Smart Contract
Security Audit
V1

Crypto Culinary Club Pass Smart Contract

7/12/2022

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**: 0xA0C5ff8415733208115Fde57bB2a1D0FE85afd30
- **Code:**

https://github.com/Saferico/Smart-Contracts-for-Projects/blob/main/CCC.sol

NFT Information

- Name: Crypto Culinary Club
- MAX Supply: 10,000
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum)

Crypto Culinary Club smart contract on Ethereum test net to test every function by the auditor.

https://goerli.etherscan.io/address/0xa0c5ff8415733208115fde57bb2a1d0fe85afd30

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, 3 low, 0 very low-level issues and 0 note in all solidity files of the contract

The files:

CCC.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
CryptoCulinaryClub_NFT.sol	**c3885bb57519945d6b0b8b5874e597695e751b5f3172a8fa70c386b4bea6ba64**	0xA0C5ff8415733208115Fde57bB2a1D0FE85afd30

- Contract: CryptoCulinaryClub_NFT
- Inherit: ERC721A, 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**
baseExtension	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
cost	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
owners	✓	Read / public	**Passed**
public_paused	✓	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**
mint	✓	Write / payable	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
setBaseExtension	✓	Write / public	**Passed**
withdraw	✓	Write / payable	**Passed**
setCost	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
ownerMint	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
startPublicSale	✓	Write / public	**Passed**
stopPublicSale	✓	Write / public	**Passed**
setBaseURI	✓	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:

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, and the same for the main mint function.

```
function mint(address _to, uint _amount) public payable {
        uint256 supply = totalSupply();
        require(!public_paused, "Sale has not started yet");
        require(supply + _amount <= maxSupply);
        require(msg.value >= cost * _amount);

        _mint(_to, _amount);
    }

    function ownerMint(address _to, uint _amount) public onlyOwner {
        uint256 supply = totalSupply();
        require(supply + _amount <= maxSupply);
        _mint(_to, _amount);
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version 2.

#Multiple pragma statements

Line	Pragma
6	pragma solidity ^0.8.0;
79	pragma solidity ^0.8.0;
105	pragma solidity ^0.8.0;
183	pragma solidity ^0.8.4;
457	pragma solidity ^0.8.4;
1402	pragma solidity ^0.8.7;

Description
There are multiple pragma statements in the code. The newest compiler version 0.8.17 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 mint NFT to any address.
The owner can pause and un pause the contract.
The owner can change the price at any time.

```
function ownerMint(address _to, uint _amount) public onlyOwner {
        uint256 supply = totalSupply();
        require(supply + _amount <= maxSupply);
        _mint(_to, _amount);
    }
function setCost(uint256 _newCost) public onlyOwner {
        cost = _newCost;
    }
    function startPublicSale() public onlyOwner {
        public_paused = false;
    }
    function stopPublicSale() public onlyOwner {
        public_paused = true;
    }
```

Remediation
Make these functions internal in next version or the team should announce the investors before doing anything 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.

Very Low:

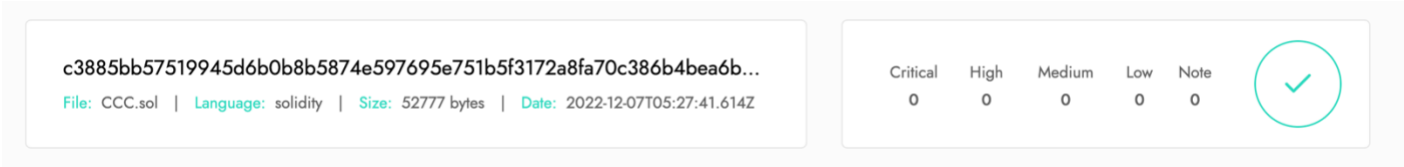
No Very Low severity vulnerabilities were found.

Notes:

No Notes vulnerabilities were found.

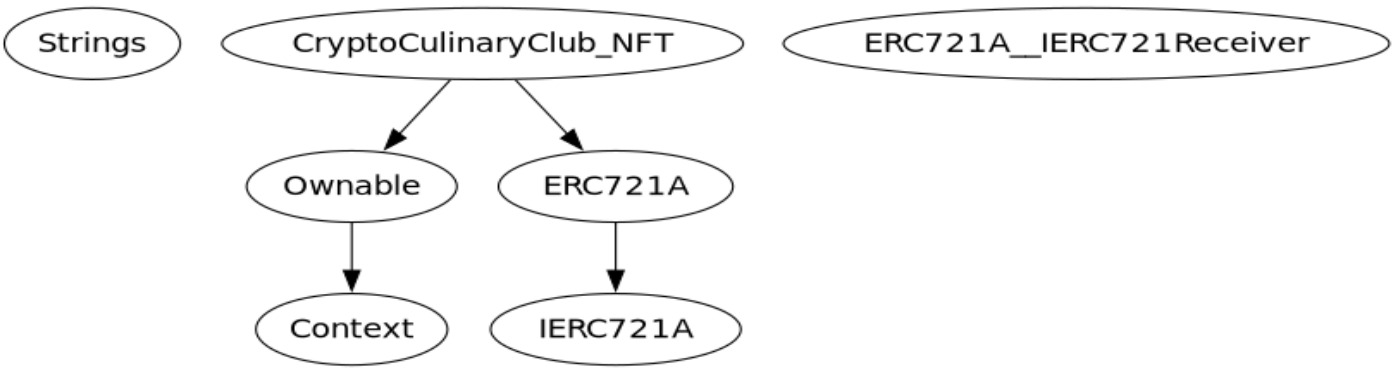
Automatic Testing

1- Check for security

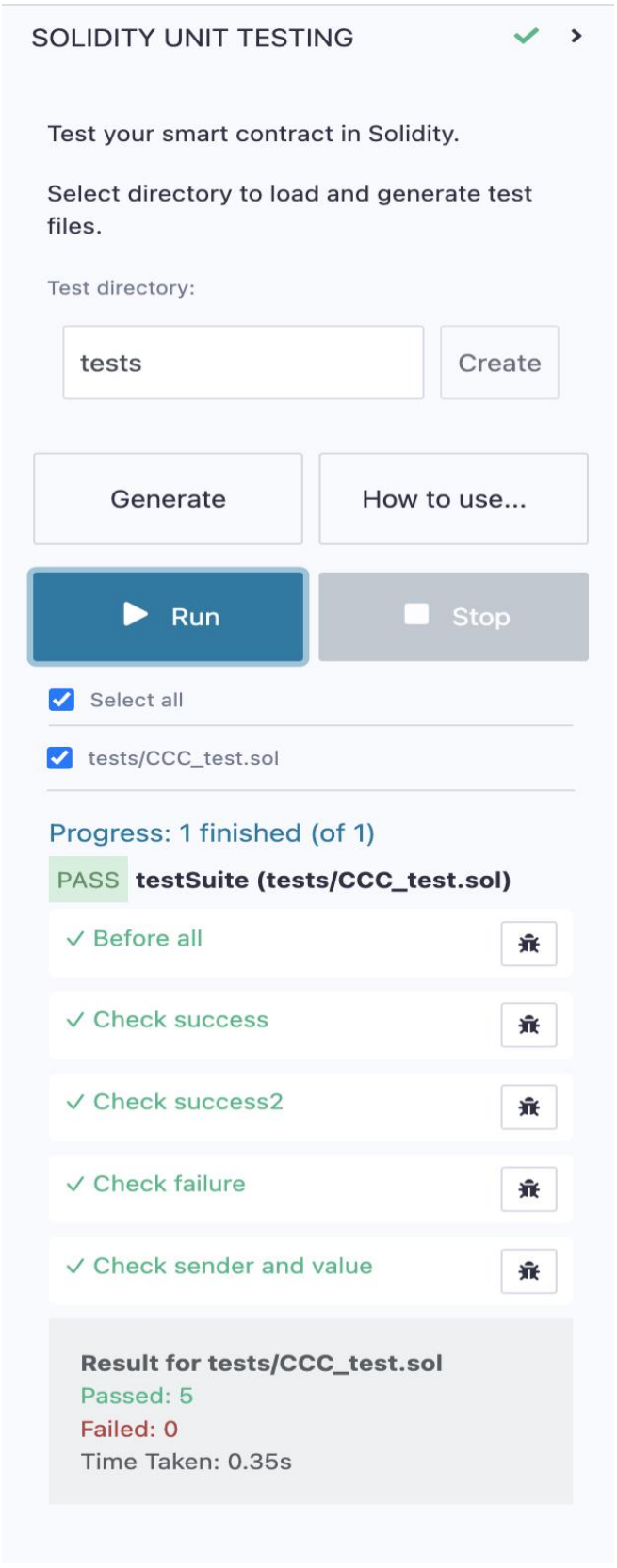


2- SOLIDITY STATIC ANALYSIS

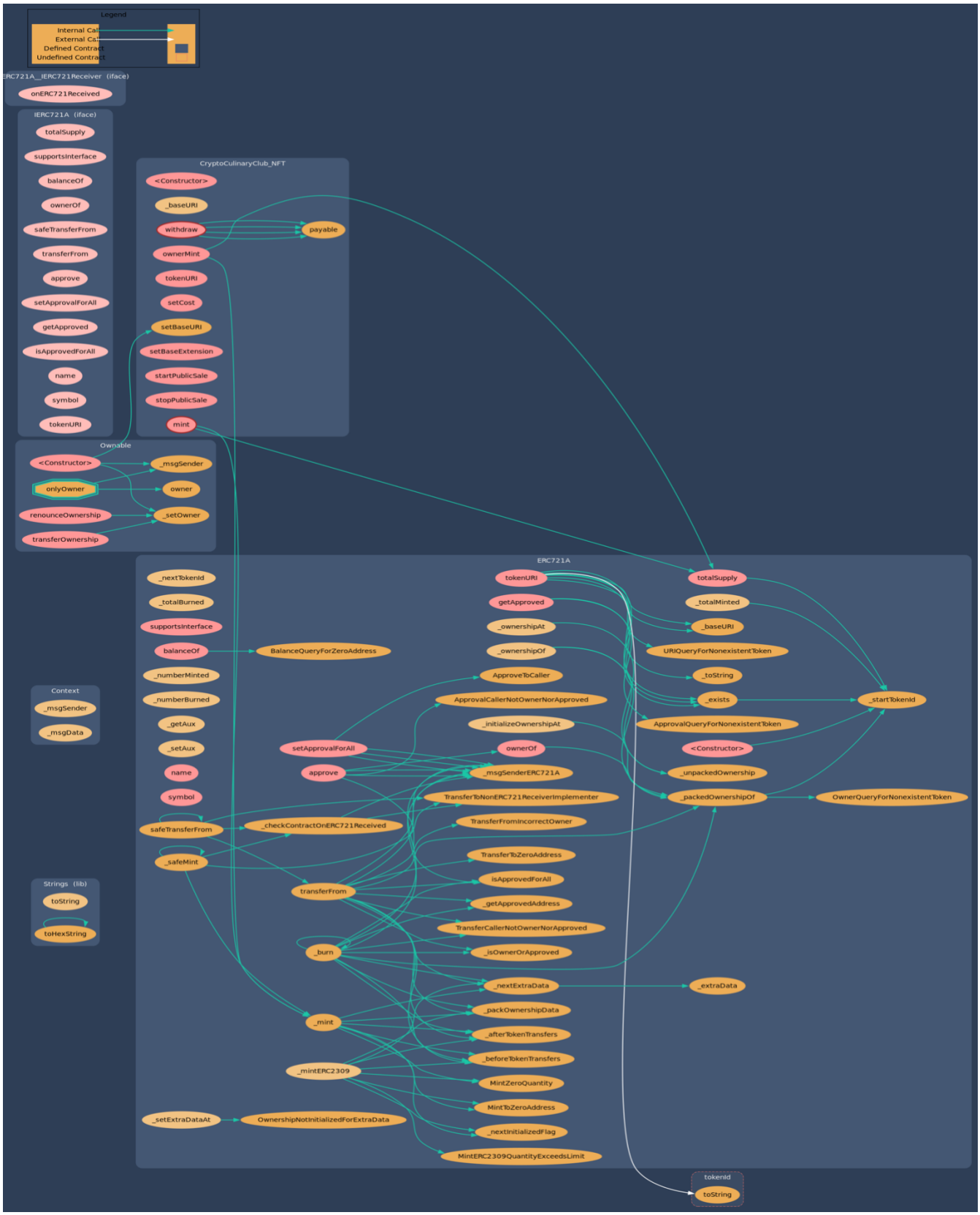
3- Inheritance graph



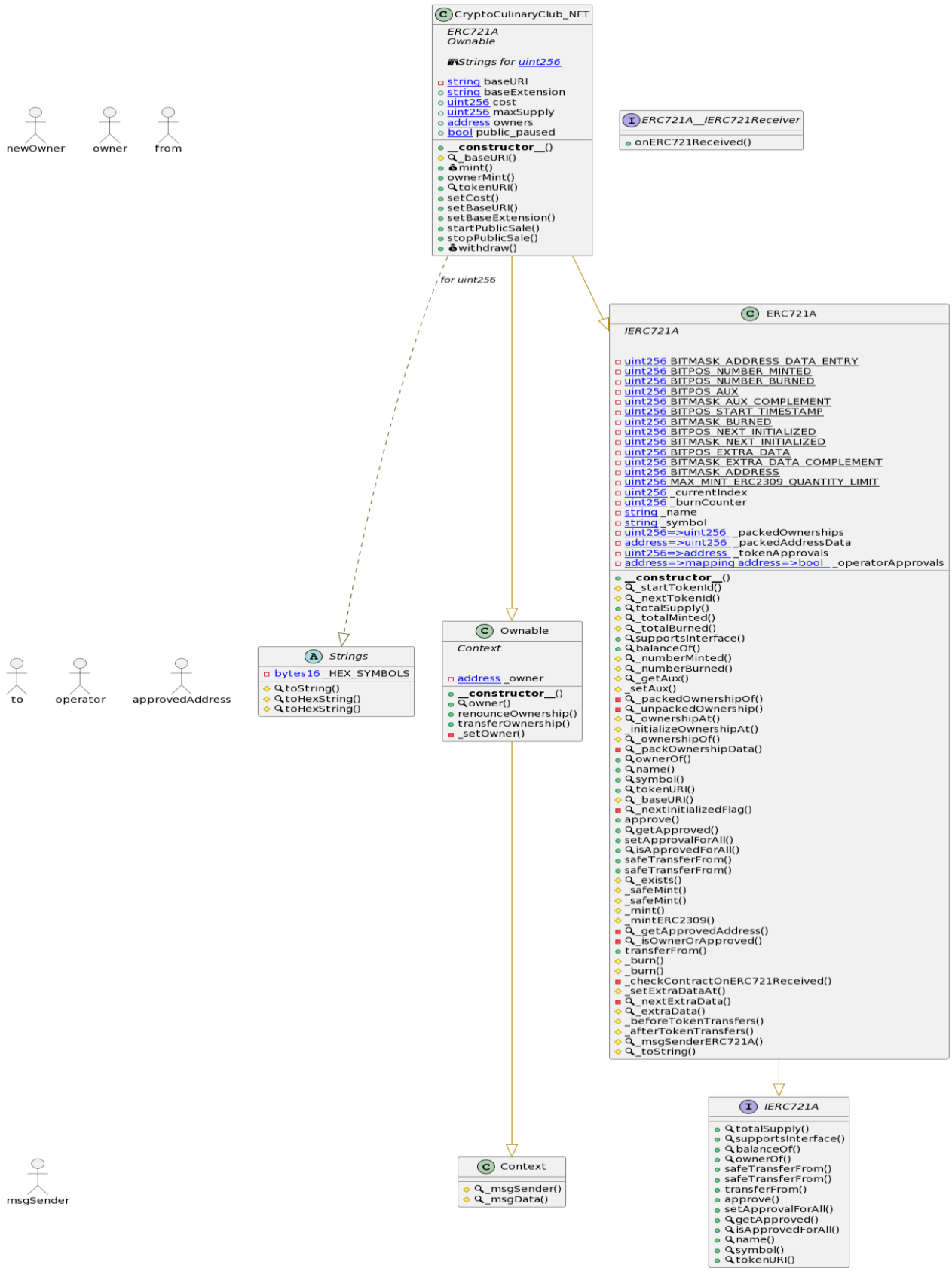
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC721A__IERC721Receiver (iface)
onERC721Received
IERC721A (iface)
totalSupply
supportsInterface
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
setApprovalForAll
getApproved
isApprovedForAll
name
symbol
tokenURI
CryptoCulinaryClub_NFT
<Constructor>
_baseURI
withdraw
payable
ownerMint
tokenURI
setCost
setBaseURI
setBaseExtension
startPublicSale
stopPublicSale
mint
Ownable
<Constructor>
_msgSender
onlyOwner
owner
renounceOwnership
_setOwner
transferOwnership
ERC721A
_nextTokenId
_totalBurned
supportsInterface
balanceOf
BalanceQueryForZeroAddress
_numberMinted
_numberBurned
_getAux
_setAux
name
symbol
safeTransferFrom
_checkContractOnERC721Received
_safeMint
transferFrom
_burn
_mint
_mintERC2309
_setExtraDataAt
OwnershipNotInitializedForExtraData
tokenURI
getApproved
_ownershipAt
_ownershipOf
ApproveToCaller
ApprovalCallerNotOwnerNorApproved
_initializeOwnershipAt
setApprovalForAll
approve
ownerOf
_msgSenderERC721A
TransferToNonERC721ReceiverImplementer
TransferFromIncorrectOwner
TransferToZeroAddress
isApprovedForAll
_getApprovedAddress
TransferCallerNotOwnerNorApproved
_isOwnerOrApproved
_nextExtraData
_packOwnershipData
_afterTokenTransfers
_beforeTokenTransfers
MintZeroQuantity
MintToZeroAddress
_nextInitializedFlag
MintERC2309QuantityExceedsLimit
totalSupply
_totalMinted
_baseURI
URIQueryForNonexistentToken
_toString
_exists
_startTokenId
ApprovalQueryForNonexistentToken
<Constructor>
_unpackedOwnership
_packedOwnershipOf
OwnerQueryForNonexistentToken
_extraData
Context
_msgSender
_msgData
Strings (lib)
toString
toHexString
tokenId
toString

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
6900a3ae  =>  toString(uint256)
8fba8d5c  =>  toHexString(uint256)
63e1cbea  =>  toHexString(uint256,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
fc201122  =>  _setOwner(address)
18160ddd  =>  totalSupply()
01ffc9a7  =>  supportsInterface(bytes4)
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)
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)
444996c1  =>  _packedOwnershipOf(uint256)
4fe3c13e  =>  _unpackedOwnership(uint256)
cbaf28ce  =>  _ownershipAt(uint256)
f2d31624  =>  _initializeOwnershipAt(uint256)
fb372cf2  =>  _ownershipOf(uint256)
bf460657  =>  _packOwnershipData(address,uint256)
743976a0  =>  _baseURI()
e0e30f80  =>  _nextInitializedFlag(uint256)
f8e76cc0  =>  _exists(uint256)
b3e1c718  =>  _safeMint(address,uint256)
6a4f832b  =>  _safeMint(address,uint256,bytes)
4e6ec247  =>  _mint(address,uint256)
4908d13b  =>  _mintERC2309(address,uint256)
56d5f541  =>  _getApprovedAddress(uint256)
be8825c4  =>  _isOwnerOrApproved(address,address,address)
9b1f9e74  =>  _burn(uint256)
834a9477  =>  _burn(uint256,bool)
d88343e2  =>  _checkContractOnERC721Received(address,address,uint256,bytes)
bd3cdd6d  =>  _setExtraDataAt(uint256,uint24)
```

```
5afe32e4  =>  _nextExtraData(address,address,uint256)
fc37bbd3  =>  _extraData(address,address,uint24)
ef435773  =>  _beforeTokenTransfers(address,address,uint256,uint256)
08c018f7  =>  _afterTokenTransfers(address,address,uint256,uint256)
b60986df  =>  _msgSenderERC721A()
f832e238  =>  _toString(uint256)
40c10f19  =>  mint(address,uint256)
484b973c  =>  ownerMint(address,uint256)
44a0d68a  =>  setCost(uint256)
55f804b3  =>  setBaseURI(string)
da3ef23f  =>  setBaseExtension(string)
0c1c972a  =>  startPublicSale()
da1b91c3  =>  stopPublicSale()
3ccfd60b  =>  withdraw()
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/CCC.sol	51dea5069c3c145d03c74295dd28c1d0160c0cca

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Strings	Library			
└	toString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IERC721A	Interface			
└	totalSupply	External ❗		NO❗
└	supportsInterface	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	ownerOf	External ❗		NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	setApprovalForAll	External ❗	🛑	NO❗
└	getApproved	External ❗		NO❗
└	isApprovedForAll	External ❗		NO❗
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	tokenURI	External ❗		NO❗
ERC721A__IERC721Receiver	Interface			
└	onERC721Received	External ❗	🛑	NO❗
ERC721A	Implementation	IERC721A		

└	<Constructor>	Public ❗	🛑	NO❗
└	_startTokenId	Internal 🔒		
└	_nextTokenId	Internal 🔒		
└	totalSupply	Public ❗		NO❗
└	_totalMinted	Internal 🔒		
└	_totalBurned	Internal 🔒		
└	supportsInterface	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	_numberMinted	Internal 🔒		
└	_numberBurned	Internal 🔒		
└	_getAux	Internal 🔒		
└	_setAux	Internal 🔒	🛑	
└	_packedOwnershipOf	Private 🔐		
└	_unpackedOwnership	Private 🔐		
└	_ownershipAt	Internal 🔒		
└	_initializeOwnershipAt	Internal 🔒	🛑	
└	_ownershipOf	Internal 🔒		
└	_packOwnershipData	Private 🔐		
└	ownerOf	Public ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	_baseURI	Internal 🔒		
└	_nextInitializedFlag	Private 🔐		
└	approve	Public ❗	🛑	NO❗
└	getApproved	Public ❗		NO❗
└	setApprovalForAll	Public ❗	🛑	NO❗
└	isApprovedForAll	Public ❗		NO❗
└	safeTransferFrom	Public ❗	🛑	NO❗
└	safeTransferFrom	Public ❗	🛑	NO❗
└	_exists	Internal 🔒		
└	_safeMint	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_mintERC2309	Internal 🔒	🛑	
└	_getApprovedAddress	Private 🔐		
└	_isOwnerOrApproved	Private 🔐		
└	transferFrom	Public ❗	🛑	NO❗
└	_burn	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_checkContractOnERC721Received	Private 🔐	🛑	
└	_setExtraDataAt	Internal 🔒	🛑	
└	_nextExtraData	Private 🔐		
└	_extraData	Internal 🔒		
└	_beforeTokenTransfers	Internal 🔒	🛑	
└	_afterTokenTransfers	Internal 🔒	🛑	
└	_msgSenderERC721A	Internal 🔒		
└	_toString	Internal 🔒		
CryptoCulinaryClub_NFT	Implementation	ERC721A, Ownable		
└	<Constructor>	Public ❗	🛑	ERC721A
└	_baseURI	Internal 🔒		
└	mint	Public ❗	💵	NO❗
└	ownerMint	Public ❗	🛑	onlyOwner
└	tokenURI	Public ❗		NO❗

└	setCost	Public ❗	🛑	onlyOwner
└	setBaseURI	Public ❗	🛑	onlyOwner
└	setBaseExtension	Public ❗	🛑	onlyOwner
└	startPublicSale	Public ❗	🛑	onlyOwner
└	stopPublicSale	Public ❗	🛑	onlyOwner
└	withdraw	Public ❗	💸	onlyOwner

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.