#HashEx
BLOCKCHAIN SECURITY

Manufactory Public Lands Sale

smart contracts final audit report

April 2022


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).

2. Overview

HashEx was commissioned by the Manufactory team to perform an audit of their smart contracts. The audit was conducted on 19/04/2022.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available at 0xad89021333aFC533c7336C758c02d0c7827B292d and 0xd19F6c8628f49F1aC4233eC04cEf0661ad5bef93 in BSC testnet.

2.1 Summary

Project name	Manufactory Public Lands Sale
URL	https://manufactory.gg
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
PublicLandsSale	

AbstractLandsSale

PCSPriceOracle

PancakeLibrary

PancakeOracleLibrary

3. Found issues

Medium	1 (25%)
Low	2 (50%)
Info	1 (25%)

C1. PublicLandsSale

ID	Severity	Title	Status
C1-01	Low	Gas optimization	Open
C1-02	Low	Price manipulation on purchase	Open

C2. AbstractLandsSale

ID	Severity	Title	Status
C2-01	Info	AccessControl misuse	Open

C3. PCSPriceOracle

ID	Severity	Title	Status
C3-01	Medium	PERIOD value	Open

4. Contracts

C1. PublicLandsSale

Overview

The contract implements the AbstractLandsSale contract with a purchase method. In the current version of the contract, a buyer sends tokens to the contract and the contract just updates its variables.

Issues

C1-01 Gas optimization

Low Open

The state variable `relativeToken` can be declared as immutable to save gas.

C1-02 Price manipulation on purchase

Low Open

The price in the `purchase()` function is calculated within a fixed window TWAP oracle with Pancakeswap pair. TWAP oracles can be manipulated if the time frame is narrow, otherwise, the oracle may suffer from incorrect answers during the significant volatility.

C2. AbstractLandsSale

Overview

An abstract contract that implements a whitelist, sale period, and all the data relative to the market like the accepted token and the price.

Issues

C2-01 AccessControl misuse

Info Open

OpenZeppelin's AccessControl authentication model assumes defining specific roles. Using only the DEFAULT_ADMIN_ROLE provides zero benefits over the classic Ownable model, but slightly reduces the safety in general as there shouldn't be multiple concurrent default admins.

C3. PCSPriceOracle

Overview

Fixed window oracle that recomputes the average price for the entire period once every period.

Issues

C3-01 PERIOD value

Medium Open

The observation time window is set to 10 seconds which can be considered a low value. In extreme cases, an attacker might be able to manipulate the price long enough (3 consecutive blocks) to fool the oracle.

Recommendation

Consider increasing the time window.

C4. PancakeLibrary

Overview

The library for performing overflow-safe math and interaction with LP-pairs. No issues were found.

C5. PancakeOracleLibrary

Overview

The library with helper methods for oracles that are concerned with computing average prices. No issues were found.

5. Conclusion

1 medium, 2 low, and 1 information severity issue were found. The PublicLandsSale contract itself only attracts users' funds without providing any output. Users have to trust the project owner.

This audit includes recommendations on improving the code and preventing potential attacks.

Appendix A. Issues severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Info.** Issues that do not impact the contract operation. Usually, info severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY