HashEx

BondAppétit

smart contracts audit report

Prepared for:

BondAppétit

Authors: HashEx audit team

June 2021

Contents

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as 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.

DISCLAIMER: 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.

Introduction

HashEx was commissioned by the BondAppétit team to perform an audit of BondAppétit's smart contracts. The audit was conducted between May 19 and May 24, 2021. The code recheck was done between May 28 and June 1, 2021.

The code located in github repository @bondappetit/bondappetit-protocol was audited after the commit `21f9f5a`. It should be noted, only the BuybackDepositaryBalanceView contract from this repo was audited.

Documentation and white paper can be found on bondappetit.io.

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 to understand 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.
Public audit by MixBytes of severalBondAppétit contracts can be found in their GitHub [1].

Update: BondAppétit team has responded to this report. Individual responses to the high severity issues were added after each item in section. Those fixes can be found in the repo after the `1bbe963` commit.

Contracts overview

`BuybackDepositaryBalanceView.sol`

Buyback contract for stable tokens.

`OwnablePausable.sol`

Derivative of OpenZeppelin's Ownable and Pausable contracts.

`Issuer.sol`

Contract to balance the volume of issued stable tokens and the cost of the collateral.

`AgregateDepositaryBalanceView.sol`

Contract to control depositaries.

`StableToken.sol`

ERC20 token with restricted mint and burn.

`AccessControl.sol`

Whitelist and modifier for it.

Found issues

ID	Title	Severity	Response
01	productAmount exceeds balance	High	Fixed
02	Negative issuerInbalance throw	Medium	Fixed
03	Checks for zero input data	Low	Fixed
04	General recommendations	Low	Fixed

#01 productAmount exceeds balance — High

BuybackDepositaryBalanceView contract L93 contains check of `amount >= balance`. Error message and transfer in L97 state that it should be `amount <= balance`.

The issue was fixed in the update.

#02 Negative issuerInbalance throw — Medium

`issuerInbalance` variable in BuybackDepositaryBalanceView contract L99 uses safe subtraction of the balance of the Issuer contract from total supply. However, `rebalance()` function of Issuer contract L65 suggests that total supply could be greater than balance. In that case, `buy()` function will throw without a specific error.

The issue was fixed in the update.

#03 Checks for zero input data — Low

BuybackDepositaryBalanceView L47 and Issuer L38, 47 lack the zero check of input data. We recommend adding this check to avoid calling functions with unset parameters.

The issue was fixed in the update.

#04 General recommendations Low

BuybackDepositaryBalanceView imports Treasury.sol contract that is never used.

Event `Buyback` in BuybackDepositaryBalanceView contract should index at least address to allow filtering by its value.

`product` variable in BuybackDepositaryBalanceView contract should be declared immutable. This will save gas for reading the variable from blockchain.

`decimals` variable in BuybackDepositaryBalanceView contract should be declared constant. This will save gas for reading the variable from blockchain.

Typos in comments in BuybackDepositaryBalanceView contract L27, 30, 54, 56, 71, 80.

Pragma version should be fixed to avoid situations when the contract is tested on one version of the compiler but is deployed on another.

AgregateDepositaryBalanceView is misspelled.

OwnablePausable L30, L38 `pause()` and `unpause()` functions should be declared external.

OwnablePausable contract uses the pauser role to pause the BuybackDepositaryBalanceView buy function. We recommend using OpenZeppelin's AccessControl contract to better role management, i.e. granting and revoking roles for several accounts if needed.

Issues were fixed in the update.

Commentary on the update:
BuybackDepositaryBalanceView contract's found issues were fixed completely.

Conclusion

At the time of the audit reviewed contract BuybackDepositaryBalanceView was not deployed. Other contracts' addresses can be found on BondAppétit website [2].

1 high severity issue was found.

Update: All high, medium, and most of the low severity issues were fixed in the update.

Audit includes recommendations on the code improving and preventing potential attacks.

References

1. Audit by MixBytes
2. Contracts' addresses

Appendix A. Issues' severity classification

We consider an issue critical, if it may cause unlimited losses or breaks the workflow of the contract and could be easily triggered.

High severity issues may lead to limited losses or break interaction with users or other contracts under very specific conditions.

Medium severity issues do not cause the full loss of functionality but break the contract logic.

Low severity issues are typically nonoptimal code, unused variables, errors in messages. Usually, these issues do not need immediate reactions.

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