Aug.17, 2023

ANTCHAIN | ZAN

Security Assessment

TokenVesting

—

Express Service

Table of Contents

1. Overview

1.1. Executive Summary

TokenVesting is an ERC20 token vesting project. This report has been prepared for TokenVesting project to discover issues and vulnerabilities in the source code as well as contract dependencies that were not part of an officially recognized library. Conducted by Static Analysis and Formal Verificaton, we have identified 1 high vulnerability and 10 informational issues in TokenVesting.sol (dea0d81e28437d04fe3a63d244b884f8). The TokenVesting team has resolved the H-01 and I-03 issues in TokenVesting.sol (b9a6ad121c8a99a7aff865f4de43246e). Regarding the other informational issues, the team has chosen to retain the current implementation without modifications.

1.2. Project Summary

Project Name	TokenVesting
Platform	Ethereum
Language	Solidity
Code Repository	
Commit	

1.3. Assessment Summary

Delivery Date	Aug.17, 2023
Audit Methodology	Static Analysis, Formal Verification

1.4. Assessment Scope

ID	File	File Hash
1	/clof_vesting/TokenVesting.sol	dea0d81e28437d04fe3a63d244b884f8

2. Checklist

2.1. Code Security

Reentrancy	DelegateCall	Integer Overflow
Input Validation	Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer	Do-while Continue
Right-To-Left-Override Character	Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values	Diamond Inheritance
ThisBalance	VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable	Divide Before Multiply
Affected by Compiler Bug		

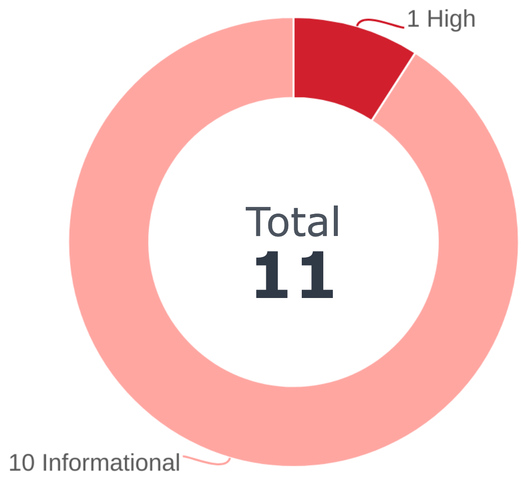
2.2. Optimization Suggestion

Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Inexistent Error Message
State Variable Defined Without Storage Location	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
Lack of SafeMath	Compiler Error/Warning
Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

2.3. Business Security

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings



Code Security

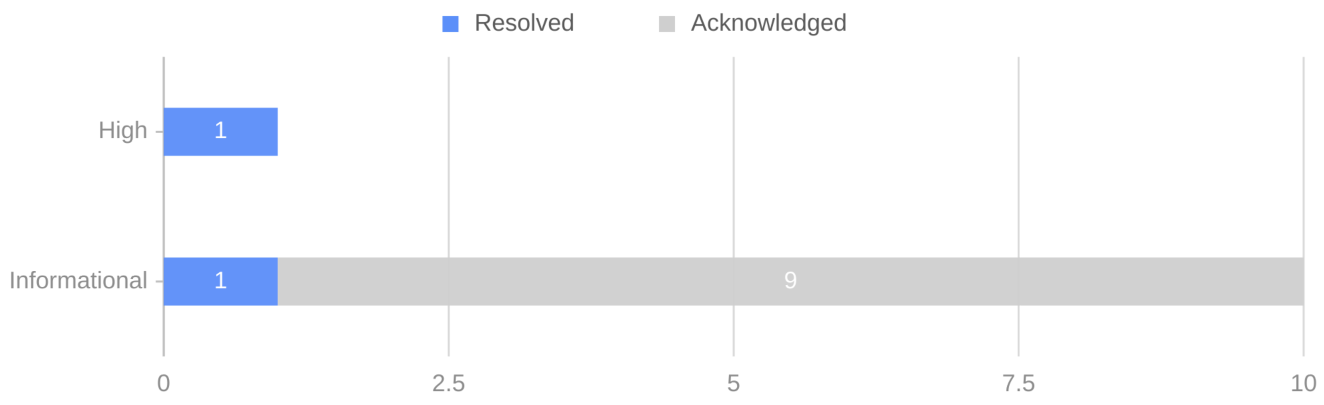
Total: 1, High: 1

Optimization Suggestion

Total: 10, Informational: 10

Business Security

Total: 0, no vulnerabilities.



ID	Title	Category	Severity	Status
H-01	Code Security - Freeze Money	Code Security	● High	Resolved
I-02	Optimization Suggestion - Function Visibility Can Be External	Optimization Suggestion	● Informational	Acknowledged
I-03	Optimization Suggestion - Floating Pragma	Optimization Suggestion	● Informational	Resolved
I-04	Optimization Suggestion - Use CustomError Instead of String	Optimization Suggestion	● Informational	Acknowledged

ID	Title	Category	Severity	Status
I-05	Optimization Suggestion - No Check of Address Params with Zero Address	Optimization Suggestion	● Informational	Acknowledged
I-06	Optimization Suggestion - Use Assembly to Check Zero Address	Optimization Suggestion	● Informational	Acknowledged
I-07	Optimization Suggestion - Use != 0 Instead of > 0 for Unsigned Integer Comparison	Optimization Suggestion	● Informational	Acknowledged
I-08	Optimization Suggestion - Lack of Error Message	Optimization Suggestion	● Informational	Acknowledged
I-09	Optimization Suggestion - Redundant Check of _totalAmount	Optimization Suggestion	● Informational	Acknowledged
I-10	Optimization Suggestion - Recommend to Use a Multi-signature Wallet as Owner	Optimization Suggestion	● Informational	Acknowledged
I-11	Optimization Suggestion - Redundant Computation When Function revoke Calls release	Optimization Suggestion	● Informational	Acknowledged

H-01|Code Security - Freeze Money

High : Code Security

File Location : /clof_vesting/TokenVesting.sol:73,79

Description

There is at least one payable function in the contract, but no transfer function(like send, transfer, call...) exists, which will cause Ether to be locked in the contract.

/clof_vesting/TokenVesting.sol

```
    /**
* @dev Fallback function is executed if none of the other functions match the
function
* identifier or no data was provided with the function call.
*/
    fallback() external payable{}
```

/clof_vesting/TokenVesting.sol

```
    /**
* @dev This function is called for plain Ether transfers, i.e. for every call
with empty calldata.
*/
    receive() external payable{}
```

Recommendation

If the contract is about to receive ether, an ether transfer function should be provided. Otherwise, we suggest to remove payable functions.

Alleviation

Resolved in TokenVesting.sol (b9a6ad121c8a99a7aff865f4de43246e). An "emergency" function is added to withdraw tokens or native coins. Meanwhile vesting token is not withdrawable in "emergency" function, in purpose of protecting the interest of beneficiary.

I-02|Optimization Suggestion - Function Visibility Can Be External

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:244,281,258,265,329,230,220

Description

Functions that are not called should be declared as external.

/clof_vesting/TokenVesting.sol

```
218         * @return the number of vesting schedules
219         */
220        function getVestingSchedulesCountByBeneficiary(
221            address _beneficiary
222        ) public view returns (uint256) {
```

/clof_vesting/TokenVesting.sol

```
228         * @return the vesting id
229         */
230        function getVestingIdAtIndex(
231            uint256 index
232        ) public view returns (bytes32) {
```

/clof_vesting/TokenVesting.sol

```
279         * @return the vested amount
280         */
281        function computeReleasableAmount(
282            bytes32 vestingScheduleId
283        )
284
```

/clof_vesting/TokenVesting.sol

```
242         * @return the vesting schedule structure information
243         */
244        function getVestingScheduleByAddressAndIndex(
245            address holder,
246            uint256 index
247
```

/clof_vesting/TokenVesting.sol

```
327         * @dev Returns the last vesting schedule for a given holder address.
328         */
329        function getLastVestingScheduleForHolder(
330            address holder
331        ) public view returns (VestingSchedule memory) {
```

/clof_vesting/TokenVesting.sol

```
     * @return the total amount of vesting schedules
     */
    function getVestingSchedulesTotalAmount() public view returns (uint256){
        return vestingSchedulesTotalAmount;
    }
```

/clof_vesting/TokenVesting.sol

```
     * @dev Returns the address of the ERC20 token managed by the vesting
contract.
     */
    function getToken() public view returns (address){
        return address(_token);
    }
```

Recommendation

Functions that are not called in the contract should be declared as external.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-03|Optimization Suggestion - Floating Pragma

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:3

Description

Contracts should be deployed with fixed compiler version which has been tested thoroughly or make sure to lock the contract compiler version in the project configuration. Locked compiler version ensures that contracts will not be compiled by untested compiler version.

/clof_vesting/TokenVesting.sol

```
// contracts/TokenVesting.sol
// SPDX-License-Identifier: Apache-2.0
pragma solidity ^0.8.19;

// OpenZeppelin dependencies
```

Recommendation

Use a fixed compiler version, and consider whether the bugs in the selected compiler version (https://github.com/ethereum/solidity/releases) will affect the contract.

Alleviation

Resolved in TokenVesting.sol (b9a6ad121c8a99a7aff865f4de43246e).

I-04|Optimization Suggestion - Use CustomError Instead of String

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:106,54-55,117,199,233,111-113,155,174,204,65,102

Description

When using require or revert, CustomError is more gas efficient than string description, as the error message described using CustomError is only compiled into four bytes. Especially when string exceeds 32 bytes, more gas will be consumed. Generally, around 250-270 gas can be saved for one CustomError replacement when compiler optimization is turned off, 60-80 gas can be saved even if compiler optimization is turned on.

/clof_vesting/TokenVesting.sol

```
53      modifier onlyIfVestingScheduleNotRevoked(bytes32 vestingScheduleId){
54          require(vestingSchedules[vestingScheduleId].initialized);
55          require(!vestingSchedules[vestingScheduleId].revoked);
56          _;
57      }
```

/clof_vesting/TokenVesting.sol

```
172       */
173      function withdraw(uint256 amount) external nonReentrant onlyOwner{
174          require(
175              getWithdrawableAmount() >= amount,
176              "TokenVesting: not enough withdrawable funds"
```

/clof_vesting/TokenVesting.sol

```
197
198          bool isReleasor = (msg.sender == owner());
199          require(
200              isBeneficiary || isReleasor,
201
     "TokenVesting: only beneficiary and owner can release vested tokens"
```

/clof_vesting/TokenVesting.sol

```
202          );
203          uint256 vestedAmount = _computeReleasableAmount(vestingSchedule);
204          require(
205              vestedAmount >= amount,
206              "TokenVesting: cannot release tokens, not enough vested tokens"
```

/clof_vesting/TokenVesting.sol

```
231          uint256 index
232      ) public view returns (bytes32) {
233          require(
            index < getVestingSchedulesCount(),
            "TokenVesting: index out of bounds"
```

/clof_vesting/TokenVesting.sol

```
    constructor(address token_){
        // Check that the token address is not 0x0.
        require(token_ != address(0x0));
        // Set the token address.
        _token = ERC20(token_);
```

/clof_vesting/TokenVesting.sol

```
        require(_duration > 0, "TokenVesting: duration must be > 0");
        require(_totalAmount > 0, "TokenVesting: amount must be > 0");
        require(
            _slicePeriodSeconds >= 1,
            "TokenVesting: slicePeriodSeconds must be >= 1"
```

/clof_vesting/TokenVesting.sol

```
        uint256 _totalAmount
    ) external onlyOwner {
        require(
            getWithdrawableAmount() >= _totalAmount + _tgeAmount,

 "TokenVesting: cannot create vesting schedule because not sufficient tokens"
```

/clof_vesting/TokenVesting.sol

```
            vestingScheduleId
        ];
        require(
            vestingSchedule.revocable,
            "TokenVesting: vesting is not revocable"
```

/clof_vesting/TokenVesting.sol

```

 "TokenVesting: cannot create vesting schedule because not sufficient tokens"
        );
        require(
            _totalAmount > _tgeAmount,
            "TokenVesting: _tgeAmount must lesser than _totalAmount"
```

/clof_vesting/TokenVesting.sol

```
            "TokenVesting: slicePeriodSeconds must be >= 1"
        );
        require(_duration >= _cliff, "TokenVesting: duration must be >= cliff"
 );
        bytes32 vestingScheduleId = computeNextVestingScheduleIdForHolder(
            _beneficiary
```

Recommendation

Use CustomError instead of string for require or revert description.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-05|Optimization Suggestion - No Check of Address Params with Zero Address

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:93

Description

The input parameter of the address type in the function does not use the zero address for verification.

/clof_vesting/TokenVesting.sol

```
90       * @param _totalAmount total amount of tokens to be released at the end of
    the vesting
91       */
92      function createVestingSchedule(
93          address _beneficiary,
94          uint256 _start,
95
```

Recommendation

It is recommended to perform zero address verification on the input parameters of the address type.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-06|Optimization Suggestion - Use Assembly to Check Zero Address

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:65

Description

Using assembly to check zero address can save gas. Under Solidity compiler 0.8.x, about 18 gas can be saved in each call.

/clof_vesting/TokenVesting.sol

```
    constructor(address token_){
        // Check that the token address is not 0x0.
        require(token_ != address(0x0));
        // Set the token address.
        _token = ERC20(token_);
```

Recommendation

It is recommended to use assembly to check zero address.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-07|Optimization Suggestion - Use != 0 Instead of > 0 for Unsigned Integer Comparison

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:111-112

Description

For unsigned integers, use !=0 for comparison, which consumes less gas than >0. Under Solidity compiler 0.8.x, when compiler optimization is turned off, about 3 gas can be saved. When compiler optimization is turned on, no gas can be saved.

/clof_vesting/TokenVesting.sol

```
110
111         require(_duration > 0, "TokenVesting: duration must be > 0");
112         require(_totalAmount > 0, "TokenVesting: amount must be > 0");
113         require(
114             _slicePeriodSeconds >= 1,
```

Recommendation

For unsigned integers, it is recommended to use !=0 instead of >0 for comparison.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-08|Optimization Suggestion - Lack of Error Message

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:54-55,65

Description

Use empty string as parameter when invoking function Revert() or Require().

/clof_vesting/TokenVesting.sol

```
63      constructor(address token_){
64          // Check that the token address is not 0x0.
65          require(token_ != address(0x0));
66          // Set the token address.
67          _token = ERC20(token_);
68      }
```

/clof_vesting/TokenVesting.sol

```
52       */
53      modifier onlyIfVestingScheduleNotRevoked(bytes32 vestingScheduleId){
54          require(vestingSchedules[vestingScheduleId].initialized);
55          require(!vestingSchedules[vestingScheduleId].revoked);
56          _;
```

Recommendation

It is recommended to provide detailed error messages in the parameters when calling require or revert functions. Alternatively, CustomError can be used to describe error messages.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-09|Optimization Suggestion - Redundant Check of _totalAmount

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:112

Description

The variables _tgeAmount and _totalAmount are both of type uint. If the condition in lines 106-109 is satisfied, then it is guaranteed that _totalAmount is greater than 0. Therefore, the condition in line 112 becomes redundant and can be removed to optimize for gas efficiency.

/clof_vesting/TokenVesting.sol

```
106  require(
107      _totalAmount > _tgeAmount,
108      "TokenVesting: _tgeAmount must lesser than _totalAmount"
109  );
110
111  require(_duration > 0, "TokenVesting: duration must be > 0");
112  require(_totalAmount > 0, "TokenVesting: amount must be > 0");
```

Recommendation

It is recommended to remove the verification of _totalAmount in line 112.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-10|Optimization Suggestion - Recommend to Use a Multi-signature Wallet as Owner

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:173

Description

The TokenVesting contract will hold a significant amount of tokens, while the contract owner has the permission to withdraw withdrawable tokens. The TokenVesting contract might be vulnerable to theft if the owner's private key get compromised.

/clof_vesting/TokenVesting.sol

```
173 function withdraw(uint256 amount) external nonReentrant onlyOwner{
174     require(
175         getWithdrawableAmount() >= amount,
176         "TokenVesting: not enough withdrawable funds"
177     );
178     /*
179 * @dev Replaced owner() with msg.sender => address of WITHDRAWER_ROLE
180 */
181     SafeERC20.safeTransfer(_token, msg.sender, amount);
182 }
```

Recommendation

The project team is recommended to carefully protect the private key of the contract owner. It is also a good practice to use a multi-signature wallet as the owner of the contract.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

I-11|Optimization Suggestion - Redundant Computation When Function revoke Calls release

Informational : Optimization Suggestion

File Location : /clof_vesting/TokenVesting.sol:159,203

Description

Redundant computation occurs when the function revoke calls the function release, as vestedAmount is calculated twice in each of these functions. This results in unnecessary gas consumption.

/clof_vesting/TokenVesting.sol

```
149  function revoke(
150      bytes32 vestingScheduleId
151  ) external onlyOwner onlyIfVestingScheduleNotRevoked(vestingScheduleId) {
152      VestingSchedule storage vestingSchedule = vestingSchedules[
153          vestingScheduleId
154      ];
155      require(
156          vestingSchedule.revocable,
157          "TokenVesting: vesting is not revocable"
158      );
159      uint256 vestedAmount = _computeReleasableAmount(vestingSchedule);
160      if (vestedAmount > 0) {
161          release(vestingScheduleId, vestedAmount);
162      }
163      uint256 unreleased = vestingSchedule.amountTotal -
164          vestingSchedule.released;
165      vestingSchedulesTotalAmount = vestingSchedulesTotalAmount - unreleased;
166      vestingSchedule.revoked = true;
167  }
```

/clof_vesting/TokenVesting.sol

```
189  function release(
190      bytes32 vestingScheduleId,
191      uint256 amount
192  ) public nonReentrant onlyIfVestingScheduleNotRevoked(vestingScheduleId) {
193      VestingSchedule storage vestingSchedule = vestingSchedules[
194          vestingScheduleId
195      ];
196      bool isBeneficiary = msg.sender == vestingSchedule.beneficiary;
197
198      bool isReleasor = (msg.sender == owner());
199      require(
200          isBeneficiary || isReleasor,
201          "TokenVesting: only beneficiary and owner can release vested tokens"
202      );
203      uint256 vestedAmount = _computeReleasableAmount(vestingSchedule);
```

Recommendation

It is recommended to create an _release function with internal visibility, which is solely responsible for releasing tokens. Both the revoke and release functions should call the _release function to release tokens.

Alleviation

Acknowledged. TokenVesting team decided to keep no change.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

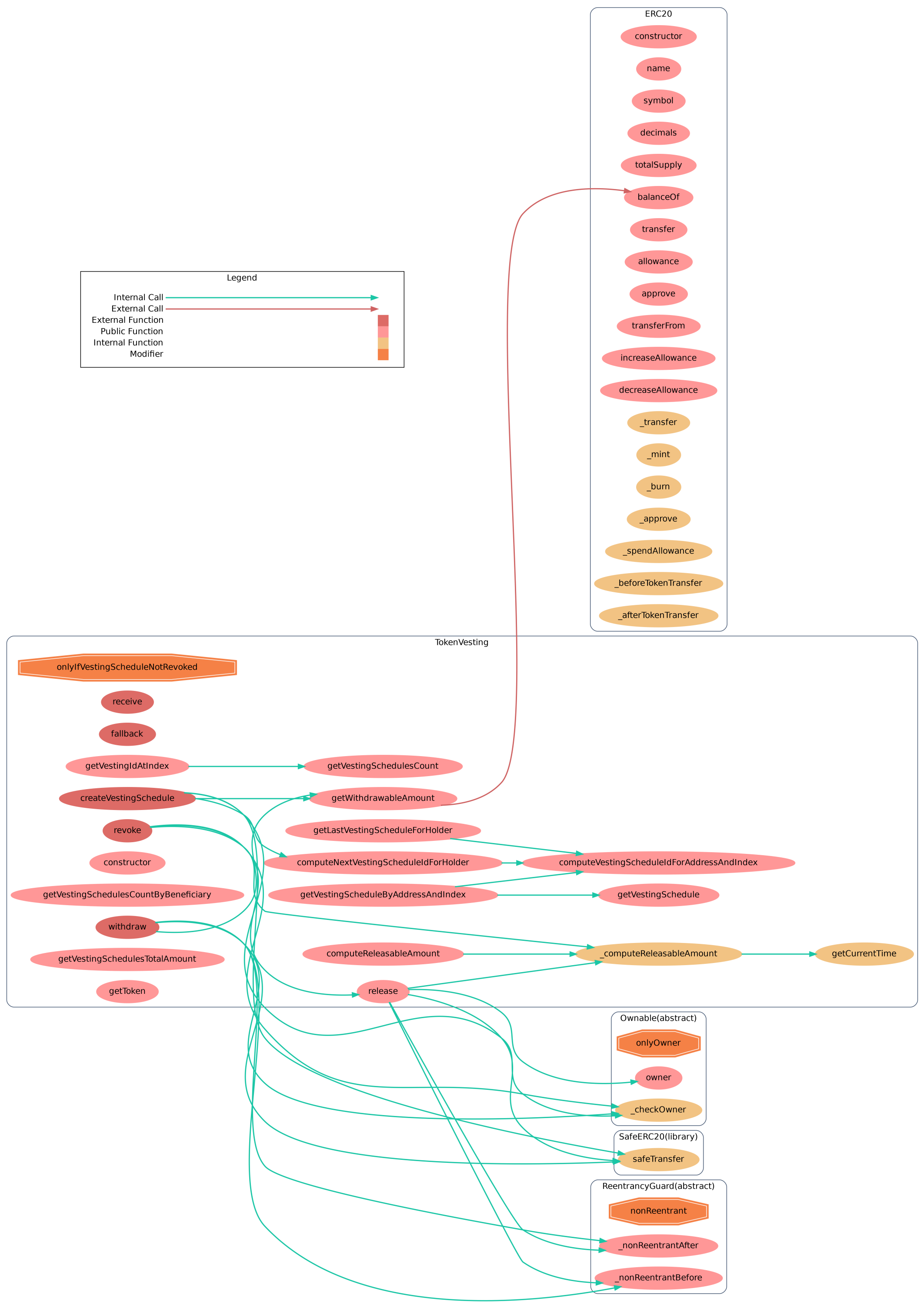
Contract	FuncName	Visibility	Mutability	Modifiers
TokenVesting	_CTOR_	public	Y	
TokenVesting	receive	external	N	
TokenVesting	fallback	external	N	
TokenVesting	createVestingSchedule	external	Y	onlyOwner
TokenVesting	revoke	external	Y	onlyOwner,onlyIfVestingScheduleNotRevoked
TokenVesting	withdraw	external	Y	nonReentrant,onlyOwner
TokenVesting	release	public	Y	nonReentrant,onlyIfVestingScheduleNotRevoked
TokenVesting	getVestingSchedulesCountByBeneficiary	public	N	
TokenVesting	getVestingIdAtIndex	public	N	
TokenVesting	getVestingScheduleByAddressAndIndex	public	N	
TokenVesting	getVestingSchedulesTotalAmount	public	N	
TokenVesting	getToken	public	N	
TokenVesting	getVestingSchedulesCount	public	N	
TokenVesting	computeReleasableAmount	public	N	onlyIfVestingScheduleNotRevoked
TokenVesting	getVestingSchedule	public	N	
TokenVesting	getWithdrawableAmount	public	N	

Contract	FuncName	Visibility	Mutability	Modifiers
TokenVesting	computeNextVestingScheduleIdForHolder	public	N	
TokenVesting	getLastVestingScheduleForHolder	public	N	
TokenVesting	computeVestingScheduleIdForAddressAndIndex	public	N	
TokenVesting	_computeReleasableAmount	internal	N	
TokenVesting	getCurrentTime	internal	N	

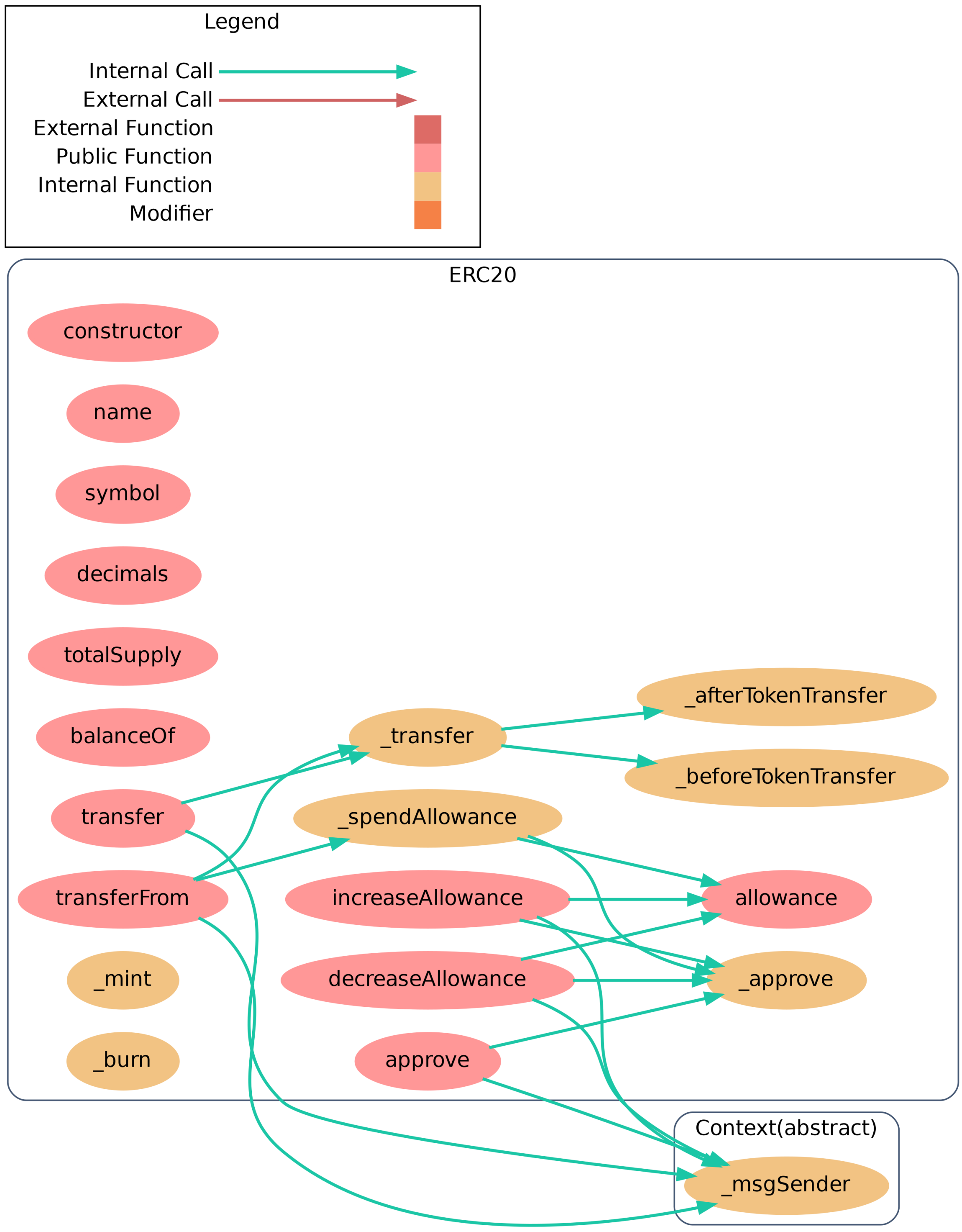
5. Appendix

5.2 Call Graph

TokenVesting



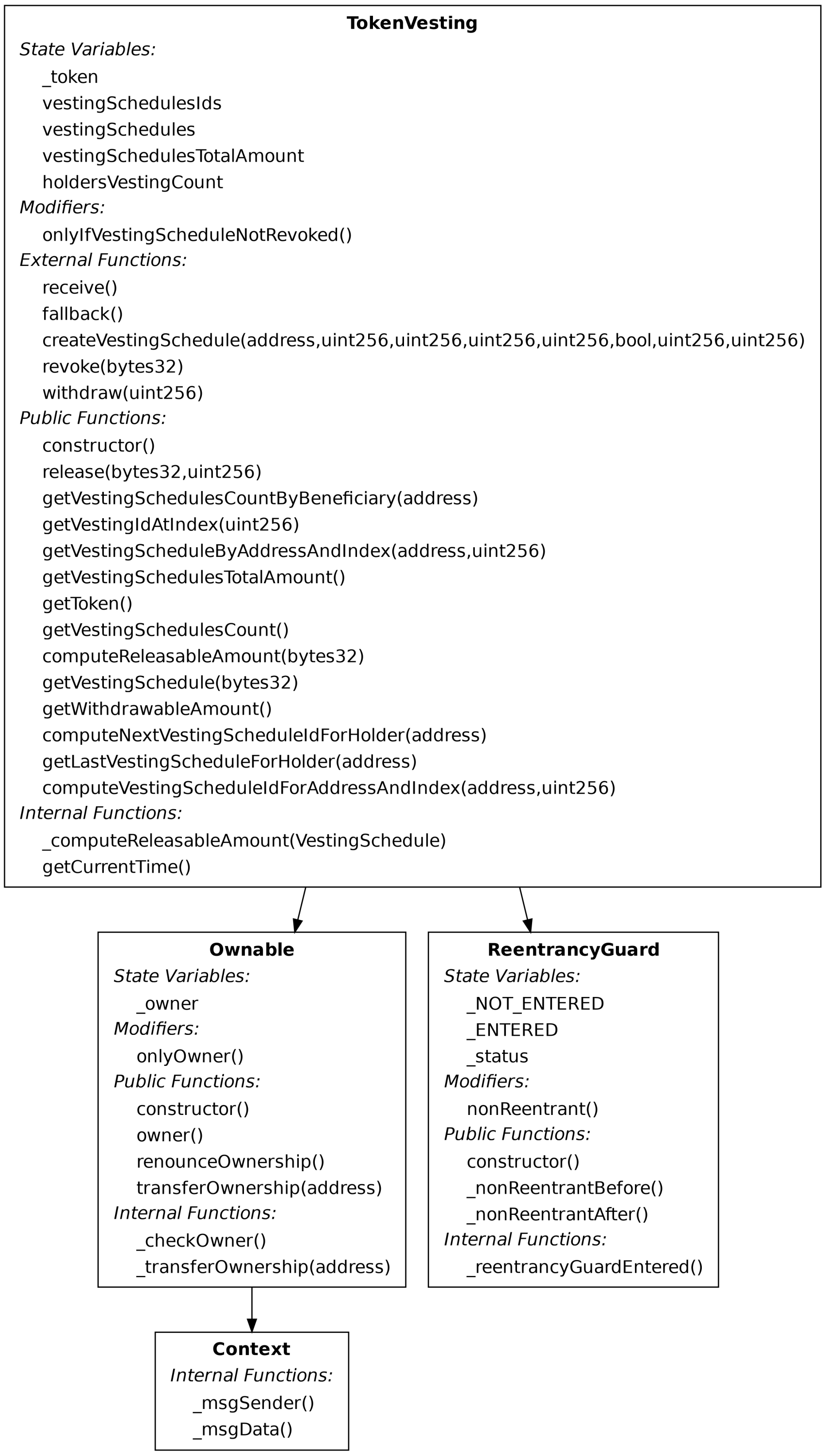
ERC20


Legend
Internal Call
External Call
External Function
Public Function
Internal Function
Modifier
ERC20
constructor
name
symbol
decimals
totalSupply
balanceOf
transfer
transferFrom
_mint
_burn
_transfer
_spendAllowance
increaseAllowance
decreaseAllowance
approve
_afterTokenTransfer
_beforeTokenTransfer
allowance
_approve
Context(abstract)
_msgSender

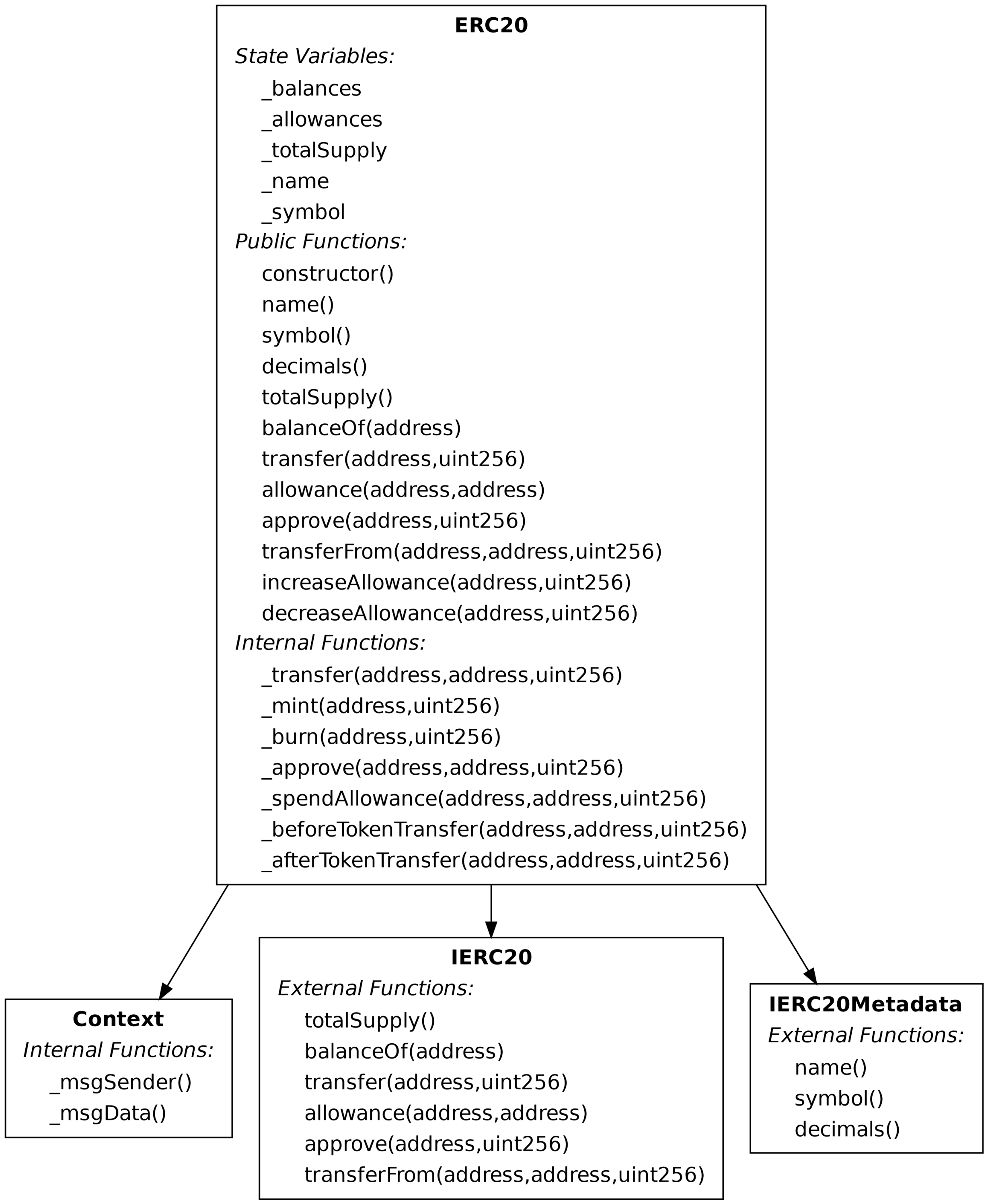
5. Appendix

5.3 Inheritance Graph

TokenVesting


TokenVesting
State Variables:
_token
vestingSchedulesIds
vestingSchedules
vestingSchedulesTotalAmount
holdersVestingCount
Modifiers:
onlyIfVestingScheduleNotRevoked()
External Functions:
receive()
fallback()
createVestingSchedule(address,uint256,uint256,uint256,uint256,bool,uint256,uint256)
revoke(bytes32)
withdraw(uint256)
Public Functions:
constructor()
release(bytes32,uint256)
getVestingSchedulesCountByBeneficiary(address)
getVestingIdAtIndex(uint256)
getVestingScheduleByAddressAndIndex(address,uint256)
getVestingSchedulesTotalAmount()
getToken()
getVestingSchedulesCount()
computeReleasableAmount(bytes32)
getVestingSchedule(bytes32)
getWithdrawableAmount()
computeNextVestingScheduleIdForHolder(address)
getLastVestingScheduleForHolder(address)
computeVestingScheduleIdForAddressAndIndex(address,uint256)
Internal Functions:
_computeReleasableAmount(VestingSchedule)
getCurrentTime()
Ownable
State Variables:
_owner
Modifiers:
onlyOwner()
Public Functions:
constructor()
owner()
renounceOwnership()
transferOwnership(address)
Internal Functions:
_checkOwner()
_transferOwnership(address)
ReentrancyGuard
State Variables:
_NOT_ENTERED
_ENTERED
_status
Modifiers:
nonReentrant()
Public Functions:
constructor()
_nonReentrantBefore()
_nonReentrantAfter()
Internal Functions:
_reentrancyGuardEntered()
Context
Internal Functions:
_msgSender()
_msgData()

ERC20


ERC20
State Variables:
_balances
_allowances
_totalSupply
_name
_symbol
Public Functions:
constructor()
name()
symbol()
decimals()
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)
increaseAllowance(address,uint256)
decreaseAllowance(address,uint256)
Internal Functions:
_transfer(address,address,uint256)
_mint(address,uint256)
_burn(address,uint256)
_approve(address,address,uint256)
_spendAllowance(address,address,uint256)
_beforeTokenTransfer(address,address,uint256)
_afterTokenTransfer(address,address,uint256)
Context
Internal Functions:
_msgSender()
_msgData()
IERC20
External Functions:
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)
IERC20Metadata
External Functions:
name()
symbol()
decimals()