QuillAudits

Audit Report
March, 2022

For

StackOS

Contents

Scope of the Audit

The scope of this audit was to analyze and document the StackOS Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20/721 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	2	1	0
Closed	0	0	2	2

Introduction

During the period of **January 04, 2021 to February 28, 2021** - QuillAudits Team performed a security audit for StackOS smart contracts.

https://github.com/stackosofficial/node-nft-contract

The code for the audit was taken from following the official link:

V	Date	Commit ID	Files
1	January	7b4999ef00c337b707699e23d5d2d48cad2cb5b8	Contracts/*
2	January	5d600a4cf3bd1399650d16adcdf908fab4048e1d	Contracts/*
3	February	098925436ba5ff6019137584804b1cbb48856c09	Contracts/*

A. Contract - StableCoinAcceptor

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

No issues were found.

Functional Tests

- Should be able to initialize the stablecoins array PASS
- supportsCoin function should return true for addresses in the array PASS

Automated Tests

```
enderphan@enderphan contracts % slither StableCoinAcceptor.sol
INFO:Detectors:
StableCoinAcceptor.supportsCoin(IERC20).i (StableCoinAcceptor.sol#22) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (StableCoinAcceptor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter StableCoinAcceptor.supportsCoin(IERC20)._address (StableCoinAcceptor.sol#21) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
supportsCoin(IERC20) should be declared external:
        - StableCoinAcceptor.supportsCoin(IERC20) (StableCoinAcceptor.sol#21-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StableCoinAcceptor.sol analyzed (2 contracts with 75 detectors), 6 result(s) found
```

Results

No major issues were found. Some false positive errors were reported by Slither. All the other issues have been categorized above according to their level of severity.

B. Contract - DarkMatter

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

1. **Missing Events for Significant Transactions**

   Description

   The missing event makes it difficult to track off-chain decimal changes. An event should be emitted for significant transactions calling the functions: **activate**

   Remediation

   We recommend emitting the appropriate events.

   Status: Fixed

Functional Tests

- StackNFT can be deposited PASS
- Minting for Dark Matter NFT should work PASS
- Only whitelisted can do the transfers PASS

Automated Tests

Results
No major issues were found. Some false positive errors were reported by Slither. All the other issues have been categorized above according to their level of severity.

C. Contract - StackOsNFTBasic

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

2. Token transfers might not return a boolean value

Line	Code

325-331	_stablecoin.transferFrom(msg.sender, address(this), usdToSpend);
	_stablecoin.approve(address(exchange), usdToSpend);
	uint256 stackAmount = exchange.swapExactTokensForTokens(
	usdToSpend,
	_stablecoin,
	stackToken
	);

Description
There are some ERC20 tokens that do not return a boolean value on transfers or only revert on failed transfers. Some might return false on failed transfers.

Remediation
Use the SafeERC20 library provided by openzeppelin to account for all the different types of tokens.

Status: Fixed

Informational issues

No issues were found.

Functional Tests

- Last gen can transfer and mint NFTs PASS
- Users should be able to mint NFTs PASS
- Users can mint using subscription and Royalty rewards PASS
- Only whitelisted can do the transfers PASS

Automated Tests

Results
No major issues were found. Some false positive errors were reported by Slither. All the other issues have been categorized above according to their level of severity.

D. Contract - StackOsNFT

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

3. Allowed NFTs can be more than maxSupply

Description
There can be a scenario where allowed NFTs for partners + 20 (lottery nfts) + auctionNfts > maxSupply .While minting these NFTs, when a new generation is deployed because totalSupply == maxSupply then lottery/ auction/partners won’t be able to claim their pending NFTs.

Remediation
Put some checks in place to make sure that allowed NFTs for mint won't be more than maxSupply.

Status: Acknowledged
Comments: Currently the admin is responsible to be careful to not exceed max supply.

Low severity issues

No issues were found.

Informational issues

No issues were found.

Functional Tests

- Whitelisted partners can mint NFTs PASS
- Only whitelisted can do the transfers PASS
- Users should be able to bid in auction PASS
- Users can participate in lottery PASS
- NFT can be transferred to the next gen PASS
- Lottery rewards can be claimed by winners PASS
- Random number should generated from VRF PASS
- Admin can withdraw the amount after timelock passed PASS

Automated Tests

Results
No major issues were found. Some false positive errors were reported by Slither. All the other issues have been categorized above according to their level of severity.

E. Contract - Royalty

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

4. Txn fees will be more than pending rewards

Description

There will be a time when the pending royalty rewards will be less than the gas fee for that txn. This can happen when there are a lot of generations and the ETH reward is to be shared between all of the holders.

Remediation

We recommend making a change in the logic, so that users can be paid more than the txn gas fee. Ensure that minEthPerCycle is high enough to be divided between all the holders.

Status: Acknowledged

Informational issues

No issues were found.

F. Contract - Market

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

5. Users can create fake listing in the marketplace

Description
The NFT holders can create a listing in the marketplace, giving the NFT approval to the market contract and then lock that NFT in the Dark Matter contract for minting. Now this will create a fake listing in the marketplace, as that token can never be bought.

Remediation
We recommend checking if the owner of that token is the seller, otherwise allow that listing to be removed from the marketplace.

Status: Fixed

Informational issues

No issues were found.

G. Contract - GenerationManager

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

No issues were found.

H. Contract - Subscription

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

6. Centralization Risks

Description

The role owner has the authority to update these critical settings:
- setOnlyFirstGeneration()
- setDripPeriod()
- setPrice()
- setMaxPrice()
- setBonusPercent()
- setTaxReductionAmount()
- setForgivenessPeriod()

As described in the business logic document:
- For every $100 paid in $STACK the user should get back $80 in STACK locked into the NFT which is distributed over 12 months.
- Subscription is a monthly payment. And there is 5 days as a forgiveness period after 30 days.

However, these settings can be changed by the administrator by calling the above-mentioned functions. For example, the forgivenessPeriod value was changed to 7 days in the hardhat test instead of 5 days.

Remediation

If the Auditee has already committed these provisions to the holders, the Auditee should be unable to update these settings.

Status: Acknowledged

Low severity issues

No issues were found.

Informational issues

7. Typos

The following typos were found:
L171: perdiod should be period
L243,L245: Generaion should be Generation
L411, L418: Transfered should be Transferred
L797: elememement should be element

Status: Fixed

I. Contract - Whitelist

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

No issues were found.

Automation Tests Results(All)

https://drive.google.com/drive/folders/1dx_urHc-m_6kKBA2rjguGja588pXgLB2?usp=sharing

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the StackOS platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the StackOS Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
March, 2022

For

StackOS

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com