HashEx
BLOCKCHAIN SECURITY

Youniq Labs Public Distribution

smart contracts
final audit report

June 2023

hashex.org

contact@hashex.org

Contents

1. Disclaimer 3
2. Overview 4
3. Found issues 6
4. Contracts 7
5. Conclusion 10
Appendix A. Issues' severity classification 11
Appendix B. List of examined issue types 12

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 Youniq Labs team to perform an audit of their smart contract. The audit was conducted between 10/06/2023 and 16/06/2023.

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 available at https://github.com/techbandorg/youniq-labs-contracts GitHub repository and was audited after the commit 6a2e1cc. The scope of the audit includes PublicDistribution and UnkwnBonesGenesis contracts with their dependencies.

Update. A recheck was done after the commit 0ecd262.

2.1 Summary

Project name	Youniq Labs Public Distribution
URL	https://youniqlabs.io
Platform	Ethereum
Language	Solidity

2.2 Contracts

Name	Address
UnkwnBonesGenesis	
PublicDistribution	
Whitelist	

3. Found issues

3
Total issues

●	Low	2 (67%)
●	Info	1 (33%)

Cac. UnkwnBonesGenesis

ID	Severity	Title	Status
CacIc4	● Low	Gas optimizations	✓ Resolved

Cad. PublicDistribution

ID	Severity	Title	Status
CadIc6	● Low	Gas optimizations	✓ Resolved
CadIc5	● Info	Typos	✓ Resolved

4. Contracts

Cac. UnkwnBonesGenesis

Overview

The Token smart contract is an ERC721 token implementation that includes the ERC2981 royalty standard, ERC721Enumerable and ERC721URIStorage, and access control features from OpenZeppelin. The token has a maximum supply which is set in the constructor.

Centralization risks

- The owner can set a URI of the token to a wrong value.

- Accounts with a MINTER role can mint new tokens until the number of the tokens reaches the maximum total supply.

- The owner can change the recipient and the amount of ERC2981 royalties.

Issues

CacIc4 Gas optimizations ● Low ✓ Resolved

- `maxTotalSupply` variable can be declared immutable

- multiple storage reads in the `updateBaseURI()` function: `baseURI`

- `for()` loop increment could be alter inside `unchecked{}` scope in the `batchSafeMintWithUri()`, `batchSafeMint()`, `updateUriBatch()`, and `userTokens()` functions

- custom errors with error messages doesn't reduce the deployed bytecode, and therefore should be avoided as custom errors may be interpreted incorrectly by user's software - using `revert(errorString)` is recommended

Cad. PublicDistribution

Overview

An NFT token sale contract where users can buy NFT token at a given price. Only whitelisted users are allowed to use contract until specific timestamp. After the sale is ended the owner of the contract distributes purchased tokens.

Centralization risk

- The owner may not distribute purchased tokens after the sale.

Issues

CadIc6 Gas optimizations ● Low ✓ Resolved

- excessive storage data: `distributionInfo.distributionDuration` is derivable from the `distributionInfo.start` and `distributionInfo.end` timestamps and vice versa

- multiple storage reads in the `mint()` function: `mintIndex`

- multiple storage reads in the `withdrawEthToOwner()` function: `owner()`

- multiple storage reads in the `updateDistributionStartTime()` function: `distributionInfo.distributionDuration`

- multiple storage reads in the `availableForClaim()` function: `distributionInfo.totalDistributionQuantity` and `distributionInfo.distributedQuantity`

- unnecessary read from storage in the `_buy()` function: `userBalance[user]`

- `for()` loop increment could be alter inside `unchecked{}` scope in the `mint()` and `_mint()` functions

- unchecked math could be used for `excessETH` calculations in the `_buy()` function

- unnecessary calculations `ethAmount -= excessETH` in the `_buy()` function

- custom errors with error messages doesn't reduce the deployed bytecode, and therefore should be avoided as custom errors may be interpreted incorrectly by user's software - using `revert(errorString)` is recommended

- `start` and `end` , `distributionDuration` fields in DistributionInfo struct may be declared as `uint64` to fit into one storage slot. `totalDistributionQuantity`, `distributedQuantity`, `quantityPerWallet` also may use smaller types depending on the anticipated scale of the quantities being handled, thus optimizing the storage and gas costs

CadIc5 Typos ● Info ✓ Resolved

Typos reduce the code's readability. Typos in 'addresss', 'availavble', 'onwer', 'the function claim...' instead of 'the function claims...'.

Cae. Whitelist

Overview

A simple owner-governed contract implementing a whitelist of privileged accounts. No issues were found.

5. Conclusion

2 low severity issues were found during the audit. 2 low issues were resolved in the update.

The reviewed contracts are highly dependent on the owner’s account. See the centralization risks chapters for each contract. Users using the project have to trust the owner and that the owner's account is properly secured.

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

@hashex_manager

blog.hashex.org

linkedin

github

twitter

HashEx
BLOCKCHAIN SECURITY