Smart Contract Security Audit V1

Doka Smart Contract Audit

April 22, 2023

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**: 0xc171be42ec8d2d515346b86189ce03bfe7d165df
- **Code:** https://github.com/Saferico/Smart-Contracts-for-Projects/blob/main/Doka.sol

NFT Information

- Name: Doka
- symbol: DOKA
- MAX Supply: 5555 (100 reserved for the team - 5455 available for reservation)
- Price: 0.088 ETH
- phases:

Phase 1 (Whitelist)

- Both public & Whitelist users can reserve tokens (2 max for each wallet).

- Whitelist have priority, capped at 5455.

- Public can only reserve if there are available spots left.

- e.g., if 5055 are reserved by Whitelist, only 400 can be reserved by the public.

Phase 2 (public)

- Reservation open to public.

- Public can only reserve if there are available spots left.

P.S: The airdrop stage after reservations phases is finished to distribute tokens.

Contracts address deployed to test net (Ethereum)

Doka smart contract on Ethereum test net to test every function by the auditor.

https://sepolia.etherscan.io/address/0xc171be42ec8d2d515346b86189ce03bfe7d165df

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:

Doka.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Doka.sol	**ca821ab65fa1a9a9ace4b0aad612562a6d5bb4a3**	0xc171be42ec8d2d515346b86189ce03bfe7d165df

- Contract: Doka
- Inherit: ERC721A, Ownable, ERC2981, OperatorFilterer
- 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**
mintEnabled	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
operatorFilteringEnabled	✓	Read / public	**Passed**
publicReserveCounter	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
publicReserveAddress	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

ownerOf	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
phase	✓	Read / public	**Passed**
reserveCounter	✓	Read / public	**Passed**
reservePrice	✓	Read / public	**Passed**
royaltyInfo	✓	Read / public	**Passed**
signer	✓	Read / public	**Passed**
totalReserveCounter	✓	Read / public	**Passed**
wlReserveCounter	✓	Read / public	**Passed**
wlReserveAdresses	✓	Read / public	**Passed**
reserve	✓	Write / payable	**Passed**
approve	✓	Write / payable	**Passed**
safeTransferFrom	✓	Write / payable	**Passed**
safeTransferFrom	✓	Write / payable	**Passed**
setOperatorFilteringEnabled	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
deleteDefaultRoyalty	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / payable	**Passed**
stopMint	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
airdrop	✓	Write / public	**Passed**
setDefaultRoyalty	✓	Write / public	**Passed**
setBaseURI	✓	Write / public	**Passed**

setReservePrice	✓	Write / public	**Passed**
setPhase	✓	Write / public	**Passed**
setSigner	✓	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:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.19 instead of ^0.8.17). 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 version 2.

#Missing zero address validation

Description

When the owner wants to airdrop some NFT to investors, and when he wants to change signer address, he has to check for the zero address to make, he didn't add the zero address. Otherwise, the airdrop for address function will act like the burn function, and make burn address as signer.

```
constructor(address signer_) ERC721A("Doka", "DOKA") {
        // initial states
        $signer = signer_;
    }
function setDefaultRoyalty(address receiver, uint96 feeNumerator) public onlyOwner
{ _setDefaultRoyalty(receiver, feeNumerator); }

function setSigner(address signer_) external onlyOwner {
        $signer = signer_; }
```

```
struct Holder {
        address addr;
        uint256 amount;}

    function airdrop(Holder[] calldata holders_) external onlyOwner {
        if (!$mintEnabled) {
            revert ErrMintDisabled();
        } for (uint256 i = 0; i < holders_.length;) {
            Holder memory __holder = holders_[i];
            _mint(__holder.addr, __holder.amount);
            unchecked {
                ++i;}}
        if (_totalMinted() > MAX_SUPPLY) {
            revert ErrExceedsSupply();
        }
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version 2.

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

Description

The owner can airdrop NFT to any address.
The owner can stop minting at any time.
The owner can change the price at any time.
The owner can change the royalty fees or delete it.

```
function setDefaultRoyalty(address receiver, uint96 feeNumerator) public onlyOwner
{
        require(receiver != address (0),"can't add zero address");
        _setDefaultRoyalty(receiver, feeNumerator);
    }

    function deleteDefaultRoyalty() public onlyOwner {
        _deleteDefaultRoyalty();
    }
function setReservePrice(uint256 reservePrice_) external onlyOwner {
        $reservePrice = reservePrice_;
    }
function stopMint() external onlyOwner {
        $mintEnabled = false;
    }

function airdrop(Holder[] calldata holders_) external onlyOwner {
        if (!$mintEnabled) {
            revert ErrMintDisabled();
        } for (uint256 i = 0; i < holders_.length;) {
            Holder memory __holder = holders_[i];
            _mint(__holder.addr, __holder.amount);
            unchecked {
```

```
            ++i;}}
        if (_totalMinted() > MAX_SUPPLY) {
            revert ErrExceedsSupply();
        }
```

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.

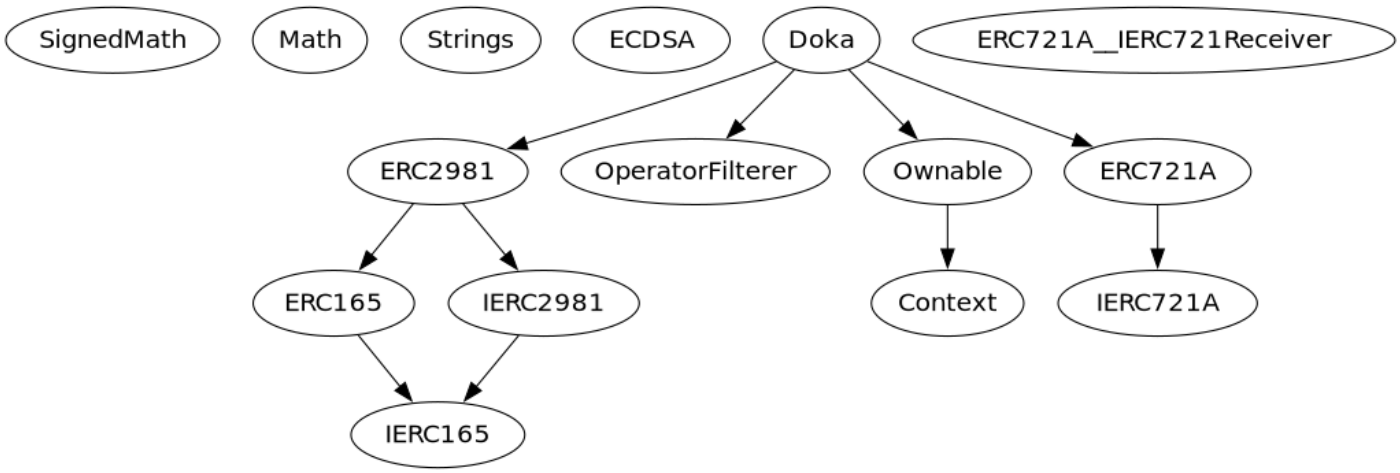
Notes:

No Notes vulnerabilities were found.

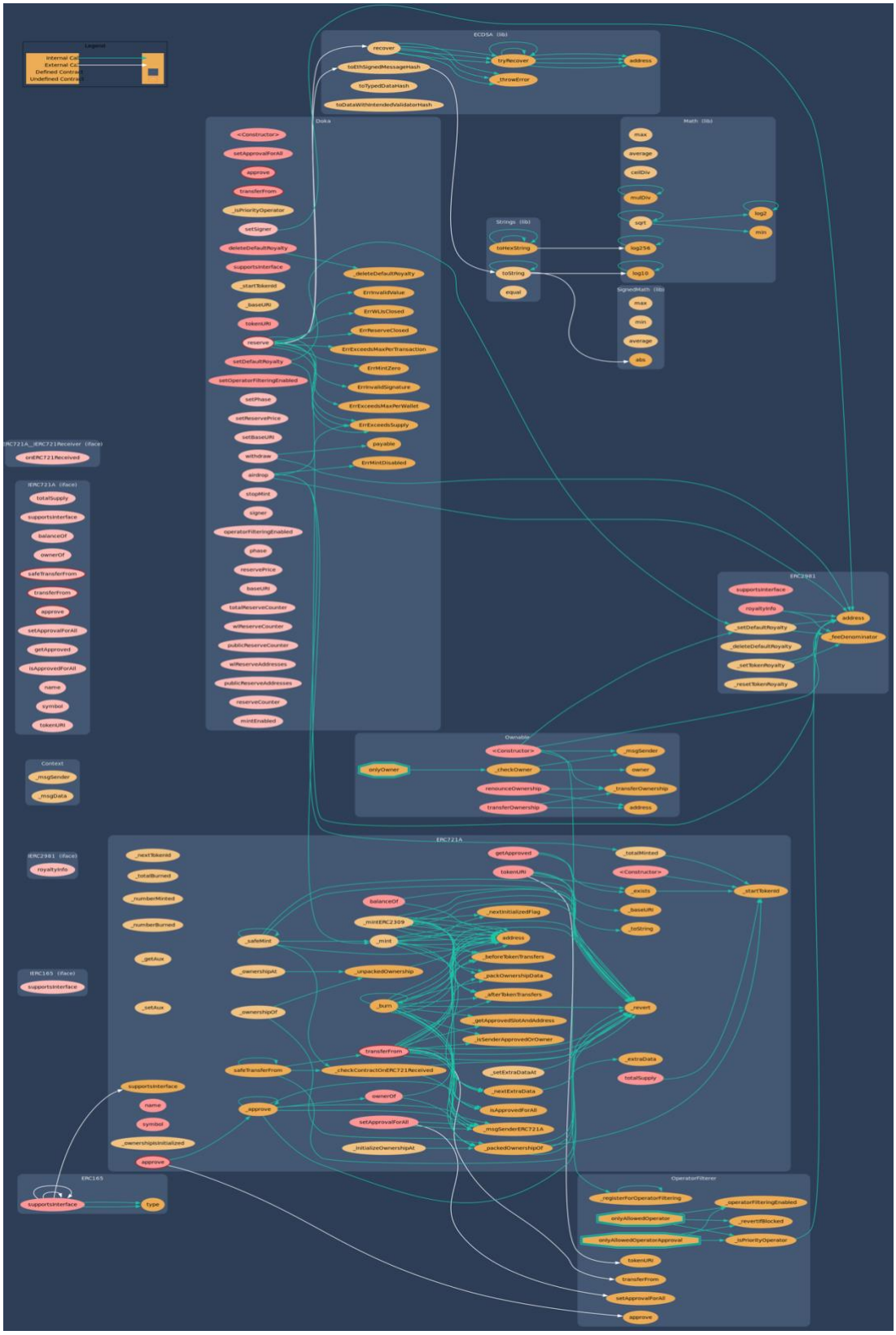
Automatic Testing

1- SOLIDITY STATIC ANALYSIS

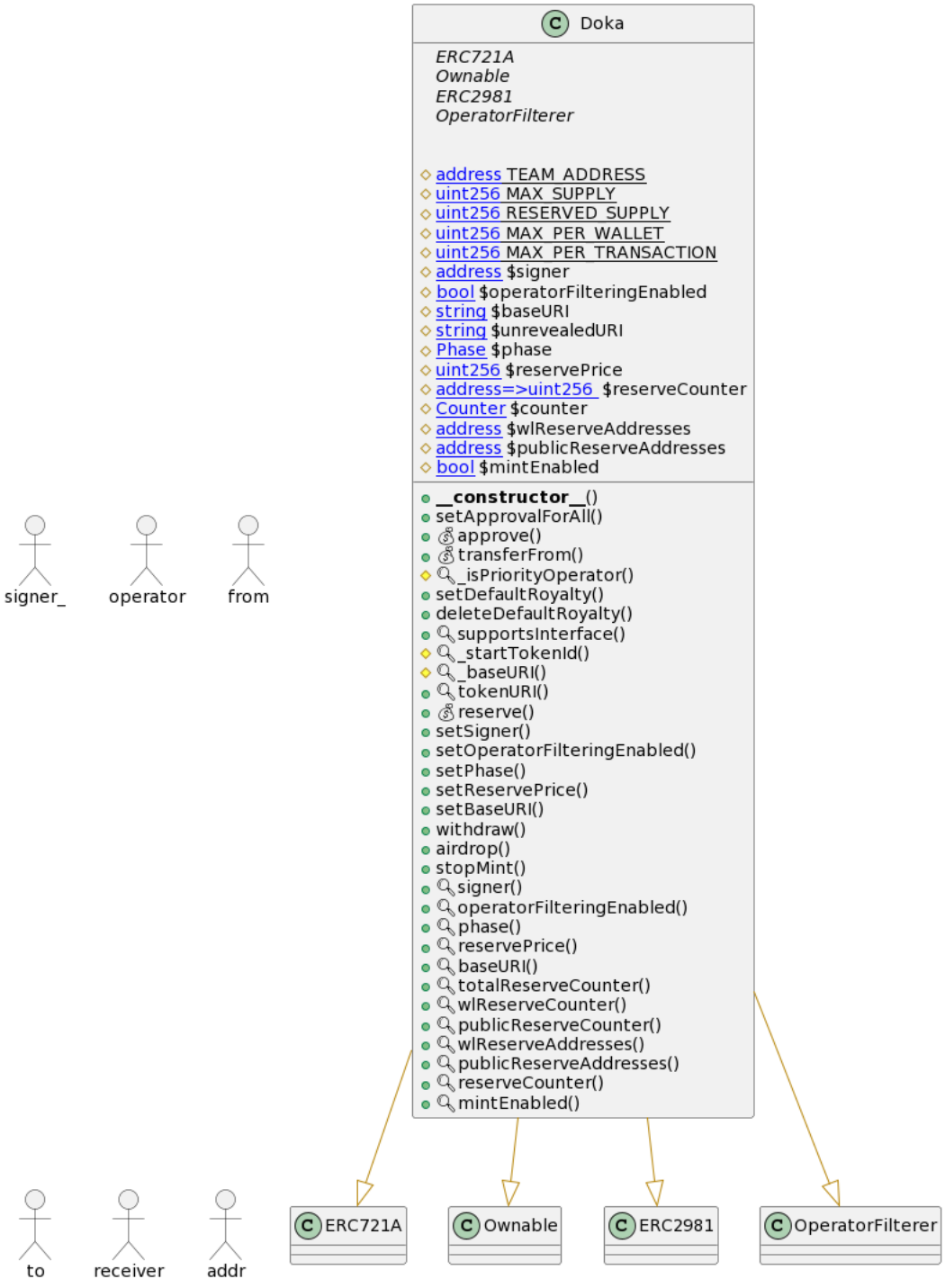
2- Inheritance graph



3- Call graph


Legend
ECDSA (lib)
recover
tryRecover
address
toEthSignedMessageHash
_throwError
toTypedDataHash
toDataWithIntendedValidatorHash
Doka
<Constructor>
setApprovalForAll
approve
transferFrom
_isPriorityOperator
setSigner
deleteDefaultRoyalty
supportsInterface
_startTokenId
_baseURI
tokenURI
reserve
setDefaultRoyalty
setOperatorFilteringEnabled
setPhase
setReservePrice
setBaseURI
withdraw
airdrop
stopMint
signer
operatorFilteringEnabled
phase
reservePrice
baseURI
totalReserveCounter
wlReserveCounter
publicReserveCounter
wlReserveAddresses
publicReserveAddresses
reserveCounter
mintEnabled
_deleteDefaultRoyalty
ErrInvalidValue
ErrWLIsClosed
ErrReserveClosed
ErrExceedsMaxPerTransaction
ErrMintZero
ErrInvalidSignature
ErrExceedsMaxPerWallet
ErrExceedsSupply
payable
ErrMintDisabled
Math (lib)
max
average
ceilDiv
mulDiv
sqrt
log2
min
Strings (lib)
toHexString
log256
toString
log10
equal
SignedMath (lib)
max
min
average
abs
IERC721A__IERC721Receiver (iface)
onERC721Received
IERC721A (iface)
totalSupply
supportsInterface
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
setApprovalForAll
getApproved
isApprovedForAll
name
symbol
tokenURI
ERC2981
supportsInterface
royaltyInfo
_setDefaultRoyalty
_deleteDefaultRoyalty
_setTokenRoyalty
_resetTokenRoyalty
address
feeDenominator
Context
_msgSender
_msgData
Ownable
<Constructor>
onlyOwner
_checkOwner
renounceOwnership
transferOwnership
_msgSender
owner
_transferOwnership
address
ERC721A
IERC2981 (iface)
royaltyInfo
IERC165 (iface)
supportsInterface
_nextTokenId
_totalBurned
_numberMinted
_numberBurned
_getAux
_setAux
supportsInterface
name
symbol
_ownershipsInitialized
approve
balanceOf
_safeMint
_ownershipAt
_ownershipOf
transferFrom
safeTransferFrom
_approve
ownerOf
setApprovalForAll
_initializeOwnershipAt
_mintERC2309
_mint
_unpackedOwnership
_burn
_checkContractOnERC721Received
getApproved
tokenURI
_nextInitializedFlag
address
_beforeTokenTransfers
_packOwnershipData
_afterTokenTransfers
_getApprovedSlotAndAddress
_isSenderApprovedOrOwner
_setExtraDataAt
_nextExtraData
isApprovedForAll
_msgSenderERC721A
_packedOwnershipOf
totalMinted
<Constructor>
_exists
_baseURI
_toString
_startTokenId
_revert
_extraData
totalSupply
ERC165
supportsInterface
type
OperatorFilterer
_registerForOperatorFiltering
onlyAllowedOperator
onlyAllowedOperatorApproval
_operatorFilteringEnabled
_revertIfBlocked
_isPriorityOperator
tokenURI
transferFrom
setApprovalForAll
approve

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
81fe5786  =>  max(int256,int256)
29aa9cbe  =>  min(int256,int256)
7f0bb292  =>  average(int256,int256)
1b5ac4b5  =>  abs(int256)
6d5433e6  =>  max(uint256,uint256)
7ae2b5c7  =>  min(uint256,uint256)
2b7423ab  =>  average(uint256,uint256)
9cb35327  =>  ceilDiv(uint256,uint256)
aa9a0912  =>  mulDiv(uint256,uint256,uint256)
1db78456  =>  mulDiv(uint256,uint256,uint256,Rounding)
677342ce  =>  sqrt(uint256)
a902bc5e  =>  sqrt(uint256,Rounding)
5456bf13  =>  log2(uint256)
2ee6af53  =>  log2(uint256,Rounding)
ebdae5f9  =>  log10(uint256)
f86799ff  =>  log10(uint256,Rounding)
36cb4c48  =>  log256(uint256)
2910b3a1  =>  log256(uint256,Rounding)
6900a3ae  =>  toString(uint256)
a322c40e  =>  toString(int256)
8fba8d5c  =>  toHexString(uint256)
63e1cbea  =>  toHexString(uint256,uint256)
1bb0c665  =>  toHexString(address)
46bdca9a  =>  equal(string,string)
5e2ffa14  =>  _throwError(RecoverError)
c6edd8a7  =>  tryRecover(bytes32,bytes)
19045a25  =>  recover(bytes32,bytes)
628f98cc  =>  tryRecover(bytes32,bytes32,bytes32)
bf2fe7fd  =>  recover(bytes32,bytes32,bytes32)
4d78da76  =>  tryRecover(bytes32,uint8,bytes32,bytes32)
c2bf17b0  =>  recover(bytes32,uint8,bytes32,bytes32)
918a15cf  =>  toEthSignedMessageHash(bytes32)
92bd87b5  =>  toEthSignedMessageHash(bytes)
7df7a71c  =>  toTypedDataHash(bytes32,bytes32)
2d9bc32a  =>  toDataWithIntendedValidatorHash(address,bytes)
01ffc9a7  =>  supportsInterface(bytes4)
2a55205a  =>  royaltyInfo(uint256,uint256)
bf8e572e  =>  _feeDenominator()
b1c1ab1b  =>  _setDefaultRoyalty(address,uint96)
36fdc63c  =>  _deleteDefaultRoyalty()
b552a471  =>  _setTokenRoyalty(uint256,address,uint96)
604ba39e  =>  _resetTokenRoyalty(uint256)
72dfb778  =>  _registerForOperatorFiltering()
ee43f83f  =>  _registerForOperatorFiltering(address,bool)
9771b304  =>  _revertIfBlocked(address)
b040adf6  =>  _operatorFilteringEnabled()
fb645b3b  =>  _isPriorityOperator(address)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
```

```
8da5cb5b  =>  owner()
53a72975  =>  _checkOwner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
18160ddd  =>  totalSupply()
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)
743976a0  =>  _baseURI()
fb372cf2  =>  _ownershipOf(uint256)
cbaf28ce  =>  _ownershipAt(uint256)
f5d51a6b  =>  _ownershipIsInitialized(uint256)
f2d31624  =>  _initializeOwnershipAt(uint256)
444996c1  =>  _packedOwnershipOf(uint256)
4fe3c13e  =>  _unpackedOwnership(uint256)
bf460657  =>  _packOwnershipData(address,uint256)
e0e30f80  =>  _nextInitializedFlag(uint256)
f8e76cc0  =>  _exists(uint256)
848b8eac  =>  _isSenderApprovedOrOwner(address,address,address)
f112a2af  =>  _getApprovedSlotAndAddress(uint256)
ef435773  =>  _beforeTokenTransfers(address,address,uint256,uint256)
08c018f7  =>  _afterTokenTransfers(address,address,uint256,uint256)
d88343e2  =>  _checkContractOnERC721Received(address,address,uint256,bytes)
4e6ec247  =>  _mint(address,uint256)
4908d13b  =>  _mintERC2309(address,uint256)
6a4f832b  =>  _safeMint(address,uint256,bytes)
b3e1c718  =>  _safeMint(address,uint256)
7b7d7225  =>  _approve(address,uint256)
4775c8cf  =>  _approve(address,uint256,bool)
9b1f9e74  =>  _burn(uint256)
834a9477  =>  _burn(uint256,bool)
bd3cdd6d  =>  _setExtraDataAt(uint256,uint24)
fc37bbd3  =>  _extraData(address,address,uint24)
5afe32e4  =>  _nextExtraData(address,address,uint256)
b60986df  =>  _msgSenderERC721A()
f832e238  =>  _toString(uint256)
20e114b2  =>  _revert(bytes4)
```

```
04634d8d  =>  setDefaultRoyalty(address,uint96)
aa1b103f  =>  deleteDefaultRoyalty()
1b321fe2  =>  reserve(uint256,bool,bytes)
6c19e783  =>  setSigner(address)
b7c0b8e8  =>  setOperatorFilteringEnabled(bool)
ed4ac487  =>  setPhase(Phase)
ce9c7c0d  =>  setReservePrice(uint256)
55f804b3  =>  setBaseURI(string)
3ccfd60b  =>  withdraw()
70e67a92  =>  airdrop(Holder[])
d5582965  =>  stopMint()
238ac933  =>  signer()
fb796e6c  =>  operatorFilteringEnabled()
b1c9fe6e  =>  phase()
db2e1eed  =>  reservePrice()
6c0360eb  =>  baseURI()
8bb2f74d  =>  totalReserveCounter()
2d251f90  =>  wlReserveCounter()
10f1bb36  =>  publicReserveCounter()
bfe9517b  =>  wlReserveAddresses()
7a90f77e  =>  publicReserveAddresses()
2456c1ac  =>  reserveCounter(address)
d1239730  =>  mintEnabled()
```

Automatic general report

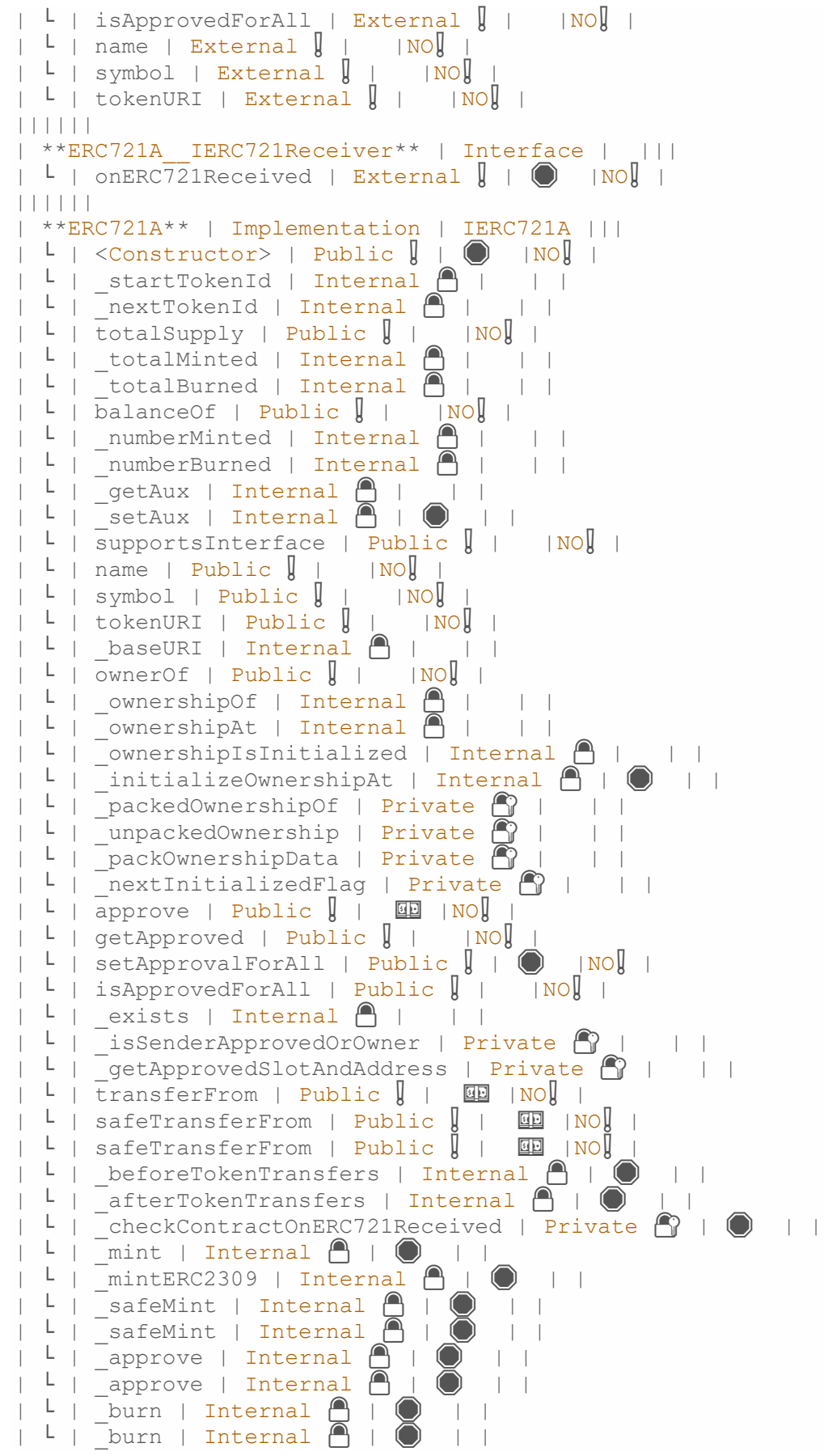
Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Doka.sol	ca821ab65fa1a9a9ace4b0aad612562a6d5bb4a3

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
SignedMath	Library			
└	max	Internal 🔒		
└	min	Internal 🔒		
└	average	Internal 🔒		
└	abs	Internal 🔒		
Math	Library			
└	max	Internal 🔒		
└	min	Internal 🔒		
└	average	Internal 🔒		
└	ceilDiv	Internal 🔒		
└	mulDiv	Internal 🔒		
└	mulDiv	Internal 🔒		
└	sqrt	Internal 🔒		
└	sqrt	Internal 🔒		
└	log2	Internal 🔒		
└	log2	Internal 🔒		
└	log10	Internal 🔒		
└	log10	Internal 🔒		
└	log256	Internal 🔒		
└	log256	Internal 🔒		
Strings	Library			
└	toString	Internal 🔒		
└	toString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
└	equal	Internal 🔒		
ECDSA	Library			
└	_throwError	Private 🔐		
└	tryRecover	Internal 🔒		
└	recover	Internal 🔒		
└	tryRecover	Internal 🔒		
└	recover	Internal 🔒		

└	tryRecover	Internal 🔒		
└	recover	Internal 🔒		
└	toEthSignedMessageHash	Internal 🔒		
└	toEthSignedMessageHash	Internal 🔒		
└	toTypedDataHash	Internal 🔒		
└	toDataWithIntendedValidatorHash	Internal 🔒		
IERC165	Interface			
└	supportsInterface	External ❗		NO❗
ERC165	Implementation	IERC165		
└	supportsInterface	Public ❗		NO❗
IERC2981	Interface	IERC165		
└	royaltyInfo	External ❗		NO❗
ERC2981	Implementation	IERC2981, ERC165		
└	supportsInterface	Public ❗		NO❗
└	royaltyInfo	Public ❗		NO❗
└	_feeDenominator	Internal 🔒		
└	_setDefaultRoyalty	Internal 🔒	🛑	
└	_deleteDefaultRoyalty	Internal 🔒	🛑	
└	_setTokenRoyalty	Internal 🔒	🛑	
└	_resetTokenRoyalty	Internal 🔒	🛑	
OperatorFilterer	Implementation			
└	_registerForOperatorFiltering	Internal 🔒	🛑	
└	_registerForOperatorFiltering	Internal 🔒	🛑	
└	_revertIfBlocked	Private 🔐		
└	_operatorFilteringEnabled	Internal 🔒		
└	_isPriorityOperator	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
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 🔒		
└	balanceOf	Public ❗		NO❗
└	_numberMinted	Internal 🔒		
└	_numberBurned	Internal 🔒		
└	_getAux	Internal 🔒		
└	_setAux	Internal 🔒	🛑	
└	supportsInterface	Public ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	_baseURI	Internal 🔒		
└	ownerOf	Public ❗		NO❗
└	_ownershipOf	Internal 🔒		
└	_ownershipAt	Internal 🔒		
└	_ownershipIsInitialized	Internal 🔒		
└	_initializeOwnershipAt	Internal 🔒	🛑	
└	_packedOwnershipOf	Private 🔐		
└	_unpackedOwnership	Private 🔐		
└	_packOwnershipData	Private 🔐		
└	_nextInitializedFlag	Private 🔐		
└	approve	Public ❗	💸	NO❗
└	getApproved	Public ❗		NO❗
└	setApprovalForAll	Public ❗	🛑	NO❗
└	isApprovedForAll	Public ❗		NO❗
└	_exists	Internal 🔒		
└	_isSenderApprovedOrOwner	Private 🔐		
└	_getApprovedSlotAndAddress	Private 🔐		
└	transferFrom	Public ❗	💸	NO❗
└	safeTransferFrom	Public ❗	💸	NO❗
└	safeTransferFrom	Public ❗	💸	NO❗
└	_beforeTokenTransfers	Internal 🔒	🛑	
└	_afterTokenTransfers	Internal 🔒	🛑	
└	_checkContractOnERC721Received	Private 🔐	🛑	
└	_mint	Internal 🔒	🛑	
└	_mintERC2309	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	

└	_setExtraDataAt	Internal 🔒	🛑	
└	_extraData	Internal 🔒		
└	_nextExtraData	Private 🔐		
└	_msgSenderERC721A	Internal 🔒		
└	_toString	Internal 🔒		
└	_revert	Internal 🔒		
Doka	Implementation	ERC721A, Ownable, ERC2981, OperatorFilterer		
└	<Constructor>	Public ❗	🛑	ERC721A
└	setApprovalForAll	Public ❗	🛑	onlyAllowedOperatorApproval
└	approve	Public ❗	💵	onlyAllowedOperatorApproval
└	transferFrom	Public ❗	💵	onlyAllowedOperator
└	_isPriorityOperator	Internal 🔒		
└	setDefaultRoyalty	Public ❗	🛑	onlyOwner
└	deleteDefaultRoyalty	Public ❗	🛑	onlyOwner
└	supportsInterface	Public ❗		NO❗
└	_startTokenId	Internal 🔒		
└	_baseURI	Internal 🔒		
└	tokenURI	Public ❗		NO❗
└	reserve	External ❗	💵	NO❗
└	setSigner	External ❗	🛑	onlyOwner
└	setOperatorFilteringEnabled	Public ❗	🛑	onlyOwner
└	setPhase	External ❗	🛑	onlyOwner
└	setReservePrice	External ❗	🛑	onlyOwner
└	setBaseURI	External ❗	🛑	onlyOwner
└	withdraw	External ❗	🛑	onlyOwner
└	airdrop	External ❗	🛑	onlyOwner
└	stopMint	External ❗	🛑	onlyOwner
└	signer	External ❗		NO❗
└	operatorFilteringEnabled	External ❗		NO❗
└	phase	External ❗		NO❗
└	reservePrice	External ❗		NO❗
└	baseURI	External ❗		NO❗
└	totalReserveCounter	External ❗		NO❗
└	wlReserveCounter	External ❗		NO❗
└	publicReserveCounter	External ❗		NO❗
└	wlReserveAddresses	External ❗		NO❗
└	publicReserveAddresses	External ❗		NO❗
└	reserveCounter	External ❗		NO❗
└	mintEnabled	External ❗		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.

✓ 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.