The fun part on the second method is that it just runs over two blocks.

Therefore, if a miner/validator has control of two blocks in a row, the attack works without any cost, as no arbitragers can get in the way.

The way this works differ between PoW and PoS systems.

Controlling Two Blocks in a Row: PoW

In PoW, no one knows who will mine the block. However, a miner with hash rate share of p will have a probability p of mining a block.

Via some math with Markov Chains, we crack the numbers

- The goal is to mine two blocks faster than the others
- If there's one block mined, there are uncle block rewards for grabs

Controlling Two Blocks in a Row: PoW

Denoting S as the number of blocks until the miner controls two blocks,

$$E(S) = \frac{1 + 2p - p^2}{2p^2 - p^3}$$

and the number of expected uncle blocks generated is $\approx E(S) \cdot p$.

Therefore, given the two parameters

- the uncle block rewards
- the cost for renting hashrate

one can solve for the optimal p for the attack

Controlling Two Blocks in a Row: PoS

The validators to propose the blocks are known at least one epoch before.

Therefore, we have to consider

- two validators directly colluding with each other
- validator bribes the other validator who proposes the block before
- increasing stake to have better probability

Alternates to TWAP: Median

The main issue is that the average is manipulated to easily by outliers.

Therefore, we could use the well-known statistic that is not, the median.

- easier multi-block attack by 50%
- impossible to perform single-block attack
- may be difficult to implement in a smart contract

Alternates to TWAP: Geometric Mean

Another method is to make the capital requirement extremely high.
This can be done with Geometric TWAP, the native UniswapV3 method.

Instead of $(1 + N\epsilon)p$, the single-block attack needs $(1 + \epsilon)^N p$.
This is infeasible as large N requires exponentially large capital.

Michael Bentley of euler finance has analyzed the cost of manipulation.
See https://oracle.euler.finance/ for further details.

Thank You! Any Questions?

people@haechi.io