Jul.25, 2023

ANTCHAIN | ZAN

Security Assessment

MITx-v2

—

Express Service

Table of Contents

1. Overview

1.1. Executive Summary

MITx is a standard ERC20 token project. This report has been prepared for MITx 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 11 informational issues. MITx-v2 team acknowledged all those issues. The high vulnerability and 2 informational issues are fixed while the others are retained as before.

1.2. Project Summary

Project Name	MITx-v2
Platform	Ethereum
Language	Solidity
Code Repository	https://github.com/Morpheuslabs-io/MITx-v2
Commit	3232630c0c3400dc672bef3a3afe2be5b2a58599

1.3. Assessment Summary

Delivery Date	Jul.25, 2023
Audit Methodology	Static Analysis, Formal Verification

1.4. Assessment Scope

ID	File	File Hash
1	/MITx-v2/contracts/Distribution.sol	7c045951e256408286e026bfbd2ed370
2	/MITx-v2/contracts/DistributionManager.sol	9a2b089b293c93397b285401023ba864
3	/MITx-v2/contracts/interfaces/IDistributionManager.sol	e0a995213c294ef8247799c63066774a

1.4. Assessment Scope

ID	File	File Hash
4	/MITx-v2/contracts/MITx.sol	701be6b9fd6f7cb7232cef3d73b53727
5	/MITx-v2/contracts/Swap.sol	63c3f0e8c23dbd42cbed192e8d75c7c2
6	/MITx-v2/contracts/utils/Blockable.sol	8ff0bee045b96972d74694e55dfb3e87

2. Checklist

2.1. General Vulnerability

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

2.2. Code Conventions

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

2.3. Gas Optimization

Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

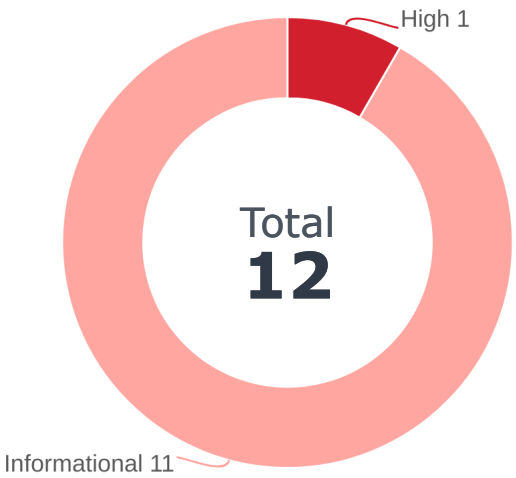
2.4. Compiler Bug

Affected by Compiler Bug

2.5. Logical Issue

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

3. Findings



Code Security

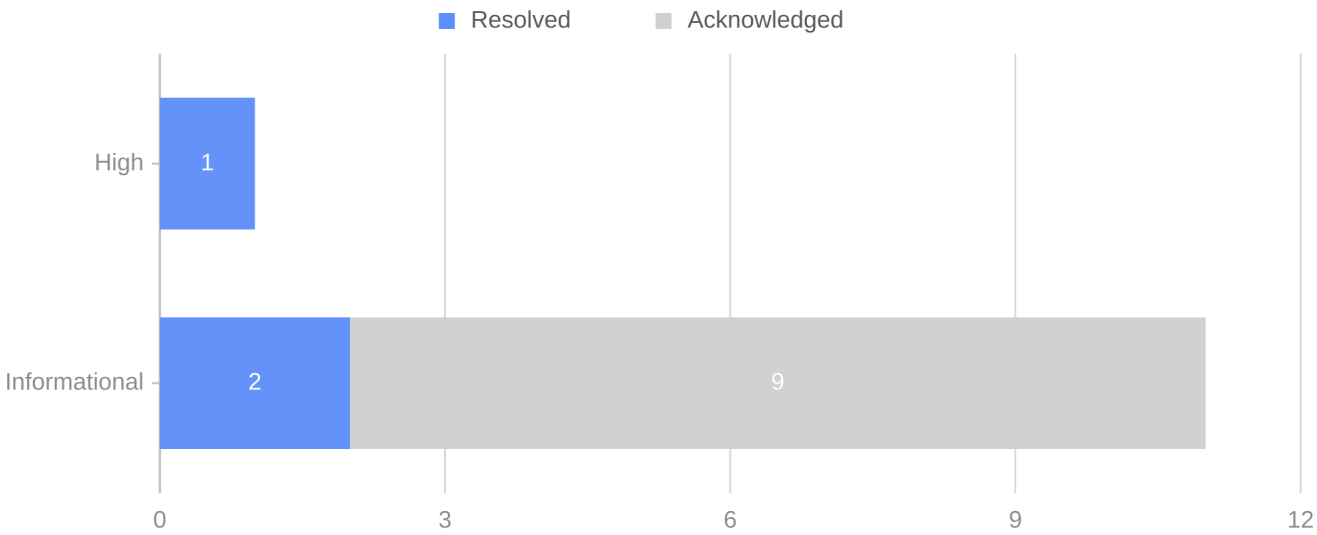
Total: 1, High: 1

Optimization Suggestion

Total: 11, Informational: 11

Business Security

Total: 0, no vulnerabilities.



ID	Title	Category	Severity	Status
H-01	Code Security - Unauthenticated Access of emergency Makes swap Not Work	Code Security	High	Resolved
I-02	Optimization Suggestion - Parameters in External Function should declared as Calldata	Optimization Suggestion	Informational	Acknowledged

ID	Title	Category	Severity	Status
I-03	Optimization Suggestion - Recommended to Follow Code layout Conventions	Optimization Suggestion	Informational	Acknowledged
I-04	Optimization Suggestion - No Check of Address Params with Zero Address	Optimization Suggestion	Informational	Resolved
I-05	Optimization Suggestion - Use ++i/--i instead of i++/i--	Optimization Suggestion	Informational	Acknowledged
I-06	Optimization Suggestion - Use != 0 Instead of > 0 for Unsigned Integer Comparison	Optimization Suggestion	Informational	Acknowledged
I-07	Optimization Suggestion - Interface Defined Not Inherited	Optimization Suggestion	Informational	Acknowledged
I-08	Optimization Suggestion - Floating Pragma	Optimization Suggestion	Informational	Acknowledged
I-09	Optimization Suggestion - Unused Import Contract	Optimization Suggestion	Informational	Resolved
I-10	Optimization Suggestion - Use CustomError Instead of String	Optimization Suggestion	Informational	Acknowledged
I-11	Optimization Suggestion - Get Contract Balance of ETH in Assembly	Optimization Suggestion	Informational	Acknowledged
I-12	Optimization Suggestion - Use Assembly to Check Zero Address	Optimization Suggestion	Informational	Acknowledged

H-01|Code Security - Unauthenticated Access of emergency Makes swap Not Work

High : Code Security

File Location : /MITx-v2/contracts/Swap.sol:128,112

Description

Since the ‘emergency’ function lacks authority control, anyone can call ‘emergency’ to transfer ERC20 or native tokens held by the contract to the owner. This will indirectly cause the transaction to fail as there are insufficient tokens transferred to the caller when the ‘swap’ function are called.

/MITx-v2/contracts/Swap.sol

```
 99      function swap(uint256 iAmount) external{
100          // First, calculate receiving amount of the new Token
101
    // calculate() already check whether `iAmount` exceeds a current balance of
    `msg.sender`
102          address caller = msg.sender;
103          uint256 oAmount = calculate(caller, iAmount);
104          require(oAmount > 0, "Receiving amount is 0");
105
106          // burn `iAmount` of the old Token
107          // Note: MITx-V1 Token didn't have burning function
108          // thus, transfer to 0x....dEaD will be used as alternative instead
109          OLD_TOKEN.safeTransferFrom(caller, BURN_ADDR, iAmount);
110
111          // after burning, transfer `oAmount` to `msg.sender`
112          NEW_TOKEN.safeTransfer(caller, oAmount);
113
114          emit Swapped(caller, iAmount, oAmount);
115      }
```

/MITx-v2/contracts/Swap.sol

```
126          - Three types of token can be withdrawed: Native Coin, ERC-20, and ERC
    -721
127      */
128      function emergency(address token, uint256 amount) external{
129          address receiver = owner();
130          if (token == address(0)) Address.sendValue(payable(receiver), amount);
```

Recommendation

Authenticate msg.sender when crucial state will be updated. e.g. require(msg.sender == ...).

Alleviation

Fixed in commit 3232630. MITx-v2 team applied the modifier onlyOwner to the function emergency to restrict its use to the owner.

I-02|Optimization Suggestion - Parameters in External Function should declared as Calldata

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/MITx.sol:17-20

Description

When the compiler parses the external function, it can directly read the function parameters from calldata. Setting it to other storage locations may waste gas. Under Solidity compiler 0.8.17, about 300-400 gas can be saved with optimization turned off while 120-150 gas can be saved vice versa.

/MITx-v2/contracts/MITx.sol

```
14      constructor(
15          uint256 maxSupply,
16          address tokenOwner,
17          address[] memory accounts,
18          uint256[] memory amounts,
19          string memory name,
20          string memory symbol
21      ) ERC20Capped(maxSupply) ERC20(name, symbol) {
```

Recommendation

In external functions, the storage location of function parameters should be set to calldata to save gas.

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-03|Optimization Suggestion - Recommended to Follow Code layout Conventions

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/DistributionManager.sol:8,/MITx-v2/contracts/Swap.sol:8

Description

In the solidity document(https://docs.soliditylang.org/en/v0.8.17/style-guide.html), there are the following conventions for code layout:
Layout contract elements in the following order: 1. Pragma statements, 2. Import statements, 3. Interfaces, 4. Libraries, 5. Contracts.
Inside each contract, library or interface, use the following order: 1. Type declarations, 2. State variables, 3. Events, 4. Modifiers, 5. Functions.
Functions should be grouped according to their visibility and ordered: 1. constructor, 2. receive function (if exists), 3. fallback function (if exists), 4. external, 5. public, 6. internal, 7. private.

/MITx-v2/contracts/DistributionManager.sol

```
6   import {Distribution} from "./Distribution.sol";
7
8   contract DistributionManager is Ownable{
9       uint256 private _counter;
10
```

/MITx-v2/contracts/Swap.sol

```
6   import {Ownable} from "@openzeppelin/contracts/access/Ownable.sol";
7
8   contract Swap is Ownable{
9       using SafeERC20 for IERC20;
10
```

Recommendation

It is recommended to follow the above code layout conventions to improve code readability.

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-04|Optimization Suggestion - No Check of Address Params with Zero Address

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/DistributionManager.sol:21,/MITx-v2/contracts/Swap.sol:30

Description

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

/MITx-v2/contracts/Swap.sol

```
27      );
28
29      constructor(
30          address contractOwner,
31          address oldToken,
32
```

/MITx-v2/contracts/DistributionManager.sol

```
19      );
20
21      constructor(address owner_){
22          // Ownable() set `msg.sender` as Owner by default
23          // Setting non-sender as `owner` is available in the version v5
```

Recommendation

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

Alleviation

Fixed in commit 3232630.

I-05|Optimization Suggestion - Use ++i/--i instead of i++/i--

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/DistributionManager.sol:63,/MITx-v2/contracts/MITx.sol:33,/MITx-v2/contracts/utils/Blockable.sol:64,53

Description

Compared with i++, ++i can save about 5 gas per use. Compared with i--, --i can save about 3 gas per use in for loop.

/MITx-v2/contracts/utils/Blockable.sol

```
51      function blacklist(address[] calldata accounts) external onlyOwner{
52          uint256 len = accounts.length;
53          for (uint256 i; i < len; i++) _update(accounts[i], true);
54      }
55
```

/MITx-v2/contracts/utils/Blockable.sol

```
62      function unblock(address[] calldata accounts) external onlyOwner{
63          uint256 len = accounts.length;
64          for (uint256 i; i < len; i++) _update(accounts[i], false);
65      }
66
```

/MITx-v2/contracts/DistributionManager.sol

```
61          list = new address[](len);
62
63          for (uint256 i; i < len; i++) list[i] = _vestingByCounter[from + i];
64      }
65
```

/MITx-v2/contracts/MITx.sol

```
31          uint256 len = accounts.length;
32          require(amounts.length == len, "Length mismatch");
33          for (uint256 i; i < len; i++) _mint(accounts[i], amounts[i]);
34
35          // Ownable() set `msg.sender` as Owner by default
```

Recommendation

It is recommended to use ++i/--i instead of i++/i-- in for loop.

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-06|Optimization Suggestion - Use != 0 Instead of > 0 for Unsigned Integer Comparison

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/Swap.sol:39,40

Description

For unsigned integers, use !=0 for comparison, which consumes less gas than >0. Under Solidity compiler 0.8.17, when compiler optimization is turned off, about 3 gas can be saved. When compiler optimization is turned on, no gas can be saved.

/MITx-v2/contracts/Swap.sol

```
37              oldToken != address(0) &&
38                  newToken != address(0) &&
39                  iRate > 0 &&
40                  oRate > 0,
41              "Invalid settings"
```

/MITx-v2/contracts/Swap.sol

```
38                  newToken != address(0) &&
39                  iRate > 0 &&
40                  oRate > 0,
41              "Invalid settings"
42          );
```

Recommendation

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

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-07|Optimization Suggestion - Interface Defined Not Inherited

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/DistributionManager.sol:8

Description

There is an interface definition, but the interface is not inherited when the contract is defined.

/MITx-v2/contracts/DistributionManager.sol

```
6  import {Distribution} from "./Distribution.sol";
7
8  contract DistributionManager is Ownable{
9      uint256 private _counter;
10
```

Recommendation

It is recommended that the contract inherits the relevant interface when it is defined.

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-08|Optimization Suggestion - Floating Pragma

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/Distribution.sol:1,/MITx-v2/contracts/DistributionManager.sol:1,/MITx-v2/contracts/MITx.sol:1,/MITx-v2/contracts/Swap.sol:1,/MITx-v2/contracts/interfaces/IDistributionManager.sol:1,/MITx-v2/contracts/utils/Blockable.sol:1

Description

Contracts should be deployed with the fixed compiler version that they have been tested thoroughly or make sure to pin the contract compiler version in the project config. Pinning the compiler version helps ensure that contracts are not compiled by untested compiler versions.

/MITx-v2/contracts/interfaces/IDistributionManager.sol

```
1 pragma solidity ^0.8.17;
2
3 interface IDistributionManager{
```

/MITx-v2/contracts/DistributionManager.sol
/MITx-v2/contracts/utils/Blockable.sol

```
1 pragma solidity ^0.8.17;
2
3 import {Ownable} from "@openzeppelin/contracts/access/Ownable.sol";
```

/MITx-v2/contracts/MITx.sol
/MITx-v2/contracts/Swap.sol

```
1 pragma solidity ^0.8.17;
2
3 import {SafeERC20, IERC20, Address} from
  "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

/MITx-v2/contracts/Distribution.sol

```
1 pragma solidity ^0.8.17;
2
3 import {VestingWallet, SafeERC20, IERC20, Address} from
  "@openzeppelin/contracts/finance/VestingWallet.sol";
```

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

Acknowledged. MITx-v2 team decided to keep no change.

I-09|Optimization Suggestion - Unused Import Contract

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/Swap.sol:5

Description

Import's contract is not used, which will increase gas consumption.

/MITx-v2/contracts/Swap.sol

```
3
4  import {SafeERC20, IERC20, Address} from
   "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
5  import {ERC20} from "@openzeppelin/contracts/token/ERC20/ERC20.sol";
6  import {Ownable} from "@openzeppelin/contracts/access/Ownable.sol";
7
```

Recommendation

It is recommended to delete unused Import contracts.

Alleviation

Fixed in commit 3232630.

I-10|Optimization Suggestion - Use CustomError Instead of String

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/Distribution.sol:86,82,/MITx-v2/contracts/DistributionManager.sol:119,153,/MITx-v2/contracts/MITx.sol:24,32,/MITx-v2/contracts/Swap.sol:36,89,104,/MITx-v2/contracts/utils/Blockable.sol:25,36

Description

When using revert, using CustomError is more gas efficient than string description. Because the error message described using CustomError is only compiled into four bytes. Under Solidity compiler 0.8.17, in a reverted transaction, when compiler optimization is turned off, about 250-270 gas can be saved. When compiler optimization is turned on, about 60-80 gas can be saved.

/MITx-v2/contracts/Distribution.sol

```
80      */
81      function revoke(address token, uint256 amount) external{
82          require(
83              _msgSender() == address(DISTRIBUTION_MANAGER),
84              "Distribution: unauthorized request"
```

/MITx-v2/contracts/DistributionManager.sol

```
151      ) external onlyOwner {
152          address vesting = vestingAddress(beneficiary);
153          require(vesting != address(0), "Beneficiary not found");
154          Distribution(payable(vesting)).revoke(token, amount);
155
```

/MITx-v2/contracts/Swap.sol

```
102          address caller = msg.sender;
103          uint256 oAmount = calculate(caller, iAmount);
104          require(oAmount > 0, "Receiving amount is 0");
105
106          // burn `iAmount` of the old Token
```

/MITx-v2/contracts/Swap.sol

```
34          uint256 oRate
35      ) {
36          require(
37              oldToken != address(0) &&
38                  newToken != address(0) &&
```

/MITx-v2/contracts/MITx.sol

```
30
    // Wouldn't check sum of `amounts` and MAX_SUPPLY. It could be checked off-
    chain instead
```

```
31          uint256 len = accounts.length;
32          require(amounts.length == len, "Length mismatch");
33          for (uint256 i; i < len; i++) _mint(accounts[i], amounts[i]);
34
```

/MITx-v2/contracts/MITx.sol

```
22          // Only need to check `tokenOwner` must be NOT 0x00
23          // ERC20Capped already check `maxSupply` must be non-zero
24          require(tokenOwner != address(0), "Set 0x00 as Owner");
25
26          // set a max number of MITx-V2 Token
```

/MITx-v2/contracts/Distribution.sol

```
84              "Distribution: unauthorized request"
85          );
86          require(REVOCABLE, "Distribution: contract not revocable");
87          address receiver = DISTRIBUTION_MANAGER.owner();
88
```

/MITx-v2/contracts/Swap.sol

```
87      ) public view returns (uint256 oAmount) {
88          uint256 balance = OLD_TOKEN.balanceOf(account);
89          require(balance >= iAmount, "Swapping amount exceeds balance");
90          oAmount = (iAmount * OUT_RATE) / IN_RATE;
91      }
```

/MITx-v2/contracts/utils/Blockable.sol

```
34    */
35      function disable() external onlyOwner{
36          require(blockable(), "Blockable: not enabled yet");
37          delete _blockable;
38
```

/MITx-v2/contracts/utils/Blockable.sol

```
23    */
24      function enable() external onlyOwner{
25          require(!blockable(), "Blockable: already enabled");
26          _blockable = true;
27
```

/MITx-v2/contracts/DistributionManager.sol

```
117          bool isRevocable
118      ) external onlyOwner returns (address vesting) {
119          require(
120              vestingAddress(beneficiary) == address(0),
121              "Beneficiary registered already"
```

Recommendation

When reverting, it is recommended to use CustomError instead of ordinary strings to describe the error message.

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-11|Optimization Suggestion - Get Contract Balance of ETH in Assembly

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/Distribution.sol:62

Description

Using the selfbalance and balance opcodes to get the ETH balance of the contract in assembly saves gas compared to getting the ETH balance through address(this).balance and xx.balance. Under Solidity compiler 0.8.17, when compiler optimization is turned off, about 210-250 gas can be saved, and when compiler optimization is turned on, about 50-100 gas can be saved.

/MITx-v2/contracts/Distribution.sol

```
60              released_ = released();
61              releasable_ = releasable();
62              remain = address(this).balance;
63          } else {
64              released_ = released(token);
```

Recommendation

It is recommended to get the contract ETH balance in assembly.

Alleviation

Acknowledged. MITx-v2 team decided to keep no change.

I-12|Optimization Suggestion - Use Assembly to Check Zero Address

Informational : Optimization Suggestion

File Location : /MITx-v2/contracts/Distribution.sol:89,59,/MITx-v2/contracts/DistributionManager.sol:120,94,153,/MITx-v2/contracts/MITx.sol:24,84,/MITx-v2/contracts/Swap.sol:37,38,130

Description

Using assembly to check zero address can save gas. Under Solidity compiler 0.8.17, about 18 gas can be saved in each call.

/MITx-v2/contracts/DistributionManager.sol

```
151      ) external onlyOwner {
152          address vesting = vestingAddress(beneficiary);
153          require(vesting != address(0), "Beneficiary not found");
154          Distribution(payable(vesting)).revoke(token, amount);
155
```

/MITx-v2/contracts/MITx.sol

```
22          // Only need to check `tokenOwner` must be NOT 0x00
23          // ERC20Capped already check `maxSupply` must be non-zero
24          require(tokenOwner != address(0), "Set 0x00 as Owner");
25
26          // set a max number of MITx-V2 Token
```

/MITx-v2/contracts/DistributionManager.sol

```
118      ) external onlyOwner returns (address vesting) {
119          require(
120              vestingAddress(beneficiary) == address(0),
121              "Beneficiary registered already"
122          );
```

/MITx-v2/contracts/DistributionManager.sol

```
92      {
93          address vesting = vestingAddress(beneficiary);
94          if (vesting != address(0))
95              (, start, end, released, releasable, remain) = Distribution(
96                  payable(vesting)
```

/MITx-v2/contracts/Distribution.sol

```
57          start_ = start();
58          end = start_ + duration();
59          if (token == address(0)) {
60              released_ = released();
61              releasable_ = releasable();
```

/MITx-v2/contracts/Swap.sol

```
35      ) {
36          require(
37              oldToken != address(0) &&
38                  newToken != address(0) &&
39                  iRate > 0 &&
```

/MITx-v2/contracts/MITx.sol
/MITx-v2/contracts/Swap.sol

```
1       function emergency(address token, uint256 amount) external{
2           address receiver = owner();
3           if (token == address(0)) Address.sendValue(payable(receiver), amount);
4           else IERC20(token).safeTransfer(receiver, amount);
5       }
```

/MITx-v2/contracts/Swap.sol

```
36          require(
37              oldToken != address(0) &&
38                  newToken != address(0) &&
39                  iRate > 0 &&
40                  oRate > 0,
```

/MITx-v2/contracts/Distribution.sol

```
87          address receiver = DISTRIBUTION_MANAGER.owner();
88
89          if (token == address(0)) Address.sendValue(payable(receiver), amount);
90          else IERC20(token).safeTransfer(receiver, amount);
91
```

Recommendation

It is recommended to use assembly to check zero address.

Alleviation

Acknowledged. MITx-v2 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

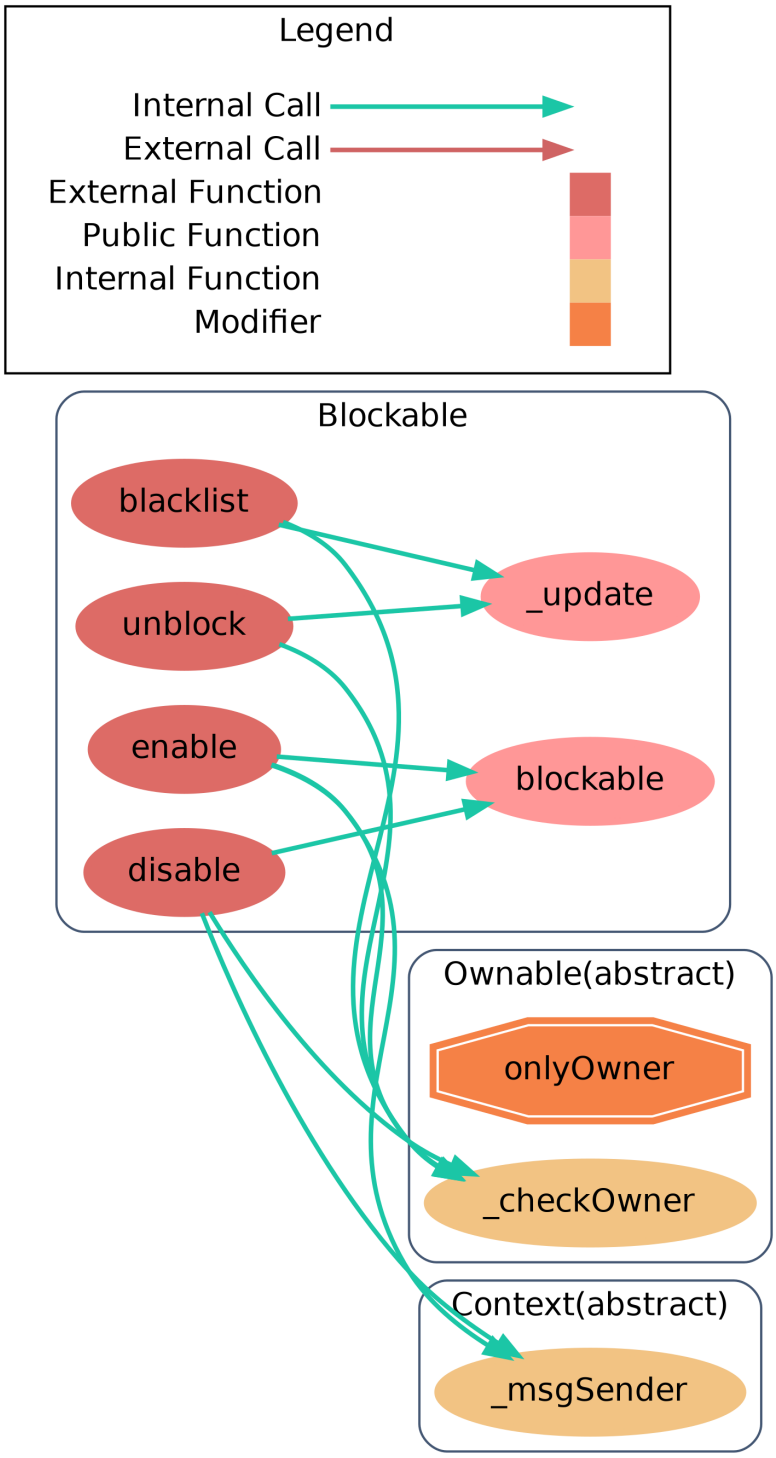
Contract	FuncName	Visibility	Mutability	Modifiers
Blockable	enable	external	Y	onlyOwner
Blockable	disable	external	N	onlyOwner
Blockable	blacklist	external	Y	onlyOwner
Blockable	unblock	external	Y	onlyOwner
Blockable	blockable	public	N	
Blockable	_update	private	N	
Distribution	_CTOR_	public	Y	VestingWallet
Distribution	info	external	N	
Distribution	revoke	external	N	
DistributionManager	_CTOR_	public	Y	
DistributionManager	counter	public	N	
DistributionManager	vestingAddress	public	N	
DistributionManager	vestingAddresses	external	N	
DistributionManager	vestingInfo	external	N	
DistributionManager	addBeneficiary	external	Y	onlyOwner
DistributionManager	revoke	external	N	onlyOwner

Contract	FuncName	Visibility	Mutability	Modifiers
MITx	_CTOR_	public	Y	ERC20Capped,ERC20
MITx	mint	external	Y	onlyOwner
MITx	burn	external	Y	
MITx	emergency	external	N	
Swap	_CTOR_	public	Y	
Swap	info	external	N	
Swap	calculate	public	N	
Swap	swap	external	N	
Swap	emergency	external	N	onlyOwner

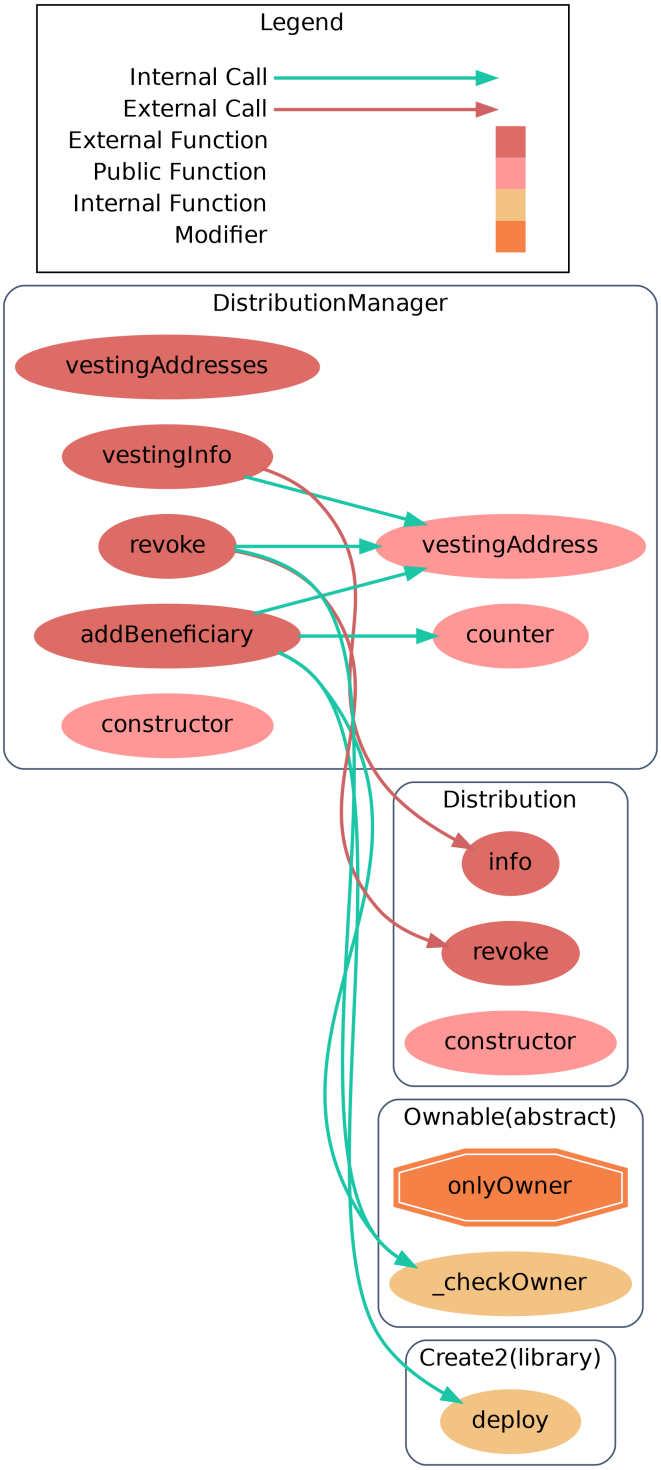
5. Appendix

5.2 Call Graph

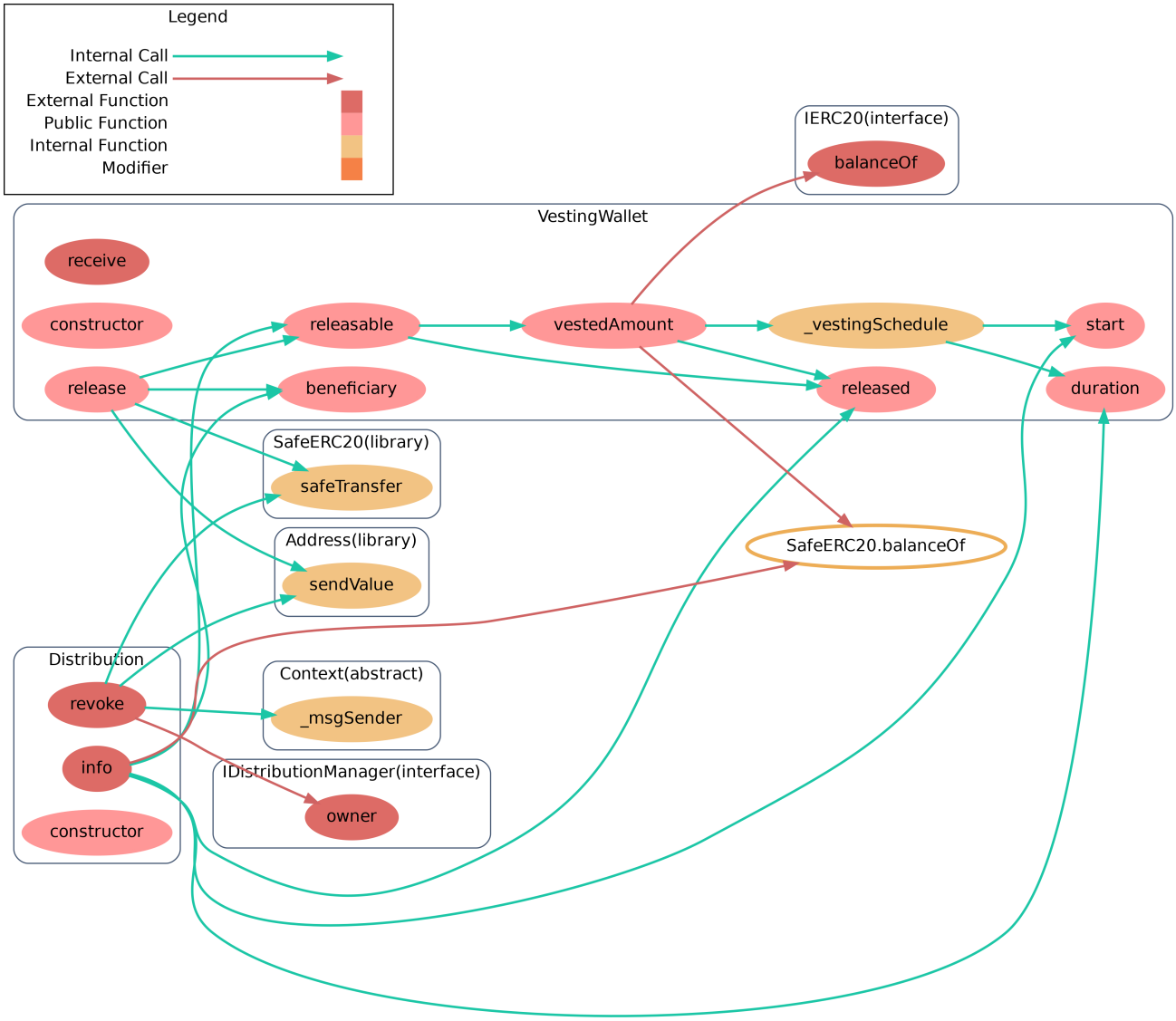
Blockable



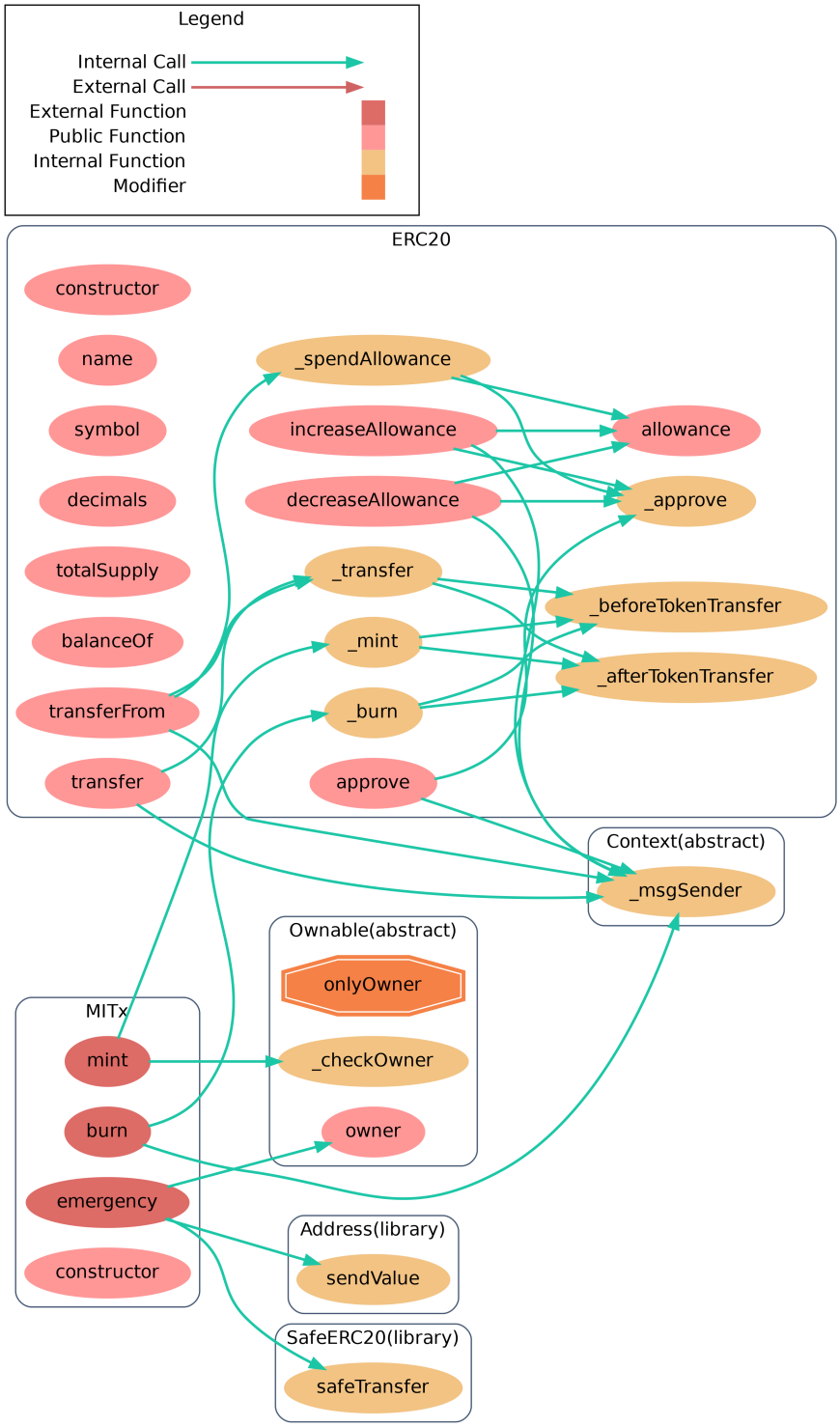
DistributionManager



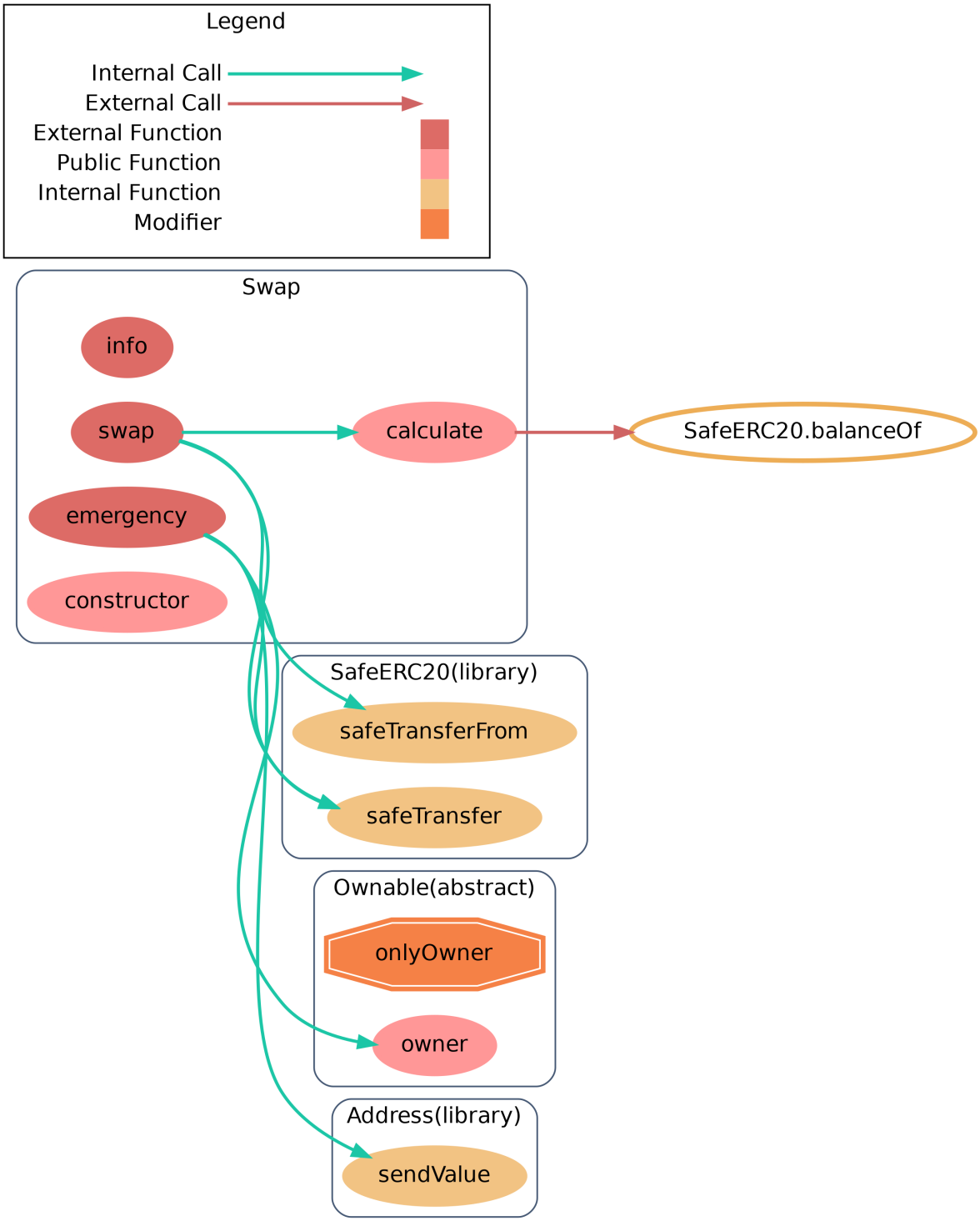
Distribution



MITx


Legend
Internal Call
External Call
External Function
Public Function
Internal Function
Modifier
ERC20
constructor
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
increaseAllowance
decreaseAllowance
_transfer
_mint
_burn
approve
allowance
_approve
_beforeTokenTransfer
_afterTokenTransfer
Context(abstract)
_msgSender
Ownable(abstract)
onlyOwner
_checkOwner
owner
MITx
mint
burn
emergency
constructor
Address(library)
sendValue
SafeERC20(library)
safeTransfer

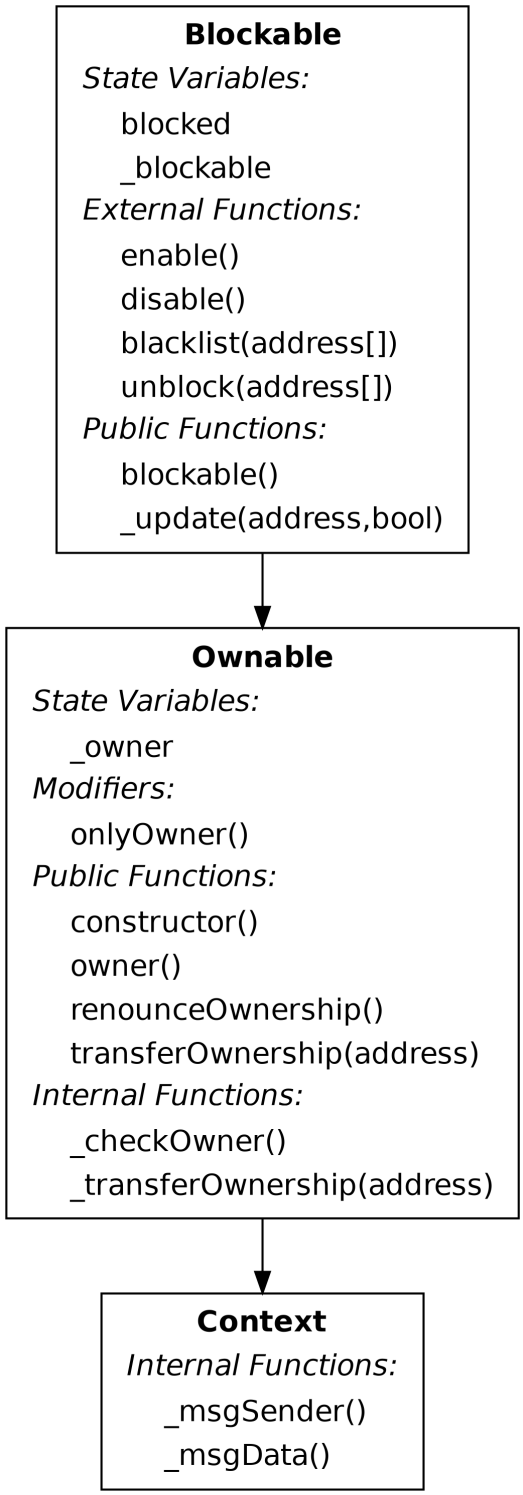
Swap


Legend
Internal Call
External Call
External Function
Public Function
Internal Function
Modifier
Swap
info
swap
calculate
SafeERC20.balanceOf
emergency
constructor
SafeERC20(library)
safeTransferFrom
safeTransfer
Ownable(abstract)
onlyOwner
owner
Address(library)
sendValue

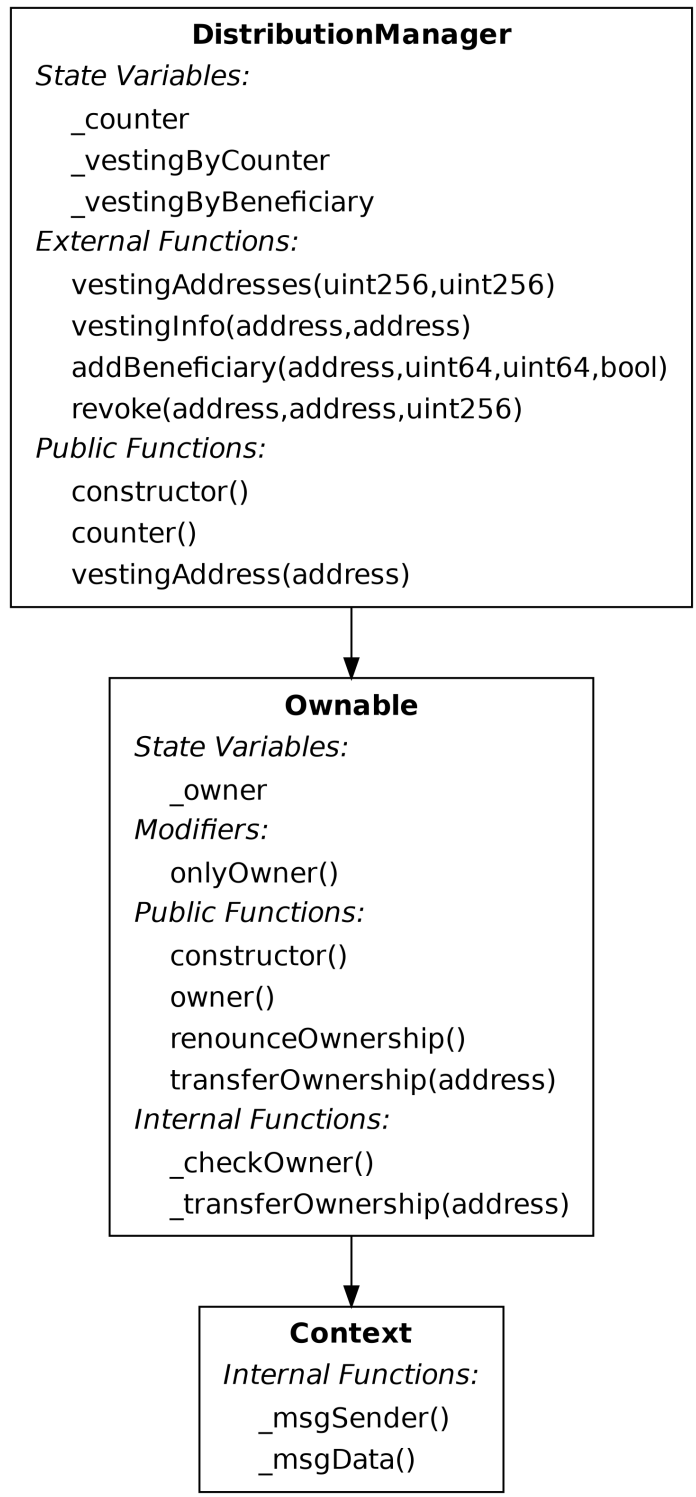
5. Appendix

5.3 Inheritance Graph

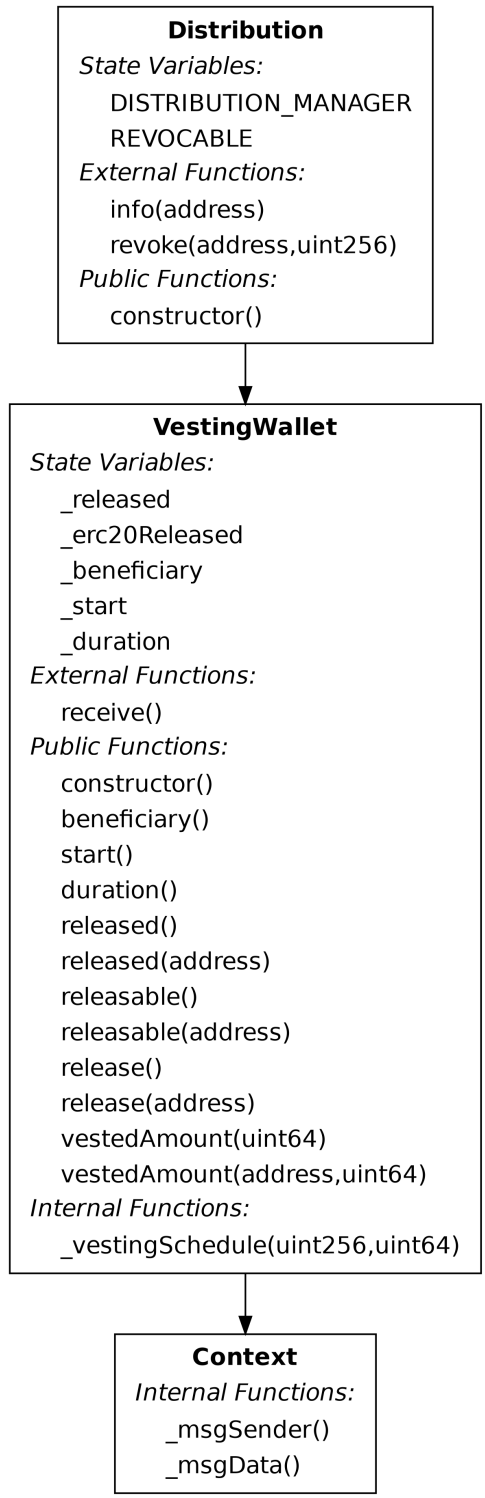
Blockable



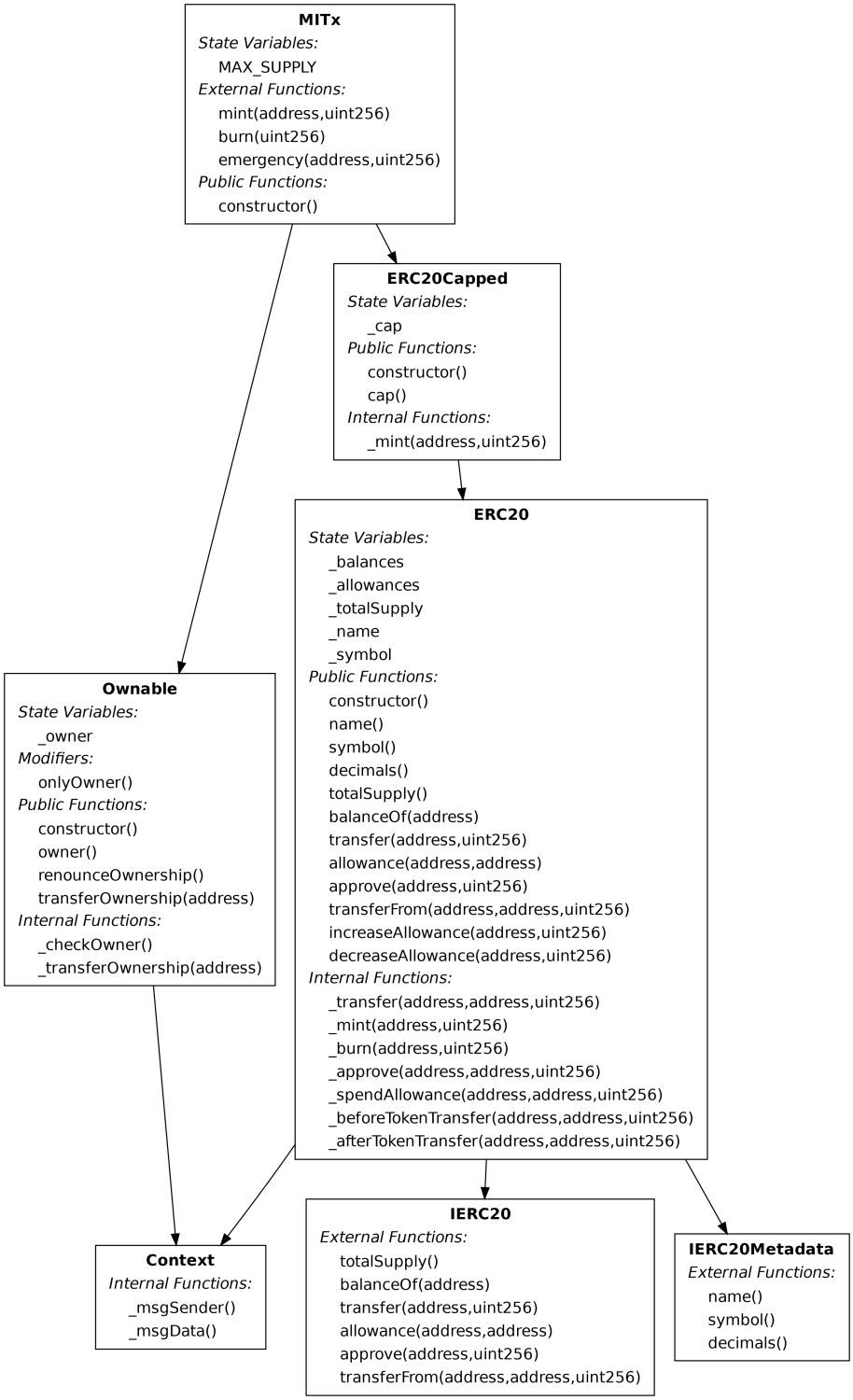
DistributionManager



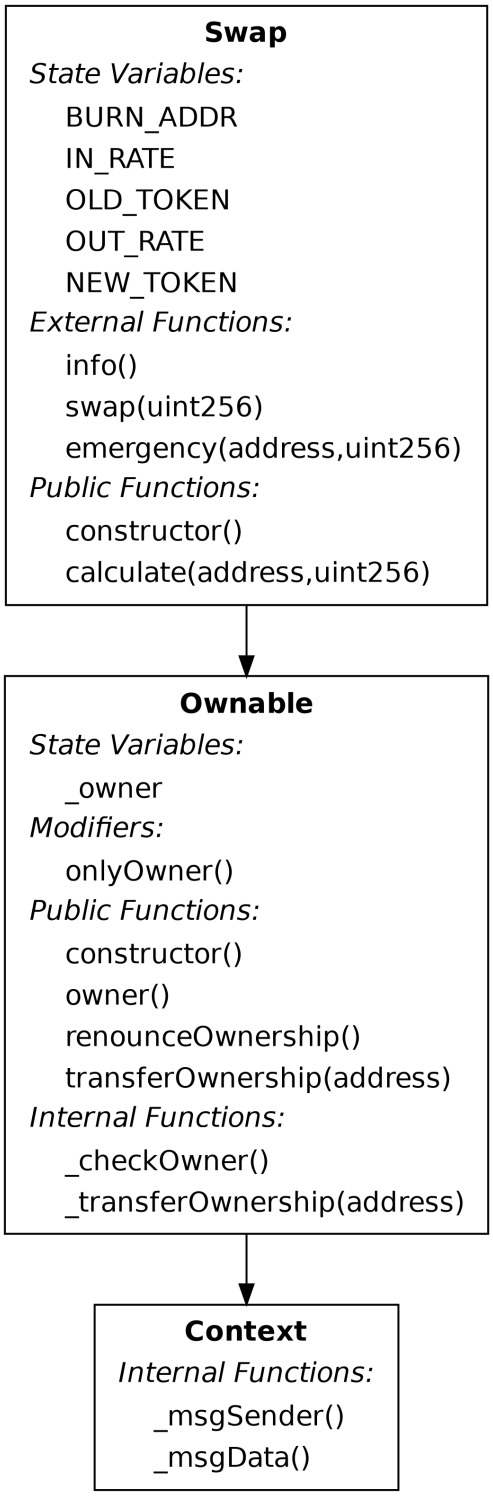
Distribution



MITx


MITx
State Variables:
MAX_SUPPLY
External Functions:
mint(address,uint256)
burn(uint256)
emergency(address,uint256)
Public Functions:
constructor()
ERC20Capped
State Variables:
_cap
Public Functions:
constructor()
cap()
Internal Functions:
_mint(address,uint256)
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)
Ownable
State Variables:
_owner
Modifiers:
onlyOwner()
Public Functions:
constructor()
owner()
renounceOwnership()
transferOwnership(address)
Internal Functions:
_checkOwner()
_transferOwnership(address)
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()

Swap


Swap
State Variables:
BURN_ADDR
IN_RATE
OLD_TOKEN
OUT_RATE
NEW_TOKEN
External Functions:
info()
swap(uint256)
emergency(address,uint256)
Public Functions:
constructor()
calculate(address,uint256)
Ownable
State Variables:
_owner
Modifiers:
onlyOwner()
Public Functions:
constructor()
owner()
renounceOwnership()
transferOwnership(address)
Internal Functions:
_checkOwner()
_transferOwnership(address)
Context
Internal Functions:
_msgSender()
_msgData()