HashEx

ToolApe

smart contract audit report

Prepared for:

ToolApe

Authors: HashEx audit team

May 2021

Contents

Disclaimer 3

Introduction 4

Contracts overview 4

Found issues 5

Conclusion 6

References 6

Appendix. Issues' severity classification 7

Appendix B. List of examined issue types 7

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 security is purely based on smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Introduction

HashEx was commissioned by the ToolApe team to perform an audit of SuperBid token smart contracts. The audit was conducted between May 09 and May 11, 2021.

The audited smart contract is deployed to Binance Smart Chain (BSC) at the address of [0x993660CD4b6529ac122571cE434219bDB506E0fb](0x993660CD4b6529ac122571cE434219bDB506E0fb).

The audited contract is an ERC20 token with minor extensions.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts.
- Ensure that smart contract functions perform as intended.

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.

Contracts overview

`ToolApe`

Implementation of ERC20 token standard [1]. Supports burns: a user can burn it's tokens.

`Ownable`

Standard OpenZeppelin contract without the `renounceOwnership()` function.

`Pausable`

Custom implementation based on OpenZeppelin contract. Owner of the contract can call the function `pause()` up to May 20 2021 15:00:00 GMT+00.

`IBEP20`

Custom interface of BEP20 standard [2]. Lacks getter functions `decimals()`, `symbol()`, `name()`, `getOwner()`.

`HasForeignAsset`

Transfers any ERC20 token from the balance of the contract. Intended to be used if any ERC20 token was accidentally sent to the token address.

Found issues

ID	Title	Severity	Response
01	BEP20 standard violation	Medium	
02	Transfer to contract is restricted	Low	
03	No error messages in pause	Low	
04	Redundant receive() function	Low	
05	General recommendations	Low	

#01 BEP20 standard violation Medium

Interface `IBEP20` lacks some getters. Token contract has no `getOwner()` function required by BEP20 standard [2].

#02 Transfer to contract is restricted Low

`_transfer()` function contains a restriction for transfers to this contract. This seems excessive as the function to retrieve ERC20 tokens was implemented.

#03 No error messages in pause Low

`whenPaused` and `whenNotPaused` modifiers should revert with error messages to provide a better user experience.

#04 Redundant receive() function Low

Payable `receive()` function with revert is not needed for this pragma version.

#05 General recommendations Low

Address library is not in use and may be removed.
Pragma version should be fixed.
No need to use SafeMath library as the contract is compiled with v0.8.4+commit.c7e474f2 Solidity compile version which includes overflow/underflow checks.
Generally, for retrieving any ERC20 token SafeERC20 library should be used.

Conclusion

Reviewed contract is deployed at 0x993660CD4b6529ac122571cE434219bDB506E0fb in BSC. The audited contract is an ERC20 token made with standard OpenZeppelin templates with minor modifications.

No critical nor high severity issues were found.

Audit includes recommendations on code improvement.

References

1. EIP-20: ERC-20 Token Standard
2. BEP20 Standard by Binance

Appendix. 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 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