HashEx
Blockchain Security

Outrace

smart contracts final audit report

December 2021

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. 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 (hashex.org).

2. Overview

HashEx was commissioned by the Outrace team to perform an audit of their smart contract. The audit was conducted between December 06 and December 08, 2021.

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 for understanding 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.

The code is deployed on BSC network at address 0x91f006ee672f8f39c6e63ca75b1ca14067b3c366. The whitepaper is available at the project's website.

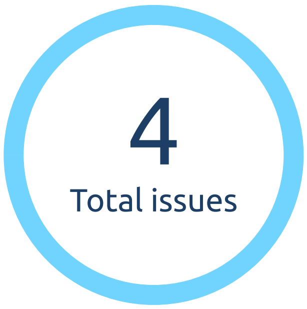
2.1 Summary

Project name	Outrace
URL	https://outrace.game/
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
BurnableTeamToken	0x91f006ee672f8f39c6e63ca75b1ca14067b3c366

3. Found issues



- Low 4 (100%)

BurnableTeamToken

ID	Title	Severity	Status
01	Proper use of external an public function types	Low	Acknowledged
02	Unnecessary check in checkIsAddressValid modifier	Low	Acknowledged
03	Pragma version is not fixed	Low	Acknowledged
04	SafeMath library is not used in the TeamToken contstructor	Low	Acknowledged

4. Contracts

4.1 BurnableTeamToken

4.1.1 Overview

The audited token is a standard ERC20 token with burning functionality. A user can burn their tokens or tokens can be burnt by a previously given allowance. Most of the functionality is inherited from well-tested OpenZeppelin contracts.

4.1.2 Issues

01. Proper use of external an public function types

- Low ⓘ Acknowledged

Functions burn(), burnFrom(), name(), symbol(), decimals(), totalSupply(), balanceOf(), transfer(), approve(), transferFrom(), increaseAllowance(), decreaseAllowance() can be declared external. This will save gas on calling them.

02. Unnecessary check in checkIsAddressValid modifier

- Low ⓘ Acknowledged

The modifier checkIsAddressValid in the TeamToken contract checks that ethAddress == address(ethAddress) which is always true.

Recommendation

We recommend removing unnecessary checks to reduce the contact's code size and save gas.

03. Pragma version is not fixed

- Low ⓘ Acknowledged

The contract declares a too wide range of possible Solidity versions:

```
pragma solidity >=0.6.2 <0.8.0;
```

We recommend fixing the Solidity version to be sure that the contract is deployed with the same compiler version it was tested with.

04. SafeMath library is not used in the TeamToken contstructor

- Low ⓘ Acknowledged

The constructor of the TeamToken performs unchecked uint computations:

```
_mint(owner, supply * 995/1000);
_mint(feeWallet, supply * 5/1000);
```

Recommendation

We recommend using the SafeMath library for all computations in the smart contracts.

5. Conclusion

No critical, high or medium severity issues were found. The audited contract is a standard ERC-20 token with an initial supply of 1,000,000,000 tokens and a burning mechanism.

At the time of the audit, 92.5% of tokens are on a vesting contract (out of scope of the current audit, deployed as upgradeable proxy).

This audit includes recommendations on the gas usage reduction.

The audited contract is deployed to the mainnet of Binance Smart Chain: 0x91f006ee672f8f39c6e63ca75b1ca14067b3c366.

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

contact@hashex.org

@hashexbot

blog.hashex.org

linkedin

github

twitter

HashEx
Blockchain Security