HashEx
Blockchain Security

Spectre Protocol

smart contracts final audit report

January 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.

2. Overview

This audit report was generated for Spectre Protocol with CryptEx token constructor [1].

The audited code is deployed at 0x76893e9dF37f65eD538774624D5D9092899fcC8c in Binance Smart Chain (BSC).

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 hereby verify that the generated token has identical bytecode with the original audited token. The external audit of the same code was conducted by PaladinSec [2].

2.1 Summary

Project name	Spectre Protocol
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
ReflectToken	`0x76893e9dF37f65eD538774624D5D9092899fcC8c`

3. Found issues

Medium	2 (67%)
Low	1 (33%)

ReflectToken

ID	Title	Severity	Status
01	addLiquidity() recipient	Medium	Acknowledged
02	No slippage checks on swaps and adding liquidity	Medium	Acknowledged
03	General recommendations	Low	Acknowledged

4. Contracts

4.1 ReflectToken

4.1.1 Overview

Implementation of ERC20 token standard with the custom functionality of auto-yield by burning tokens and distributing the fees on transfers. Also has a marketing fee. Default fees values are: distributed between users (0%), automatic addition to liquidity(4%), burning (1%). 10000000000000 tokens were minted to the token creator 0x3301dC96Be657fe64851b0cfe1C7117C58FF3220 during the contract creation.

4.1.2 Issues

01. addLiquidity() recipient

- Medium ⓘ Acknowledged

addLiquidity() function calls for swapRouter.addLiquidityETH() function with the parameter of LP tokens recipient set to the liquidityAddress. With time the liquidityAddress may accumulate a significant amount of LP tokens which may be dangerous for token economics if the owner acts maliciously or their account gets compromised. The owner can change the liquidityAddress address.

Recommendation

Investors should check if the liquidity is actually locked.

02. No slippage checks on swaps and adding liquidity

- Medium ⓘ Acknowledged

The functions _swapTokensForBNB() and addLiquidity() do not perform slippage checks. The transactions may be front-runned.

Recommendation

This is an architectural decision, but the owner of the token should be aware that if the liqThreshold parameter is set to a big value it creates incentives on frontrun attacks.

03. General recommendations

- Low ⓘ Acknowledged

We recommend adding a documentation section to the Project website to track any changes in token parameters made by the owner.

5. Conclusion

The audited contract is ERC20 token with a Reflect.finance [3] auto-yield model with some changes such as the ability to swap itself to BNB and to add liquidity. The audited contract was generated with CryptEx token constructor [1].

No high severity issues were found.

The audited code is deployed at 0x76893e9dF37f65eD538774624D5D9092899fcC8c in Binance Smart Chain (BSC).

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

6. References

1. CryptEx token constructor
2. Audit by PaladinSec
3. Reflect.finace github repo

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.

Informational. Issues that do not impact the contract operation. Usually, informational 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

HashEx
Blockchain Security