Smart Contract Security Audit V1

Special K Glass Smart Contract

24/6/2022

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**: 0x0D89Df0e8051346d39121ad6FeEdCac50bbD8DEE
- **Code:**

https://ropsten.etherscan.io/address/0x7b039374cc422595a9473b4e88d3d3c382da53a9#code

NFT Information

- Name: SKG
- MAX Supply: 10000
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum)

Special K Glass smart contract on ETH test net to test every function by the auditor.

https://rinkeby.etherscan.io/address/0x0d89df0e8051346d39121ad6feedcac50bbd8dee

Executive Summary

According to our assessment, the customer`s solidity smart contract is **“WELL SECURED”**. The team has fixed the critical and 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 1 critical, 0 high, 0 medium, 4 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

SpecialKGlass.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
SpecialKGlass.sol	**f8eed8f9f88a275b1eae090ddd46ae40e5aa86e274ca7fd23958addb8bede183**	0x0D89Df0e8051346d39121ad6FeEdCac50bbD8DEE

- Contract: BlowingGlass
- Inherit: ERC721A, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
preSaleMintRate	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
mintRate	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

whitelistMerkleRoot	✓	Read / public	Passed
presale	✓	Read / public	Passed
uriPrefix	✓	Read / public	Passed
paused	✓	Read / public	Passed
uriSuffix	✓	Read / public	Passed
mint	✓	Write / public	Passed
approve	✓	Write / public	Passed
safeTransferFrom	✓	Write / public	Passed
safeTransferFrom	✓	Write / public	Passed
setPaused	✓	Write / public	Passed
withdraw	✓	Write / payable	Passed
mintForAddress	✓	Write / public	Passed
transferOwnership	✓	Write / public	Passed
setApprovalForAll	✓	Write / public	Passed
transferFrom	✓	Write / public	Passed
mintWhitelist	✓	Write / payable	Passed
setMaxMintAmountPerTx	✓	Write / public	Passed
renounceOwnership	✓	Write / public	Passed
setUriSuffix	✓	Write / public	Passed
setMaxMints	✓	Write / public	Passed
setMintRate	✓	Write / public	Passed
setPresaleMintRate	✓	Write / public	Passed
setMaxSupply	✓	Write / public	Passed
setUriPrefix	✓	Write / public	Passed
setWhitelistMerkleRoot	✓	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 after fixing the errors**
6	Timestamp dependence.	**Passed with Notes**
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 after fixing the errors**
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:

#Logic errors

Description

According to the smart contract functionality, the team should start the presale mint for the whitelist and then close it to start the public mint for the others, the auditor had found these logic errors on the smart contract.
The initial status of the contract is the presale is closed.

```
bool public presale = false;
```

so, the whitelist of the investors can't mint in the WL stage because it requires the presale to be active which can happen because the missing function.

```
function mintWhitelist(bytes32[] calldata merkleProof, uint256 quantity)
        public
        payable
        isValidMerkleProof(merkleProof, whitelistMerkleRoot)
    {
        require(!paused, "The contract is paused");
        require(presale, "Not in presale mode!");
        if (msg.sender != owner()) {
            if (presale == true) {
                require(quantity + _numberMinted(msg.sender) <= MAX_MINTS,
"Exceeded the limit");
                require(totalSupply() + quantity <= MAX_SUPPLY, "Not enough tokens
left");
                require(msg.value >= preSaleMintRate * quantity, "Not enough ether
sent");
            }
        }
        _safeMint(msg.sender, quantity);
    }
```

The investors can only mint in the public stage because the presale in the normal stage is closed,

```
function mint(uint256 quantity) external payable {
        // _safeMint's second argument now takes in a quantity, not a tokenId.
        if (msg.sender != owner()) {
            require(!paused, "Minting is paused!");
            require(!presale, "Presale is active");
            require(quantity + _numberMinted(msg.sender) <= MAX_MINTS, "Exceeded
the limit");
            require(totalSupply() + quantity <= MAX_SUPPLY, "Not enough tokens
left");
            require(msg.value >= (mintRate * quantity), "Not enough ether sent");}
        _safeMint(msg.sender, quantity);
    }
```

Remediation
The team should add a presale status function in the smart contract to allow the owner open and close the presale stage at any time he wants.

Status: Closed. Fixed In version 2

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found

Low:

#Multiple pragma statements

Line	Pragma
7	pragma solidity ^0.8.0;
75	pragma solidity ^0.8.0;
145	pragma solidity ^0.8.0;
175	pragma solidity ^0.8.0;
250	pragma solidity ^0.8.1;
475	pragma solidity ^0.8.0;
505	pragma solidity ^0.8.0;
533	pragma solidity ^0.8.0;
564	pragma solidity ^0.8.0;
709	pragma solidity ^0.8.0;
741	pragma solidity ^0.8.4;
1981	pragma solidity ^0.8.4;

Description
There are multiple pragma statements in the code. The newest compiler version 0.8.14 will work with the code, but keeping only one pragma statement helps in maintaining readability of the code.

Remediation
Keep a single pragma statement.

Status: Closed. Fixed In version 2

#Missing zero address validation

Description

When the owner wants to mint NFTs for investors, he has to check for the zero address to make, he didn’t mint for the zero address. Otherwise, the mint function will act like burn function.

```
function mintForAddress( uint256 _mintAmount, address _reciever) public onlyOwner {

        _safeMint(_reciever, _mintAmount);

    }
```

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 change the price in WL stage and public stage.
The owner can change the max supply of NFTs.
The owner can pause and un pause the contract.

```
function setPresaleMintRate(uint256 _cost) public onlyOwner {
        preSaleMintRate = _cost;
    }
    // Set Paused
    function setPaused(bool _state) public onlyOwner {
        paused = _state;
    }
function setMaxSupply(uint256 _amount)  public onlyOwner {
        MAX_SUPPLY = _amount;
    }
    // Set Max Mints
    function setMaxMints(uint256 _amount) public onlyOwner {
        MAX_MINTS = _amount;
    }
    // Set Mint Rate
    function setMintRate(uint256 _cost) public onlyOwner {
        mintRate = _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.

#Use of block.timestamp for comparisons

Description

The value of block.timestamp can be manipulated by the miner.
And conditions with strict equality is difficult to achieve -
block.timestamp

Remediation
Avoid use of block.timestamp
Status: Acknowledged

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Missing SPDX-License-Identifier:

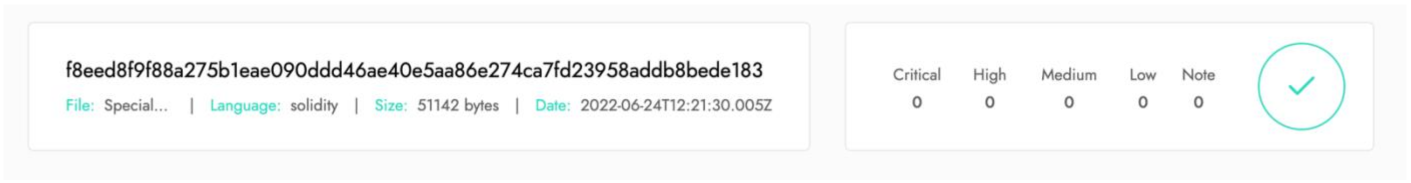
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information .

Remediation
Add License Identifier
// SPDX-License-Identifier: MIT

Status: Closed. Fixed In version 2

Automatic Testing

1- Check for security

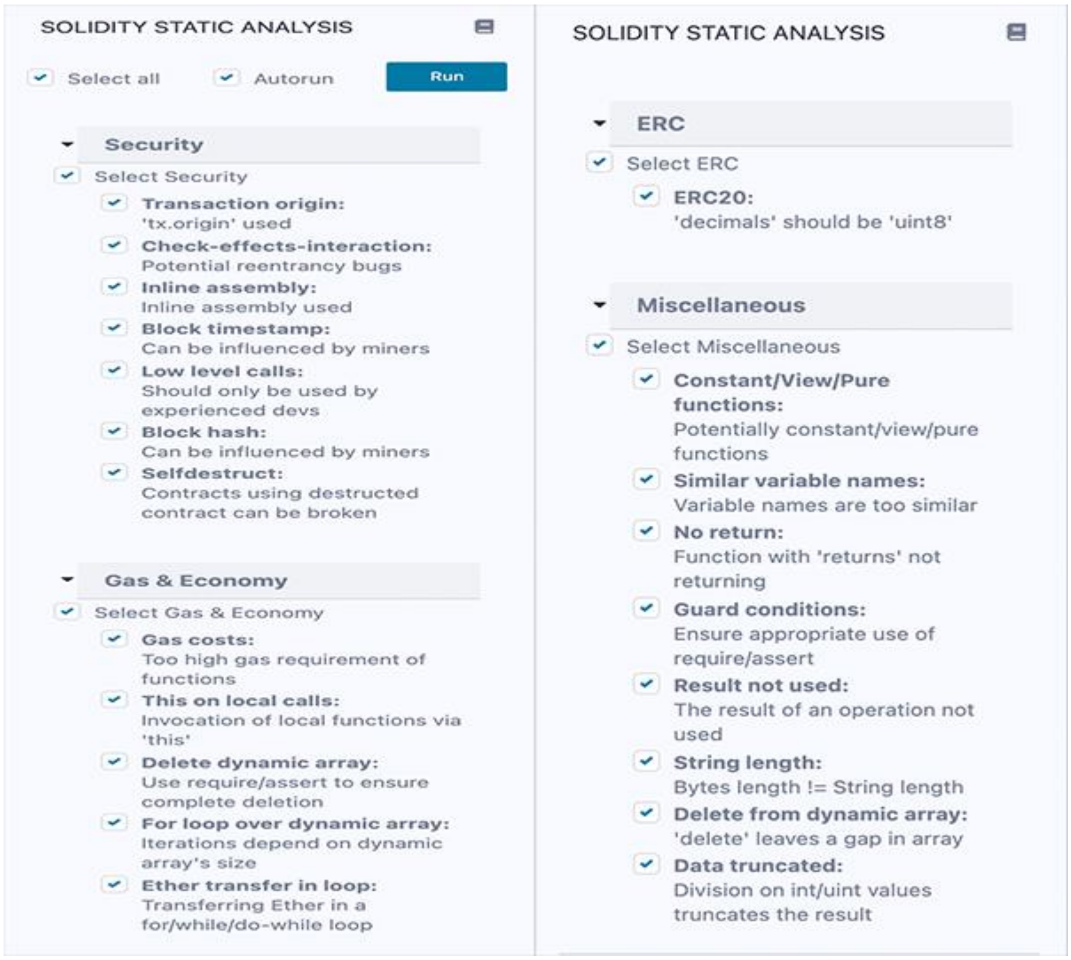


f8eed8f9f88a275b1eae090ddd46ae40e5aa86e274ca7fd23958addb8bede183

File: Special... | Language: solidity | Size: 51142 bytes | Date: 2022-06-24T12:21:30.005Z

Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS



SOLIDITY STATIC ANALYSIS

Select all ✓ Autorun ✓ Run

Security

✓ Select Security

✓ **Transaction origin:** 'tx.origin' used

✓ **Check-effects-interaction:** Potential reentrancy bugs

✓ **Inline assembly:** Inline assembly used

✓ **Block timestamp:** Can be influenced by miners

✓ **Low level calls:** Should only be used by experienced devs

✓ **Block hash:** Can be influenced by miners

✓ **Selfdestruct:** Contracts using destructed contract can be broken

Gas & Economy

✓ Select Gas & Economy

✓ **Gas costs:** Too high gas requirement of functions

✓ **This on local calls:** Invocation of local functions via 'this'

✓ **Delete dynamic array:** Use require/assert to ensure complete deletion

✓ **For loop over dynamic array:** Iterations depend on dynamic array's size

✓ **Ether transfer in loop:** Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

ERC

✓ Select ERC

✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

✓ Select Miscellaneous

✓ **Constant/View/Pure functions:** Potentially constant/view/pure functions

✓ **Similar variable names:** Variable names are too similar

✓ **No return:** Function with 'returns' not returning

✓ **Guard conditions:** Ensure appropriate use of require/assert

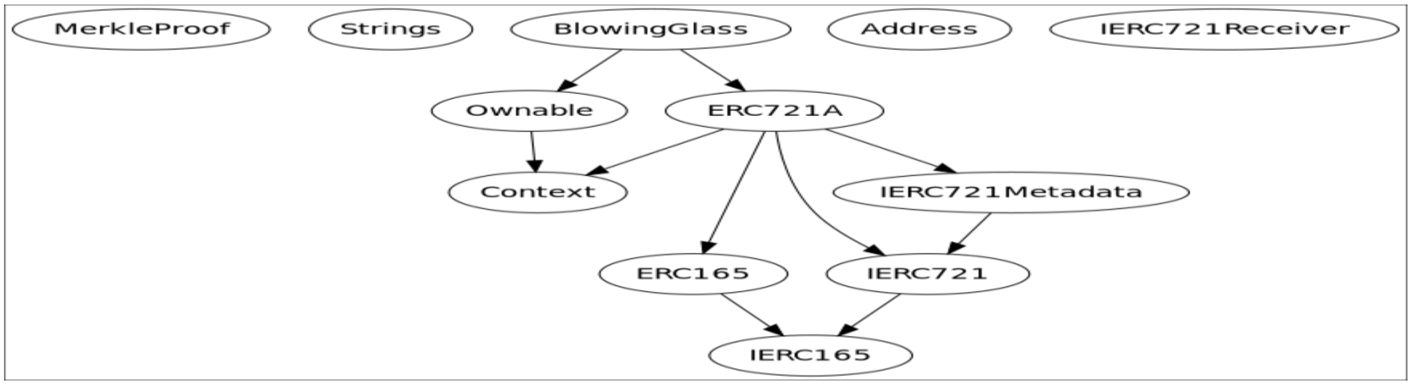
✓ **Result not used:** The result of an operation not used

✓ **String length:** Bytes length != String length

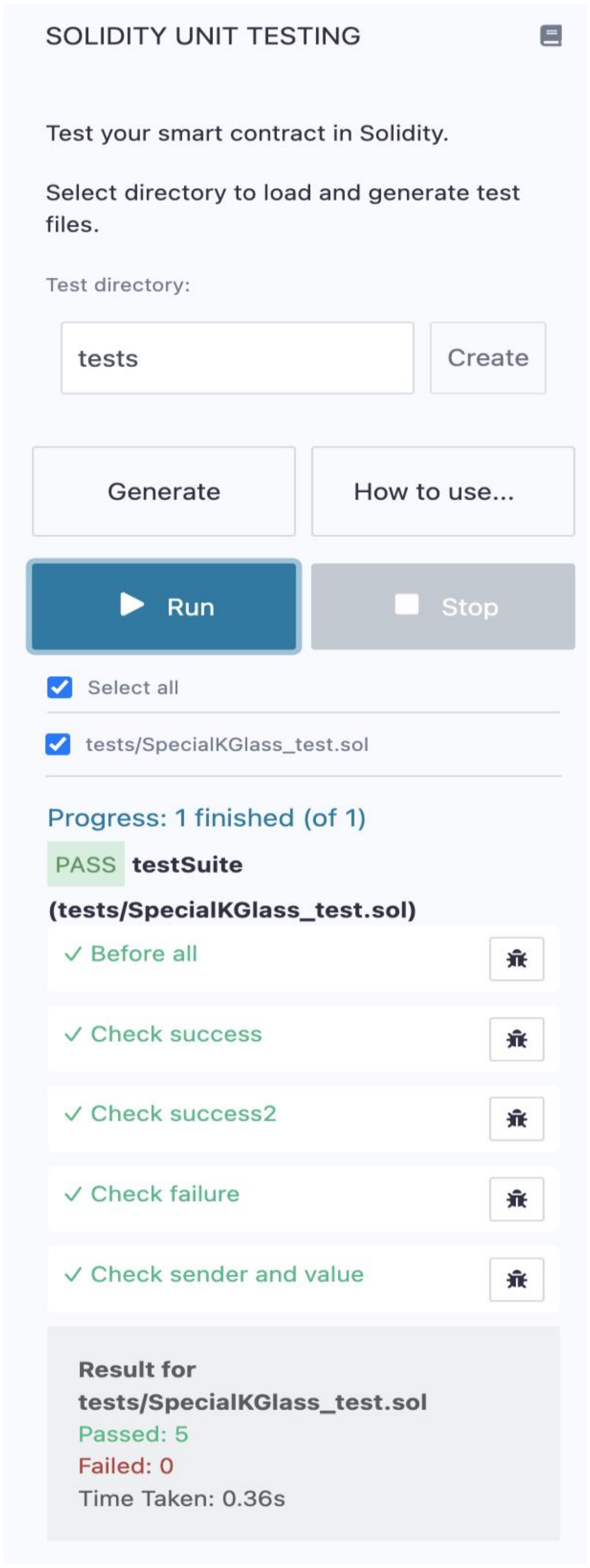
✓ **Delete from dynamic array:** 'delete' leaves a gap in array

✓ **Data truncated:** Division on int/uint values truncates the result

3- Inheritance graph



4- SOLIDITY UNIT TESTING


SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

☑ Select all

☑ tests/SpecialKGlass_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/SpecialKGlass_test.sol)**

✓ Before all

✓ Check success

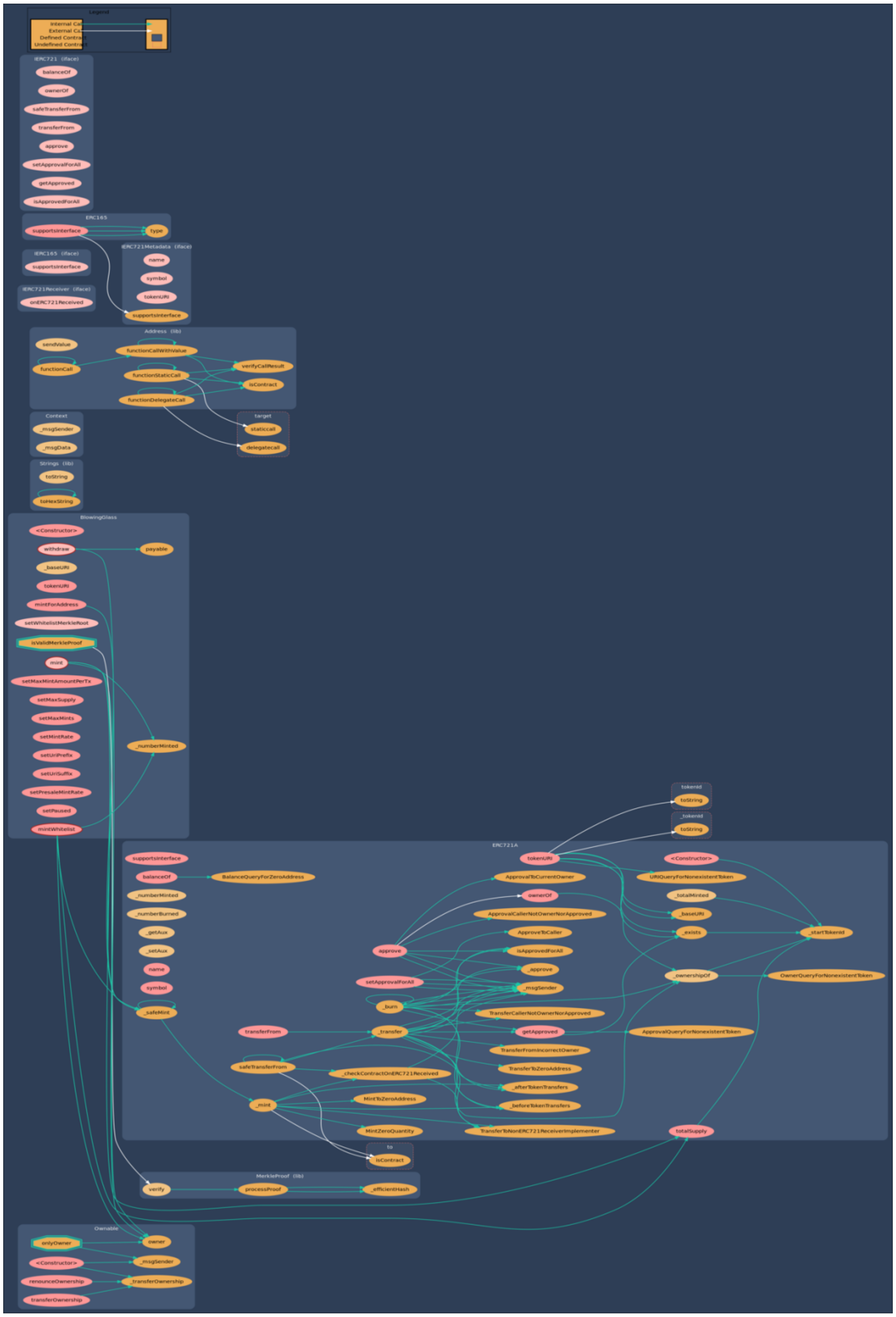
✓ Check success2

✓ Check failure

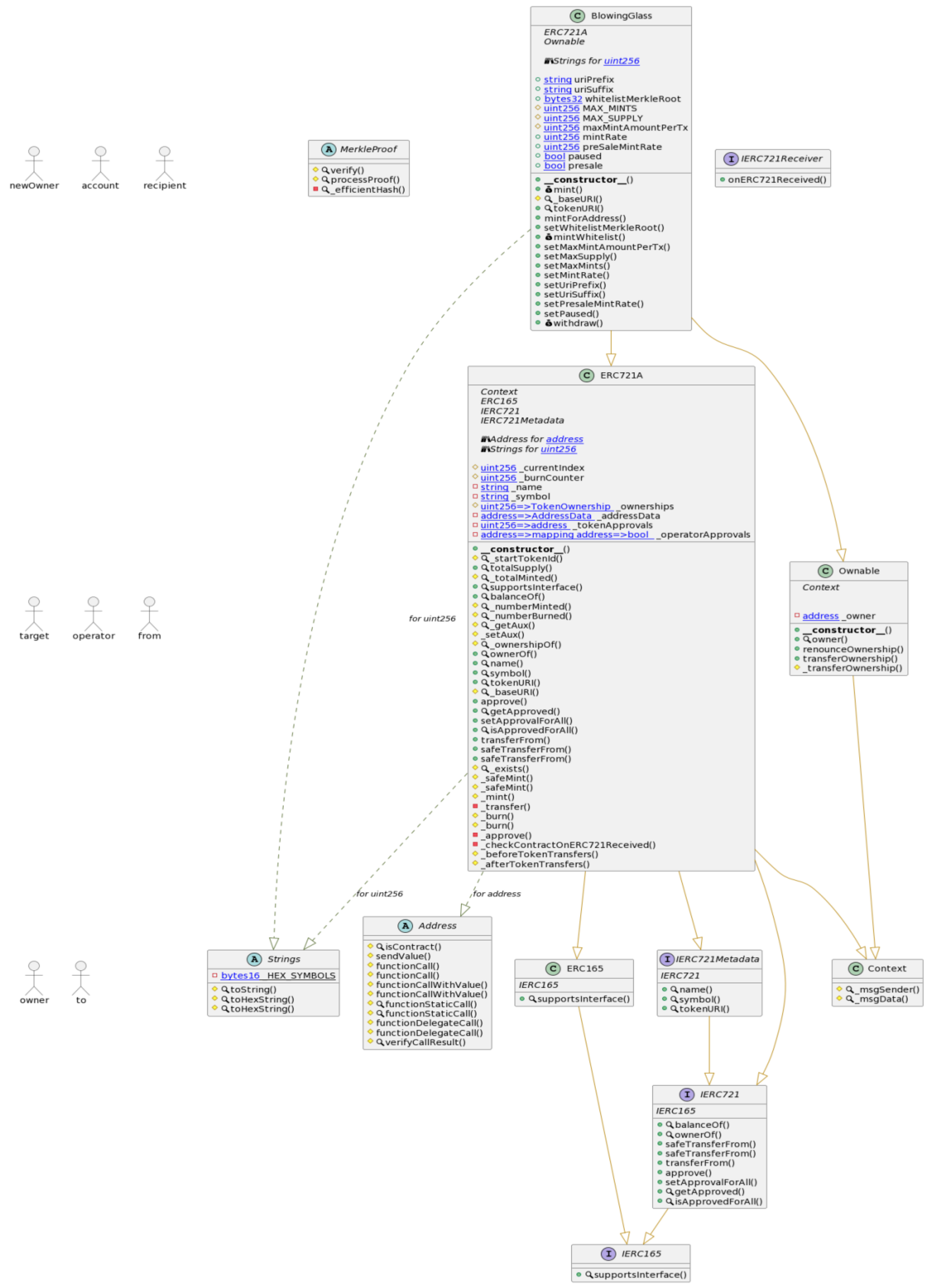
✓ Check sender and value

Result for tests/SpecialKGlass_test.sol
Passed: 5
Failed: 0
Time Taken: 0.36s

5- Call graph



Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
5a9a49c7  =>  verify(bytes32[],bytes32,bytes32)
62702a6b  =>  processProof(bytes32[],bytes32)
41ed615b  =>  _efficientHash(bytes32,bytes32)
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)
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)
98995f77  =>  _startTokenId()
18160ddd  =>  totalSupply()
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)
de0d9900  =>  _mint(address,uint256,bytes,bool)
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)
a0712d68  =>  mint(uint256)
efbd73f4  =>  mintForAddress(uint256,address)
bd32fb66  =>  setWhitelistMerkleRoot(bytes32)
a6d612f9  =>  mintWhitelist(bytes32[],uint256)
b071401b  =>  setMaxMintAmountPerTx(uint256)
6f8b44b0  =>  setMaxSupply(uint256)
79c9cb7b  =>  setMaxMints(uint256)
dbe2193f  =>  setMintRate(uint256)
7ec4a659  =>  setUriPrefix(string)
16ba10e0  =>  setUriSuffix(string)
6417d5b2  =>  setPresaleMintRate(uint256)
16c38b3c  =>  setPaused(bool)
3ccfd60b  =>  withdraw()
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/SpecialKGlass.sol |
1f46a0b0e711e06e73afd6c9e35feb141e0ba36e |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **MerkleProof** | Library |  |||
| └ | verify | Internal 🔒 |   | |
| └ | processProof | Internal 🔒 |   | |
| └ | _efficientHash | Private 🔐 |   | |
||||||
| **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❗️ |
| └ | safeTransferFrom | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
| └ | approve | External ❗️ | 🛑  |NO❗️ |
| └ | setApprovalForAll | External ❗️ | 🛑  |NO❗️ |
| └ | getApproved | External ❗️ |   |NO❗️ |
| └ | isApprovedForAll | External ❗️ |   |NO❗️ |
||||||
| **IERC721Metadata** | Interface | IERC721 |||
| └ | name | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | tokenURI | External ❗️ |   |NO❗️ |
||||||
| **ERC721A** | Implementation | Context, ERC165, IERC721, IERC721Metadata |||
| └ | <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 🔒 | 🛑  | |
||||||
| **BlowingGlass** | Implementation | ERC721A, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  | ERC721A |
| └ | mint | External ❗️ |  💵 |NO❗️ |
| └ | _baseURI | Internal 🔒 |   | |
| └ | tokenURI | Public ❗️ |   |NO❗️ |
| └ | mintForAddress | Public ❗️ | 🛑  | onlyOwner |
| └ | setWhitelistMerkleRoot | External ❗️ | 🛑  | onlyOwner |
| └ | mintWhitelist | Public ❗️ |  💵 | isValidMerkleProof |
| └ | setMaxMintAmountPerTx | Public ❗️ | 🛑  | onlyOwner |
| └ | setMaxSupply | Public ❗️ | 🛑  | onlyOwner |
| └ | setMaxMints | Public ❗️ | 🛑  | onlyOwner |
| └ | setMintRate | Public ❗️ | 🛑  | onlyOwner |
| └ | setUriPrefix | Public ❗️ | 🛑  | onlyOwner |
| └ | setUriSuffix | Public ❗️ | 🛑  | onlyOwner |
| └ | setPresaleMintRate | Public ❗️ | 🛑  | onlyOwner |
| └ | setPaused | Public ❗️ | 🛑  | onlyOwner |
| └ | withdraw | External ❗️ |  💵 | onlyOwner |

 Legend

|  Symbol  |  Meaning  |
|:--------:|-----------|
|    🛑    | Function can modify state |
|    💵    | Function is payable |
```

Conclusion

The contracts are written systematically. Team found no critical issues after fixing the errors. 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 many high severity issues were found.

✓ Low (or very low) level issues have been fixed.

Disclaimer