QuillAudits

Audit Report
June, 2022

For

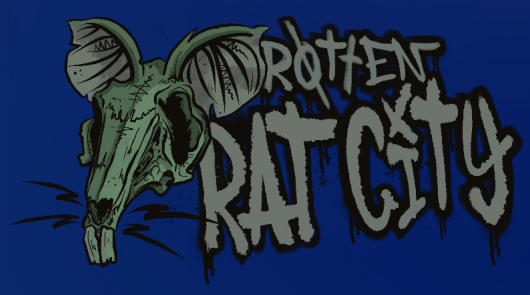
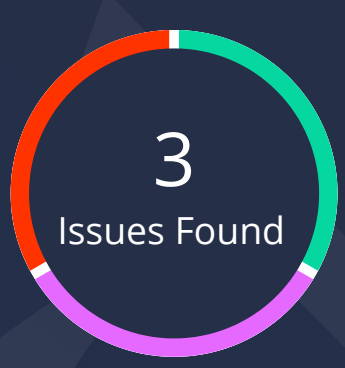


Table of Content

Executive Summary

Project Name	Rotten Rat City
Overview	Rotten Rat City is The collection uses ERC 721 standards, with well-known and secure parts of the contract from OpenZeppelin.
Timeline	22 June, 2022 to 23 June, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Rotten Rat City codebase for quality, security, and correctness.
Sourcecode	*https://github.com/NFT-FABRIC/NFT_SC/blob/main/single_NFT.sol*
Commit Hash	a2259d3412bcae8ef020af1b6d364dcf45b1d1c7
Fixed in	9e775f88c93e751151b76022436d69ec3dacc87a



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	1
Partially Resolved Issues	0	0	0	0
Resolved Issues	1	0	1	0

Types of Severities

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.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- 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 leve

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 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.

Manual Analysis

Contract - NFT.sol

High Severity Issues

1. Reentrancy when minting NFT

Line#1245-1267

```
function mint(uint256 _mintAmount) public payable {
  uint256 supply = totalSupply();
  require(!paused);
  require(_mintAmount > 0);
  require(_mintAmount <= maxMintAmount);
  require(supply + _mintAmount <= maxSupply);

  if (msg.sender != owner()) {
    require(msg.value >= cost * _mintAmount);
  }

  for (uint256 i = 1; i <= _mintAmount; i++) {
    _safeMint(msg.sender, supply + i);
  }
}
```

Line#832-842

```
function _safeMint(
    address to,
    uint256 tokenId,
    bytes memory _data
) internal virtual {
    _mint(to, tokenId);
    require(
        _checkOnERC721Received(address(0), to, tokenId, _data),
        "ERC721: transfer to non ERC721Receiver implementer"
    );
}
```

Description

The mint and white_mint functions use the _safeMint function to mint NFTs to users. The _safeMint function makes a callback to the recipient to check if the recipient implements the correct interface. Any exploiter can design a contract in such a way to reenter the mint function again, inside the callback. This method can be used to exploit the limit on the total supply of the NFTs. An exploiter can use this to mint more than the allowed limit.

Remediation

It is recommended that you use reentrancy guards for such functions, or do not use safeMint versions of the mint function. *Read more*.

Status

Fixed

Medium Severity Issues

No issues were found

Low Severity Issues

2. Multiple solidity versions

Description

The contract has multiple pragma statements to specify the solidity versions. There is no check for overflow and underflow in the mathematical calculations. Using solidity compiler version < 0.8 will create issues.

Remediation

Lock the pragma version to something >= 0.8

Status

Fixed

Informational Issues

3. Missing Events for Significant Transactions

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the function:

- setCost
- setmaxMintAmount
- setBaseURI
- setBaseExtension
- pause

Remediation

We recommend emitting an event for the above-mentioned transactions.

Status

Acknowledged

Functional Testing

Some of the tests performed are mentioned below

- Minting of NFTs should increase the totalSupply
- White minting of NFTs should not exceed 77
- List of tokens should be correct for every wallet

Automated Tests

```
NFT.walletOfOwner(address).i (sc.sol#1296) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (sc.sol#943-964) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (sc.sol#958-960)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

NFT.constructor(string,string,string)._name (sc.sol#1228) shadows:
        - ERC721._name (sc.sol#857) (state variable)
NFT.constructor(string,string,string)._symbol (sc.sol#1229) shadows:
        - ERC721._symbol (sc.sol#858) (state variable)
NFT.walletOfOwner(address)._owner (sc.sol#1288) shadows:
        - Ownable._owner (sc.sol#1137) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

ERC721._checkOnERC721Received(address,address,uint256,bytes) (sc.sol#943-964) has external calls inside a loop: IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (sc.sol#958-960)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (sc.sol#958)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (sc.sol#943-964) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (sc.sol#961)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (sc.sol#957)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (sc.sol#943-964) potentially used before declaration: reason.length == 0 (sc.sol#962)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (sc.sol#957)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (sc.sol#943-964) potentially used before declaration: revert(uint256,uint256)(32 + reason_pcvoud(uint256)(reason)) (sc.sol#957)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.isContract(address) (sc.sol#353-365) uses assembly
        - INLINE ASM (sc.sol#359-363)
Address.verifyCallResult(bool,bytes,string) (sc.sol#482-502) uses assembly
        - INLINE ASM (sc.sol#494-496)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (sc.sol#943-964) uses assembly
        - INLINE ASM (sc.sol#956-958)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['>=0.7.0<0.9.0', '^0.8.0']
        - >=0.8.0 (sc.sol#4)
        - >=0.8.0 (sc.sol#27)
        - >=0.8.0 (sc.sol#105)
        - >=0.8.0 (sc.sol#221)
        - >=0.8.0 (sc.sol#278)
        - >=0.8.0 (sc.sol#295)
```

```
ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (sc.sol#1130-1148) has costly operations inside a loop:
        - delete _allTokensIndex[tokenId] (sc.sol#1146)
ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (sc.sol#1130-1148) has costly operations inside a loop:
        - _allTokens.pop() (sc.sol#1147)
ERC721Enumerable._removeTokenFromOwnerEnumeration(address,uint256) (sc.sol#1105-1123) has costly operations inside a loop:
        - delete _ownedTokensIndex[tokenId] (sc.sol#1121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Address.functionCall(address,bytes) (sc.sol#368-370) is never used and should be removed
Address.functionCall(address,bytes,string) (sc.sol#378-382) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (sc.sol#393-401) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (sc.sol#409-420) is never used and should be removed
Address.functionDelegateCall(address,bytes) (sc.sol#455-457) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (sc.sol#465-476) is never used and should be removed
Address.functionStaticCall(address,bytes) (sc.sol#428-430) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (sc.sol#438-447) is never used and should be removed
Address.sendValue(address,uint256) (sc.sol#381-384) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (sc.sol#482-502) is never used and should be removed
Context._msgData() (sc.sol#129-131) is never used and should be removed
ERC721._baseURI() (sc.sol#878-880) is never used and should be removed
ERC721._burn(uint256) (sc.sol#919-940) is never used and should be removed
Strings.toHexString(uint256) (sc.sol#257-268) is never used and should be removed
Strings.toHexString(uint256,uint256) (sc.sol#273-283) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (sc.sol#4) allows old versions
Pragma version^0.8.0 (sc.sol#27) allows old versions
Pragma version^0.8.0 (sc.sol#105) allows old versions
Pragma version^0.8.0 (sc.sol#221) allows old versions
Pragma version^0.8.0 (sc.sol#278) allows old versions
Pragma version^0.8.0 (sc.sol#295) allows old versions
Pragma version^0.8.0 (sc.sol#317) allows old versions
Pragma version^0.8.0 (sc.sol#361) allows old versions
Pragma version^0.8.0 (sc.sol#386) allows old versions
Pragma version^0.8.0 (sc.sol#991) allows old versions
Pragma version^0.8.0 (sc.sol#1113) allows old versions
Pragma version>=0.7.0<0.9.0 (sc.sol#1220) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (sc.sol#365-366):
        - (success) = recipient.call{value: amount}() (sc.sol#364)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (sc.sol#409-420):
        - (success,returndata) = target.call{value: value}(data) (sc.sol#418)
Low level call in Address.functionStaticCall(address,bytes,string) (sc.sol#438-447):
        - (success,returndata) = target.staticcall(data) (sc.sol#445)
Low level call in Address.functionDelegateCall(address,bytes,string) (sc.sol#465-476):
        - (success,returndata) = target.delegatecall(data) (sc.sol#473)
Low level call in NFT.withdraw() (sc.sol#1292-1295):
        - (os) = address(owner()).call{value: address(this).balance}() (sc.sol#1293)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (sc.sol#703) is not in mixedCase
```

```
Parameter NFT.mint(uint256)._mintAmount (sc.sol#1249) is not in mixedCase
Function NFT.white_mint(uint256) (sc.sol#1269-1287) is not in mixedCase
Parameter NFT.white_mint(uint256)._mintAmount (sc.sol#1269) is not in mixedCase
Parameter NFT.walletOfOwner(address)._owner (sc.sol#1289) is not in mixedCase
Parameter NFT.setCost(uint256)._newCost (sc.sol#1322) is not in mixedCase
Parameter NFT.setmaxMintAmount(uint256)._newmaxMintAmount (sc.sol#1326) is not in mixedCase
Parameter NFT.setBaseURI(string)._newBaseURI (sc.sol#1330) is not in mixedCase
Parameter NFT.setBaseExtension(string)._newBaseExtension (sc.sol#1334) is not in mixedCase
Parameter NFT.pause(bool)._state (sc.sol#1338) is not in mixedCase
Variable NFT.white_cost (sc.sol#1228) is not in mixedCase
Variable NFT.white_maxSupply (sc.sol#1230) is not in mixedCase
Variable NFT.white_paused (sc.sol#1233) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

NFT.maxSupply (sc.sol#1229) should be constant
NFT.white_cost (sc.sol#1228) should be constant
NFT.white_maxSupply (sc.sol#1230) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

name() should be declared external:
        - ERC721.name() (sc.sol#652-654)
symbol() should be declared external:
        - ERC721.symbol() (sc.sol#659-661)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (sc.sol#666-671)
        - NFT.tokenURI(uint256) (sc.sol#1302-1318)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (sc.sol#685-695)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (sc.sol#709-714)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (sc.sol#726-735)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (sc.sol#740-746)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (sc.sol#1038-1041)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (sc.sol#1200-1202)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (sc.sol#1208-1211)
mint(uint256) should be declared external:
        - NFT.mint(uint256) (sc.sol#1249-1267)
white_mint(uint256) should be declared external:
        - NFT.white_mint(uint256) (sc.sol#1269-1287)
walletOfOwner(address) should be declared external:
        - NFT.walletOfOwner(address) (sc.sol#1289-1300)
setCost(uint256) should be declared external:
        - NFT.setCost(uint256) (sc.sol#1322-1324)
setmaxMintAmount(uint256) should be declared external:
        - NFT.setmaxMintAmount(uint256) (sc.sol#1326-1328)
setBaseExtension(string) should be declared external:
        - NFT.setBaseExtension(string) (sc.sol#1334-1336)
pause(bool) should be declared external:
        - NFT.pause(bool) (sc.sol#1338-1340)
withdraw() should be declared external:
```

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

Closing Summary

In this report, we have considered the security of the Rotten Rat City project. We performed our audit according to the procedure described above.

Some issues of High, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the End and Rotten Rat Team Resolved High and Low severity issue and Acknowledged Informational Issue.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Rotten Rat City 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 Rotten Rat City Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

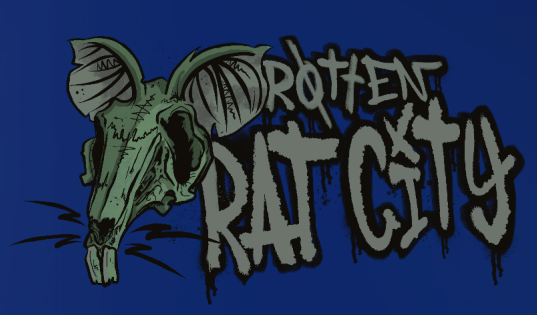
$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
June, 2022

For



QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com