Smart Contract
Security Audit
V1

theNFtreasury Smart Contract

3/11/2022

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**: 0x6102fac38d441bbe316987af49dedb84a7f0ffd8
- **Code:**

https://goerli.etherscan.io/address/0x6102fac38d441bbe316987af49dedb84a7f0ffd8#code

NFT Information

- Name: theNFtreasury
- Total Supply:
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum)

theNFtreasury smart contract on ETH test net to test every function by the auditor.

https://thirdweb.com/goerli/0x6102fAC38d441bbe316987Af49DeDB84a7f0FFD8/nfts

https://goerli.etherscan.io/address/0x6102fac38d441bbe316987af49dedb84a7f0ffd8

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

P.S: the smart contract is import thirdweb libraries https://thirdweb.com/ which is one of most secure libraries And you can check the audit report of thirdweb smart contract here: https://gateway.ipfscdn.io/ipfs/QmNgNaLwzgMxcx9r6qDvJmTFam6xxUxX7Vp8E99oRt7i74/

The files:

theNFtreasury.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
contract.sol	**3fdad492b34107d1544c7e752e96d618a682c5f3705d5c64254adb0bab086fa6**	0x6102fac38d441bbe316987af49dedb84a7f0ffd8

- Contract: contract
- Inherit: ERC721Drop
- 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**
approvedMarketplaces	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
baseExtension	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
cost	✓	Read / public	**Passed**
baseURI	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**

claimCondition	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
contractURI	✓	Read / public	**Passed**
encryptDecrypt	✓	Read / public	**Passed**
encryptedData	✓	Read / public	**Passed**
getBaseURICount	✓	Read / public	**Passed**
getBatchIdAtIndex	✓	Read / public	**Passed**
getClaimTimestamp	✓	Read / public	**Passed**
getDefaultRoyaltyInfo	✓	Read / public	**Passed**
getRevealURI	✓	Read / public	**Passed**
getRoyaltyInfoForToken	✓	Read / public	**Passed**
isEncryptedBatch	✓	Read / public	**Passed**
nextTokenIdToClaim	✓	Read / public	**Passed**
maxSupply	✓	Read / public	**Passed**
nextTokenIdToMint	✓	Read / public	**Passed**
pause	✓	Read / public	**Passed**
primarySaleRecipient	✓	Read / public	**Passed**
revealed	✓	Read / public	**Passed**
royaltyInfo	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
verifyClaimMerkleProof	✓	Read / public	**Passed**
verifyClaim	✓	Read / public	**Passed**
burn	✓	Write / public	**Passed**
claim	✓	Write / payable	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
lazyMint	✓	Write / public	**Passed**

multiCall	✓	Write / public	**Passed**
reveal	✓	Write / public	**Passed**
toggleContractPause	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
setClaimConditions	✓	Write / public	**Passed**
setCost	✓	Write / public	**Passed**
toggleContractStart	✓	Write / public	**Passed**
setBaseURI	✓	Write / public	**Passed**
setConractURI	✓	Write / public	**Passed**
setDefaultRoyaltyInfo	✓	Write / public	**Passed**
setPrimarySaleRecipient	✓	Write / public	**Passed**
setOwner	✓	Write / public	**Passed**
setMaxSupply	✓	Write / public	**Passed**
setRoyaltyInfoForToken	✓	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:

#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 contract.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. The team will use to enable optimization at 200 to avoid this issue.

#Pragam version not fixed

Description
It is a good practice to lock the solidity version for a live deployment (use 0.8.17 instead of ^0.8.0). 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: Acknowledged.

Very Low:

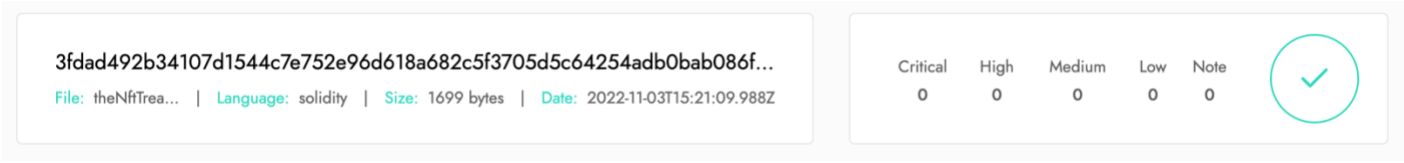
No Very Low severity vulnerabilities were found.

Notes:

No Notes were found.

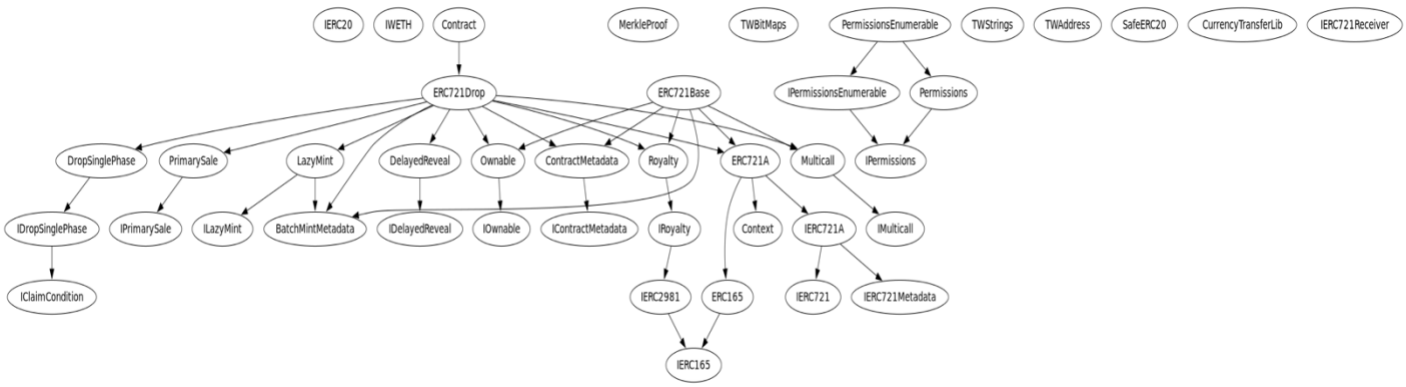
Automatic Testing

1- Check for security

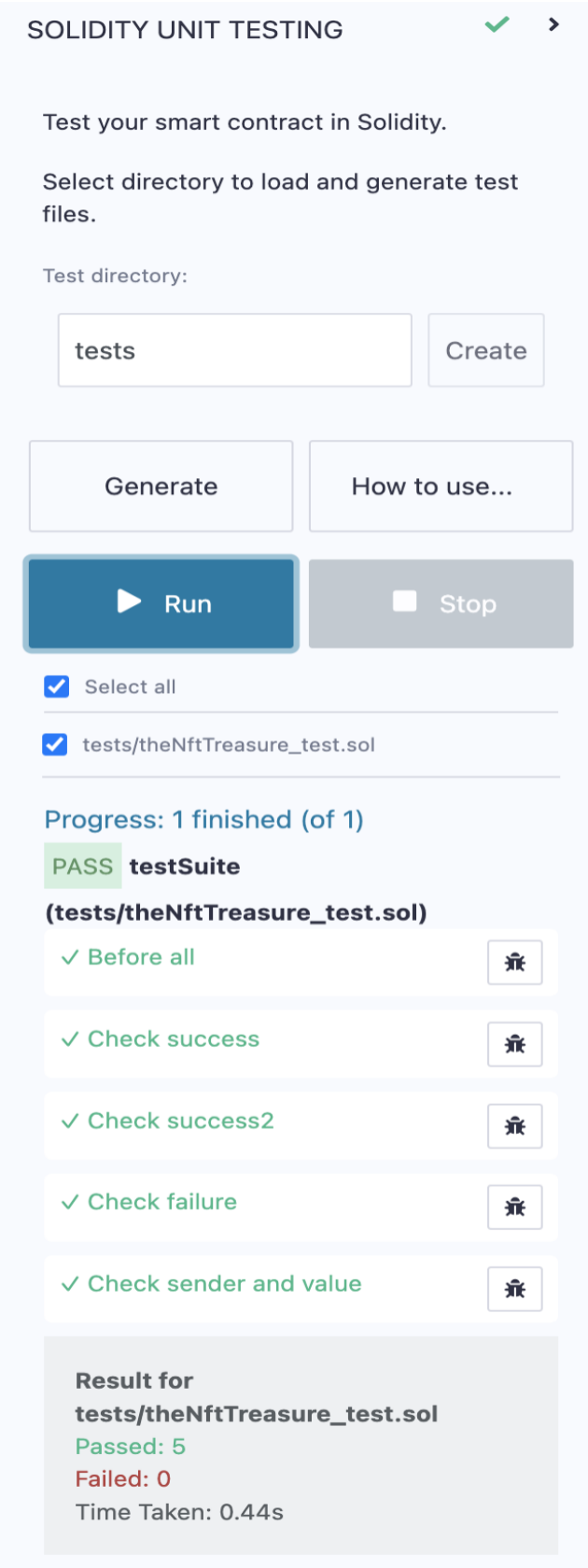


2- SOLIDITY STATIC ANALYSIS

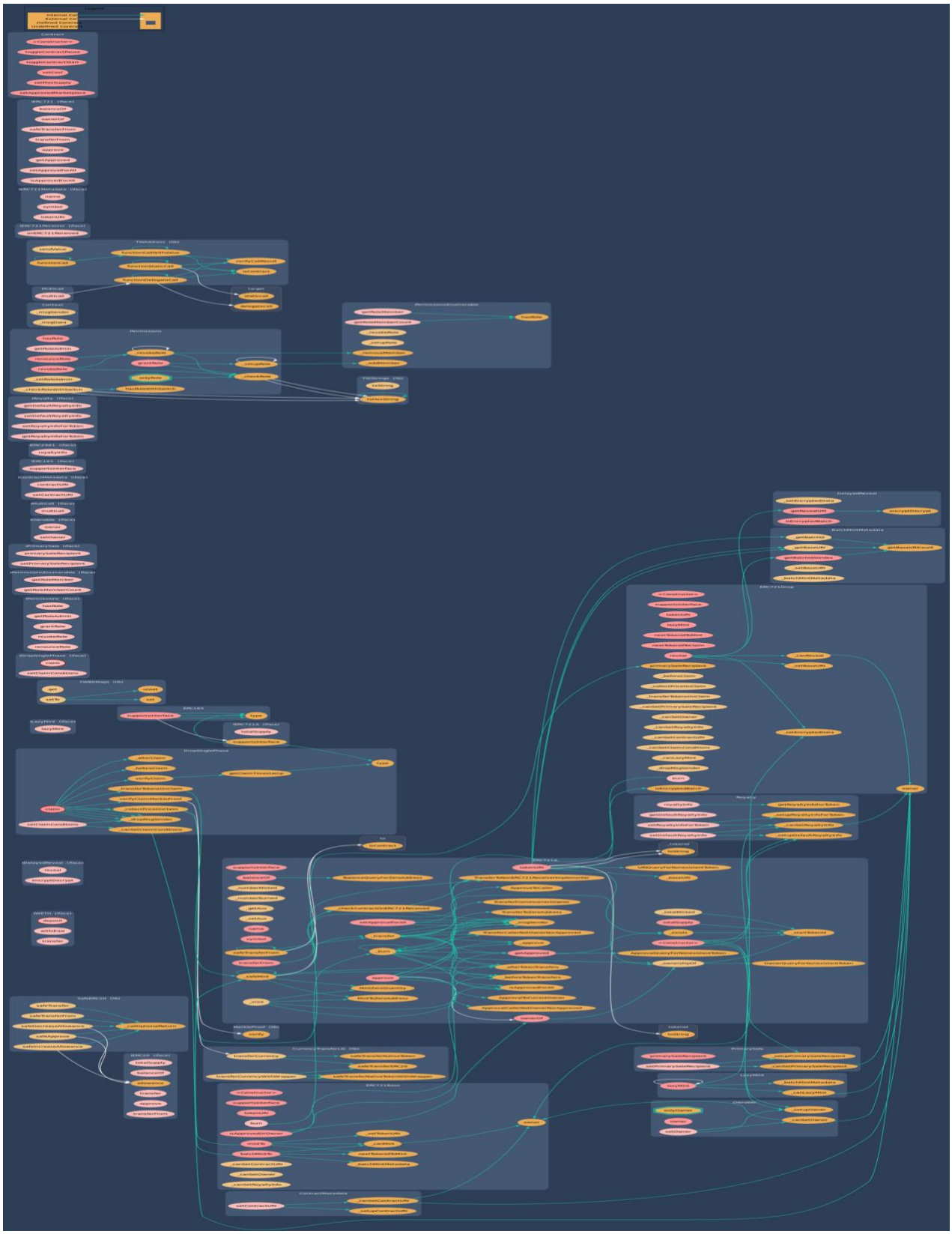
3- Inheritance graph



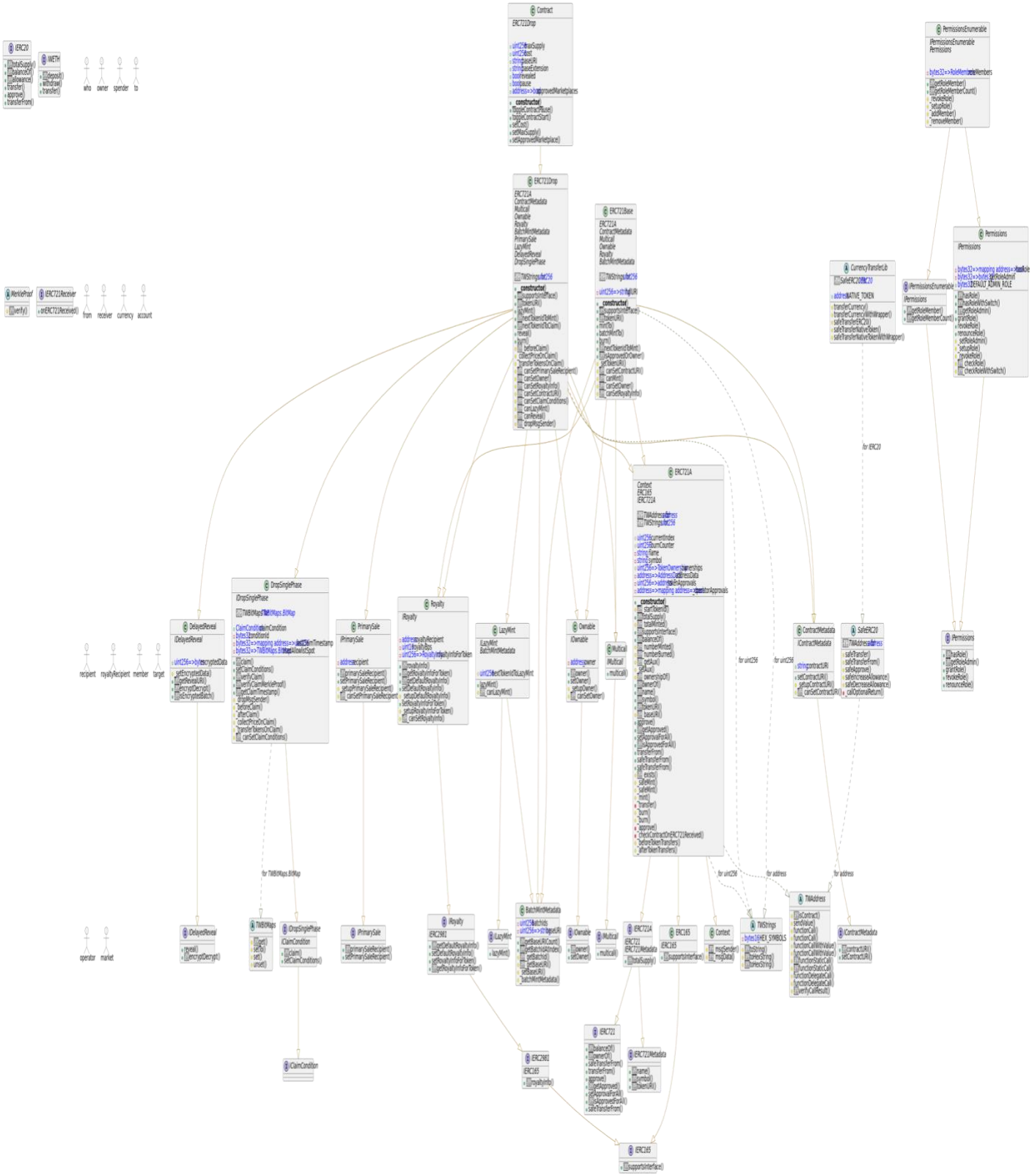
4- SOLIDITY UNIT TESTING



5- Call graph



Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
dd62ed3e  =>  allowance(address,address)
a9059cbb  =>  transfer(address,uint256)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
d0e30db0  =>  deposit()
2e1a7d4d  =>  withdraw(uint256)
ce805642  =>  reveal(uint256,bytes)
e7150322  =>  encryptDecrypt(bytes,bytes)
25eaac09  =>  _setEncryptedData(uint256,bytes)
9fc4d68f  =>  getRevealURI(uint256,bytes)
492e224b  =>  isEncryptedBatch(uint256)
d37c353b  =>  lazyMint(uint256,string,bytes)
5a9a49c7  =>  verify(bytes32[],bytes32,bytes32)
2d8ae2af  =>  get(BitMap,uint256)
68183e20  =>  setTo(BitMap,uint256,bool)
d321d5bd  =>  set(BitMap,uint256)
1c4af00b  =>  unset(BitMap,uint256)
d47b9562  =>  claim(address,uint256,address,uint256,AllowlistProof,bytes)
b6932e45  =>  setClaimConditions(ClaimCondition,bool)
a72e157d  =>  verifyClaim(address,uint256,address,uint256,bool)
e431487c  =>  verifyClaimMerkleProof(address,uint256,AllowlistProof)
b67875ce  =>  getClaimTimestamp(address)
a0242a63  =>  _dropMsgSender()
2b244303  =>  _beforeClaim(address,uint256,address,uint256,AllowlistProof,bytes)
2264aefb  =>  _afterClaim(address,uint256,address,uint256,AllowlistProof,bytes)
be9967fe  =>  _collectPriceOnClaim(address,uint256,address,uint256)
8ca3f676  =>  _transferTokensOnClaim(address,uint256)
c4b1b7c9  =>  _canSetClaimConditions()
91d14854  =>  hasRole(bytes32,address)
248a9ca3  =>  getRoleAdmin(bytes32)
2f2ff15d  =>  grantRole(bytes32,address)
d547741f  =>  revokeRole(bytes32,address)
36568abe  =>  renounceRole(bytes32,address)
9010d07c  =>  getRoleMember(bytes32,uint256)
ca15c873  =>  getRoleMemberCount(bytes32)
079fe40e  =>  primarySaleRecipient()
6f4f2837  =>  setPrimarySaleRecipient(address)
281c506b  =>  _setupPrimarySaleRecipient(address)
e585cb64  =>  _canSetPrimarySaleRecipient()
63b45e2d  =>  getBaseURICount()
2419f51b  =>  getBatchIdAtIndex(uint256)
269c228a  =>  _getBatchId(uint256)
9fe1b3d9  =>  _getBaseURI(uint256)
35574feb  =>  _setBaseURI(uint256,string)
b63fb51b  =>  _batchMintMetadata(uint256,uint256,string)
314b465b  =>  _canLazyMint()
8da5cb5b  =>  owner()
13af4035  =>  setOwner(address)
```

```
fae9bc1c  =>  _setupOwner(address)
c2158105  =>  _canSetOwner()
ac9650d8  =>  multicall(bytes[])
e8a3d485  =>  contractURI()
938e3d7b  =>  setContractURI(string)
e907c8bf  =>  _setupContractURI(string)
ba01a8af  =>  _canSetContractURI()
01ffc9a7  =>  supportsInterface(bytes4)
2a55205a  =>  royaltyInfo(uint256,uint256)
b24f2d39  =>  getDefaultRoyaltyInfo()
600dd5ea  =>  setDefaultRoyaltyInfo(address,uint256)
9bcf7a15  =>  setRoyaltyInfoForToken(uint256,address,uint256)
4cc157df  =>  getRoyaltyInfoForToken(uint256)
536a425f  =>  _setupDefaultRoyaltyInfo(address,uint256)
d2bebab6  =>  _setupRoyaltyInfoForToken(uint256,address,uint256)
1911eab7  =>  _canSetRoyaltyInfo()
6900a3ae  =>  toString(uint256)
8fba8d5c  =>  toHexString(uint256)
63e1cbea  =>  toHexString(uint256,uint256)
a32fa5b3  =>  hasRoleWithSwitch(bytes32,address)
7612997d  =>  _setRoleAdmin(bytes32,bytes32)
4fa943a6  =>  _setupRole(bytes32,address)
2c95bd23  =>  _revokeRole(bytes32,address)
5b7b2c38  =>  _checkRole(bytes32,address)
49a9f91a  =>  _checkRoleWithSwitch(bytes32,address)
07222cad  =>  _addMember(bytes32,address)
b931ea5a  =>  _removeMember(bytes32,address)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
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)
d0c407e1  =>  safeTransfer(IERC20,address,uint256)
5beae096  =>  safeTransferFrom(IERC20,address,address,uint256)
d6dcec8d  =>  safeApprove(IERC20,address,uint256)
390cc046  =>  safeIncreaseAllowance(IERC20,address,uint256)
5164ffed  =>  safeDecreaseAllowance(IERC20,address,uint256)
becc5a20  =>  _callOptionalReturn(IERC20,bytes)
31c13bd8  =>  transferCurrency(address,address,address,uint256)
02b63f89  =>  transferCurrencyWithWrapper(address,address,address,uint256,address)
557b00f3  =>  safeTransferERC20(address,address,address,uint256)
3e167aaf  =>  safeTransferNativeToken(address,uint256)
f4f0ca3e  =>  safeTransferNativeTokenWithWrapper(address,uint256,address)
150b7a02  =>  onERC721Received(address,address,uint256,bytes)
06fdde03  =>  name()
95d89b41  =>  symbol()
c87b56dd  =>  tokenURI(uint256)
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)
98995f77  =>  _startTokenId()
736bf591  =>  _totalMinted()
4d388a98  =>  _numberMinted(address)
6ba1b8d0  =>  _numberBurned(address)
f4a540c5  =>  _getAux(address)
4ff8c452  =>  _setAux(address,uint64)
fb372cf2  =>  _ownershipOf(uint256)
743976a0  =>  _baseURI()
f8e76cc0  =>  _exists(uint256)
b3e1c718  =>  _safeMint(address,uint256)
6a4f832b  =>  _safeMint(address,uint256,bytes)
4e6ec247  =>  _mint(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
9b1f9e74  =>  _burn(uint256)
834a9477  =>  _burn(uint256,bool)
f272404d  =>  _approve(address,uint256,address)
d88343e2  =>  _checkContractOnERC721Received(address,address,uint256,bytes)
ef435773  =>  _beforeTokenTransfers(address,address,uint256,uint256)
08c018f7  =>  _afterTokenTransfers(address,address,uint256,uint256)
3b1475a7  =>  nextTokenIdToMint()
acd083f8  =>  nextTokenIdToClaim()
42966c68  =>  burn(uint256)
ce2ee880  =>  _canReveal()
0075a317  =>  mintTo(address,string)
754a81d9  =>  batchMintTo(address,uint256,string,bytes)
430c2081  =>  isApprovedOrOwner(address,uint256)
01538868  =>  _setTokenURI(uint256,string)
b7b98830  =>  _canMint()
86f9c864  =>  toggleContractPause()
ce901a06  =>  toggleContractStart()
44a0d68a  =>  setCost(uint256)
6f8b44b0  =>  setMaxSupply(uint256)
bbf46b84  =>  setApprovedMarketplace(address,bool)
```

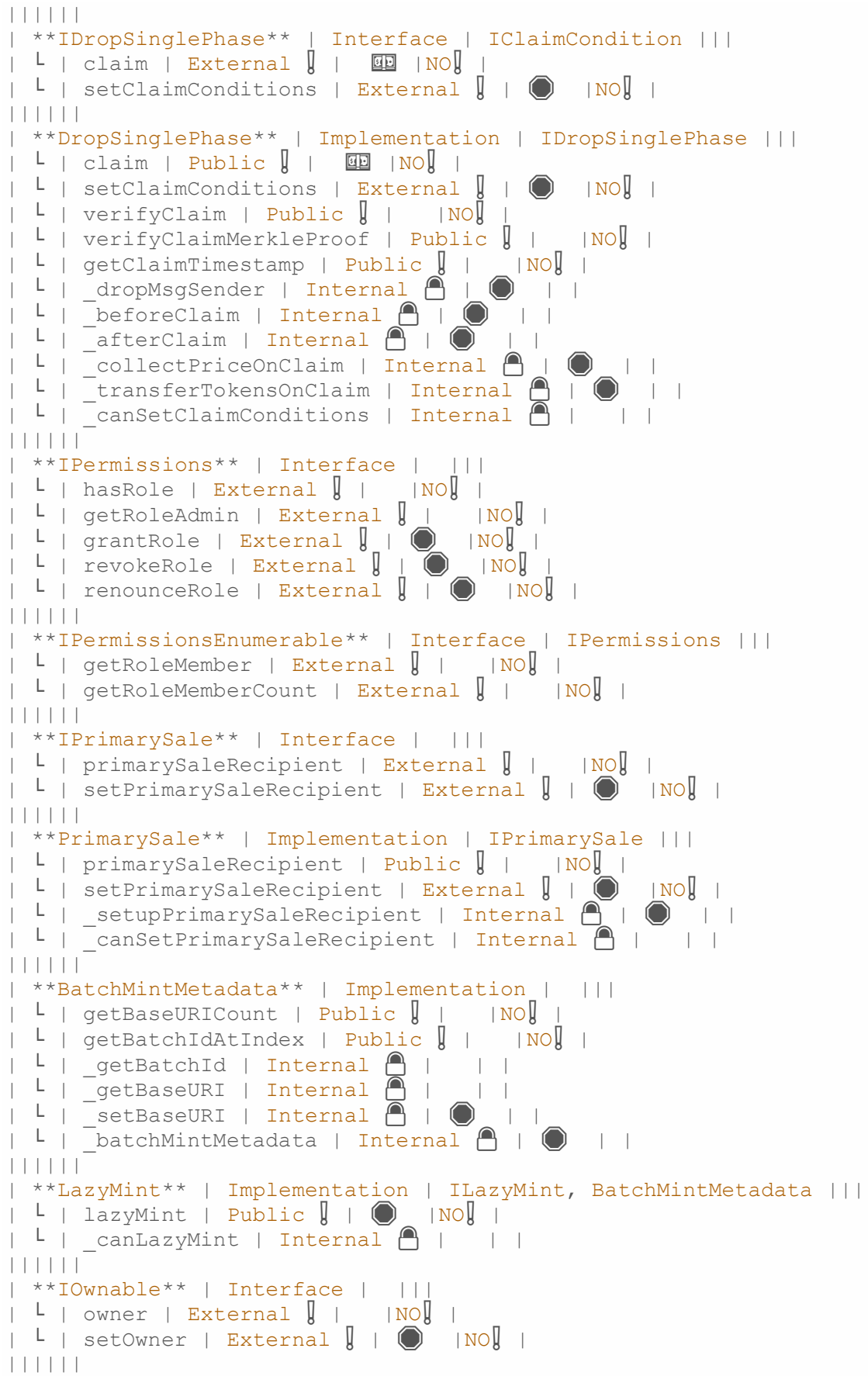
Automatic general report

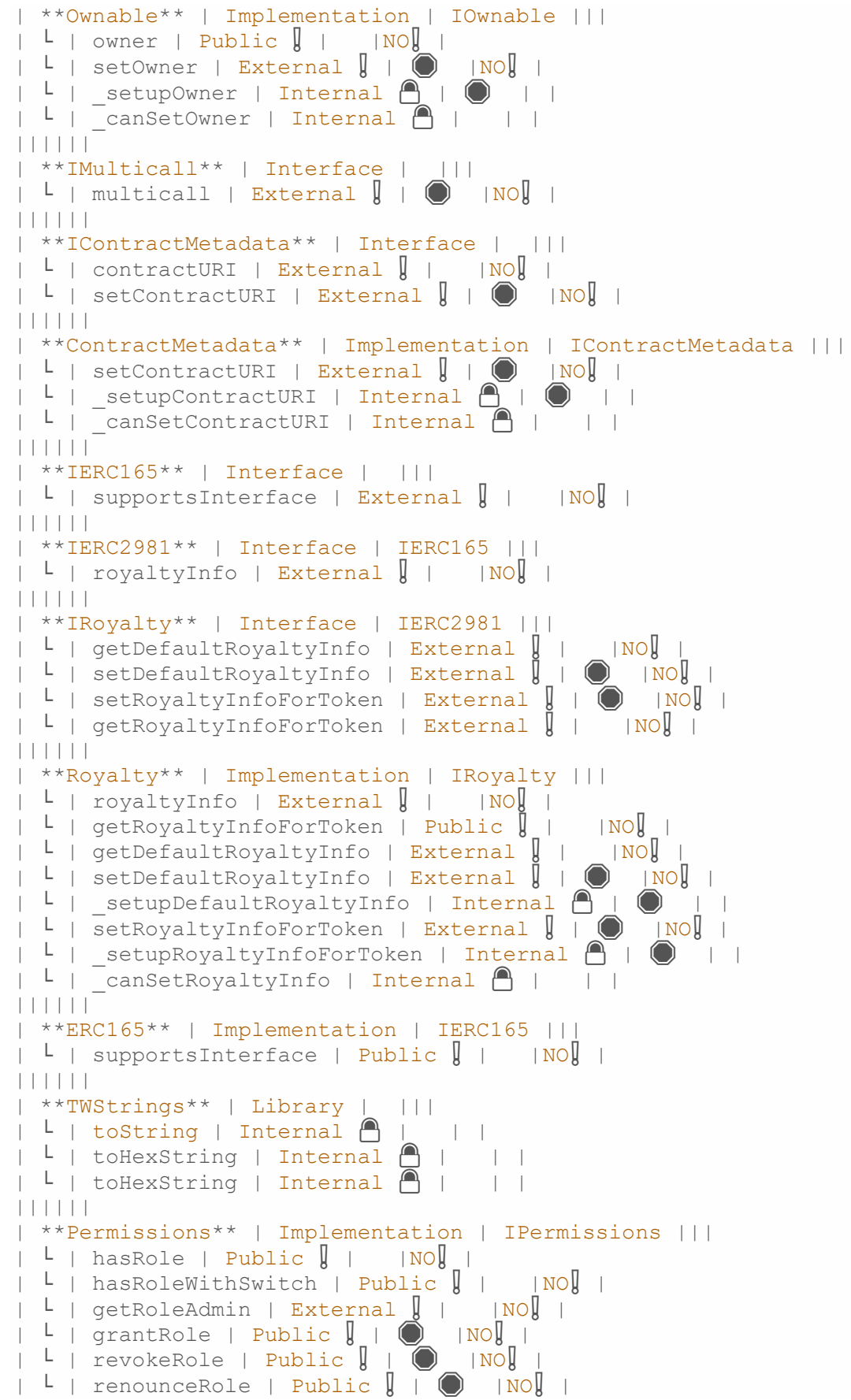
Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/theNFtreasury.sol	00466eee6d090e72e4b0e801bc0c32d46b712aa2

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IWETH	Interface			
└	deposit	External ❗	💵	NO❗
└	withdraw	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
IDelayedReveal	Interface			
└	reveal	External ❗	🛑	NO❗
└	encryptDecrypt	External ❗		NO❗
DelayedReveal	Implementation	IDelayedReveal		
└	_setEncryptedData	Internal 🔒	🛑	
└	getRevealURI	Public ❗		NO❗
└	encryptDecrypt	Public ❗		NO❗
└	isEncryptedBatch	Public ❗		NO❗
ILazyMint	Interface			
└	lazyMint	External ❗	🛑	NO❗
MerkleProof	Library			
└	verify	Internal 🔒		
TWBitMaps	Library			
└	get	Internal 🔒		
└	setTo	Internal 🔒	🛑	
└	set	Internal 🔒	🛑	
└	unset	Internal 🔒	🛑	
IClaimCondition	Interface			


IDropSinglePhase	Interface	IClaimCondition		
└	claim	External		NO	
└	setClaimConditions	External		NO	
DropSinglePhase	Implementation	IDropSinglePhase		
└	claim	Public		NO	
└	setClaimConditions	External		NO	
└	verifyClaim	Public		NO	
└	verifyClaimMerkleProof	Public		NO	
└	getClaimTimestamp	Public		NO	
└	_dropMsgSender	Internal			
└	_beforeClaim	Internal			
└	_afterClaim	Internal			
└	_collectPriceOnClaim	Internal			
└	_transferTokensOnClaim	Internal			
└	_canSetClaimConditions	Internal			
IPermissions	Interface			
└	hasRole	External		NO	
└	getRoleAdmin	External		NO	
└	grantRole	External		NO	
└	revokeRole	External		NO	
└	renounceRole	External		NO	
IPermissionsEnumerable	Interface	IPermissions		
└	getRoleMember	External		NO	
└	getRoleMemberCount	External		NO	
IPrimarySale	Interface			
└	primarySaleRecipient	External		NO	
└	setPrimarySaleRecipient	External		NO	
PrimarySale	Implementation	IPrimarySale		
└	primarySaleRecipient	Public		NO	
└	setPrimarySaleRecipient	External		NO	
└	_setupPrimarySaleRecipient	Internal			
└	_canSetPrimarySaleRecipient	Internal			
BatchMintMetadata	Implementation			
└	getBaseURICount	Public		NO	
└	getBatchIdAtIndex	Public		NO	
└	_getBatchId	Internal			
└	_getBaseURI	Internal			
└	_setBaseURI	Internal			
└	_batchMintMetadata	Internal			
LazyMint	Implementation	ILazyMint, BatchMintMetadata		
└	lazyMint	Public		NO	
└	_canLazyMint	Internal			
IOwnable	Interface			
└	owner	External		NO	
└	setOwner	External		NO	


Ownable	Implementation	IOwnable		
└	owner	Public ❗		NO❗
└	setOwner	External ❗	🛑	NO❗
└	_setupOwner	Internal 🔒	🛑	
└	_canSetOwner	Internal 🔒		
IMulticall	Interface				
└	multicall	External ❗	🛑	NO❗
IContractMetadata	Interface				
└	contractURI	External ❗		NO❗
└	setContractURI	External ❗	🛑	NO❗
ContractMetadata	Implementation	IContractMetadata		
└	setContractURI	External ❗	🛑	NO❗
└	_setupContractURI	Internal 🔒	🛑	
└	_canSetContractURI	Internal 🔒		
IERC165	Interface				
└	supportsInterface	External ❗		NO❗
IERC2981	Interface	IERC165		
└	royaltyInfo	External ❗		NO❗
IRoyalty	Interface	IERC2981		
└	getDefaultRoyaltyInfo	External ❗		NO❗
└	setDefaultRoyaltyInfo	External ❗	🛑	NO❗
└	setRoyaltyInfoForToken	External ❗	🛑	NO❗
└	getRoyaltyInfoForToken	External ❗		NO❗
Royalty	Implementation	IRoyalty		
└	royaltyInfo	External ❗		NO❗
└	getRoyaltyInfoForToken	Public ❗		NO❗
└	getDefaultRoyaltyInfo	External ❗		NO❗
└	setDefaultRoyaltyInfo	External ❗	🛑	NO❗
└	_setupDefaultRoyaltyInfo	Internal 🔒	🛑	
└	setRoyaltyInfoForToken	External ❗	🛑	NO❗
└	_setupRoyaltyInfoForToken	Internal 🔒	🛑	
└	_canSetRoyaltyInfo	Internal 🔒		
ERC165	Implementation	IERC165		
└	supportsInterface	Public ❗		NO❗
TWStrings	Library				
└	toString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
Permissions	Implementation	IPermissions		
└	hasRole	Public ❗		NO❗
└	hasRoleWithSwitch	Public ❗		NO❗
└	getRoleAdmin	External ❗		NO❗
└	grantRole	Public ❗	🛑	NO❗
└	revokeRole	Public ❗	🛑	NO❗
└	renounceRole	Public ❗	🛑	NO❗

└	_setRoleAdmin	Internal 🔒	🛑		
└	_setupRole	Internal 🔒	🛑		
└	_revokeRole	Internal 🔒	🛑		
└	_checkRole	Internal 🔒			
└	_checkRoleWithSwitch	Internal 🔒			
PermissionsEnumerable	Implementation	IPermissionsEnumerable, Permissions		
└	getRoleMember	External ❗			NO❗
└	getRoleMemberCount	External ❗			NO❗
└	_revokeRole	Internal 🔒	🛑		
└	_setupRole	Internal 🔒	🛑		
└	_addMember	Internal 🔒	🛑		
└	_removeMember	Internal 🔒	🛑		
Context	Implementation			
└	_msgSender	Internal 🔒			
└	_msgData	Internal 🔒			
TWAddress	Library			
└	isContract	Internal 🔒			
└	sendValue	Internal 🔒	🛑		
└	functionCall	Internal 🔒	🛑		
└	functionCall	Internal 🔒	🛑		
└	functionCallWithValue	Internal 🔒	🛑		
└	functionCallWithValue	Internal 🔒	🛑		
└	functionStaticCall	Internal 🔒			
└	functionStaticCall	Internal 🔒			
└	functionDelegateCall	Internal 🔒	🛑		
└	functionDelegateCall	Internal 🔒	🛑		
└	verifyCallResult	Internal 🔒			
SafeERC20	Library			
└	safeTransfer	Internal 🔒	🛑		
└	safeTransferFrom	Internal 🔒	🛑		
└	safeApprove	Internal 🔒	🛑		
└	safeIncreaseAllowance	Internal 🔒	🛑		
└	safeDecreaseAllowance	Internal 🔒	🛑		
└	_callOptionalReturn	Private 🔐	🛑		
CurrencyTransferLib	Library			
└	transferCurrency	Internal 🔒	🛑		
└	transferCurrencyWithWrapper	Internal 🔒	🛑		
└	safeTransferERC20	Internal 🔒	🛑		
└	safeTransferNativeToken	Internal 🔒	🛑		
└	safeTransferNativeTokenWithWrapper	Internal 🔒	🛑		
Multicall	Implementation	IMulticall	
└	multicall	External ❗	🛑		NO❗
IERC721Receiver	Interface			
└	onERC721Received	External ❗	🛑		NO❗
IERC721Metadata	Interface			
└	name	External ❗			NO❗

└	symbol	External ❗		NO❗
└	tokenURI	External ❗		NO❗
IERC721	Interface				
└	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❗
IERC721A	Interface	IERC721, IERC721Metadata		
└	totalSupply	External ❗		NO❗
ERC721A	Implementation	Context, ERC165, IERC721A		
└	<Constructor>	Public ❗	🛑	NO❗
└	_startTokenId	Internal 🔒			
└	totalSupply	Public ❗		NO❗
└	_totalMinted	Internal 🔒			
└	supportsInterface	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	_numberMinted	Internal 🔒			
└	_numberBurned	Internal 🔒			
└	_getAux	Internal 🔒			
└	_setAux	Internal 🔒	🛑		
└	_ownershipOf	Internal 🔒			
└	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❗
└	_exists	Internal 🔒			
└	_safeMint	Internal 🔒	🛑		
└	_safeMint	Internal 🔒	🛑		
└	_mint	Internal 🔒	🛑		
└	_transfer	Private 🔐	🛑		
└	_burn	Internal 🔒	🛑		
└	_burn	Internal 🔒	🛑		
└	_approve	Private 🔐	🛑		
└	_checkContractOnERC721Received	Private 🔐	🛑		
└	_beforeTokenTransfers	Internal 🔒	🛑		
└	_afterTokenTransfers	Internal 🔒	🛑		
ERC721Drop	Implementation	ERC721A, ContractMetadata, Multicall, Ownable, Royalty, BatchMintMetadata, PrimarySale, LazyMint, DelayedReveal, DropSinglePhase

└	<Constructor>	Public ❗	🛑	ERC721A
└	supportsInterface	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	lazyMint	Public ❗	🛑	NO❗
└	nextTokenIdToMint	Public ❗		NO❗
└	nextTokenIdToClaim	Public ❗		NO❗
└	reveal	Public ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	_beforeClaim	Internal 🔒		
└	_collectPriceOnClaim	Internal 🔒	🛑	
└	_transferTokensOnClaim	Internal 🔒	🛑	
└	_canSetPrimarySaleRecipient	Internal 🔒		
└	_canSetOwner	Internal 🔒		
└	_canSetRoyaltyInfo	Internal 🔒		
└	_canSetContractURI	Internal 🔒		
└	_canSetClaimConditions	Internal 🔒		
└	_canLazyMint	Internal 🔒		
└	_canReveal	Internal 🔒		
└	_dropMsgSender	Internal 🔒		
ERC721Base	Implementation	ERC721A, ContractMetadata, Multicall, Ownable, Royalty, BatchMintMetadata		
└	<Constructor>	Public ❗	🛑	ERC721A
└	supportsInterface	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	mintTo	Public ❗	🛑	NO❗
└	batchMintTo	Public ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	nextTokenIdToMint	Public ❗		NO❗
└	isApprovedOrOwner	Public ❗		NO❗
└	_setTokenURI	Internal 🔒	🛑	
└	_canSetContractURI	Internal 🔒		
└	_canMint	Internal 🔒		
└	_canSetOwner	Internal 🔒		
└	_canSetRoyaltyInfo	Internal 🔒		
Contract	Implementation	ERC721Drop		
└	<Constructor>	Public ❗	🛑	ERC721Drop
└	toggleContractPause	Public ❗	🛑	onlyOwner
└	toggleContractStart	Public ❗	🛑	onlyOwner
└	setCost	Public ❗	🛑	onlyOwner
└	setMaxSupply	Public ❗	🛑	onlyOwner
└	setApprovedMarketplace	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.

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.