Smart Contract
Security Audit
V1

Tru or Die Smart Contract

3/5/2022

https://saferico.com/

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

–

Table of Contents

Background

The purpose of the audit was to achieve the following:

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

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

Project Information

- **Platform**: Ethereum
- **Contract Address**: 0xEE058cc8E47c72F4795f9DB26769ac2445E50ae0
- **Code:**

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

NFT Information

- Name: TD
- Total Supply: 10000

0xe381Fefad57f20908BF0d0cDEBaa585d594a2B9a

This is an Ethereum network address. Do not send ETH over any other network to this address or your funds may be lost. Do not send NFTs or Wrapped ETH (ETH) to this address.

Contracts address deployed to test net (ETH)

Tru or Die Smart contract on ETH test net to test write functions by the auditor.

https://rinkeby.etherscan.io/address/0xee058cc8e47c72f4795f9db26769ac2445e50ae0

Executive Summary

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

The files:

TruorDie.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
TruorDie.sol	**b9eba29a3288744cbd84b74f01f70a7256a72c5a09179cdf6b91bf45558d0307**	0xee058cc8e47c72f4795f9db26769ac2445e50ae0

- Contract: TruorDie
- Inherit: ERC721, 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**
presaleCost	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
isInWhitelist	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
presale	✓	Read / public	**Passed**
maxMintAmountPerTx	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

tokenURI	✓	Read / public	**Passed**
uriSuffix	✓	Read / public	**Passed**
maxSupplyLimits	✓	Read / public	**Passed**
onlyWhitelisted	✓	Read / public	**Passed**
nftPublicPerAddressLimit	✓	Read / public	**Passed**
whitelistedAddressesList	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
revealed	✓	Read / public	**Passed**
maxSupply	✓	Read / public	**Passed**
WalletOfOwner	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
nftPresalePerAddressLimit	✓	Read / public	**Passed**
publicCost	✓	Read / public	**Passed**
uriPrefix	✓	Read / public	**Passed**
addressPresaleMintedBalance	✓	Read / public	**Passed**
addressPublicMintedBalance	✓	Read / public	**Passed**
hiddenMetadataUri	✓	Read / public	**Passed**
reveal	✓	Write / public	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
setUriPrefix	✓	Write / public	**Passed**
mint	✓	Write / payable	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**

setPaused	✓	Write / public	**Passed**
setPublicCost	✓	Write / public	**Passed**
setUriSuffix	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
withdraw	✓	Write / payable	**Passed**
setPresaleCost	✓	Write / public	**Passed**
setNftPublicPerAddressLimit	✓	Write / public	**Passed**
setOnlyWhitelisted	✓	Write / public	**Passed**
setNotRevealedURI	✓	Write / public	**Passed**
ownerMintSpecific	✓	Write / public	**Passed**
addArrayToWhiteList	✓	Write / public	**Passed**
addToWhiteList	✓	Write / public	**Passed**
ownerMint	✓	Write / public	**Passed**
mintForAddress	✓	Write / public	**Passed**
setHiddenMetadataUri	✓	Write / public	**Passed**
removeFromWhitelist	✓	Write / public	**Passed**
setNFTPresalePerAddressLimit	✓	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:

#Possibility of losing all funds

Description

The most important function is not checked if the owner is the caller or not . the withdraw function should the owner is the only controller but in this case the developer make it open means any one can withdraw all funds of the project at any time which make the project fails to protect the funds and the project will not complete without these funds.

```
function withdraw() public payable {
        (bool os, ) = payable(owner()).call{value: address(this).balance}("");
        require(os);
    }
```

Remediation

You have to make sure the owner is the only controller.

Status: Closed. Fixed in version2.

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 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 mintForAddress(uint256 _mintAmount, address _receiver) public
mintCompliance(_mintAmount) onlyOwner {
        _mintLoop(_receiver, _mintAmount);
    }
```

Remediation

Use the require statement to check for zero addresses.

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 0.8.13 instead of >= 0.7.0 < 0.9.0;). 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 pause and un Pause mint.
The owner can add / remove any address to whitelist mint stage.
The owner can change the price in WL or non WL stage.

```
  function addToWhiteList(address _addr) public onlyOwner {
        whitelistedAddressesList[_addr] = true;
    }

    function addArrayToWhiteList(address[] memory _addrs) public onlyOwner {

        for (uint256 i = 0;i< _addrs.length; i++) {
            whitelistedAddressesList[_addrs[i]] = true;
        }
    }
function removeFromWhiteList(address _addr) public onlyOwner {
        whitelistedAddressesList[_addr] = false;
    }

function setPaused(bool _state) public onlyOwner {
        paused = _state;
    }

    // Set Presale
    function setPresale(bool _state) public onlyOwner {
        presale = _state;
    }
function setPresaleCost(uint256 _cost) public onlyOwner {
        presaleCost = _cost;
    }

    // Publicsale Cost
    function setPublicsaleCost(uint256 _cost) public onlyOwner {
        publicsaleCost = _cost;
    }
```

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:

#Missing zero address validation for whitelist functions

Description

When the owner wants to add an address or multi addresses to the whitelist it will much better if he has to check for the zero address to make, he didn't add the burn address to the whitelist for the burn address. Otherwise, the whitelist functions will cost more gas.

```
function addToWhiteList(address _addr) public onlyOwner {
        whitelistedAddressesList[_addr] = true;
    }

    function addArrayToWhiteList(address[] memory _addrs) public onlyOwner {

        for (uint256 i = 0;i< _addrs.length; i++) {
            whitelistedAddressesList[_addrs[i]] = true;
        }
    }
```

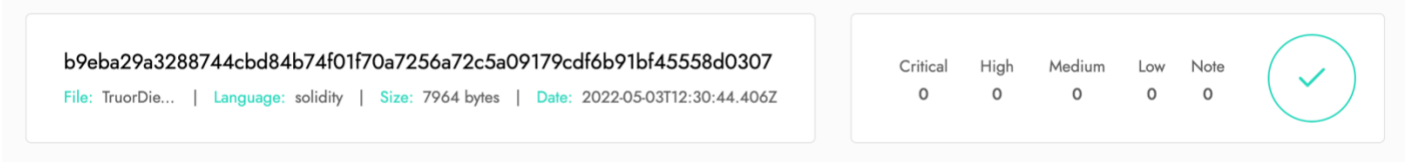
Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version2.

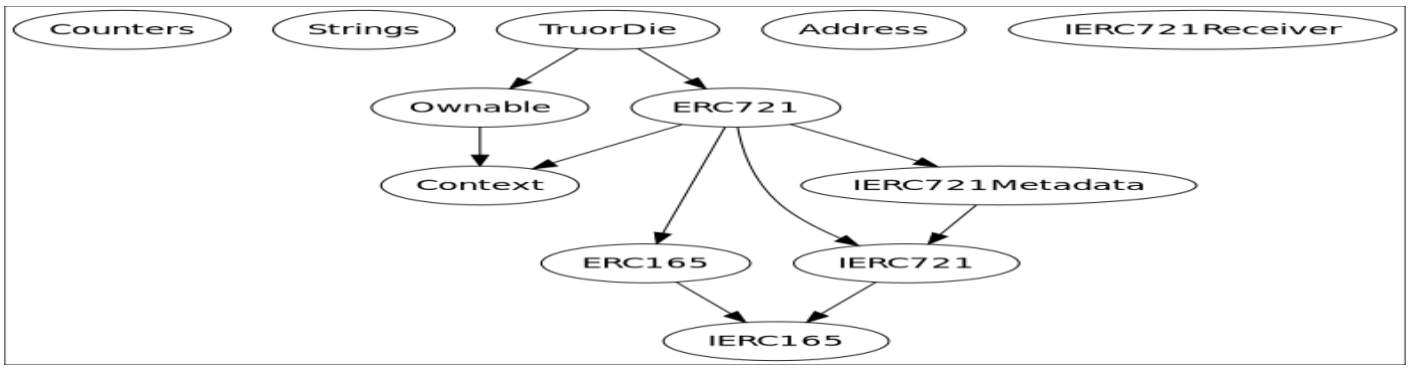
Automatic Testing

1- Check for security

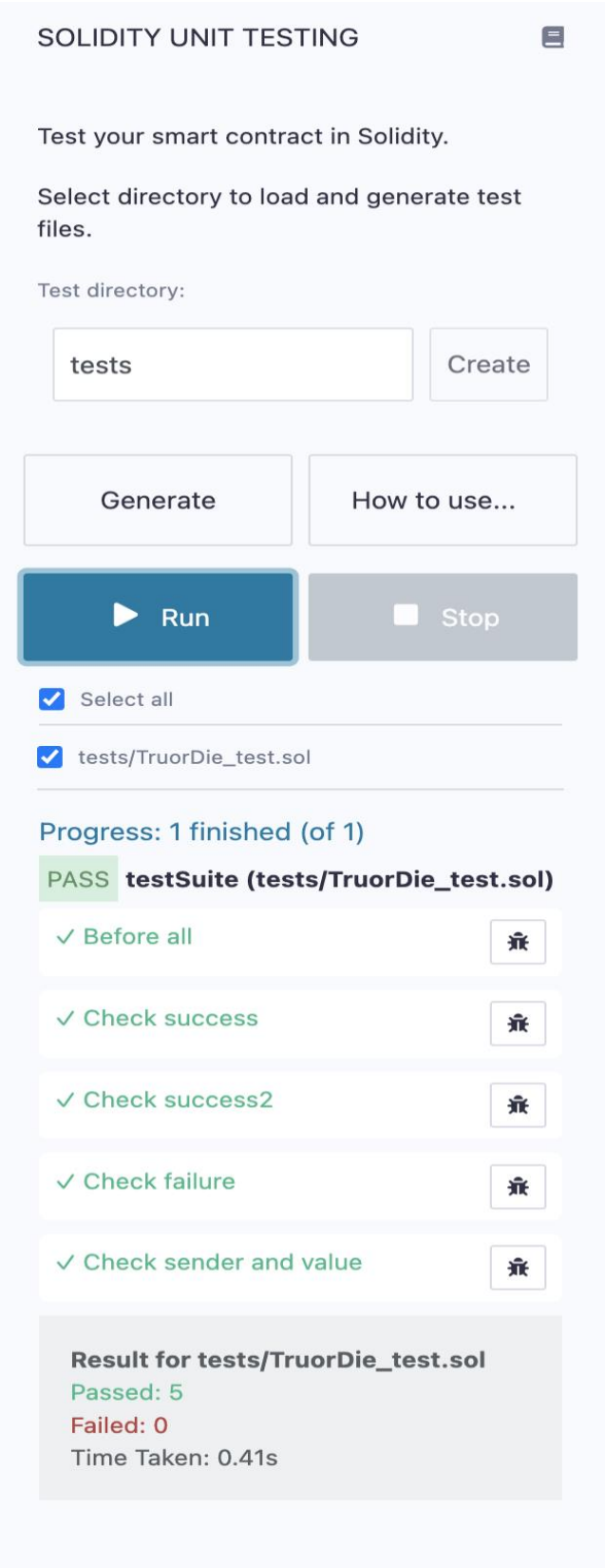


2- SOLIDITY STATIC ANALYSIS

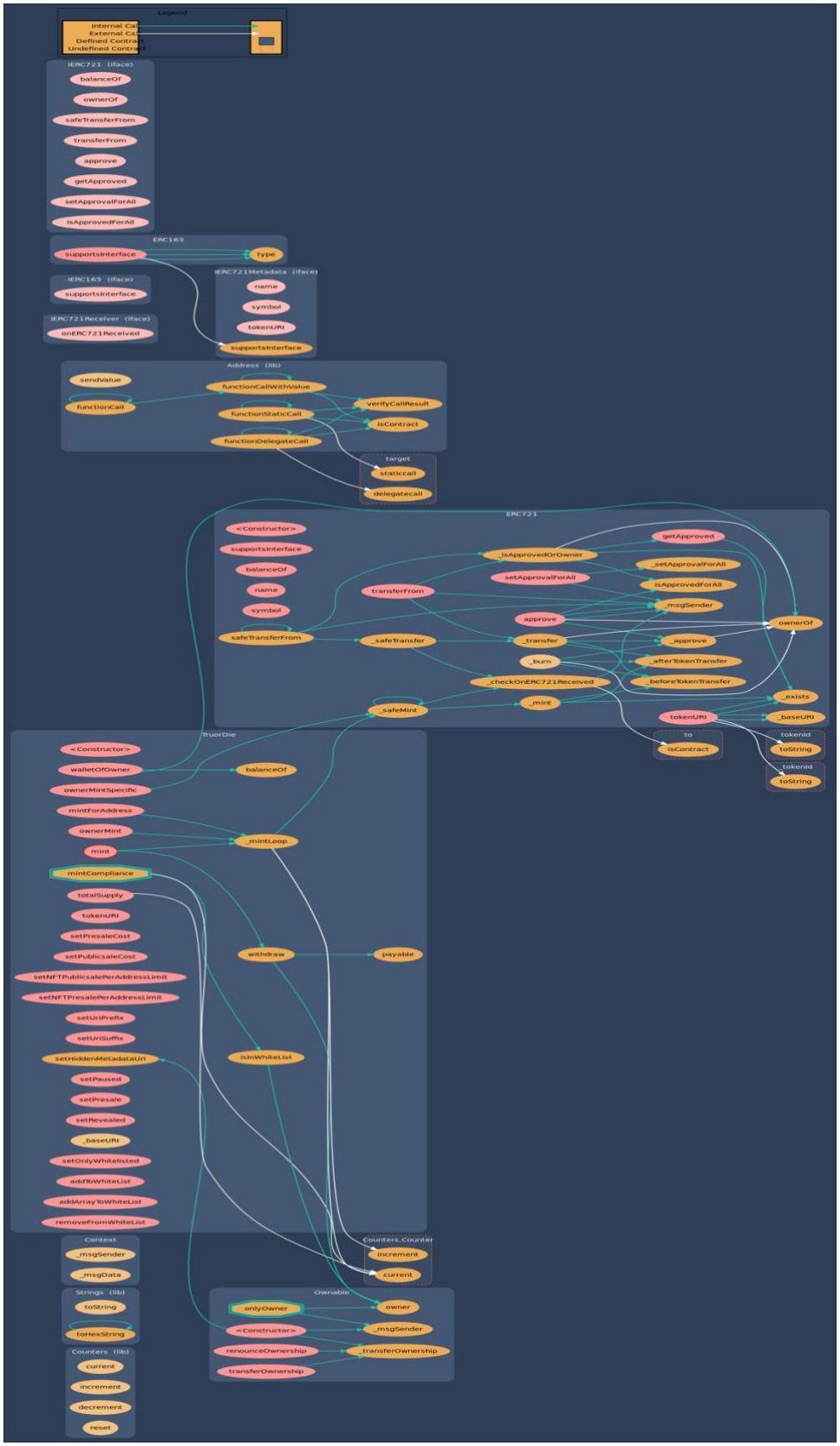
3- Inheritance graph



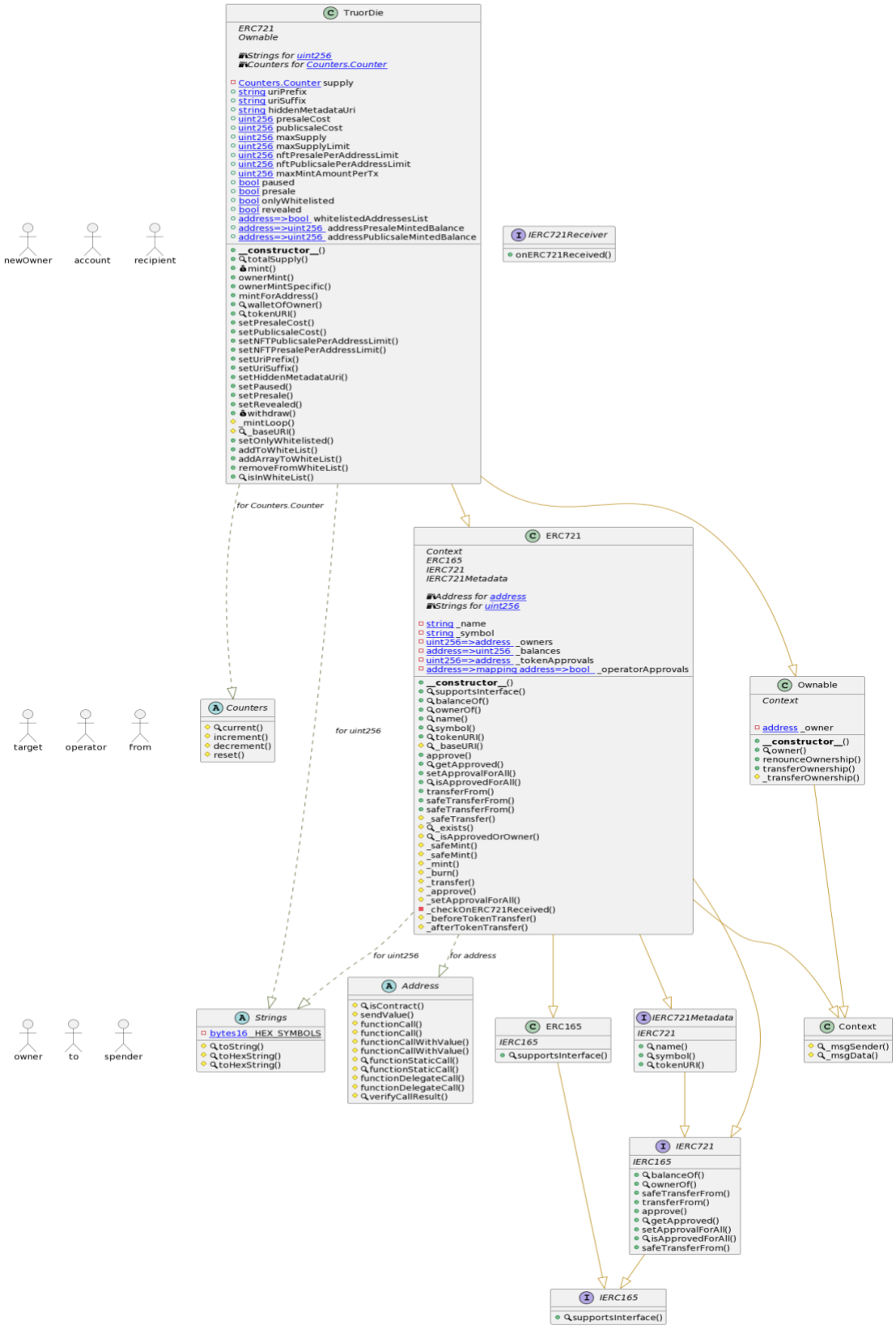
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
IERC721 (iface)
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
getApproved
setApprovalForAll
isApprovedForAll
ERC165
supportsInterface
type
IERC165 (iface)
supportsInterface
IERC721Metadata (iface)
name
symbol
tokenURI
supportsInterface
IERC721Receiver (iface)
onERC721Received
Address (lib)
sendValue
functionCallWithValue
functionCall
functionStaticCall
verifyCallResult
isContract
functionDelegateCall
target
staticcall
delegatecall
ERC721
<Constructor>
supportsInterface
balanceOf
name
symbol
safeTransferFrom
_safeTransfer
transferFrom
_isApprovedOrOwner
setApprovalForAll
approve
_transfer
_burn
_checkOnERC721Received
_safeMint
_mint
getApproved
_setApprovalForAll
isApprovedForAll
_msgSender
ownerOf
_approve
_afterTokenTransfer
_beforeTokenTransfer
_exists
tokenURI
_baseURI
to
isContract
tokenId
toString
tokenId
toString
TruorDie
<Constructor>
walletOfOwner
balanceOf
ownerMintSpecific
mintForAddress
ownerMint
mint
_mintLoop
mintCompliance
totalSupply
tokenURI
setPresaleCost
setPublicsaleCost
setNFTPublicsalePerAddressLimit
setNFTPresalePerAddressLimit
setUriPrefix
setUriSuffix
setHiddenMetadataUri
setPaused
setPresale
setRevealed
_baseURI
setOnlyWhitelisted
addToWhiteList
addArrayToWhiteList
removeFromWhiteList
withdraw
payable
isInWhiteList
Context
_msgSender
_msgData
Strings (lib)
toString
toHexString
Counters (lib)
current
increment
decrement
reset
Counters.Counter
Increment
current
Ownable
onlyOwner
<Constructor>
renounceOwnership
transferOwnership
owner
_msgSender
_transferOwnership

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
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)
18160ddd  =>  totalSupply()
a0712d68  =>  mint(uint256)
f19e75d4  =>  ownerMint(uint256)
e08823c5  =>  ownerMintSpecific(uint256)
efbd73f4  =>  mintForAddress(uint256,address)
438b6300  =>  walletOfOwner(address)
8fdcf942  =>  setPresaleCost(uint256)
6905b184  =>  setPublicsaleCost(uint256)
ae137307  =>  setNFTPublicsalePerAddressLimit(uint256)
e9058c37  =>  setNFTPresalePerAddressLimit(uint256)
7ec4a659  =>  setUriPrefix(string)
16ba10e0  =>  setUriSuffix(string)
4fdd43cb  =>  setHiddenMetadataUri(string)
16c38b3c  =>  setPaused(bool)
c54e73e3  =>  setPresale(bool)
e0a80853  =>  setRevealed(bool)
3ccfd60b  =>  withdraw()
0d43db94  =>  _mintLoop(address,uint256)
3c952764  =>  setOnlyWhitelisted(bool)
47ee0394  =>  addToWhiteList(address)
c243c6c1  =>  addArrayToWhiteList(address[])
01bf6648  =>  removeFromWhiteList(address)
96bfc229  =>  isInWhiteList(address)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/TruorDie.sol	3c7d15d8034f3c1c3e32386f5d569b0a3f4dc002

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 🔒	🛑	
TruorDie	Implementation	ERC721, Ownable		

└	<Constructor>	Public ❗	🛑	ERC721
└	totalSupply	Public ❗		NO❗
└	mint	Public ❗	💵	mintCompliance
└	ownerMint	Public ❗	🛑	onlyOwner
└	ownerMintSpecific	Public ❗	🛑	onlyOwner
└	mintForAddress	Public ❗	🛑	mintCompliance onlyOwner
└	walletOfOwner	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	setPresaleCost	Public ❗	🛑	onlyOwner
└	setPublicsaleCost	Public ❗	🛑	onlyOwner
└	setNFTPublicsalePerAddressLimit	Public ❗	🛑	onlyOwner
└	setNFTPresalePerAddressLimit	Public ❗	🛑	onlyOwner
└	setUriPrefix	Public ❗	🛑	onlyOwner
└	setUriSuffix	Public ❗	🛑	onlyOwner
└	setHiddenMetadataUri	Public ❗	🛑	onlyOwner
└	setPaused	Public ❗	🛑	onlyOwner
└	setPresale	Public ❗	🛑	onlyOwner
└	setRevealed	Public ❗	🛑	onlyOwner
└	withdraw	Public ❗	💵	NO❗
└	_mintLoop	Internal 🔒	🛑	
└	_baseURI	Internal 🔒		
└	setOnlyWhitelisted	Public ❗	🛑	onlyOwner
└	addToWhiteList	Public ❗	🛑	onlyOwner
└	addArrayToWhiteList	Public ❗	🛑	onlyOwner
└	removeFromWhiteList	Public ❗	🛑	onlyOwner
└	isInWhiteList	Public ❗		NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues after the team had fixed all critical and low 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.