HashEx

Elk Finance

smart contracts audit report

Prepared for:

elk.finance

Authors: HashEx audit team

July 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, 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 Elk Finance team to perform an audit of their smart contracts. The audit was conducted between July 09 and July 20, 2021.

The code located in the @ elkfinance/exchange-contracts github repository was audited after the d776510 commit. The whitepaper is available on the team's website.
The same contracts are deployed to Huobi Eco Chain (HECO):
Factory 0x997fCE9164D630CC58eE366d4D275B9D773d54A4,
Router 0x62710D18596c808c70864695c77480De252DD9D5.

The documentation is available on docs.elk.finance.

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.
We found out that the code is a fork of UniSwapV2. An audit for Uniswap is available [1]. For this reason we focused on the unaudited parts of code, as well as modifications made by Elk Finance.

Contracts overview

`ElkERC20.sol`

Implementation of ERC20 token standard with additional permission functionality.
Allows the unlimited token allowance if set to `uint(-1)`. The same as UniswapV2ERC20.

`ElkFactory.sol`

The same as UniswapV2Factory.

`ElkPair.sol`

The same as UniswapV2Pair.

`ElkRouter.sol`

Same as UniswapV2Router02.

`Libraries & Interfaces`

The same as UniswapV2. elk-lib is a fork of the uniswap-lib repository after `7b00ef2` commit.

Found issues

ID	Title	Severity	Response
01	`ElkERC20: ERC20 implementation`	Low	Acknowledged
02	`General recommendations`	Low	Acknowledged

`#01` `ElkERC20: ERC20 implementation` `Low`

ElkERC20.sol lacks `increaseApproval()` and `decreaseApproval()` functions. These functions mitigate frontrun attacks on the `approve()` function if a user wants to alter previously approved amounts in one transaction (see [2]).

`#02` `General recommendations` `Low`

IElkFactory interface is imported in ElkLibrary but not in use.

Conclusion

Audited contracts are copies of the UniswapV2. No high or medium severity issues were found.
The contracts are deployed to HECO network at addresses:
Factory `0x997fCE9164D630CC58eE366d4D275B9D773d54A4`,
Router `0x62710D18596c808c70864695c77480De252DD9D5`.

References

1. Uniswap audit
2. Attack vector on approve method

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