

Smart Contract Security Audit V1

Lazynaire Smart Contract Audit

26/7/2023



<https://saferico.com/>

business@saferico.com

https://t.me/SFI_ANN

—

Table of Contents

Table of Contents

Background

Project Information

NFT Information

Executive Summary

File and Function Level Report

File in Scope:

Issues Checking Status

Severity Definitions

Audit Findings

Automatic testing

Testing proves

Inheritance graph

Call graph

Unified Modeling Language (UML)

Functions signature

Automatic general report

Conclusion

Disclaimer

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:** 0x2097991b51527e5e14691114B66622EbCBB95B3a
- **Code:**

<https://github.com/Saferico/Smart-Contracts-for-Projects/blob/main/Lazynaire.sol>

- **Website:** <https://www.lazynaire.com/>

Executive Summary

According to our assessment, the customer`s solidity smart contract is **“WELL SECURED”**. The team has fixed the low-level 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, 2 low, 0 very low-level issues and 0 note in all solidity files of the contract

The files:

Lazynaire.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Lazynaire.sol	e956ece9adea61e5b896097a86d59a27720fbb7a	0x2097991b51527e5e14691114B66622EbCBB95B3a

- Contract: Lazynaire
- Inherit: ERC721A, ERC2981, Ownable, ERC721AQueryable, OperatorFilterer
- 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
royaltyInfo	✓	Read / public	Passed
supportsInterface	✓	Read / public	Passed
getCurrentPhase	✓	Read / public	Passed
balanceOf	✓	Read / public	Passed
Owner	✓	Read / public	Passed
calculateTotalMintPrice	✓	Read / public	Passed
isPaused	✓	Read / public	Passed
getApprovedForAll	✓	Read / public	Passed
maxSupply	✓	Read / public	Passed
getApproved	✓	Read / public	Passed

ownerOf	✓	Read / public	Passed
tokenURI	✓	Read / public	Passed
totalSupply	✓	Read / public	Passed
getMintable	✓	Read / public	Passed
getMintEligibilityAtCurrentPhase	✓	Read / public	Passed
getRevealedBool	✓	Read / public	Passed
getRole	✓	Read / public	Passed
getRoleFromProofs	✓	Read / public	Passed
getSupplyInfo	✓	Read / public	Passed
maxMintPerWallet	✓	Read / public	Passed
numberMintedPublicSales	✓	Read / public	Passed
operatorFilteringEnabled	✓	Read / public	Passed
tokenByIndex	✓	Read / public	Passed
tokenOfOwnerByIndex	✓	Read / public	Passed
walletOfOwner	✓	Read / public	Passed
transferFrom	✓	Write / payable	Passed
approve	✓	Write / public	Passed
safeTransferFrom	✓	Write / payable	Passed
safeTransferFrom	✓	Write / payable	Passed
setBaseExtension	✓	Write / public	Passed
withdraw	✓	Write / payable	Passed
devMint	✓	Write / public	Passed
transferOwnership	✓	Write / public	Passed
setApprovalForAll	✓	Write / public	Passed
isRevealed	✓	Write / public	Passed
mint	✓	Write / payable	Passed

renounceOwnership	✓	Write / public	Passed
setCurrentPhase	✓	Write / public	Passed
setDefaultRoyalty	✓	Write / public	Passed
setBaseURI	✓	Write / public	Passed
setMerkleRoot	✓	Write / public	Passed
setOperatorFilteringEnabled	✓	Write / public	Passed
setPauseContract	✓	Write / public	Passed
setPhaseConfig	✓	Write / public	Passed
setpreRevealURI	✓	Write / public	Passed
updateCollectionSize	✓	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 with Notes
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed with Notes
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 mint some NFT to investors, he has to check for the zero address to make, he didn't add the zero address. Otherwise, the mint for address function will act like the burn function, the same when he wants to change the Royalty address otherwise, he can't receive the royalty, and for withdraw all function too.

```
function devMint(address to_, uint256 amount_) external onlyOwner {
    // Check if the total supply does not exceed the collection size
    if (!(totalSupply() + amount_ <= _collectionSize)) {
        revert ExceedsCollectionSize(
            totalSupply(),
            amount_,
            _collectionSize);
    }
    _mint(to_, amount_);
}

function withdrawAll(address payable to_) external onlyOwner {
    to_.transfer(address(this).balance);
}

function setDefaultRoyalty(
    address receiver,
    uint96 feeNumerator
) public onlyOwner {
    _setDefaultRoyalty(receiver, feeNumerator);
}
```

Remediation

Use the require statement to check for zero addresses.

Status: **Closed**. Fixed in version 2.

#Pragma version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.21 instead of ^0.8.4). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version.

Status: **Closed**. Fixed in version 2.

Very Low:

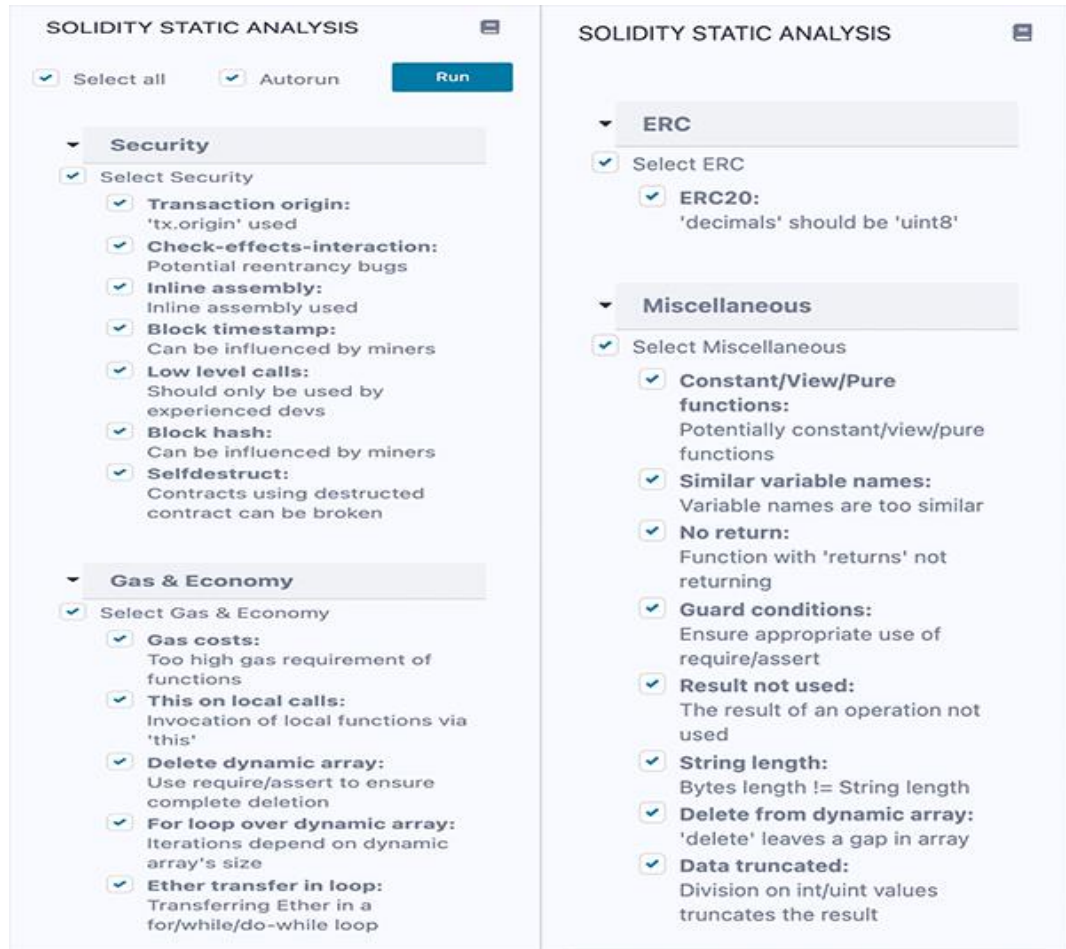
No Very Low severity vulnerabilities were found.

Notes:

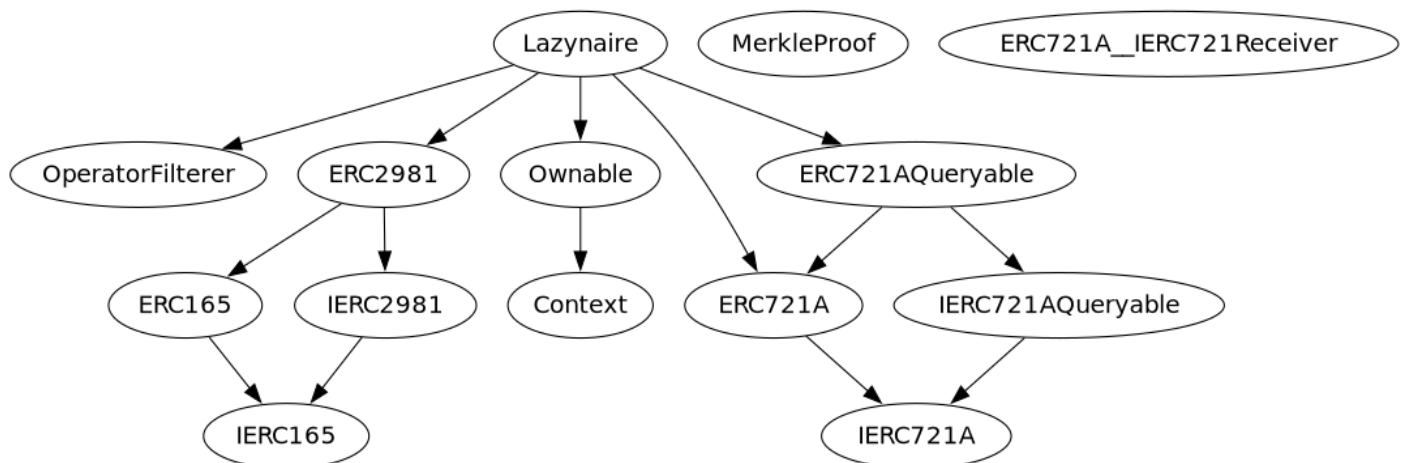
No Notes vulnerabilities were found.

Automatic Testing

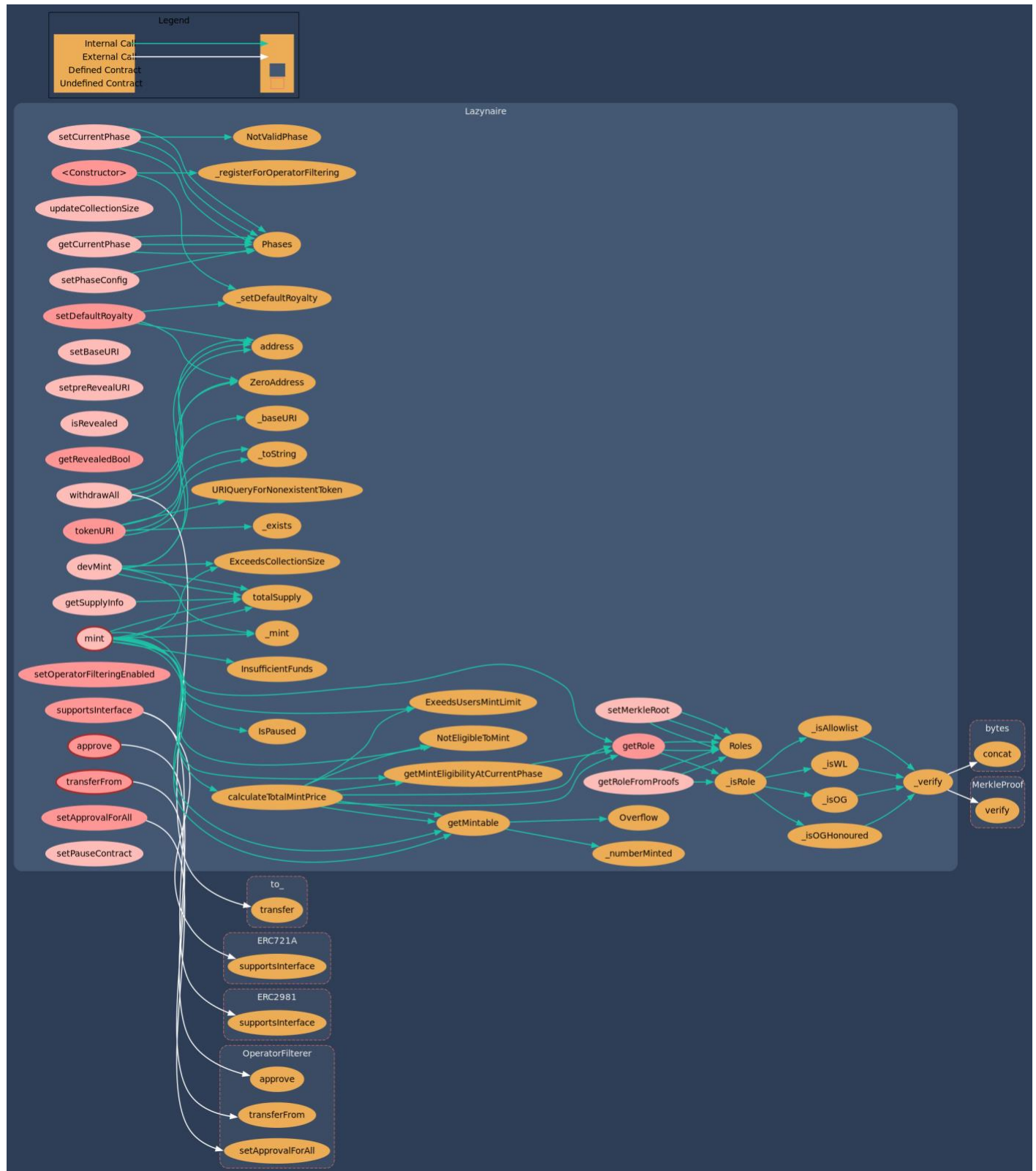
1- SOLIDITY STATIC ANALYSIS



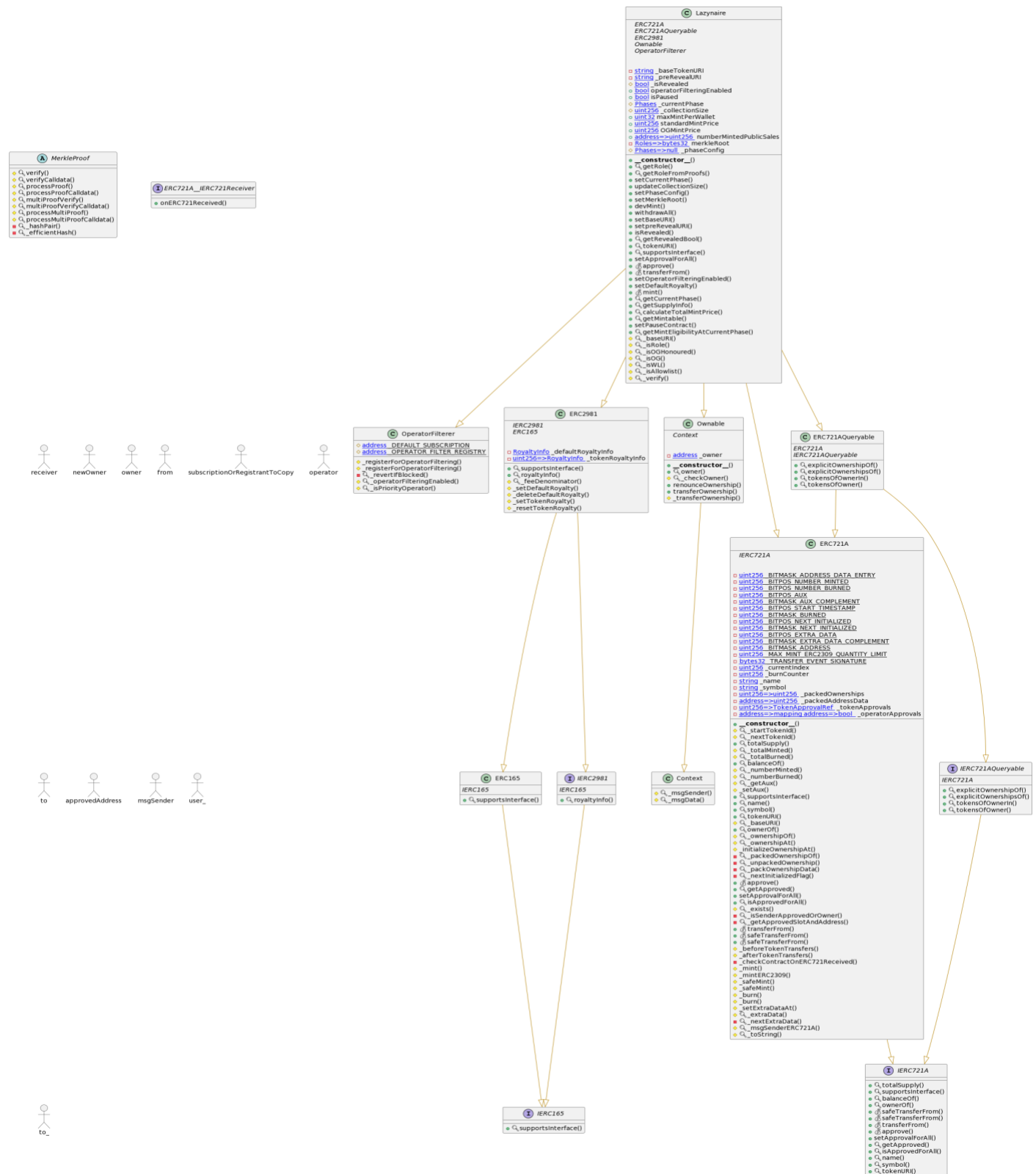
2- Inheritance graph



3- Call graph



Unified Modeling Language (UML)



Functions signature

Sighash		Function Signature
=====		
86333816	=>	multiProofVerify(bytes32[],bool[],bytes32,bytes32[])
91401607	=>	_verify(bytes32[],address,bytes32)
72dfb778	=>	_registerForOperatorFiltering()
ee43f83f	=>	_registerForOperatorFiltering(address,bool)
9771b304	=>	_revertIfBlocked(address)
b040adf6	=>	_operatorFilteringEnabled()
fb645b3b	=>	_isPriorityOperator(address)
01ffc9a7	=>	supportsInterface(bytes4)
2a55205a	=>	royaltyInfo(uint256,uint256)
bf8e572e	=>	_feeDenominator()
b1c1ab1b	=>	_setDefaultRoyalty(address,uint96)
36fdc63c	=>	_deleteDefaultRoyalty()
b552a471	=>	_setTokenRoyalty(uint256,address,uint96)
604ba39e	=>	_resetTokenRoyalty(uint256)
5a9a49c7	=>	verify(bytes32[],bytes32,bytes32)
7ffe9d8c	=>	verifyCalldata(bytes32[],bytes32,bytes32)
62702a6b	=>	processProof(bytes32[],bytes32)
53b0f85f	=>	processProofCalldata(bytes32[],bytes32)
862aac96	=>	multiProofVerifyCalldata(bytes32[],bool[],bytes32,bytes32[])
ea5d3eb6	=>	processMultiProof(bytes32[],bool[],bytes32[])
32712d22	=>	processMultiProofCalldata(bytes32[],bool[],bytes32[])
00a4cf28	=>	_hashPair(bytes32,bytes32)
41ed615b	=>	_efficientHash(bytes32,bytes32)
119df25f	=>	_msgSender()
8b49d47e	=>	_msgData()
8da5cb5b	=>	owner()
53a72975	=>	_checkOwner()
715018a6	=>	renounceOwnership()
f2fde38b	=>	transferOwnership(address)
d29d44ee	=>	_transferOwnership(address)
18160ddd	=>	totalSupply()
70a08231	=>	balanceOf(address)
6352211e	=>	ownerOf(uint256)
b88d4fde	=>	safeTransferFrom(address,address,uint256,bytes)
42842e0e	=>	safeTransferFrom(address,address,uint256)
23b872dd	=>	transferFrom(address,address,uint256)
095ea7b3	=>	approve(address,uint256)
a22cb465	=>	setApprovalForAll(address,bool)
081812fc	=>	getApproved(uint256)

```
e985e9c5 => isApprovedForAll (address, address)
06fdde03 => name ()
95d89b41 => symbol ()
c87b56dd => tokenURI (uint256)
c23dc68f => explicitOwnershipOf (uint256)
5bbb2177 => explicitOwnershipsOf (uint256[])
99a2557a => tokensOfOwnerIn (address, uint256, uint256)
8462151c => tokensOfOwner (address)
150b7a02 => onERC721Received (address, address, uint256, bytes)
98995f77 => _startTokenId ()
4a60f620 => _nextTokenId ()
736bf591 => _totalMinted ()
fd01bd4c => _totalBurned ()
4d388a98 => _numberMinted (address)
6ba1b8d0 => _numberBurned (address)
f4a540c5 => _getAux (address)
4ff8c452 => _setAux (address, uint64)
743976a0 => _baseURI ()
fb372cf2 => _ownershipOf (uint256)
cbaf28ce => _ownershipAt (uint256)
f2d31624 => _initializeOwnershipAt (uint256)
444996c1 => _packedOwnershipOf (uint256)
4fe3c13e => _unpackedOwnership (uint256)
bf460657 => _packOwnershipData (address, uint256)
e0e30f80 => _nextInitializedFlag (uint256)
f8e76cc0 => _exists (uint256)
848b8eac => _isSenderApprovedOrOwner (address, address, address)
f112a2af => _getApprovedSlotAndAddress (uint256)
ef435773 => _beforeTokenTransfers (address, address, uint256, uint256)
08c018f7 => _afterTokenTransfers (address, address, uint256, uint256)
d88343e2 => _checkContractOnERC721Received (address, address, uint256, bytes)
4e6ec247 => _mint (address, uint256)
4908d13b => _mintERC2309 (address, uint256)
6a4f832b => _safeMint (address, uint256, bytes)
b3e1c718 => _safeMint (address, uint256)
9b1f9e74 => _burn (uint256)
834a9477 => _burn (uint256, bool)
bd3cdd6d => _setExtraDataAt (uint256, uint24)
fc37bbd3 => _extraData (address, address, uint24)
5afe32e4 => _nextExtraData (address, address, uint256)
b60986df => _msgSenderERC721A ()
f832e238 => _toString (uint256)
4b1097d2 => getRole (address, bytes32[])
e2a26d73 => getRoleFromProofs (address, bytes32[][4])
7d3e1ee4 => setCurrentPhase (uint256)
fd6ce9e4 => updateCollectionSize (uint256)
```



```
e25aca32 => setPhaseConfig(uint256,uint32,uint32)
320d90a1 => setMerkleRoot(Roles,bytes32)
627804af => devMint(address,uint256)
fa09e630 => withdrawAll(address)
55f804b3 => setBaseURI(string)
57f9f9a9 => setpreRevealURI(string)
301f8cfc => isRevealed(bool)
3fecf819 => getRevealedBool()
b7c0b8e8 => setOperatorFilteringEnabled(bool)
04634d8d => setDefaultRoyalty(address,uint96)
ba41b0c6 => mint(uint256,bytes32[])
a3a40ea5 => getCurrentPhase()
20edeaf3 => getSupplyInfo()
f2992187 => calculateTotalMintPrice(address,uint256,bytes32[])
ca286de7 => getMintable(address)
e601af73 => setPauseContract(bool)
593df74a => getMintEligibilityAtCurrentPhase(address,bytes32[])
d8c16553 => _isRole(Roles,address,bytes32[])
362b41dc => _isOGHonoured(address,bytes32[])
61eda8fa => _isOG(address,bytes32[])
0691441a => _isWL(address,bytes32[])
d4c67da1 => _isAllowlist(address,bytes32[])
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Lazynaire.sol	e956ece9adea61e5b896097a86d59a27720fbb7a

Contract	Type	Bases
Contracts Description Table		
L	**Function Name**	**Visibility** **Mutability**
Modifiers		
OperatorFilterer	Implementation	
L _registerForOperatorFiltering	Internal	🔒 🔒
L _registerForOperatorFiltering	Internal	🔒 🔒
L _revertIfBlocked	Private	🔒
L _operatorFilteringEnabled	Internal	🔒
L _isPriorityOperator	Internal	🔒
IERC165	Interface	
L supportsInterface	External	! NO!
ERC165	Implementation	IERC165
L supportsInterface	Public	! NO!
IERC2981	Interface	IERC165
L royaltyInfo	External	! NO!
ERC2981	Implementation	IERC2981, ERC165
L supportsInterface	Public	! NO!
L royaltyInfo	Public	! NO!
L _feeDenominator	Internal	🔒
L _setDefaultRoyalty	Internal	🔒 🔒
L _deleteDefaultRoyalty	Internal	🔒 🔒
L _setTokenRoyalty	Internal	🔒 🔒
L _resetTokenRoyalty	Internal	🔒 🔒
MerkleProof	Library	
L verify	Internal	🔒
L verifyCalldata	Internal	🔒
L processProof	Internal	🔒
L processProofCalldata	Internal	🔒
L multiProofVerify	Internal	🔒

```

| L | multiProofVerifyCalldata | Internal 🔒 | | |
| L | processMultiProof | Internal 🔒 | | |
| L | processMultiProofCalldata | Internal 🔒 | | |
| L | _hashPair | Private 🔑 | | |
| L | _efficientHash | Private 🔑 | | |
| **Context** | Implementation | | |
| L | _msgSender | Internal 🔒 | | |
| L | _msgData | Internal 🔒 | | |
| **Ownable** | Implementation | Context | | |
| L | <Constructor> | Public ! | 🔒 | NO! |
| L | owner | Public ! | | NO! |
| L | _checkOwner | Internal 🔒 | | |
| L | renounceOwnership | Public ! | 🔒 | onlyOwner |
| L | transferOwnership | Public ! | 🔒 | onlyOwner |
| L | _transferOwnership | Internal 🔒 | 🔒 | |
| **IERC721A** | Interface | | |
| L | totalSupply | External ! | | NO! |
| L | supportsInterface | External ! | | NO! |
| L | balanceOf | External ! | | NO! |
| L | ownerOf | External ! | | NO! |
| L | safeTransferFrom | External ! | 🔒 | NO! |
| L | safeTransferFrom | External ! | 🔒 | NO! |
| L | transferFrom | External ! | 🔒 | NO! |
| L | approve | External ! | 🔒 | NO! |
| L | setApprovalForAll | External ! | 🔒 | NO! |
| L | getApproved | External ! | | NO! |
| L | isApprovedForAll | External ! | | NO! |
| L | name | External ! | | NO! |
| L | symbol | External ! | | NO! |
| L | tokenURI | External ! | | NO! |
| **IERC721AQueryable** | Interface | IERC721A | | |
| L | explicitOwnershipOf | External ! | | NO! |
| L | explicitOwnershipsOf | External ! | | NO! |
| L | tokensOfOwnerIn | External ! | | NO! |
| L | tokensOfOwner | External ! | | NO! |
| **ERC721A__IERC721Receiver** | Interface | | | |
| L | onERC721Received | External ! | 🔒 | NO! |
| **ERC721A** | Implementation | IERC721A | | |
| L | <Constructor> | Public ! | 🔒 | NO! |
| L | _startTokenId | Internal 🔒 | | |
| L | _nextTokenId | Internal 🔒 | | |
| L | totalSupply | Public ! | | NO! |
| L | _totalMinted | Internal 🔒 | | |
| L | _totalBurned | Internal 🔒 | | |
| L | balanceOf | Public ! | | NO! |





```

```

| L | _numberMinted | Internal 🔒 | | |
| L | _numberBurned | Internal 🔒 | | |
| L | _getAux | Internal 🔒 | | |
| L | _setAux | Internal 🔒 | 🔒 | |
| L | supportsInterface | Public ! | | NO! |
| L | name | Public ! | | NO! |
| L | symbol | Public ! | | NO! |
| L | tokenURI | Public ! | | NO! |
| L | _baseURI | Internal 🔒 | | |
| L | ownerOf | Public ! | | NO! |
| L | _ownershipOf | Internal 🔒 | | |
| L | _ownershipAt | Internal 🔒 | | |
| L | _initializeOwnershipAt | Internal 🔒 | 🔒 | |
| L | _packedOwnershipOf | Private 🔒 | | |
| L | _unpackedOwnership | Private 🔒 | | |
| L | _packOwnershipData | Private 🔒 | | |
| L | _nextInitializedFlag | Private 🔒 | | |
| L | approve | Public ! | 🗑️ NO! |
| L | getApproved | Public ! | | NO! |
| L | setApprovalForAll | Public ! | 🔒 NO! |
| L | isApprovedForAll | Public ! | | NO! |
| L | _exists | Internal 🔒 | | |
| L | _isSenderApprovedOrOwner | Private 🔒 | | |
| L | _getApprovedSlotAndAddress | Private 🔒 | | |
| L | transferFrom | Public ! | 🗑️ NO! |
| L | safeTransferFrom | Public ! | 🗑️ NO! |
| L | safeTransferFrom | Public ! | 🗑️ NO! |
| L | _beforeTokenTransfers | Internal 🔒 | 🔒 | |
| L | _afterTokenTransfers | Internal 🔒 | 🔒 | |
| L | _checkContractOnERC721Received | Private 🔒 | 🔒 | |
| L | _mint | Internal 🔒 | 🔒 | |
| L | _mintERC2309 | Internal 🔒 | 🔒 | |
| L | _safeMint | Internal 🔒 | 🔒 | |
| L | _safeMint | Internal 🔒 | 🔒 | |
| L | _burn | Internal 🔒 | 🔒 | |
| L | _burn | Internal 🔒 | 🔒 | |
| L | _setExtraDataAt | Internal 🔒 | 🔒 | |
| L | _extraData | Internal 🔒 | | |
| L | _nextExtraData | Private 🔒 | | |
| L | _msgSenderERC721A | Internal 🔒 | | |
| L | _toString | Internal 🔒 | | |
| **ERC721AQueryable** | Implementation | ERC721A, IERC721AQueryable |||
| L | explicitOwnershipOf | Public ! | | NO! |
| L | explicitOwnershipsOf | External ! | | NO! |
| L | tokensOfOwnerIn | External ! | | NO! |



```

```

| L | tokensOfOwner | External ! | |NO! |
| **Lazynaire** | Implementation | ERC721A, ERC721AQueryable, ERC2981, Ownable,
OperatorFilterer |||
| L | <Constructor> | Public ! |  | ERC721A |
| L | getRole | Public ! | |NO! |
| L | getRoleFromProofs | External ! | |NO! |
| L | setCurrentPhase | External ! |  | onlyOwner |
| L | updateCollectionSize | External ! |  | onlyOwner |
| L | setPhaseConfig | External ! |  | onlyOwner |
| L | setMerkleRoot | External ! |  | onlyOwner |
| L | devMint | External ! |  | onlyOwner |
| L | withdrawAll | External ! |  | onlyOwner |
| L | setBaseURI | External ! |  | onlyOwner |
| L | setpreRevealURI | External ! |  | onlyOwner |
| L | isRevealed | External ! |  | onlyOwner |
| L | getRevealedBool | Public ! | |NO! |
| L | tokenURI | Public ! | |NO! |
| L | supportsInterface | Public ! | |NO! |
| L | setApprovalForAll | Public ! |  | onlyAllowedOperatorApproval |
| L | approve | Public ! |  | onlyAllowedOperatorApproval |
| L | transferFrom | Public ! |  | onlyAllowedOperator |
| L | setOperatorFilteringEnabled | Public ! |  | onlyOwner |
| L | setDefaultRoyalty | Public ! |  | onlyOwner |
| L | mint | External ! |  |NO! |
| L | getCurrentPhase | External ! | |NO! |
| L | getSupplyInfo | External ! | |NO! |
| L | calculateTotalMintPrice | Public ! | |NO! |
| L | getMintable | Public ! | |NO! |
| L | setPauseContract | External ! |  | onlyOwner |
| L | getMintEligibilityAtCurrentPhase | Public ! | |NO! |
| L | _baseURI | Internal  | | |
| L | _isRole | Internal  | | |
| L | _isOGHonoured | Internal  | | |
| L | _isOG | Internal  | | |
| L | _isWL | Internal  | | |
| L | _isAllowlist | Internal  | | |
| L | _verify | 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.
- ✓ No high severity issues were found.
- ✓ Low (or very low) level issues have been fixed.

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.