Smart Contract
Security Audit
V1

Joker Charlie Genesis NFT Smart Contract

26/4/2022

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**: 0x4893aeC8981A555A420629E77dF3fBe9dfa673E6
- **Code:**

https://rinkeby.etherscan.io/address/0xaCb8D0e5863863f63954A50B91E9675aE44C51D5#code

NFT Information

- Name: JCCGENESIS
- MAX Supply: 555
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum)

Joker Charlie Genesis NFT contract on ETH test net to test every function by the auditor.

https://rinkeby.etherscan.io/address/0x4893aec8981a555a420629e77df3fbe9dfa673e6

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well-Secured**. Because the team fix all high and 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, 1 high, 0 medium, 3 low, 0 very low-level issues and 2 notes in all solidity files of the contract

The files:

JCCGENESIS.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
JCCGENESIS.sol	**2da34ab8406e88eaa4547c22cce9f6e34ce767c277940a0ec2c9e89916f63562**	0x4893aeC8981A555A420629E77dF3fBe9dfa673E6

- Contract: JCCGenesis
- Inherit: ERC721S,PublicPrivateVoucherMinter,ERC2981,Pausable
- 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**
_proofLength	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
baseExtension	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
beneficiary	✓	Read / public	**Passed**
tokenByIndex	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
isRevealed	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

ownerOf	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
getAmountUsed	✓	Read / public	**Passed**
lockerContract	✓	Read / public	**Passed**
metadataContract	✓	Read / public	**Passed**
numberMinted	✓	Read / public	**Passed**
notRevealedURI	✓	Read / public	**Passed**
minterConfig	✓	Read / public	**Passed**
minterStatus	✓	Read / public	**Passed**
royaltyInfo	✓	Read / public	**Passed**
ownerBy	✓	Read / public	**Passed**
totalPrivateSold	✓	Read / public	**Passed**
totalPublicSold	✓	Read / public	**Passed**
totalSold	✓	Read / public	**Passed**
totalVoucherClaimed	✓	Read / public	**Passed**
totalVoucherIssued	✓	Read / public	**Passed**
baseTokenURI	✓	Read / public	**Passed**
tokenOfOwnerByIndex	✓	Read / public	**Passed**
maxSupply	✓	Read / public	**Passed**
donate	✓	Write / payable	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
setBaseTokenURI	✓	Write / public	**Passed**
setnotRevealedUri	✓	Write / public	**Passed**
privateMint	✓	Write / payable	**Passed**

publicMint	✓	Write / payable	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
pause	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
setBaseURI	✓	Write / public	**Passed**
unpause	✓	Write / public	**Passed**
setBeneficiary	✓	Write / public	**Passed**
setLockerContract	✓	Write / public	**Passed**
setMaxTotalSupply	✓	Write / public	**Passed**
setMetadataContract	✓	Write / public	**Passed**
setMinterConfig	✓	Write / public	**Passed**
setNotRevealedURI	✓	Write / public	**Passed**
voucherMint	✓	Write / payable	**Passed**
withdraw	✓	Write / public	**Passed**
transferERC20	✓	Write / public	**Passed**
setOperator	✓	Write / public	**Passed**
setPrivateMintDetail	✓	Write / public	**Passed**
setPublicMintDetail	✓	Write / public	**Passed**
setRoyaltyInfo	✓	Write / public	**Passed**
setVoucherDetail	✓	Write / public	**Passed**
transferERC20	✓	Write / public	**Passed**
toggleVoucherMintActive	✓	Write / public	**Passed**
togglePrivateMintActive	✓	Write / public	**Passed**
togglePublicMintActive	✓	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 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:

#Contract code size exceeds 24576 bytes

Description
Contract implementation is too large in size to be deployed on main net. Ethereum with its spurious dragon release limited the size of the contracts deployable on main net to 24576 bytes.
The size of the contract JCCGenesis.sol goes way above this value.
You can read more here:
https://github.com/ethereum/EIPs/issues/170

Remediation
Define and use libraries for pure and view functions e.g. We can create a library which contains all the mathematical operations.

Status: Closed. Fixed in version 2.

Medium:

No Medium severity vulnerabilities were found

Low:

#Multiple pragma statements

Line	Pragma
8	pragma solidity ^0.8.0;
93	pragma solidity ^0.8.0;
162	pragma solidity ^0.8.9;
200	pragma solidity ^0.8.1;
425	pragma solidity ^0.8.0;
455	pragma solidity ^0.8.0;
482	pragma solidity ^0.8.0;
560	pragma solidity ^0.8.0;
653	pragma solidity ^0.8.0;
681	pragma solidity ^0.8.9;

948	pragma solidity ^0.8.0;
1093	pragma solidity ^0.8.0;
1124	pragma solidity ^0.8.0;
1153	pragma solidity ^0.8.0;
1161	pragma solidity ^0.8.0;
1188	pragma solidity ^0.8.0;
1219	pragma solidity ^0.8.0;
1751	pragma solidity ^0.8.0;
1869	pragma solidity >=0.8.10 <0.9.0

Description
There are multiple pragma statements in the code. Only the compiler version 0.8.13 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

#Missing zero address validation

Description
When the owner adds or changes Royalty receiver, operator, and beneficiary addresses it has to check for the zero address to make, he didn't send funds or fees for the burn address. Otherwise, the funds and fees will transfer to the burn address.

```
constructor (address payable beneficiary, address payable royaltyReceiver)
        ERC721S("Joker Charlie Genesis", "JCCGENESIS")
        PublicPrivateVoucherMinter(
            PublicPrivateVoucherMinter.MinterConfig({
                isPublicMintActive : false,       // can call publicMint only when
isPublicMintActive is true
                isPrivateMintActive: false,      // can call privateMint only when
isPrivateMintActive is true
                isVoucherMintActive: false,      // can call voucherMint only when
isVoucherMintActive is true
                publicMintPrice: 0.555 ether,
                privateMintPrice: 0.555 ether,      //      = 0.01 ether;      //
price for privateMint
                maxPublicMintAmount:  2,    //   = 2;  // maximum amount per
publicMint transaction
                maxPrivateMintAmount: 1,  //  = 1;  // maximum amount per
privateMint transaction
                maxTotalSupply: 555,        //        = 555; // maximum supply for
current public round
                maxPrivateMintSupply: 300
            }),
            beneficiary
```

```
        )
    {
        // setting initial royalty fee
        _setDefaultRoyalty(royaltyReceiver, 1000);
    }

    function setRoyaltyInfo(address receiver, uint96 feeBasisPoints)
        external
        onlyOwner
    {        _setDefaultRoyalty(receiver, feeBasisPoints);
    }
function setOperator(address operator) public onlyOwner {
        _operator = operator;
    }
function setBeneficiary(address payable _beneficiary) public onlyOwner {
        beneficiary = _beneficiary;
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version2.

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

Description
The owner can pause and un Pause mint.
The owner can change the price in Presale, public, and vouchers mint stage.
The owner can open / close Presale, public, and vouchers mint stage.
The owner can change Royalty Fees.

```
function setRoyaltyInfo(address receiver, uint96 feeBasisPoints)
        external
        onlyOwner
    {
        require(receiver != address(0), "Not the zero address");
        _setDefaultRoyalty(receiver, feeBasisPoints);
    }function pause() external onlyOwner {
        _pause();
    }

    function unpause() external onlyOwner {
        _unpause();
    }
 function setMaxTotalSupply(uint256 supply) external onlyOwnerAndOperator {
        minterConfig.maxTotalSupply = supply;
    }

    // set parameter for public mint
    function setPublicMintDetail(uint256 price, uint256 amount) external
onlyOwnerAndOperator {
        require(!minterConfig.isPublicMintActive, "Public mint is active");
        minterConfig.publicMintPrice = price;
```

```
        minterConfig.maxPublicMintAmount = amount;
    }

    // set parameter for private mint
    function setPrivateMintDetail(uint256 price, uint256 amount, uint256 supply)
external onlyOwnerAndOperator {
        require(!minterConfig.isPrivateMintActive, "Private mint is active");
        minterConfig.privateMintPrice = price;
        minterConfig.maxPrivateMintAmount = amount;
        minterConfig.maxPrivateMintSupply = supply;
    }

    // set parameter for voucher/private mint
    function setVoucherDetail(bytes32 merkleRoot, uint256 proofLength, uint256
voucherAmount) external onlyOwnerAndOperator {
        _merkleRoot = merkleRoot;
        _proofLength = proofLength;
        totalVoucherIssued = voucherAmount;
    }
```

Remediation

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

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

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Missing SPDX-License-Identifier:

Description

Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information .

Remediation

Add License Identifier

// SPDX-License-Identifier: UNLICENSE

#Naming Conventions

Description

The contract follows a consistent naming convention where we are private variables with leading"_" and public variables without it. But we have missed to comply to the condition for certain variable names “_proofLength” which is public.

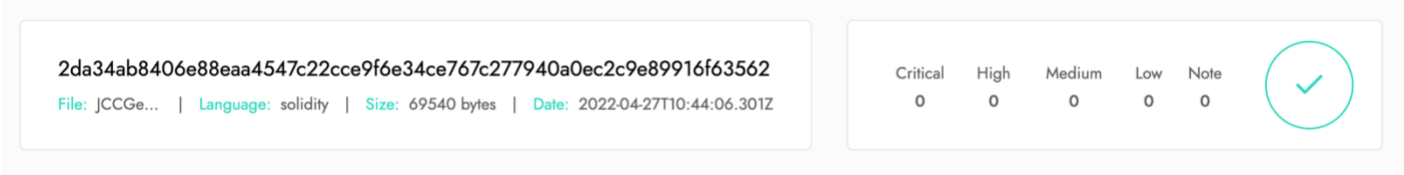
Remediation

Remove "_" from external variable names and add it to private variable names.

Status: Acknowledged

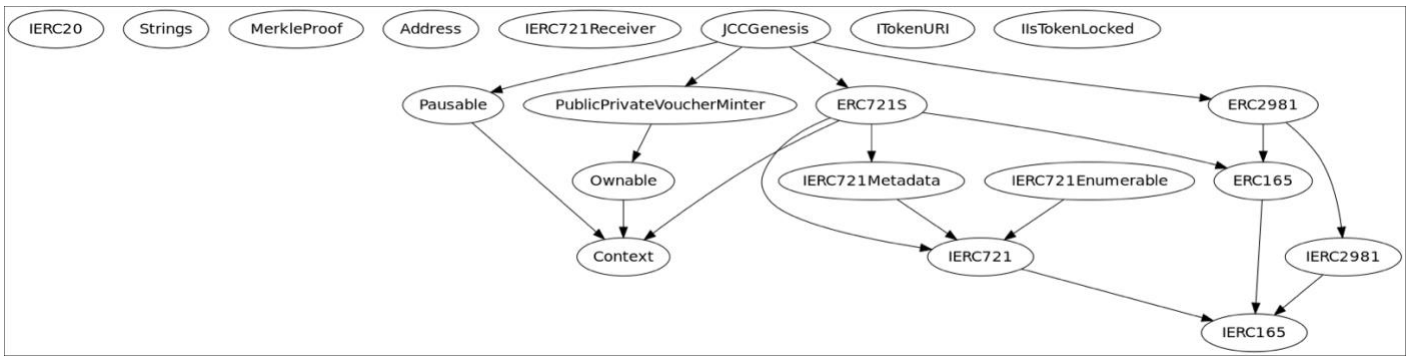
Automatic Testing

1- Check for security

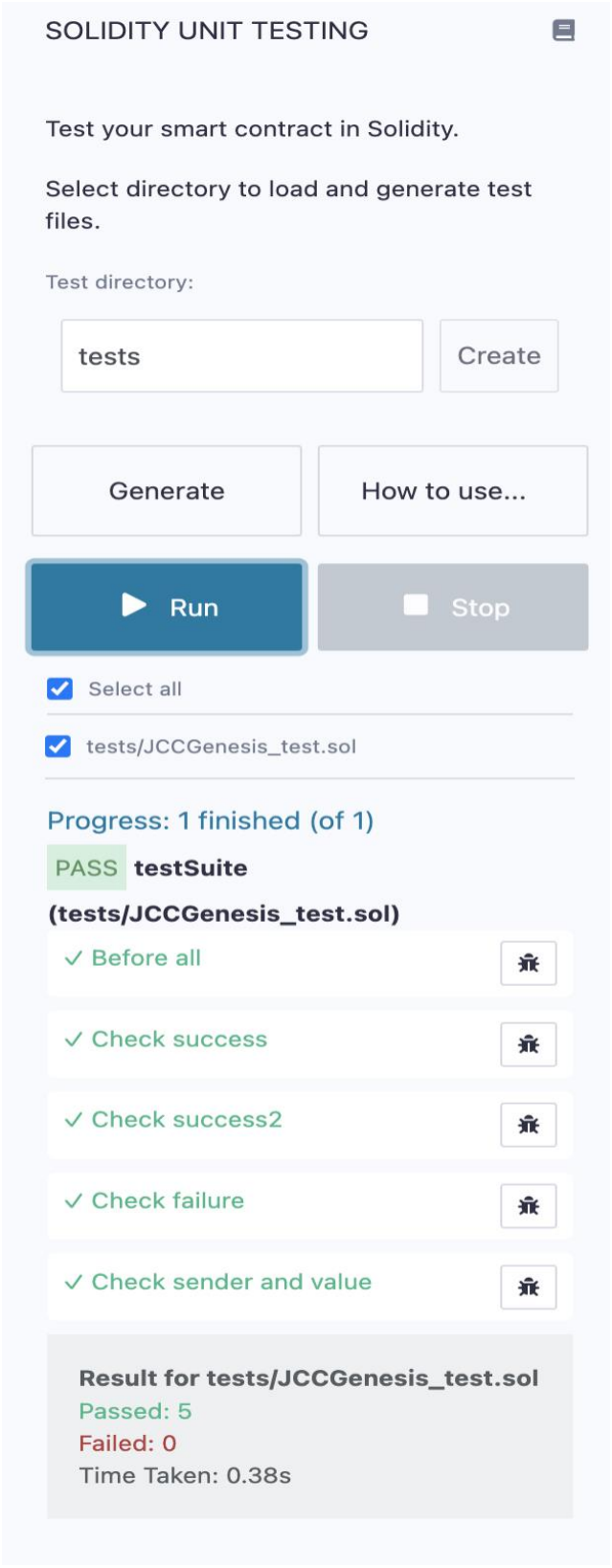


2- SOLIDITY STATIC ANALYSIS

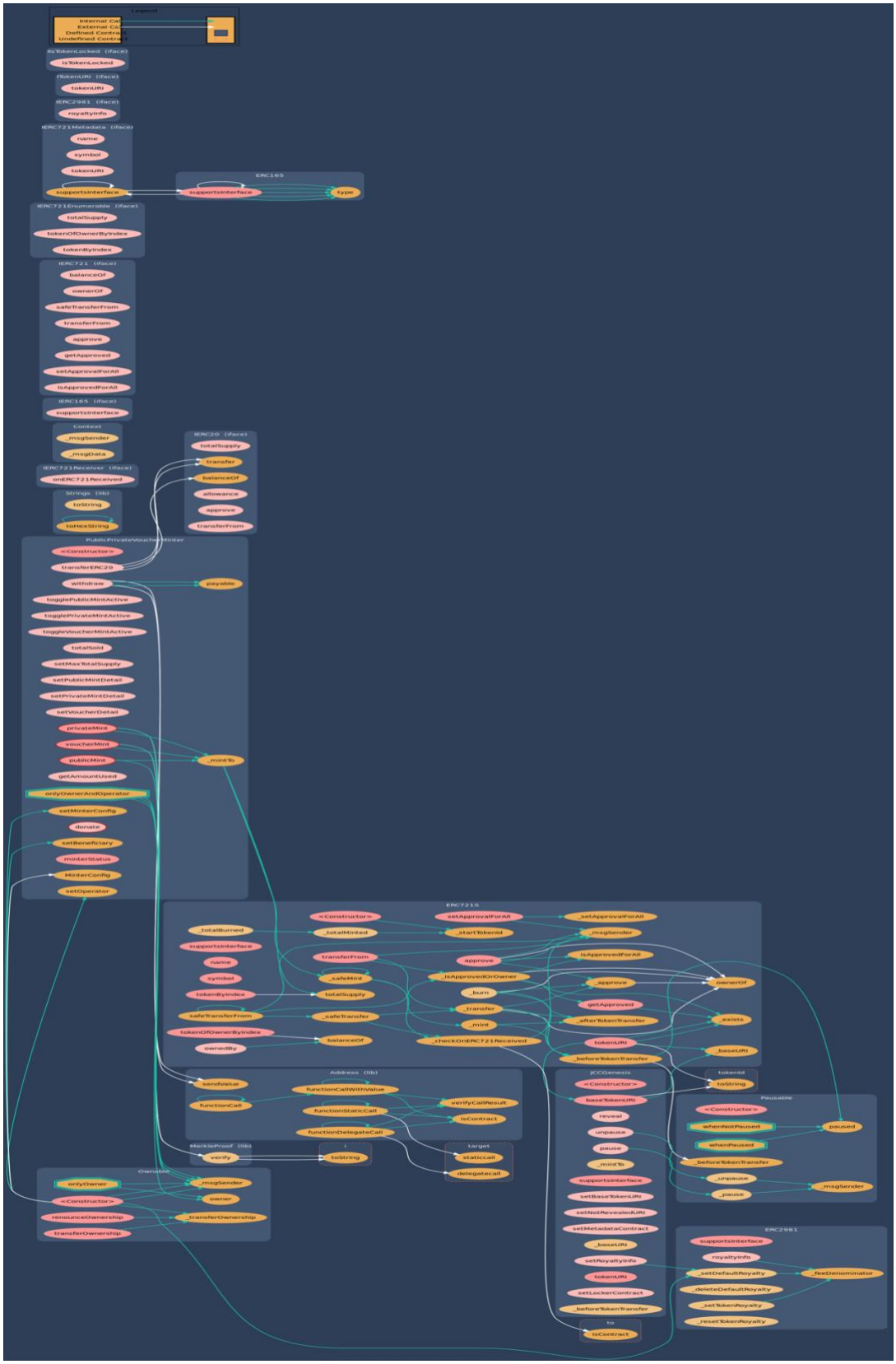
3- Inheritance graph



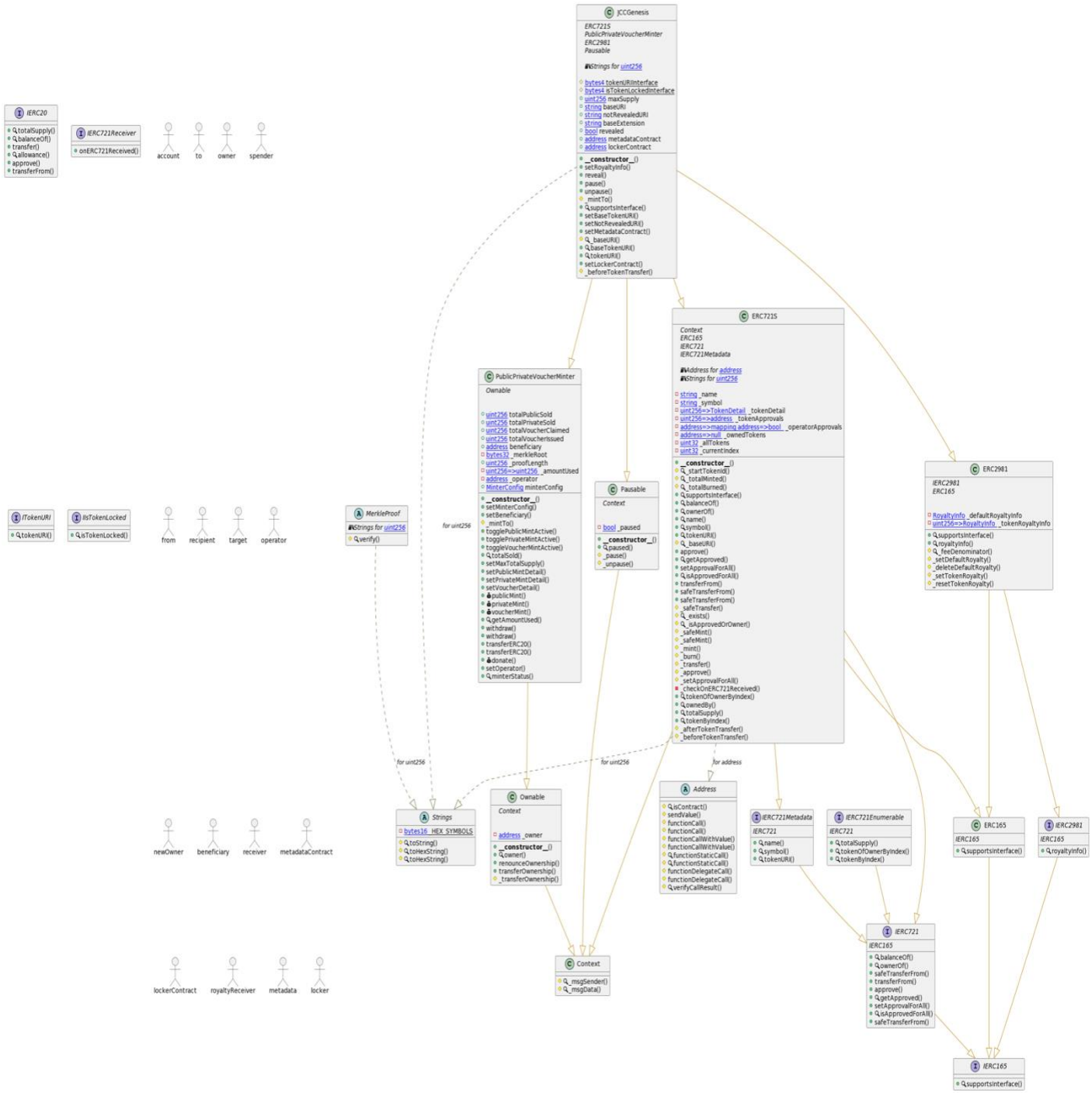
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IIsTokenLocked (iface)
isTokensLocked
ITokenURI (iface)
tokenURI
IERC2981 (iface)
royaltyInfo
IERC721Metadata (iface)
name
symbol
tokenURI
supportsInterface
ERC165
supportsInterface
type
IERC721Enumerable (iface)
totalSupply
tokenOfOwnerByIndex
tokenByIndex
IERC721 (iface)
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
getApproved
setApprovalForAll
isApprovedForAll
IERC165 (iface)
supportsInterface
Context
_msgSender
_msgData
IERC721Receiver (iface)
onERC721Received
Strings (lib)
toString
toHexString
IERC20 (iface)
totalSupply
transfer
balanceOf
allowance
approve
transferFrom
PublicPrivateVoucherMinter
<Constructor>
transferERC20
withdraw
payable
togglePublicMintActive
togglePrivateMintActive
toggleVoucherMintActive
totalSold
setMaxTotalSupply
setPublicMintDetail
setPrivateMintDetail
setVoucherDetail
privateMint
voucherMint
publicMint
getAmountUsed
_mintTo
onlyOwnerAndOperator
setMinterConfig
donate
setBeneficiary
minterStatus
MinterConfig
setOperator
ERC7215
<Constructor>
_totalBurned
_totalMinted
supportsInterface
name
symbol
tokenByIndex
safeTransferFrom
tokenOfOwnerByIndex
ownedBy
_startTokenId
transferFrom
_safeMint
totalSupply
_safeTransfer
balanceOf
setApprovalForAll
approve
_isApprovedOrOwner
_burn
_transfer
_mint
_checkOnERC721Received
_setApprovalForAll
_msgSender
isApprovedForAll
_approve
ownerOf
getApproved
_afterTokenTransfer
_exists
tokenURI
_beforeTokenTransfer
_baseURI
Address (lib)
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
verifyCallResult
isContract
MerkleProof (lib)
verify
toString
target
staticcall
delegatecall
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
_msgSender
owner
_transferOwnership
jCCGenesis
<Constructor>
baseTokenURI
reveal
unpause
pause
_mintTo
supportsInterface
setBaseTokenURI
setNotRevealedURI
setMetadataContract
_baseURI
setRoyaltyInfo
tokenURI
setLockerContract
_beforeTokenTransfer
to
isContract
tokenId
toString
Pausable
<Constructor>
whenNotPaused
paused
whenPaused
_beforeTokenTransfer
_unpause
_pause
_msgSender
ERC2981
supportsInterface
royaltyInfo
_setDefaultRoyalty
_feeDenominator
_deleteDefaultRoyalty
_setTokenRoyalty
_resetTokenRoyalty

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
76527485  =>  togglePublicMintActive()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
6900a3ae  =>  toString(uint256)
8fba8d5c  =>  toHexString(uint256)
63e1cbea  =>  toHexString(uint256,uint256)
5a9a49c7  =>  verify(bytes32[],bytes32,bytes32)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
946b5793  =>  verifyCallResult(bool,bytes,string)
150b7a02  =>  onERC721Received(address,address,uint256,bytes)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
5c975abb  =>  paused()
320b2ad9  =>  _pause()
fc8234cb  =>  _unpause()
01ffc9a7  =>  supportsInterface(bytes4)
affce55e  =>  setMinterConfig(MinterConfig)
1c31f710  =>  setBeneficiary(address)
d0820d87  =>  _mintTo(address,uint256)
fb5e28f3  =>  togglePrivateMintActive()
b2ad32e5  =>  toggleVoucherMintActive()
9106d7ba  =>  totalSold()
3f3e4c11  =>  setMaxTotalSupply(uint256)
a2a71c35  =>  setPublicMintDetail(uint256,uint256)
0475a470  =>  setPrivateMintDetail(uint256,uint256,uint256)
4df30e97  =>  setVoucherDetail(bytes32,uint256,uint256)
2db11544  =>  publicMint(uint256)
3f04fba4  =>  privateMint(uint256,uint256,uint256,bytes32[])
14bb6daf  =>  voucherMint(uint256,uint256,uint256,uint256,bytes32[])
8e909e0b  =>  getAmountUsed(uint256)
3ccfd60b  =>  withdraw()
2e1a7d4d  =>  withdraw(uint256)
93ca2d7d  =>  transferERC20(IERC20)
f7673645  =>  transferERC20(IERC20,uint256)
```

```
ed88c68e  =>  donate()
b3ab15fb  =>  setOperator(address)
2ecd28ab  =>  minterStatus()
b8abff04  =>  createMinter(address)
6352211e  =>  ownerOf(uint256)
42842e0e  =>  safeTransferFrom(address,address,uint256)
081812fc  =>  getApproved(uint256)
a22cb465  =>  setApprovalForAll(address,bool)
e985e9c5  =>  isApprovedForAll(address,address)
b88d4fde  =>  safeTransferFrom(address,address,uint256,bytes)
2f745c59  =>  tokenOfOwnerByIndex(address,uint256)
4f6ccce7  =>  tokenByIndex(uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
c87b56dd  =>  tokenURI(uint256)
2a55205a  =>  royaltyInfo(uint256,uint256)
98995f77  =>  _startTokenId()
736bf591  =>  _totalMinted()
fd01bd4c  =>  _totalBurned()
743976a0  =>  _baseURI()
24b6b8c0  =>  _safeTransfer(address,address,uint256,bytes)
f8e76cc0  =>  _exists(uint256)
4cdc9549  =>  _isApprovedOrOwner(address,uint256)
a031c832  =>  _safeMint(address)
d79573e0  =>  _safeMint(address,bytes)
09aa7b67  =>  _mint(address)
9b1f9e74  =>  _burn(uint256)
30e0789e  =>  _transfer(address,address,uint256)
7b7d7225  =>  _approve(address,uint256)
8c4e3f32  =>  _setApprovalForAll(address,address,bool)
1fd01de1  =>  _checkOnERC721Received(address,address,uint256,bytes)
b8377644  =>  ownedBy(address)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
bf8e572e  =>  _feeDenominator()
b1c1ab1b  =>  _setDefaultRoyalty(address,uint96)
36fdc63c  =>  _deleteDefaultRoyalty()
b552a471  =>  _setTokenRoyalty(uint256,address,uint96)
604ba39e  =>  _resetTokenRoyalty(uint256)
276a28a3  =>  isTokenLocked(uint256)
02fa7c47  =>  setRoyaltyInfo(address,uint96)
a475b5dd  =>  reveal()
8456cb59  =>  pause()
3f4ba83a  =>  unpause()
30176e13  =>  setBaseTokenURI(string)
f2c4ce1e  =>  setNotRevealedURI(string)
e5187f43  =>  setMetadataContract(address)
a689a914  =>  baseTokenURI(uint256)
b19b6f79  =>  setLockerContract(address)
```

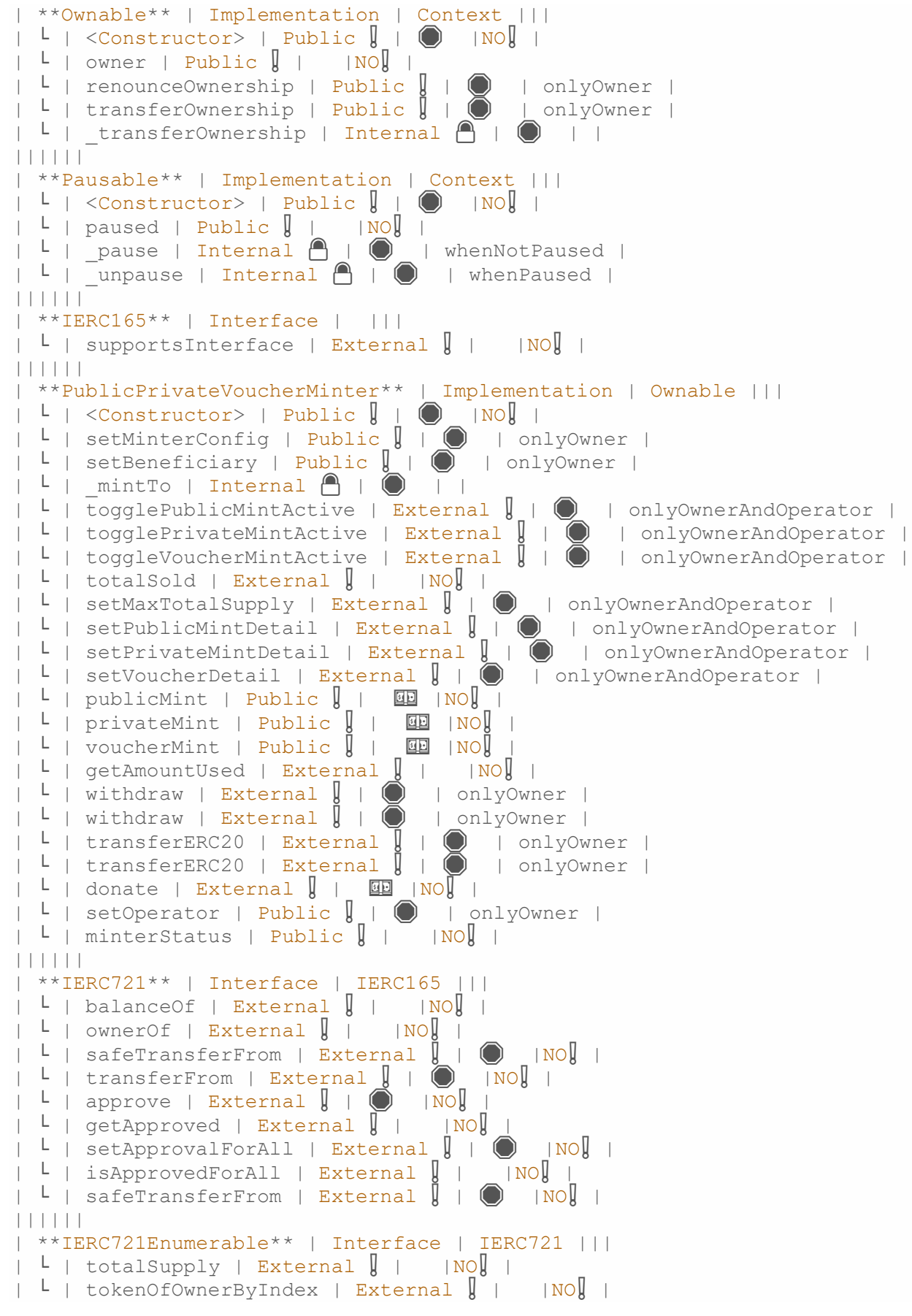
Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/JCCGenesis.sol	879c95b1563ab95b540ee5a9c5a46edb50e0d500

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Strings	Library			
└	toString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
MerkleProof	Library			
└	verify	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	verifyCallResult	Internal 🔒		
IERC721Receiver	Interface			
└	onERC721Received	External ❗	🛑	NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		


Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
Pausable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	paused	Public ❗		NO❗
└	_pause	Internal 🔒	🛑	whenNotPaused
└	_unpause	Internal 🔒	🛑	whenPaused
IERC165	Interface			
└	supportsInterface	External ❗		NO❗
PublicPrivateVoucherMinter	Implementation	Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	setMinterConfig	Public ❗	🛑	onlyOwner
└	setBeneficiary	Public ❗	🛑	onlyOwner
└	_mintTo	Internal 🔒	🛑	
└	togglePublicMintActive	External ❗	🛑	onlyOwnerAndOperator
└	togglePrivateMintActive	External ❗	🛑	onlyOwnerAndOperator
└	toggleVoucherMintActive	External ❗	🛑	onlyOwnerAndOperator
└	totalSold	External ❗		NO❗
└	setMaxTotalSupply	External ❗	🛑	onlyOwnerAndOperator
└	setPublicMintDetail	External ❗	🛑	onlyOwnerAndOperator
└	setPrivateMintDetail	External ❗	🛑	onlyOwnerAndOperator
└	setVoucherDetail	External ❗	🛑	onlyOwnerAndOperator
└	publicMint	Public ❗	💸	NO❗
└	privateMint	Public ❗	💸	NO❗
└	voucherMint	Public ❗	💸	NO❗
└	getAmountUsed	External ❗		NO❗
└	withdraw	External ❗	🛑	onlyOwner
└	withdraw	External ❗	🛑	onlyOwner
└	transferERC20	External ❗	🛑	onlyOwner
└	transferERC20	External ❗	🛑	onlyOwner
└	donate	External ❗	💸	NO❗
└	setOperator	Public ❗	🛑	onlyOwner
└	minterStatus	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❗
IERC2981	Interface	IERC165		
└	royaltyInfo	External ❗		NO❗
ERC165	Implementation	IERC165		
└	supportsInterface	Public ❗		NO❗
ERC721S	Implementation	Context, ERC165, IERC721, IERC721Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	_startTokenId	Internal 🔒		
└	_totalMinted	Internal 🔒		
└	_totalBurned	Internal 🔒		
└	supportsInterface	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	ownerOf	Public ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	_baseURI	Internal 🔒		
└	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 🔒	🛑	
└	_setApprovalForAll	Internal 🔒	🛑	
└	_checkOnERC721Received	Private 🔐	🛑	
└	tokenOfOwnerByIndex	Public ❗		NO❗
└	ownedBy	External ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	tokenByIndex	Public ❗		NO❗
└	_afterTokenTransfer	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
ERC2981	Implementation	IERC2981, ERC165		
└	supportsInterface	Public ❗		NO❗
└	royaltyInfo	External ❗		NO❗
└	_feeDenominator	Internal 🔒		
└	_setDefaultRoyalty	Internal 🔒	🛑	

└	_deleteDefaultRoyalty	Internal 🔒	🛑	
└	_setTokenRoyalty	Internal 🔒	🛑	
└	_resetTokenRoyalty	Internal 🔒	🛑	
ITokenURI	Interface				
└	tokenURI	External ❗		NO❗
IIsTokenLocked	Interface				
└	isTokenLocked	External ❗		NO❗
JCCGenesis	Implementation	ERC721S, PublicPrivateVoucherMinter, ERC2981, Pausable			
└	<Constructor>	Public ❗	🛑	ERC721S PublicPrivateVoucherMinter
└	setRoyaltyInfo	External ❗	🛑	onlyOwner
└	reveal	External ❗	🛑	onlyOwner
└	pause	External ❗	🛑	onlyOwner
└	unpause	External ❗	🛑	onlyOwner
└	_mintTo	Internal 🔒	🛑	
└	supportsInterface	Public ❗		NO❗
└	setBaseTokenURI	External ❗	🛑	onlyOwner
└	setNotRevealedURI	External ❗	🛑	onlyOwner
└	setMetadataContract	External ❗	🛑	onlyOwner
└	_baseURI	Internal 🔒		
└	baseTokenURI	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	setLockerContract	External ❗	🛑	onlyOwner
└	_beforeTokenTransfer	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.