Smart Contract
Security Audit
V1

Meta Real Estate Smart Contract

12/4/2022

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**: 0xeB836f821b8cB794Ee9e0802054CACa13A320a23
- **Code:**

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

NFT Information

- Name: MRE
- MAX Supply: 5000 , MAX Supply for presale: 5000
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum)

Meta Real Estate Smart contract on ETH test net to test every function by the auditor.

https://rinkeby.etherscan.io/address/0xeb836f821b8cb794ee9e0802054caca13a320a23

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well Secured**.

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 2 notes in all solidity files of the contract

The files:

MetaRealEstate.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
MetaRealEstate.sol	**c39602f84840453e55396e2b27bd2ff6fe188b33072525b0cf6d21ea4f3b2478**	0xeB836f821b8cB794Ee9e0802054CACa13A320a23

- Contract: MetaRealEstate
- Inherit: ERC721, ERC721URIStorage, 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**
FEE	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
MAX_TOKENS	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
baseExtension	✓	Read / public	**Passed**
getCounter	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

ownerOf	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
FEEpresales	✓	Read / public	**Passed**
revealed	✓	Read / public	**Passed**
presale	✓	Read / public	**Passed**
MAX_TOKENS_PRESALE	✓	Read / public	**Passed**
publicSale	✓	Read / public	**Passed**
MAXNFTperADDR	✓	Read / public	**Passed**
MAXNFTperTX	✓	Read / public	**Passed**
isWhitelisted	✓	Read / public	**Passed**
_tokenidreserved	✓	Read / public	**Passed**
decreaseCounter	✓	Write / public	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
setReveal	✓	Write / public	**Passed**
Pause	✓	Write / public	**Passed**
safeMint	✓	Write / payable	**Passed**
setCost	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
whitelistUserBulk	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
increaseCounter	✓	Write / public	**Passed**

setBaseURI	✓	Write / public	**Passed**
setPreSaleCost	✓	Write / public	**Passed**
setBaseExtension	✓	Write / public	**Passed**
safeMintAirdrop	✓	Write / payable	**Passed**
setMaxNFTperBuyer	✓	Write / public	**Passed**
setMaxNFTperTX	✓	Write / public	**Passed**
setMax_TOKEN	✓	Write / public	**Passed**
setMax_TOKEN_Presale	✓	Write / public	**Passed**
setprerevealURI	✓	Write / public	**Passed**
removeWhitelistUser	✓	Write / public	**Passed**
setPresaleCost	✓	Write / public	**Passed**
whitelistUser	✓	Write / public	**Passed**
startPresale	✓	Write / public	**Passed**
startPublicsale	✓	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 airdrop 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 safeMintAirdrop(uint minttimes, address toAddress) public payable
onlyOwner {require(!paused);
        require(minttimes <= MAXNFTperTX,"Max 10 NFTs per tx!");
                for(uint i=0;i<minttimes;i++){
                    _safeMint(toAddress, _tokenIdCounter.current());
                    _tokenIdCounter.increment();}}
```

Remediation

Use the require statement to check for zero addresses.

```
require(_toAddress!= address(0), "Not Mint for the zero address");
```

Status: Closed. Fixed in version2.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.7 instead of ^0.8.2). 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 version2

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

Description

The owner can change pause/ un pause the smart contract.
The owner can add / remove any address to White List.
The owner can change the presale price and public sale.

```
function whitelistUser(address _user) public onlyOwner {
        whitelisted[_user] = true;

    }
    function whitelistUserBulk(address[] memory _users) public onlyOwner {
        uint i;
        for(i=0;i<_users.length;i++){
            whitelisted[_users[i]] = true;
            }
    }

    function setCost(uint256 _newCost) public onlyOwner {
        FEE = _newCost;
    }
function setPresaleCost(uint256 _newCost) public onlyOwner {
        FEEpresales = _newCost;
    }

    function removeWhitelistUser(address _user) public onlyOwner {
        whitelisted[_user] = false;
    }
function pause(bool _state) public onlyOwner {
        paused = _state;
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change anything and 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:

#Unnecessary import of ERC721 library

Description
The main contract inherits: ERC721, ERC721URIStorage, and Ownable, ERC721URIStorage which is already import ERC721 library, so no need to import it again in the main contract.

Remediation
Remove unnecessary library from the main contract save some gas fees.

Status: Closed. Fixed in version2.

#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 "__ tokenidreserved " which is public.

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

Status: Closed. Fixed in version2

Automatic Testing

1- Check for security

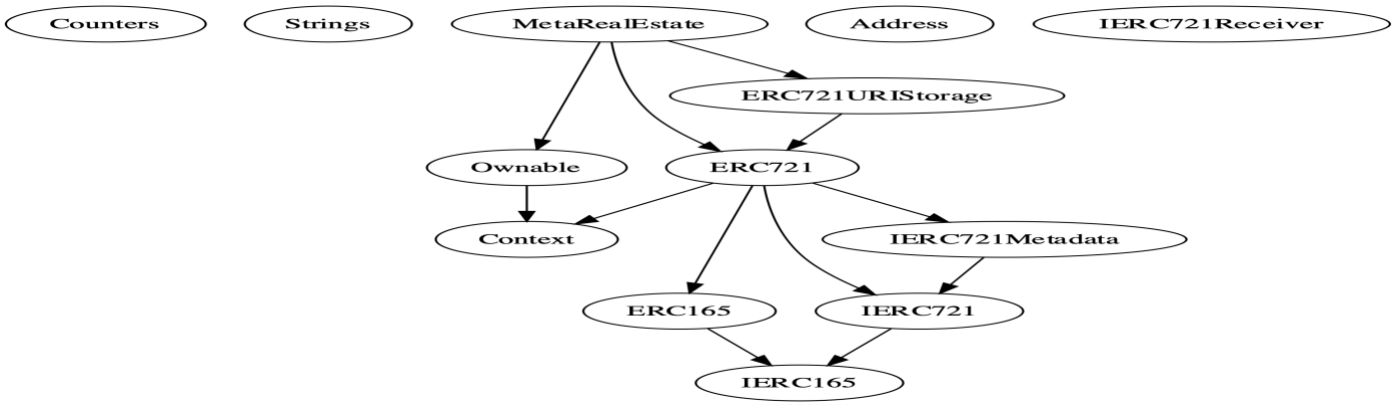
c39602f84840453e55396e2b27bd2ff6fe188b33072525b0cf6d21ea4f3b2478

File: MetaReal... | Language: solidity | Size: 7689 bytes | Date: 2022-04-12T12:13:04.277Z

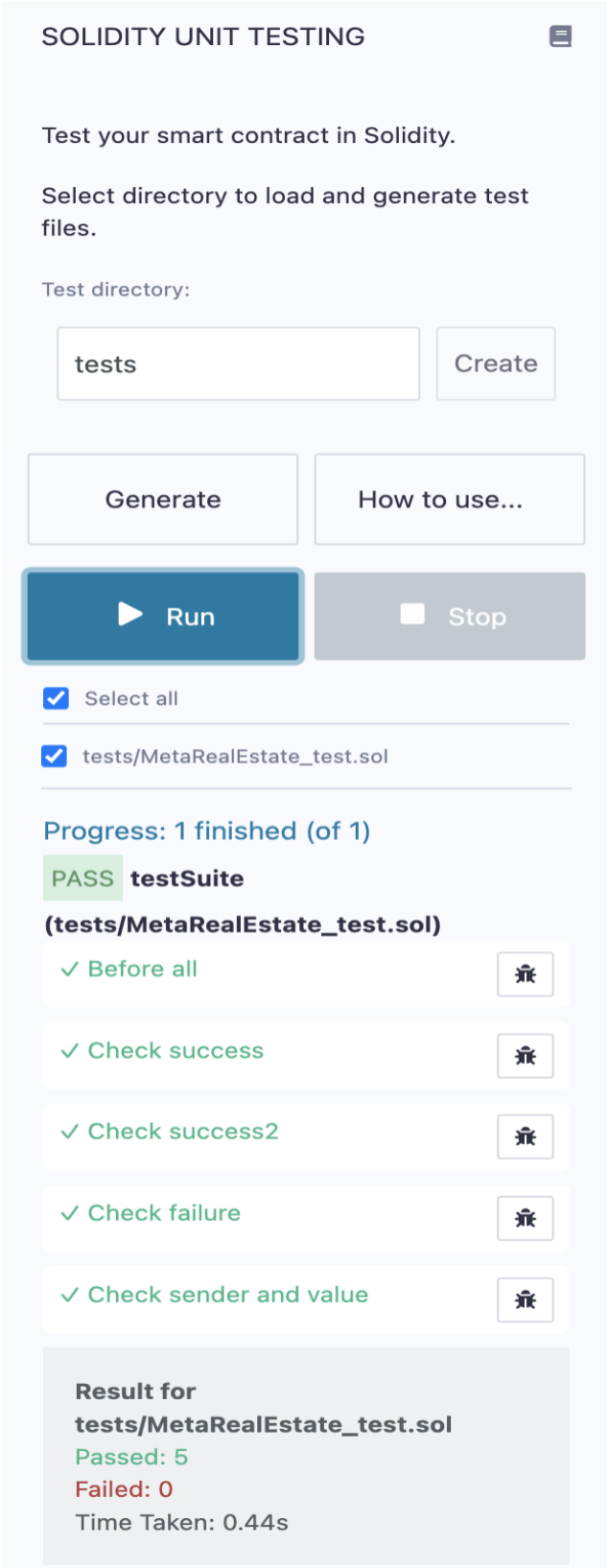
Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS

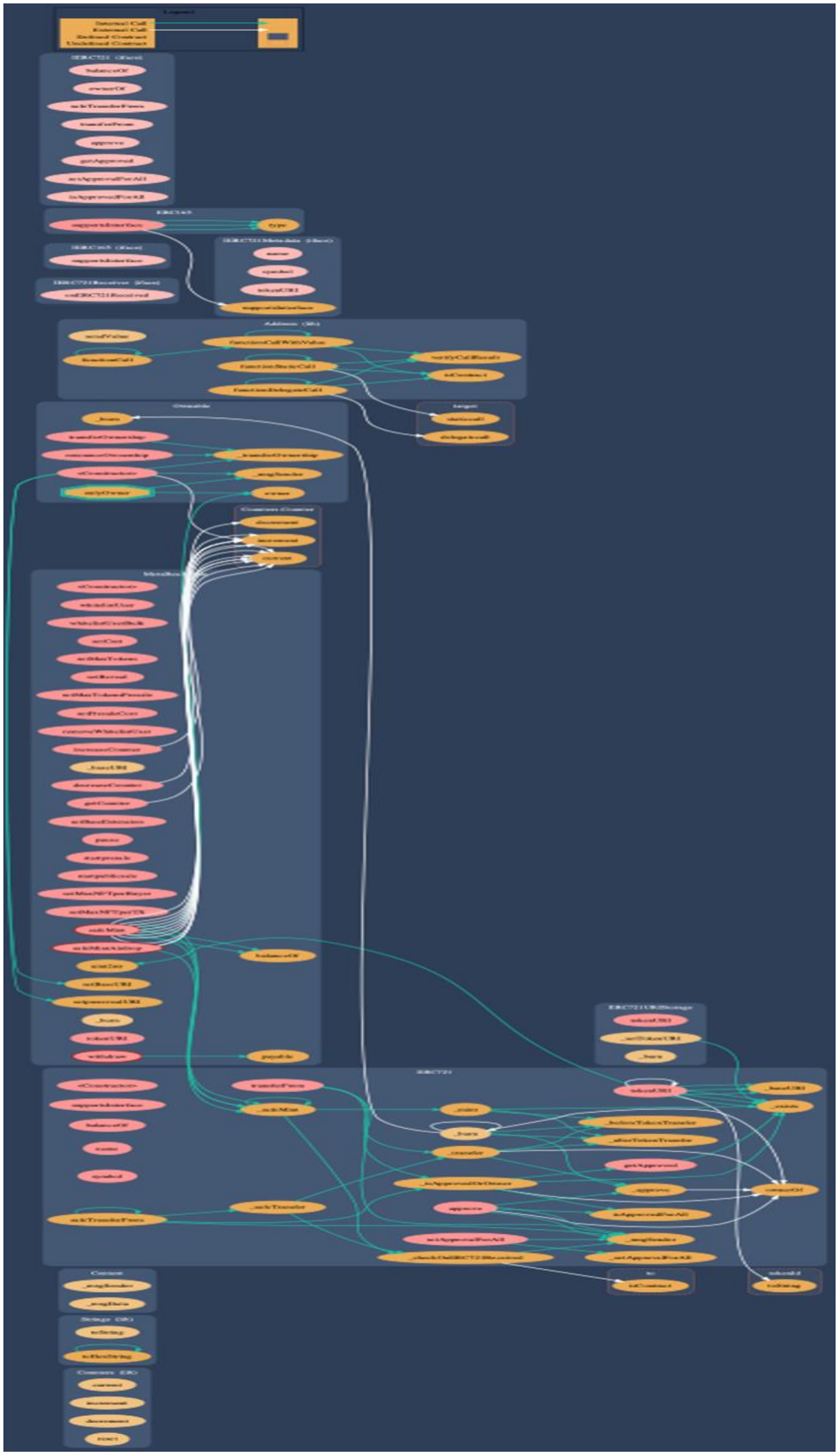
3- Inheritance graph



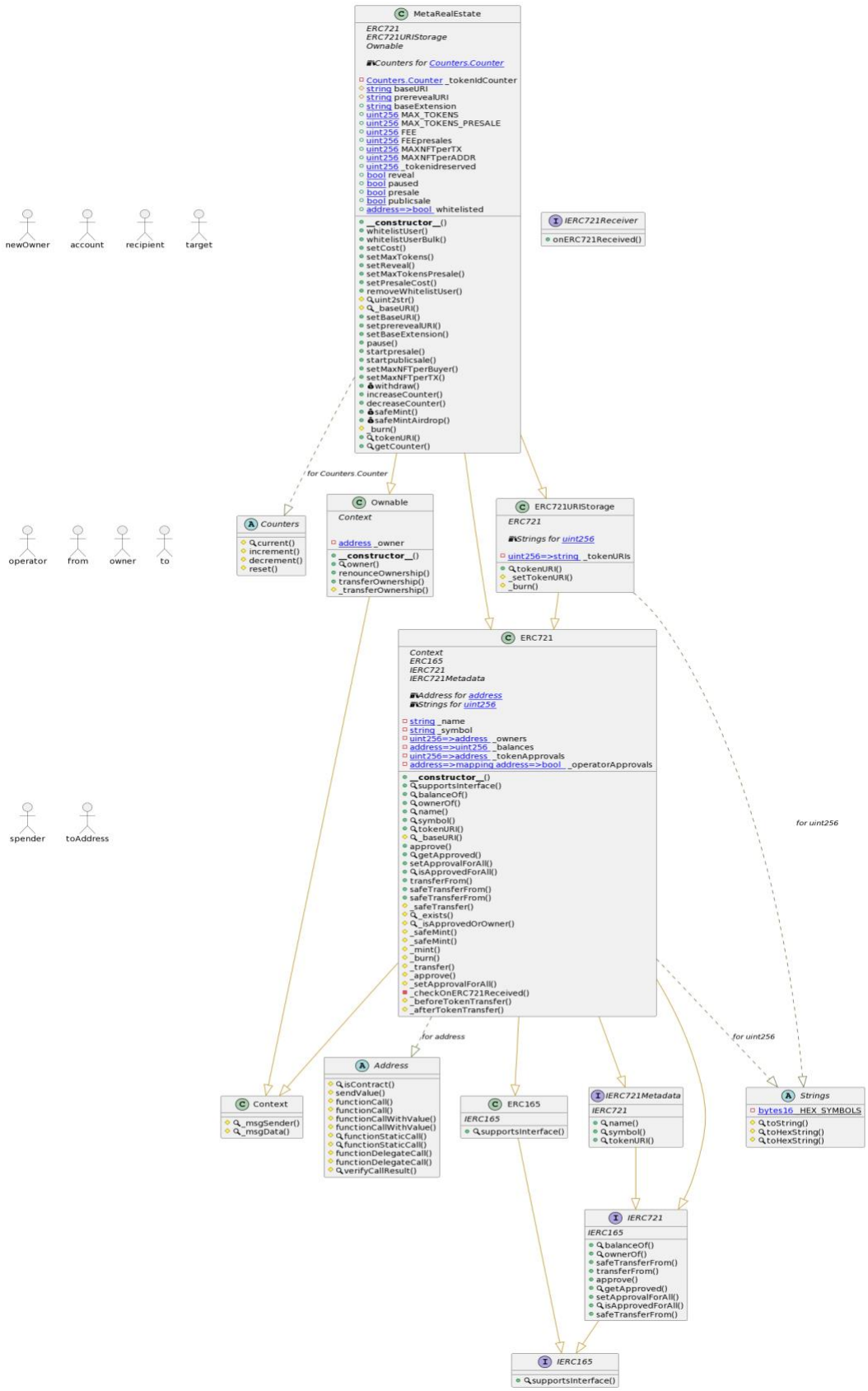
4- SOLIDITY UNIT TESTING



5- Call graph



Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
34072096  =>  setMaxNFTperTX(uint256)
ad04a8d1  =>  current(Counter)
e2bee435  =>  increment(Counter)
854ec98e  =>  decrement(Counter)
440d212a  =>  reset(Counter)
6900a3ae  =>  toString(uint256)
8fba8d5c  =>  toHexString(uint256)
63e1cbea  =>  toHexString(uint256,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
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)
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)
06fdde03  =>  name()
95d89b41  =>  symbol()
c87b56dd  =>  tokenURI(uint256)
743976a0  =>  _baseURI()
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)
8c4e3f32  =>  _setApprovalForAll(address,address,bool)
1fd01de1  =>  _checkOnERC721Received(address,address,uint256,bytes)
```

```
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
01538868  =>  _setTokenURI(uint256,string)
4a4c560d  =>  whitelistUser(address)
d4f80886  =>  whitelistUserBulk(address[])
44a0d68a  =>  setCost(uint256)
11e776fe  =>  setMaxTokens(uint256)
f89533de  =>  settokenidreserved(uint256)
2a3f300c  =>  setReveal(bool)
46ca1ab8  =>  setMaxTokensPresale(uint256)
8fdcf942  =>  setPresaleCost(uint256)
30cc7ae0  =>  removeWhitelistUser(address)
f76f950e  =>  uint2str(uint256)
55f804b3  =>  setBaseURI(string)
3078cfc6  =>  setprerevealURI(string)
da3ef23f  =>  setBaseExtension(string)
02329a29  =>  pause(bool)
18fc2871  =>  startpresale(bool)
5ccda733  =>  startpublicsale(bool)
340bb334  =>  setMaxNFTperBuyer(uint256)
3ccfd60b  =>  withdraw()
b49004e9  =>  increaseCounter()
8bba42b3  =>  decreaseCounter()
31c864e8  =>  safeMint(uint256)
8e8dc1dd  =>  safeMintAirdrop(uint256,address)
467fc5f8  =>  safeMintReserved(uint256)
8ada066e  =>  getCounter()
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/MetaRealEstate.sol |
41272fad94690e6876070b249601f2f0ec00dd20 |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **Counters** | Library |  |||
| └ | current | Internal 🔒 |   | |
| └ | increment | Internal 🔒 | 🛑 | |
| └ | decrement | Internal 🔒 | 🛑 | |
| └ | reset | Internal 🔒 | 🛑 | |
||||||
| **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 |
| └ | _transferOwnership | 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❗
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❗
IERC721Metadata	Interface	IERC721		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	tokenURI	External ❗		NO❗
ERC721	Implementation	Context, ERC165, IERC721, IERC721Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	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 🔐	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
ERC721URIStorage	Implementation	ERC721		

└	tokenURI	Public ❗		NO❗
└	_setTokenURI	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
MetaRealEstate	Implementation	ERC721, ERC721URIStorage, Ownable		
└	<Constructor>	Public ❗	🛑	ERC721
└	whitelistUser	Public ❗	🛑	onlyOwner
└	whitelistUserBulk	Public ❗	🛑	onlyOwner
└	setCost	Public ❗	🛑	onlyOwner
└	setMaxTokens	Public ❗	🛑	onlyOwner
└	setReveal	Public ❗	🛑	onlyOwner
└	setMaxTokensPresale	Public ❗	🛑	onlyOwner
└	setPresaleCost	Public ❗	🛑	onlyOwner
└	removeWhitelistUser	Public ❗	🛑	onlyOwner
└	uint2str	Internal 🔒		
└	_baseURI	Internal 🔒		
└	setBaseURI	Public ❗	🛑	onlyOwner
└	setprerevealURI	Public ❗	🛑	onlyOwner
└	setBaseExtension	Public ❗	🛑	onlyOwner
└	pause	Public ❗	🛑	onlyOwner
└	startpresale	Public ❗	🛑	onlyOwner
└	startpublicsale	Public ❗	🛑	onlyOwner
└	setMaxNFTperBuyer	Public ❗	🛑	onlyOwner
└	setMaxNFTperTX	Public ❗	🛑	onlyOwner
└	withdraw	Public ❗	💸	onlyOwner
└	increaseCounter	Public ❗	🛑	onlyOwner
└	decreaseCounter	Public ❗	🛑	onlyOwner
└	safeMint	Public ❗	💸	NO❗
└	safeMintAirdrop	Public ❗	💸	onlyOwner
└	_burn	Internal 🔒	🛑	
└	tokenURI	Public ❗		NO❗
└	getCounter	Public ❗		NO❗

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.