HashEx
Blockchain Security

StoneAgeNFT

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 StoneAgeNFT team to perform an audit of their smart contract. The audit was conducted between 2021-12-16 and 2021-12-17.

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 in StoneAgeNFT/GES GitHub repository after commit 3ad20c013.

Update: the StoneAgeNFT team has responded to this report. The updated code is located in the GitHub repository after commit 405bedfa.

The code is deployed on BSC network at address 0xa45c79ef4373b565e6dd4121ebd02b90cb82a54c.

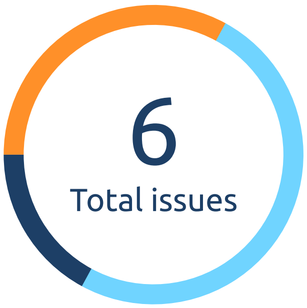
2.1 Summary

Project name	StoneAgeNFT
URL	https://stoneagenft.com/
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
GES	0xa45c79ef4373b565e6dd4121ebd02b90cb82a54c

3. Found issues



High	2 (33%)
Low	3 (50%)
Informational	1 (17%)

GES

ID	Title	Severity	Status
01	Unlimited minting	High	Resolved
02	Changed decimals value	High	Resolved
03	Mixing the Ownable and AccessControl Contracts	Low	Open
04	Redundant code	Low	Open
05	Floating pragma	Low	Acknowledged
06	No version update	Informational	Open

4. Contracts

4.1 GES

4.1.1 Overview

The contract realizes a standard ERC20 interface with upgrade functionality. It is designed to replace a token with security drawbacks and centralization risks in the proxy contract 0x01aD5C8Ca6B2470CbC81c398336F83AAE22E4471.

4.1.2 Issues

01. Unlimited minting

- High   Resolved

The contract inherits the OpenZeppelin's ERC20PresetMinterPauserUpgradeable, which allows to mint any amount of tokens to any address. Unlimited minting might be used for market manipulation.

Recommendation

We suggest restricting the maximum supply and using a multisig mechanism as a way to avoid centralized minting.

Update

The issue was fixed by overriding function mint and making it impossible to use.

02. Changed decimals value

- High
- Resolved

The function decimals is overridden and the new returned value is changed to 5 instead of 18 in the previous version of the contract. The issue can cause market rate changes and users' errors as well.

Recommendation

It's not recommended to update such contract parameters as decimals. If it's still necessary it should be announced at least in order to prevent users' errors.

03. Mixing the Ownable and AccessControl Contracts

- Low
- Open

Dependency on OwnableUpgradeable (which is the implementation of the Ownable contract) is added to the contract, while the AccessControl is already used within the ERC20PresetMinterPauserUpgradeable contract for the same purpose.

04. Redundant code

- Low   Open

Implementation of the approve function in the contract GESV2 is redundant since it just calls the parent's implementation. Also, decimals() function override repeats ERC20Upgradeable decimals() realization and brings about extra gas spending during deployment.

05. Floating pragma

- Low   Acknowledged

The general recommendation is that pragma should be fixed to the version that you are intending to deploy your contracts with. This helps to avoid deploying using an outdated compiler version and shields from possible bugs in future solidity releases.

06. No version update

- Informational   Open

Contracts' public functions modifications without further upgrading of the version may be confusing to users.

5. Conclusion

2 high severity issues were found. The reviewed contracts are still highly dependent on the owner's account and don't get rid of all their predecessors' flaws. Replacing token also overrides the decimals function which will lead to rate loss on the coin market's charts and may cause holders disturbance.

Update: In the updated code both high severity issues are fixed and a few low-grade ones were added.

All found issues are relevant for the implementation contract, which is placed in the GitHub Repository at the time of the audit. The implementation of the Token can be changed since it's deployed via a Proxy contract placed at 0x01aD5C8Ca6B2470CbC81c398336F83AAE22E4471.

This audit includes recommendations on the code improving and preventing potential attacks.

The audited contract is deployed to the mainnet of Binance Smart Chain: 0xa45c79ef4373b565e6dd4121ebd02b90cb82a54c.

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