Smart Contract
Security Audit
V1

ONRYO 888 Smart Contract

11/5/2022

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

Table of Contents

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Ethereum
- **Contract Address**: 0x3f217c1a6e7ecb5e5bb36a5728d62b5c188aaf63
- **Code:**

https://rinkeby.etherscan.io/address/0x688d2e66267EBd17966c30e33824724E9DaE27bd#code

NFT Information

- Name: ONRYO
- Total Supply: 888
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

ONRYO 888 Smart contract on ETH test net to test write functions by the auditor.

https://rinkeby.etherscan.io/address/0x3f217c1a6e7ecb5e5bb36a5728d62b5c188aaf63

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well-Secured**. Because the team fix all low issues.

Well Secured	✓
Secured	
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 4 low, 0 very low-level issues and 0 note in all solidity files of the contract

The files:

ONRYO888.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
ONRYO888.sol	**588cae89e3356e0c767f612657f512aed878eadaa461cf9cf02d5e876803dfd0**	0x3f217c1a6e7ecb5e5bb36a5728d62b5c188aaf63

- Contract: ONRYO888
- Inherit: ERC721Enumerable, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
cost	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
hiddenURL	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
maxWLMintAmountPerWallet	✓	Read / public	**Passed**
maxMintAmountPerTx	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

tokenURI	✓	Read / public	**Passed**
tokenByIndex	✓	Read / public	**Passed**
tokenOfOwnerByIndex	✓	Read / public	**Passed**
whiteListCost	✓	Read / public	**Passed**
whitelistMerkleRoot	✓	Read / public	**Passed**
nftPerWLAddress	✓	Read / public	**Passed**
nftPerAddress	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
reveal	✓	Read / public	**Passed**
maxSupply	✓	Read / public	**Passed**
WLPaused	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
setRevealed	✓	Write / public	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
setUriPrefix	✓	Write / public	**Passed**
mint	✓	Write / payable	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
setPaused	✓	Write / public	**Passed**
setWhitelistMerkleRoot	✓	Write / public	**Passed**
setWLCost	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
whitlistMint	✓	Write / payable	**Passed**

setCost	✓	Write / public	**Passed**
setWLPaused	✓	Write / public	**Passed**
Reserve	✓	Write / public	**Passed**
setHiddenMetadataUri	✓	Write / public	**Passed**
setMaxMintAmountPerTx	✓	Write / public	**Passed**
setMaxTxPerWlAddress	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found

Low:

#Missing zero address validation

Description

When the owner wants to Reserve for the investors it has to check for the zero address to make, he didn't mint for the burn address. Otherwise, the mint function will act like the burn function.

```
function Reserve(uint8 _mintAmount, address _receiver) external onlyOwner {
    uint16 totalSupply = uint8(_owners.length);
    require(totalSupply + _mintAmount <= maxSupply, "Excedes max supply.");
    for(uint16 i; i < _mintAmount; i++) {
    _mint(_receiver , totalSupply + i);
     }
     delete _mintAmount;
     delete _receiver;
     delete totalSupply;
  }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version2.

#Multiple pragma statements

Line	Pragma
7	pragma solidity ^0.8.0;
65	pragma solidity ^0.8.0;
104	pragma solidity ^0.8.0;
131	pragma solidity ^0.8.1;
209	pragma solidity ^0.8.0;
239	pragma solidity ^0.8.0;

267	pragma solidity ^0.8.0;
298	pragma solidity ^0.8.0;
443	pragma solidity ^0.8.0;
474	pragma solidity ^0.8.0;
502	pragma solidity ^0.8.7;
963	pragma solidity ^0.8.7;
1011	pragma solidity ^0.8.7;

Description

There are multiple pragma statements in the code. Only the compiler version 0.8.7 will work with the code, but keeping only one pragma statement helps in maintaining readability of the code.

Remediation
Keep a single pragma statement.

Status: Closed. Fixed In version 2

#Owner privileges (In the period when the owner isn't renounced)
Description

The owner can pause and un Pause mint.
The owner can pause and un Pause for whitelist mint.
The owner can change the price in WL or non WL stage.

```
function setRevealed() external onlyOwner {
    reveal = !reveal;
  }
   function setWLPaused() external onlyOwner {
    WLpaused = !WLpaused;
  }
function setPaused() external onlyOwner {
    paused = !paused;
  }
  function setCost(uint _cost) external onlyOwner{
      cost = _cost;
  }
```

Remediation

Make these functions internal in next version or the team should announce the investors before pause and un pause to give them time if they want to do anything.

P.S: This issue is common to the majority of NFT smart contracts.

Status: Acknowledged.

#Missing require statement in reserve

Description

When the owner wants to control the Reserve function because the developer add function allows the owner to pause/un-pause the reserved mint but the developer had missed adding the required statement to check if the owner has paused or un-paused the reserve the function.

```
function Reserve(uint8 _mintAmount, address _receiver) external onlyOwner {
    uint16 totalSupply = uint8(_owners.length);
    require(totalSupply + _mintAmount <= maxSupply, "Excedes max supply.");
    for(uint16 i; i < _mintAmount; i++) {
    _mint(_receiver , totalSupply + i);
     }
     delete _mintAmount;
     delete _receiver;
     delete totalSupply;
  }
```

Remediation

Use the require statement to check for pausing the function.

Status: Closed. Fixed in version2.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

No Notes were found.

Automatic Testing

1- Check for security

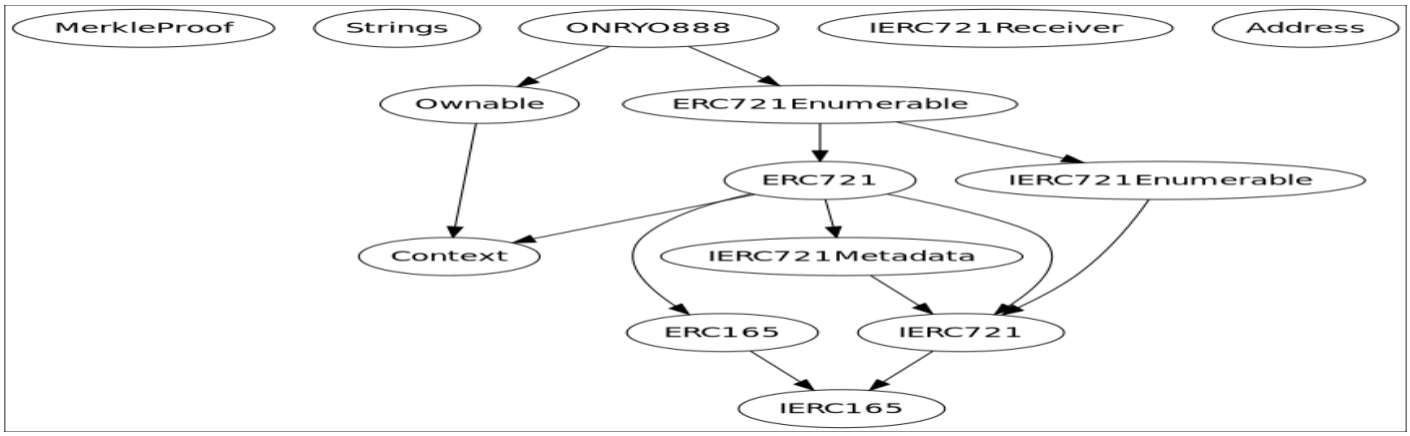
588cae89e3356e0c767f612657f512aed878eadaa461cf9cf02d5e876803dfd0

File: ONRYO... | Language: solidity | Size: 35391 bytes | Date: 2022-05-11T10:56:55.993Z

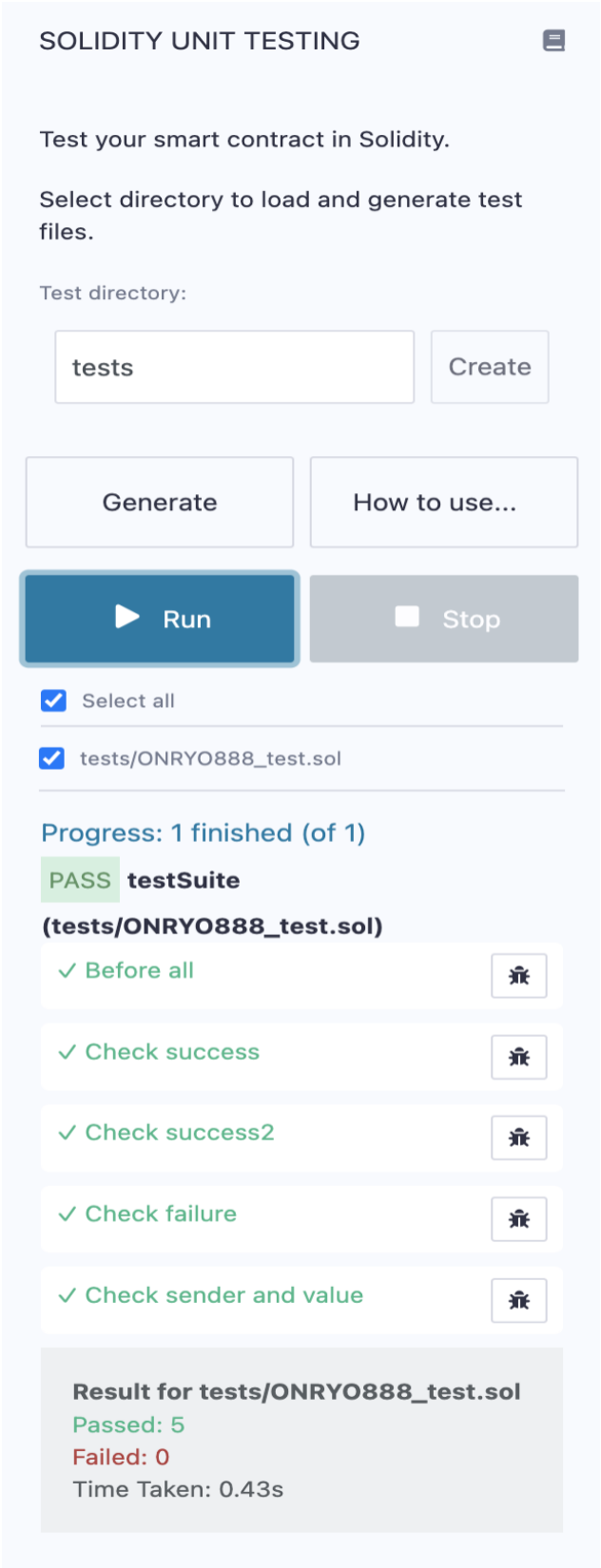
Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS

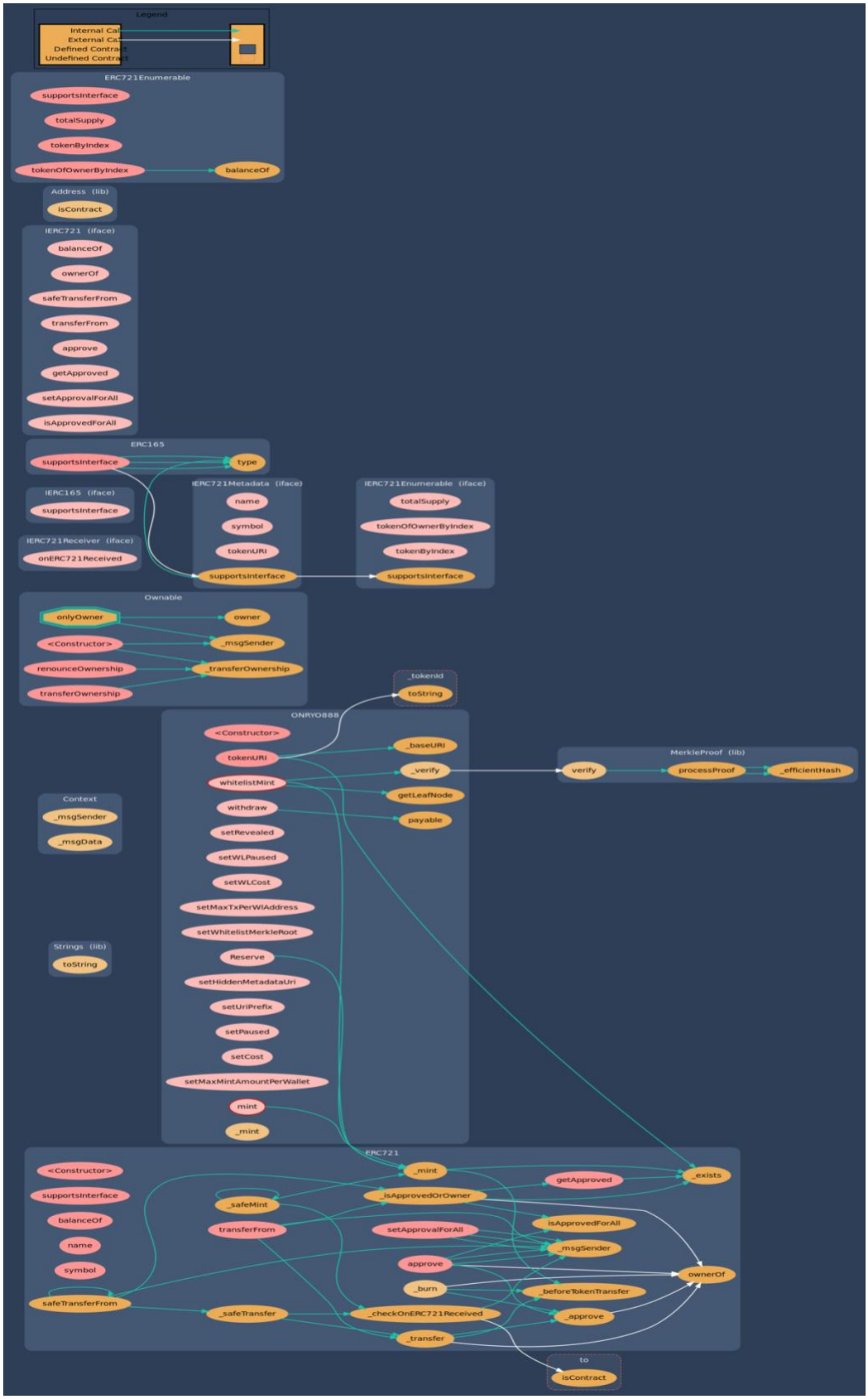
3- Inheritance graph



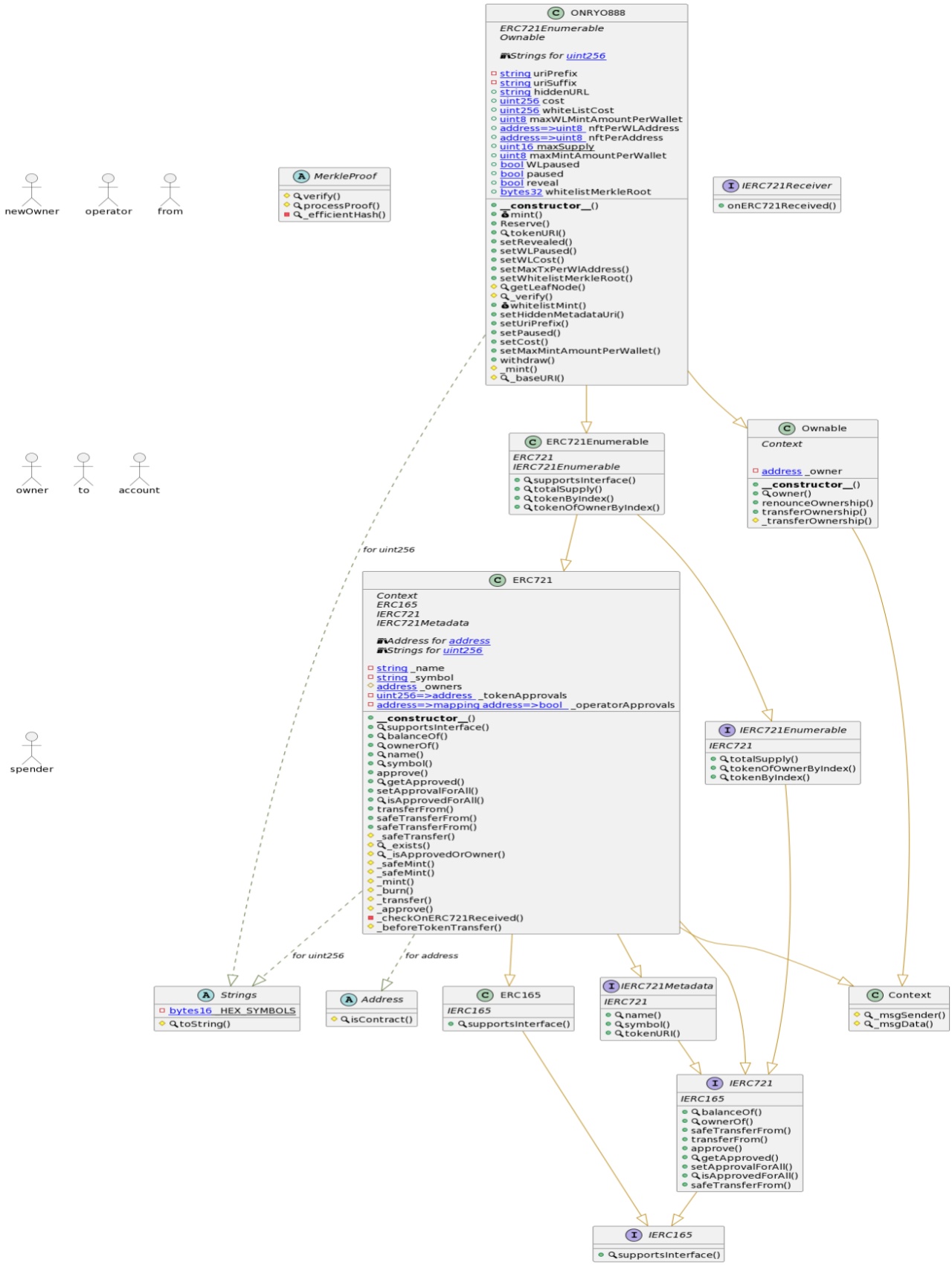
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC721Enumerable
supportsInterface
totalSupply
tokenByIndex
tokenOfOwnerByIndex
balanceOf
Address (lib)
isContract
IERC721 (iface)
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
getApproved
setApprovalForAll
isApprovedForAll
ERC165
supportsInterface
type
IERC165 (iface)
supportsInterface
IERC721Receiver (iface)
onERC721Received
IERC721Metadata (iface)
name
symbol
tokenURI
supportsInterface
IERC721Enumerable (iface)
totalSupply
tokenOfOwnerByIndex
tokenByIndex
supportsInterface
Ownable
onlyOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
_tokenId
toString
ONRYO888
<Constructor>
tokenURI
_baseURI
_verify
whitelistMint
getLeafNode
withdraw
payable
setRevealed
setWLPaused
setWLCost
setMaxTxPerWlAddress
setWhitelistMerkleRoot
Reserve
setHiddenMetadataUri
setUriPrefix
setPaused
setCost
setMaxMintAmountPerWallet
mint
_mint
MerkleProof (lib)
verify
processProof
_efficientHash
Context
_msgSender
_msgData
Strings (lib)
toString
ERC721
<Constructor>
supportsInterface
balanceOf
name
symbol
safeTransferFrom
_mint
_safeMint
transferFrom
_isApprovedOrOwner
setApprovalForAll
approve
_burn
_safeTransfer
_checkOnERC721Received
_transfer
getApproved
isApprovedForAll
_msgSender
ownerOf
_beforeTokenTransfer
_approve
_exists
to
isContract

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
58381669  =>  whitelistMint(uint8,bytes32[])
5a9a49c7  =>  verify(bytes32[],bytes32,bytes32)
62702a6b  =>  processProof(bytes32[],bytes32)
41ed615b  =>  _efficientHash(bytes32,bytes32)
6900a3ae  =>  toString(uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
150b7a02  =>  onERC721Received(address,address,uint256,bytes)
01ffc9a7  =>  supportsInterface(bytes4)
70a08231  =>  balanceOf(address)
6352211e  =>  ownerOf(uint256)
42842e0e  =>  safeTransferFrom(address,address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
095ea7b3  =>  approve(address,uint256)
081812fc  =>  getApproved(uint256)
a22cb465  =>  setApprovalForAll(address,bool)
e985e9c5  =>  isApprovedForAll(address,address)
b88d4fde  =>  safeTransferFrom(address,address,uint256,bytes)
18160ddd  =>  totalSupply()
2f745c59  =>  tokenOfOwnerByIndex(address,uint256)
4f6ccce7  =>  tokenByIndex(uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
c87b56dd  =>  tokenURI(uint256)
24b6b8c0  =>  _safeTransfer(address,address,uint256,bytes)
f8e76cc0  =>  _exists(uint256)
4cdc9549  =>  _isApprovedOrOwner(address,uint256)
b3e1c718  =>  _safeMint(address,uint256)
6a4f832b  =>  _safeMint(address,uint256,bytes)
4e6ec247  =>  _mint(address,uint256)
9b1f9e74  =>  _burn(uint256)
30e0789e  =>  _transfer(address,address,uint256)
7b7d7225  =>  _approve(address,uint256)
1fd01de1  =>  _checkOnERC721Received(address,address,uint256,bytes)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
6ecd2306  =>  mint(uint8)
169e490d  =>  Reserve(uint8,address)
3bd64968  =>  setRevealed()
093cfa63  =>  setWLPaused()
d1d19213  =>  setWLCost(uint256)
63937fe0  =>  setMaxTxPerWlAddress(uint8)
bd32fb66  =>  setWhitelistMerkleRoot(bytes32)
bbfb564a  =>  getLeafNode(address)
46f265fd  =>  _verify(bytes32,bytes32[])
4fdd43cb  =>  setHiddenMetadataUri(string)
7ec4a659  =>  setUriPrefix(string)
```

```
37a66d85  =>  setPaused()
44a0d68a  =>  setCost(uint256)
28b60d15  =>  setMaxMintAmountPerWallet(uint8)
3ccfd60b  =>  withdraw()
743976a0  =>  _baseURI()
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/ONRYO888.sol |
dda9326366bda1eff4fff6850fb0b93fe8462603 |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **MerkleProof** | Library |  |||
| └ | verify | Internal 🔒 |  | |
| └ | processProof | Internal 🔒 |  | |
| └ | _efficientHash | Private 🔐 |  | |
||||||
| **Strings** | Library |  |||
| └ | toString | Internal 🔒 |  | |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |  | |
| └ | _msgData | Internal 🔒 |  | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
| └ | owner | Public ❗ |  |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑 | |
||||||
| **IERC721Receiver** | Interface |  |||
| └ | onERC721Received | External ❗ | 🛑 |NO❗ |
||||||
| **IERC165** | Interface |  |||
| └ | supportsInterface | External ❗ |  |NO❗ |
||||||
| **ERC165** | Implementation | IERC165 |||
| └ | supportsInterface | Public ❗ |  |NO❗ |
||||||
| **IERC721** | Interface | IERC165 |||
| └ | balanceOf | External ❗ |  |NO❗ |
| └ | ownerOf | External ❗ |  |NO❗ |
| └ | safeTransferFrom | External ❗ | 🛑 |NO❗ |
| └ | transferFrom | External ❗ | 🛑 |NO❗ |
| └ | approve | External ❗ | 🛑 |NO❗ |
| └ | getApproved | External ❗ |  |NO❗ |
| └ | setApprovalForAll | External ❗ | 🛑 |NO❗ |
```


└	isApprovedForAll	External ❗		NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
IERC721Enumerable	Interface	IERC721		
└	totalSupply	External ❗		NO❗
└	tokenOfOwnerByIndex	External ❗		NO❗
└	tokenByIndex	External ❗		NO❗
IERC721Metadata	Interface	IERC721		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	tokenURI	External ❗		NO❗
Address	Library		
└	isContract	Internal 🔒		
ERC721	Implementation	Context, ERC165, IERC721, IERC721Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	supportsInterface	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	ownerOf	Public ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	getApproved	Public ❗		NO❗
└	setApprovalForAll	Public ❗	🛑	NO❗
└	isApprovedForAll	Public ❗		NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	safeTransferFrom	Public ❗	🛑	NO❗
└	safeTransferFrom	Public ❗	🛑	NO❗
└	_safeTransfer	Internal 🔒	🛑	
└	_exists	Internal 🔒		
└	_isApprovedOrOwner	Internal 🔒		
└	_safeMint	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_transfer	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_checkOnERC721Received	Private 🔐	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
ERC721Enumerable	Implementation	ERC721, IERC721Enumerable		
└	supportsInterface	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	tokenByIndex	Public ❗		NO❗
└	tokenOfOwnerByIndex	Public ❗		NO❗
ONRYO888	Implementation	ERC721Enumerable, Ownable		
└	<Constructor>	Public ❗	🛑	ERC721
└	mint	External ❗	💸	NO❗
└	Reserve	External ❗	🛑	onlyOwner
└	tokenURI	Public ❗		NO❗
└	setRevealed	External ❗	🛑	onlyOwner
└	setWLPaused	External ❗	🛑	onlyOwner

└	setWLCost	External ❗	🛑	onlyOwner
└	setMaxTxPerWlAddress	External ❗	🛑	onlyOwner
└	setWhitelistMerkleRoot	External ❗	🛑	onlyOwner
└	getLeafNode	Internal 🔒		
└	_verify	Internal 🔒		
└	whitelistMint	External ❗	💵	NO❗
└	setHiddenMetadataUri	External ❗	🛑	onlyOwner
└	setUriPrefix	External ❗	🛑	onlyOwner
└	setPaused	External ❗	🛑	onlyOwner
└	setCost	External ❗	🛑	onlyOwner
└	setMaxMintAmountPerWallet	External ❗	🛑	onlyOwner
└	withdraw	External ❗	🛑	onlyOwner
└	_mint	Internal 🔒	🛑	
└	_baseURI	Internal 🔒		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “Well-secured”.

✓ No volatile code.
✓ Not many high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of 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 the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below 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 Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico 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 Saferico 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, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, 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 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.