Smart Contract Security Audit V1

MOOLA Token Smart Contract

5/9/2022

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**: 0x0825f4a5fdfcba0791f5bfa288b0847a1101f420
- **Code Source:**

https://rinkeby.etherscan.io/address/0x0825f4a5fdfcba0791f5bfa288b0847a1101f420#code

Token Information

- Name: MOOLA
- Total Supply: 100,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

MOOLA Token smart contract on Eth test net by the auditor to test every function (ETH Test Net)

https://rinkeby.etherscan.io/address/0x0825f4a5fdfcba0791f5bfa288b0847a1101f420

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, 1 high, 0 medium, 3 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

MOOLA.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
MOOLAToken.sol	**265c300ce4bb0b2653e0d7e445e4c0b0062c7102766d668356aaf0d76769e3bc**	0x0825f4a5fdfcba0791f5bfa288b0847a1101f420

- Contract: MOOLAToken
- Inherit: ERC20, ERC20Burnable, Pausable, 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**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
totalClaimed	✓	Read / public	**Passed**
admin	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
amountToClaim	✓	Read / public	**Passed**
mint	✓	Write / public	**Passed**

approve	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
pause	✓	Write / public	**Passed**
unPause	✓	Write / public	**Passed**
burnFrom	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
burn	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
claimStuckTokens	✓	Write / public	**Passed**
claim	✓	Write / public	**Passed**
setUsersAndAmounts	✓	Write / public	**Passed**
setNewAdmin	✓	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:

#Logic errors

Description

According to the smart contract functionality, the smart contract has a total supply = of 100,000,000, but the admin can mint any token for any address which will affect the token's price. And the admin can mint when the smart contract is paused.

```
function mint(address to, uint256 amount) public onlyAdmin {
        _mint(to, amount);
    }
```

You can check these transactions:

https://rinkeby.etherscan.io/tx/0x3bf810a5c6b75aaa07f364cba96c608e71bd3172fe08941b35437ad56bb407fe

The second error the smart contract inherits the pausable library but the smart contract didn’t use it when the admin mint new tokens.

Remediation

The team should add max supply of token which can’t mint more than that and it should pause when the contract paused.
For the second error, the team has 2 chooses to delete the library to save some gas or use it in the smart contract like can’t mint or ownership when the smart contract is paused.

Status: Closed. Fixed In version 2

Medium:

No Medium severity vulnerabilities were found.

Low:

#Missing zero address validation

Description

When the owner wants to mint tokens to any address, he has to check for the zero address to make he didn’t add the zero address. otherwise, the mint function will act like a burn function. And it should do the same for set users and amounts function for calming the rewards.

```
function mint(address to, uint256 amount) public onlyAdmin {
        _mint(to, amount);
    }
function setUsersAndAmounts (address [] calldata userAddress, uint256 [] calldata
amount) external onlyAdmin {
        for (uint256 i= 0; i < userAddress.length; i++){
            amountToClaim[userAddress[i]] = amount[i];
        }
    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version 2.

#Multiple pragma statements

Line	Pragma
7	pragma solidity ^0.8.0;
34	pragma solidity ^0.8.0;
112	pragma solidity ^0.8.0;
205	pragma solidity ^0.8.0;
290	pragma solidity ^0.8.0;
320	pragma solidity ^0.8.0;
708	pragma solidity ^0.8.0;
747	pragma solidity ^0.8.16;

Description

There are multiple pragma statements in the code. The newest compiler version 0.8.16 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

#Owner and the admin privileges (In the period when the owner isn’t renounced)

Description

The admin can mint to any address.
The owner can pause / un pause the smart contract.
The admin can add any address to claim the rewards with any amount of tokens.

```
function pause() public onlyOwner {
        _pause();
    }
    function unpause() public onlyOwner {
        _unpause();}
        function setNewAdmin (address newAdmin) external onlyOwner {
        require (newAdmin != address(0), "MOOLA: new admin can't be a zero
address");
        admin = newAdmin;}
    function setUsersAndAmounts (address [] calldata userAddress, uint256 []
calldata amount) external onlyAdmin {
        for (uint256 i= 0; i < userAddress.length; i++){
            amountToClaim[userAddress[i]] = amount[i]; }
function mint(address to, uint256 amount) public onlyAdmin {
        _mint(to, amount);}
```

Remediation

Make these functions internal in next version or the team should announce the investors before change anything and give them time to do anything they want.

P.S: This issue is common to the majority of rewards smart contracts.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

# Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
   uint256 private _totalSupply = 100000000 * 10 **
decimals();//100000000000000000000000000
        _mint(msg.sender, 100000000 * 10 ** decimals()); // 100 Million initial
Supply
```

Recommendation

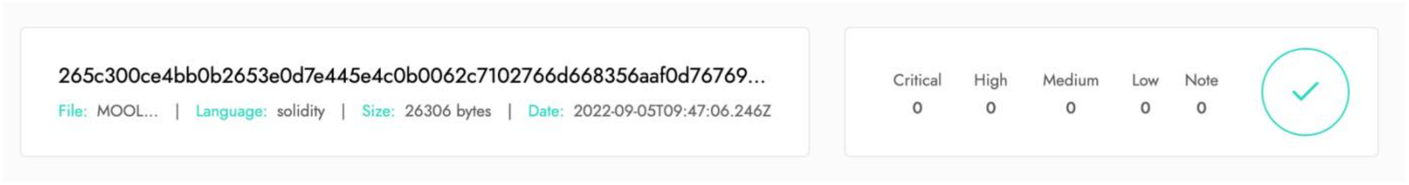
Replace the initialization as

```
   uint256 private _totalSupply = 100000000000000000000000000;
        _mint(msg.sender, 100000000000000000000000000;
```

Status Closed. Fixed in version 2.

Automatic Testing

1- Check for security

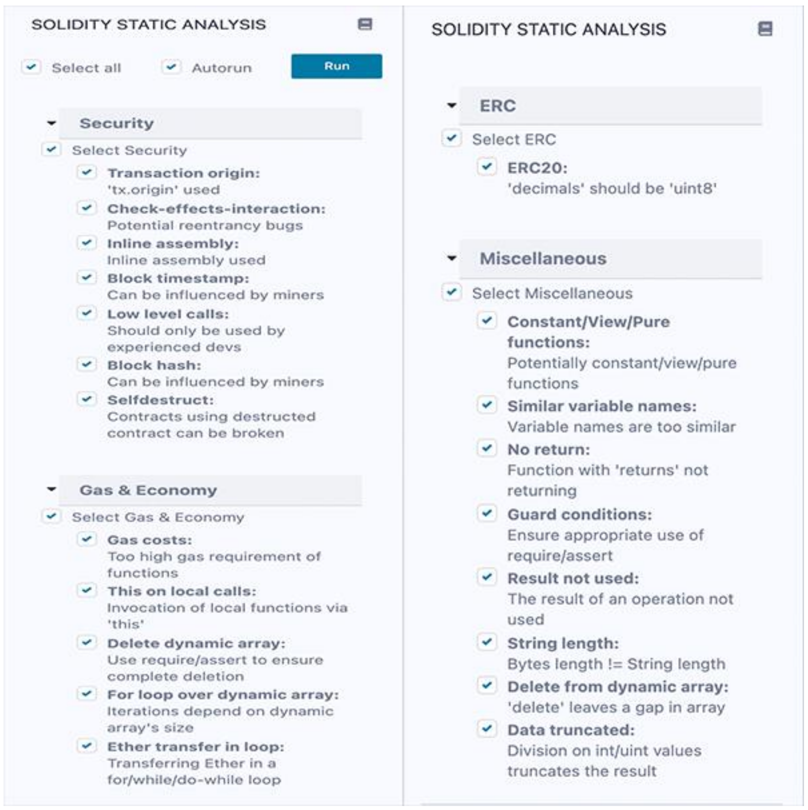


265c300ce4bb0b2653e0d7e445e4c0b0062c7102766d668356aaf0d76769...

File: MOOL... | Language: solidity | Size: 26306 bytes | Date: 2022-09-05T09:47:06.246Z

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

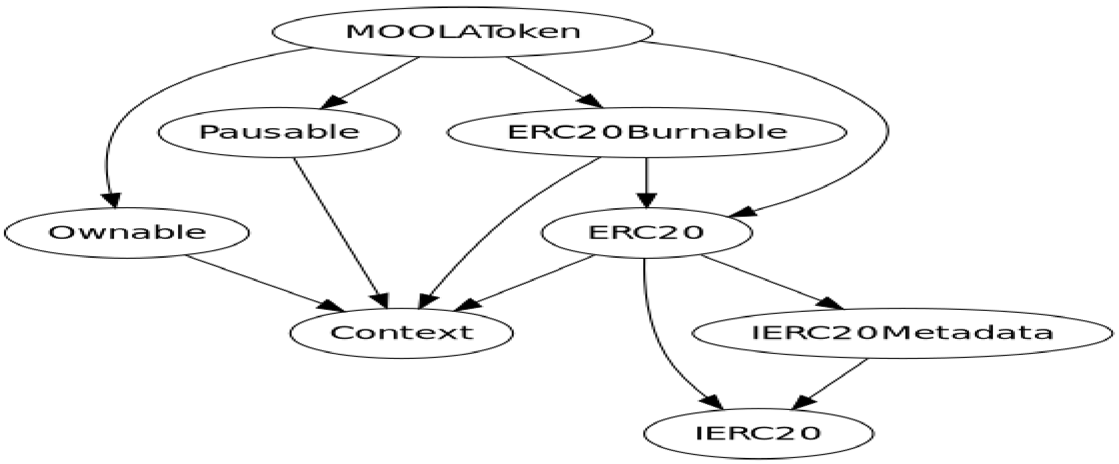
- ✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

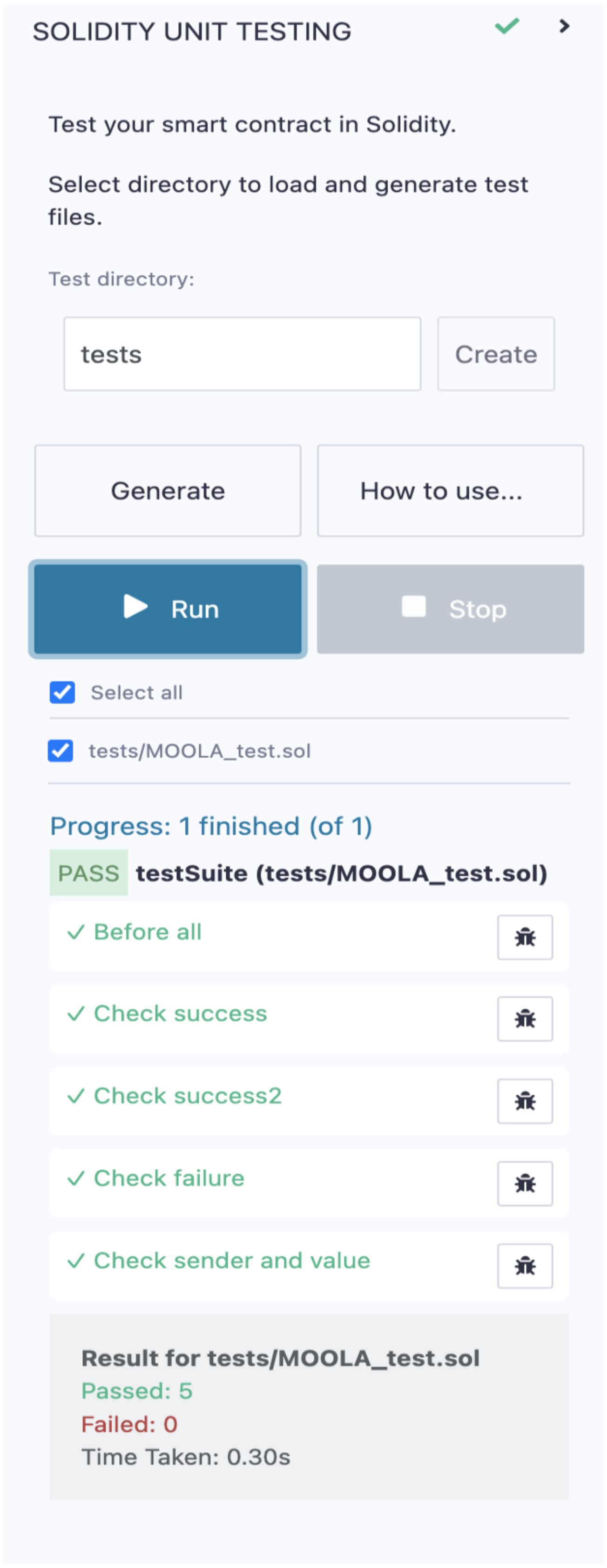
✓ Select Miscellaneous

- ✓ **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: tests

✓ tests/MOOLA_test.sol

Progress: 1 finished (of 1)

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

✓ Before all

✓ Check success

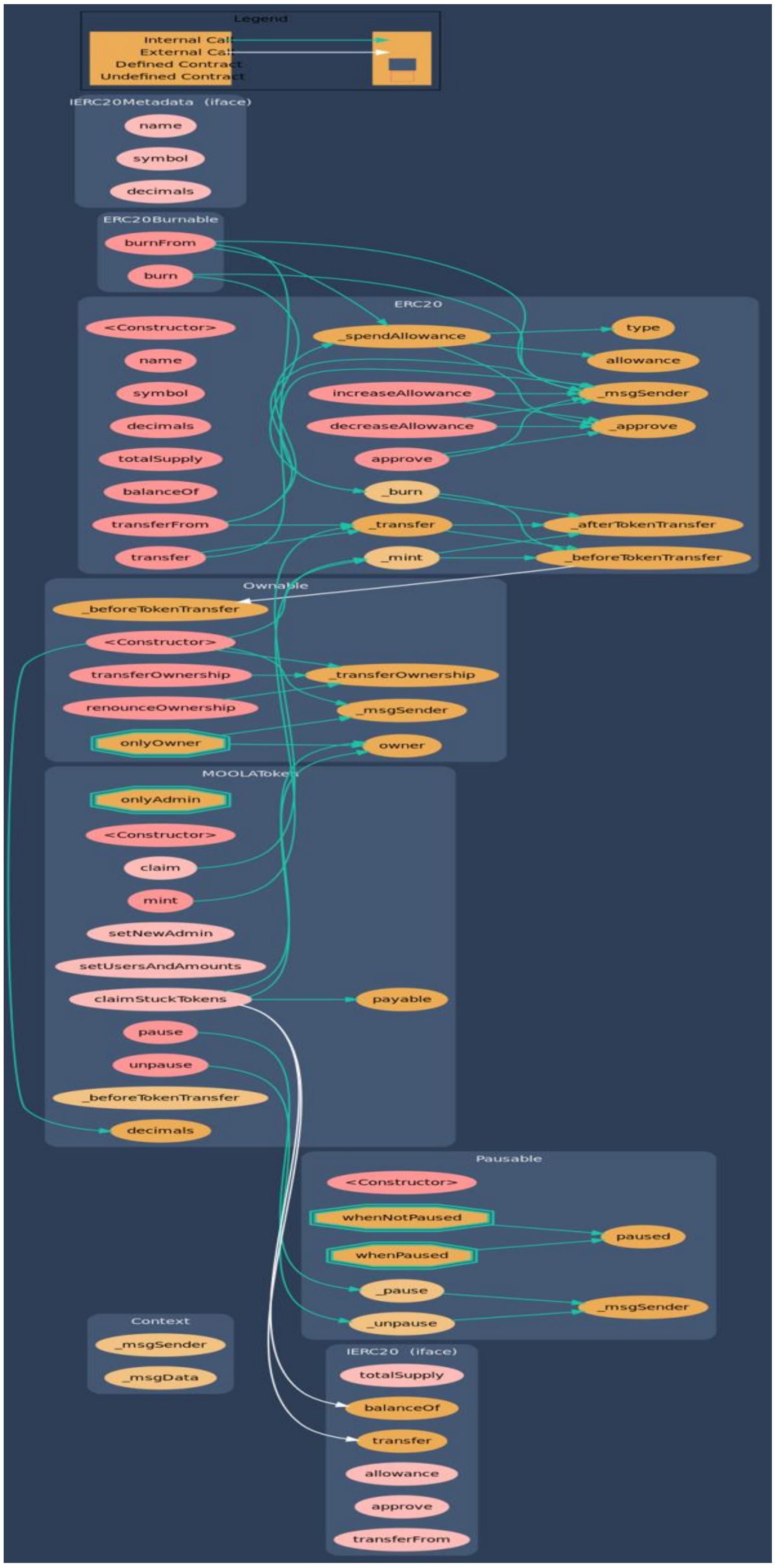
✓ Check success2

✓ Check failure

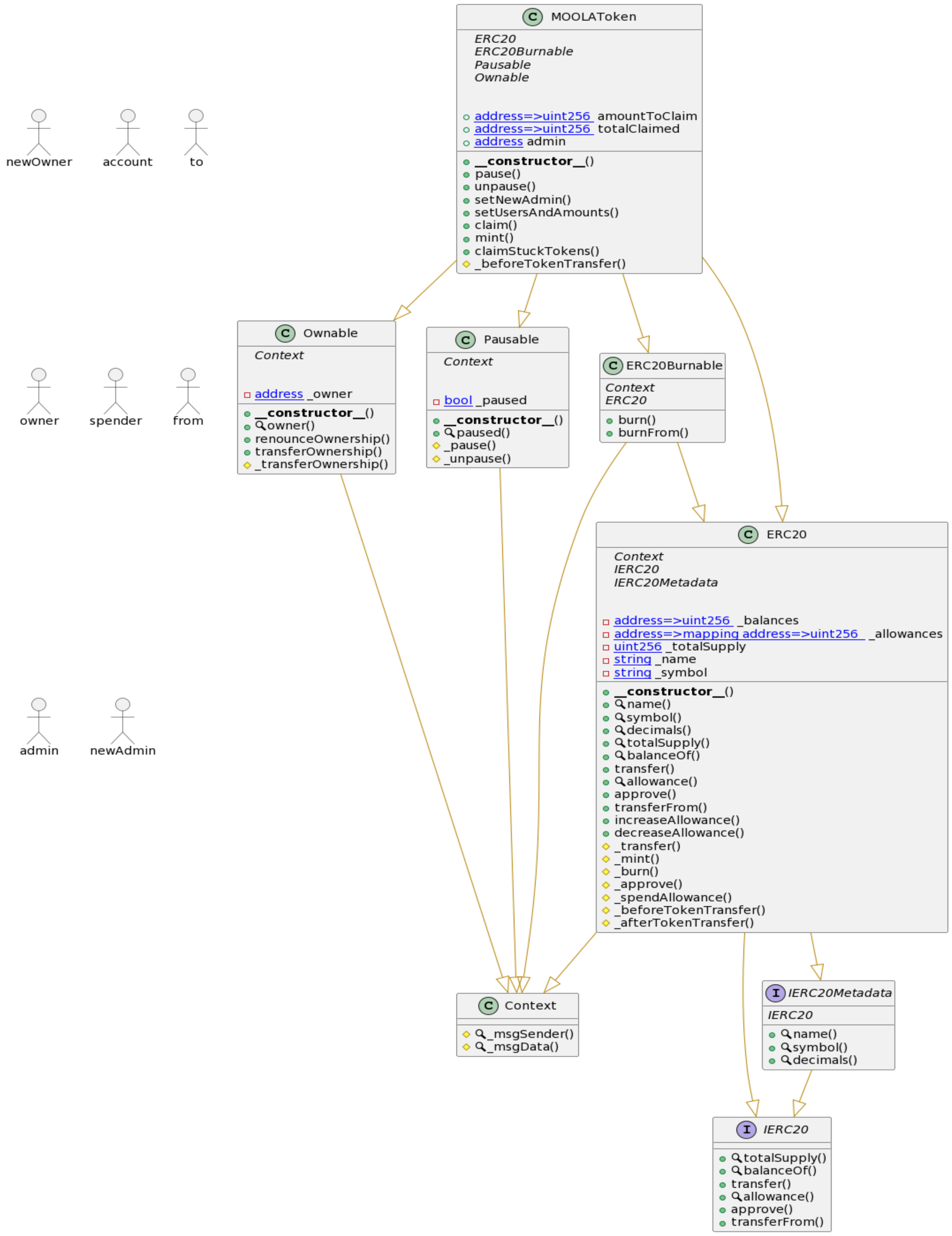
✓ Check sender and value

Result for tests/MOOLA_test.sol
Passed: 5
Failed: 0
Time Taken: 0.30s

5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Metadata (iface)
name
symbol
decimals
ERC20Burnable
burnFrom
burn
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
increaseAllowance
decreaseAllowance
approve
_burn
_transfer
_mint
type
allowance
_msgSender
_approve
_afterTokenTransfer
_beforeTokenTransfer
Ownable
_beforeTokenTransfer
<Constructor>
transferOwnership
renounceOwnership
onlyOwner
_transferOwnership
_msgSender
owner
MOOLAToken
onlyAdmin
<Constructor>
claim
mint
setNewAdmin
setUsersAndAmounts
claimStuckTokens
pause
unpause
_beforeTokenTransfer
decimals
payable
Pausable
<Constructor>
whenNotPaused
whenPaused
paused
_pause
_unpause
_msgSender
Context
_msgSender
_msgData
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
5c975abb  =>  paused()
320b2ad9  =>  _pause()
fc8234cb  =>  _unpause()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
1532335e  =>  _spendAllowance(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
42966c68  =>  burn(uint256)
79cc6790  =>  burnFrom(address,uint256)
8456cb59  =>  pause()
3f4ba83a  =>  unpause()
8eec99c8  =>  setNewAdmin(address)
cc08ce9a  =>  setUsersAndAmounts(address,uint256)
4e71d92d  =>  claim()
40c10f19  =>  mint(address,uint256)
f9d0831a  =>  claimStuckTokens(address)
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/MOOLA.sol |
c668f6b7fe1e6e276a2edee6d9824f7937ca480f |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |
|                  |
|:----------:|:-------------------:|:----------------:|:----------------
:|:----------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**
|  **Modifiers**  |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **Pausable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | paused | Public ❗ |   |NO❗ |
| └ | _pause | Internal 🔒 | 🛑  | whenNotPaused |
| └ | _unpause | Internal 🔒 | 🛑  | whenPaused |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
||||||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | name | Public ❗️ |   |NO❗️ |
| └ | symbol | Public ❗️ |   |NO❗️ |
| └ | decimals | Public ❗️ |   |NO❗️ |
| └ | totalSupply | Public ❗️ |   |NO❗️ |
| └ | balanceOf | Public ❗️ |   |NO❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO❗️ |
| └ | allowance | Public ❗️ |   |NO❗️ |
| └ | approve | Public ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _mint | Internal 🔒 | 🛑  | |
| └ | _burn | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _spendAllowance | Internal 🔒 | 🛑  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |
| └ | _afterTokenTransfer | Internal 🔒 | 🛑  | |
||||||
| **ERC20Burnable** | Implementation | Context, ERC20 |||
| └ | burn | Public ❗️ | 🛑  |NO❗️ |
| └ | burnFrom | Public ❗️ | 🛑  |NO❗️ |
||||||
| **MOOLAToken** | Implementation | ERC20, ERC20Burnable, Pausable,
Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  | ERC20 |
| └ | pause | Public ❗️ | 🛑  | onlyOwner |
| └ | unpause | Public ❗️ | 🛑  | onlyOwner |
| └ | setNewAdmin | External ❗️ | 🛑  | onlyOwner |
| └ | setUsersAndAmounts | External ❗️ | 🛑  | onlyAdmin |
| └ | claim | External ❗️ | 🛑  |NO❗️ |
| └ | mint | Public ❗️ | 🛑  | onlyAdmin |
| └ | claimStuckTokens | External ❗️ | 🛑  | onlyOwner |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | whenNotPaused |

 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.