HashEx

TIKI token

smart contracts audit report

Prepared for:

tikitoken.finance

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 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.
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 (https://hashex.org).

Introduction

HashEx was commissioned by the TIKI team to perform an audit of their smart contracts. The audit was conducted between June 14 and June 17, 2021.

The audited code was provided in .sol files without any documentation.

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.

Update: TIKI team has responded to this report. Individual responses were added after each item in the section. The Whitepaper is available here. The updated code is deployed to Binance Smart Chain (BSC): `0x9b76D1B12Ff738c113200EB043350022EBf12Ff0`.

Contracts overview

`DividendPayingToken.sol`

ERC20-like token with blocked transfers. The modified version of DividendPayingToken by Roger-Wu [1]. Further referred to as DPT.

`TIKI.sol`

ERC20 token with the 5% liquidity fee and 10% dividends to token holders. Further referred to as TIKI.

`ERC20.sol`

Implementation of ERC20 token standard with the possibility of increase/decrease allowance.

`IterableMapping.sol`

Library for iterable mapping from address to uint.

`SafeMath.sol, SafeMathInt.sol, SafeMathUint.sol`

SafeMath libraries for int and uint variables. Different from OpenZeppelin's ones.

`Various interfaces`

Found issues

ID	Title	Severity	Response
01	ERC20: unsafe math	High	Fixed
02	SafeMathInt: division requires b>0	High	Fixed
03	TIKI: swapTokensForEth uses 100% slippage	Medium	Informed
04	TIKI: hardcoded addresses	Medium	Fixed
05	TIKI: BEP20 token standard violation	Medium	P/Fixed
06	TIKI: update of DPT balances with try method	Medium	Informed
07	TIKI: tx is limited only for WETH pair	Medium	Fixed
08	DPT: BNB transfers with a low gas limit	Medium	Informed
09	TIKI: unindexed events	Low	Fixed
10	TIKI: fees should be constants	Low	Fixed
11	TIKI: multiple checks amount>0 in transfers	Low	Fixed
12	TIKI: taxFee name is misleading	Low	Fixed
13	DPT: transfers are denied	Low	Informed
14	IterableMapping: inserted[] isn’t needed	Low	Informed
15	General recommendations	Low	Informed

#01 ERC20: unsafe math — High

`mint()` and `increaseAllowance()` functions use unchecked addition. We recommend using the OpenZeppelin libraries.

The issue was fixed in the update. The functions in the update use SafeMath for addition.

#02 SafeMathInt: division requires b>0 — High

`div()` function of `a/b` requires `b>0`. We recommend using the OpenZeppelin libraries.

The issue was fixed in the update. Updated code uses SafeMathInt from Ampleforth.

#03 TIKI: swapTokensForEth uses 100% slippage — Medium

`swapTokensForEth()` function calls PancakeRouter with 100% slippage. That makes flash loan attacks possible (actually any Safemoon fork should be susceptible to these attacks). Limiting the max transaction amount should reduce the attack probability.

#04 TIKI: hardcoded addresses — Medium

Hardcoded `uniswapV2Router` address makes it impossible to migrate to a new version of DEX in case of future upgrades of PancakeSwap periphery.

The issue was fixed in the update. Setter for `uniswapV2Router` address was added.

#05 TIKI: BEP20 token standard violation — Medium

Implementation of `transfer()` function in TIKI token does not allow to input zero amount as it's demanded in the ERC-20 [2] and BEP-20 [3] standards. This issue may break the interaction with smart contracts that rely on full ERC20 support. The BEP20 standard isn't fully supported as the token lacks the `getOwner()` function.

The issue was partially fixed in the update. Zero transfers made valid, `getOwner()` function is still absent.

#06 TIKI: update of DPT balances with try method — Medium

`_transfer()` function of TIKI token calls for `dividendTracker.setBalance()` via `try` method which makes a successful transfer with unchanged balances of dividends tokens possible. The current dividendTracker implementation should not fail on setting balances. It must be noted that `dividendTracker` can be updated and in case the function `setBalance` fails, discrepancies in token balances can take place.

#07 TIKI: tx is limited only for WETH pair — Medium

Transaction amount limit is enabled only for sells in PancakeSwap pair with WETH. This doesn't prevent selling a large number of tokens on other dexes.

The issue was fixed in the update. A mapping for pair addresses was added in the updated code.

#08 DPT: BNB transfers with a low gas limit — Medium

`_withdrawDividendOfUser()` function of DividendPayingToken contract transfers ETH/BNB via `.call{value, gas}` method with 3000 of gas limit. This relatively small constant may cause problems with future ETH/BSC updates as operation gas costs may change with forks.

#09 TIKI: unindexed events — Low

All the events are completely unindexed.

The issue was fixed in the update. Parameters were set as indexed where needed.

#10 TIKI: fees should be constants — Low

Fee variables aren't changed after creation and therefore should be declared constant or immutable. Setting fees as constant/immutable will save gas on reading values from the blockchain.

The issue was fixed in the update. The parameters were set as immutable.

#11 TIKI: multiple checks amount>0 in transfers — Low

Transaction amount limit is enabled only for sells in PancakeSwap pair with WETH. This doesn't prevent selling on other dexes.

The issue was fixed in the update. Mitigation of the #05 issue has also fixed this issue.

#12 TIKI: taxFee name is misleading Low

We believe that the TIKI token is inspired by the SafeMoon model and adopts some variable names. However, `taxFee` naming is misleading as unlike SafeMoon it doesn't increase balances of TIKI holders (but used for paying dividends).

The issue was fixed in the update. Variable `taxFee` was renamed to `BNBRewardsFee`.

#13 DPT: transfers are denied Low

All the transfers of DividendPayingToken are blocked which makes it non-ERC20. It may be slightly confusing as many explorers will show TIKI_Dividend_Tracker as ERC20 token.

#14 IterableMapping: inserted[] isn't needed Low

IterableMapping library could save gas by getting rid of `inserted[]` mapping and use `indexOf[]` instead.

#15 General recommendations Low

We strongly recommend using original OpenZeppelin contracts as they are widely used and well audited. If some changes are needed to the original contracts, implement them via inheritance.

We also recommend following Solidity naming conventions [4], i.e. UPPERCASE for constants/immutable.

Conclusion

2 high severity issues were found. The issues are brought by not using the well-tested and audited library contracts but using custom implementations of ERC20 token and SafeMathInt libraries.

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

Update: TIKI team has responded to this report. All high severity issues were fixed among most of the medium and low severity issues. Individual responses to the issues were added after each item in the section. The updated code is deployed to BSC: `0x9b76D1B12Ff738c113200EB043350022EBf12Ff0`.

References

1. DividendPayingToken by Roger Wu
2. ERC-20 standard
3. BEP-20 standard
4. Solidity Docs: Naming Styles

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