Smart Contract Security Audit V1

Vesting Implementation Smart Contract Audit

Jun 30, 2025

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**: Binance Smart Chain
- **Name**: VestingImplementation
- **Language** : Solidity
- **Contract Address**: 0x3cf3d6877320131c639a0d4ac08063bbebd29dac
- **Code Source**: https://github.com/TerraDharitri/drt-pREWA/tree/main/contracts

VestingImplementation:
Token Vesting Schedule Manager

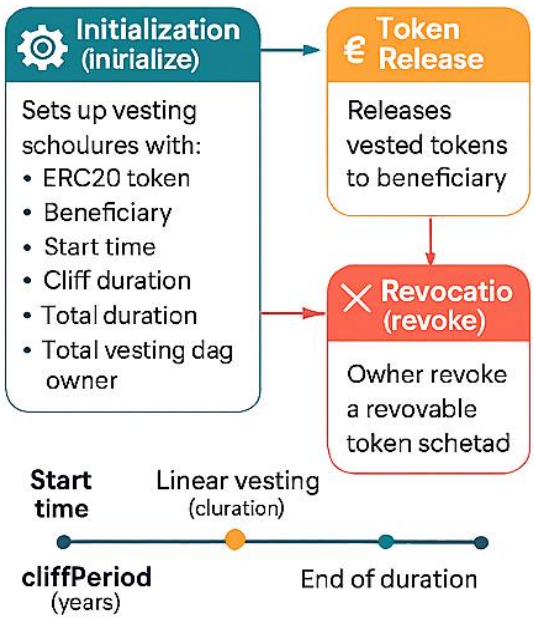
Contract Overview

- **Purpose:** Managses linear ERe+1.0 tonens for a specficiary over a specified period with eptional cliff and revocation
- **Use Case:** MIT
- **Solidity Version:** 0.8.28.

Key Components

Component	Description
Imports	Sets up vesting schedule with (ERC20)
Inhertance	Token, beneficcary, start time, total duration. total duration
Modifiers	Vesting Storage IVesting. Emergen ware, EmergencyAwareBase

Core Functionality



- **Use Case:** MIT
- **Key Features:**

I1u Initializalieiinl), struct

Inerrerat YGuerd, Upgradeaable, VstrgControloller, EmergencyAware, EmergencyController, errors, Constants, EmergencyAwareBase

- **Revocation**
  - vested Amount (timesfamp): calculates vested after cliff
  - return unvested tokens to beneficlary
  - ErmuViets/fevokd: event

Events

- OwnershipTransfered
- EmergencyControllerSet
- OracleIntegrationSet
- TokensReleased
- VestingRevoked

Security Features

- Upgradeabllity
  uses, initializable forproxy compatibility
- Reentrancy Protection
  ReennergencyGuardUpgradable
- Safe Transfers
  Safet RC20Upgradable for secure token tiransters
- Emergency Protocol
  Integration with Emergenvcystem-wide pauses

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

The files:

VestingImplementation.sol

Audit Score:

99% secure

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
VestingImplementation.sol	ca981ea3841d8d1e6371ba6f4148f0fe72524502	0x3cf3d6877320131c639a0d4ac08063bbebd29dac

- Contract: VestingImplementation
- Inherit: Initializable, ReentrancyGuardUpgradeable, VestingStorage, IVesting, EmergencyAwareBase
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
checkEmergencyStatus	✓	Read / public	**Passed**
emergencyController	✓	Read / public	**Passed**
getEmergencyController	✓	Read / public	**Passed**
getFactoryAddress	✓	Read / public	**Passed**
getTokenAddress	✓	Read / public	**Passed**
getVestingSchedule	✓	Read / public	**Passed**
isEmergencyPaused	✓	Read / public	**Passed**
oracleIntegration	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
releasableAmount	✓	Read / public	**Passed**
vestedAmount	✓	Read / public	**Passed**
emergencyShutdown	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**

initialize	✓	Write / public	**Passed**
initialize	✓	Write / public	**Passed**
release	✓	Write / public	**Passed**
revoke	✓	Write / public	**Passed**
setEmergencyController	✓	Write / public	**Passed**
setOracleIntegration	✓	Write / public	**Passed**

Issues Checking Status

SWC Attack Analysis

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) for more info check https://swcregistry.io/

No.	Issue Description	Checking Status
136	Unencrypted Private Data On-Chain	**Passed**
135	Code With No Effects	**Passed**
134	Message call with hardcoded gas amount	**Passed**
133	Hash Collisions With Multiple Variable Length Arguments	**Passed**
132	Unexpected Ether balance	**Passed**
131	Presence of unused variables	**Passed**
130	Right-To-Left-Override control character (U+202E)	**Passed**
129	Typographical Error	**Passed**
128	DoS with block gas limit.	**Passed**
127	Arbitrary Jump with Function Type Variable	**Passed**
126	Insufficient Gas Griefing	**Passed**
125	Incorrect Inheritance Order	**Passed**
124	Write to Arbitrary Storage Location	**Passed**
123	Requirement Violation	**Passed**
122	Lack of Proper Signature Verification	**Passed**
121	Missing Protection against Signature Replay Attacks	**Passed**
120	Weak Sources of Randomness from Chain Attributes	**Passed**
119	Shadowing State Variables	**Passed**

118	Incorrect Constructor Name	**Passed**
117	Signature Malleability	**Passed**
116	Block values as a proxy for time	**Not Passed**
115	Authorization through tx.origin	**Passed**
114	Transaction Order Dependence	**Passed**
113	DoS with Failed Call	**Passed**
112	Delegatecall to Untrusted Callee	**Passed**
111	Use of Deprecated Solidity Functions	**Passed**
110	Assert Violation	**Passed**
109	Uninitialized Storage Pointer	**Passed**
108	State Variable Default Visibility	**Passed**
107	Reentrancy	**Passed**
106	Unprotected SELFDESTRUCT Instruction	**Passed**
105	Unprotected Ether Withdrawal	**Passed**
104	Unchecked Call Return Value	**Passed**
103	Floating Pragma	**Not Passed**
102	Outdated Compiler Version	**Passed**
101	Integer Overflow and Underflow	**Passed**
100	Function Default Visibility	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found.

Low:

`_owner` vs. OpenZeppelin's `OwnableUpgradeable`

Description

The contract implements its own `_owner` state variable, `onlyOwner` modifier, `OwnershipTransferred` event, and `transferOwnership` function. This is a common pattern for contracts that don't inherit OpenZeppelin's `OwnableUpgradeable`. However, given the project uses OpenZeppelin extensively, it's worth noting this custom implementation. While not an immediate vulnerability, it deviates from the standard pattern and means any new owner-related features in `OwnableUpgradeable` (e.g., `renounceOwnership`) would need to be manually implemented or accounted for.

Recommendation

Current implementation is acceptable if intentional: If the design specifically requires that each vesting contract has its own _owner that is not tied to OwnableUpgradeable's standard, then the current approach is functional. The custom onlyOwner is correctly implemented.(so it is normal in large project)

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.28 instead of ^0.8.28). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors. And avoid Solidity compiler Bugs check here

https://sepolia.etherscan.io/solcbuginfo

Remediation

Remove the ^ sign to lock the pragma version.

Use of block.timestamp for comparisons

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

```
if (tokenAddress_ == address(0)) revert Vesting_TokenZero();
        if (beneficiaryAddress_ == address(0)) revert
Vesting_BeneficiaryZero();
        if (initialOwnerAddress_ == address(0)) revert
Vesting_OwnerZeroV();
        if (durationValue_ == 0) revert Vesting_DurationZero();
        if (totalVestingAmount_ == 0) revert Vesting_AmountZeroV();
        if (cliffDurationValue_ > durationValue_) revert
Vesting_CliffLongerThanDuration();
        if (startTimeValue_ < block.timestamp && startTimeValue_ !=
0) revert Vesting_StartTimeInvalid();
```

Recommendation

Avoid use of block.timestamp.

#Unused Parameter in checkEmergencyStatus

Description

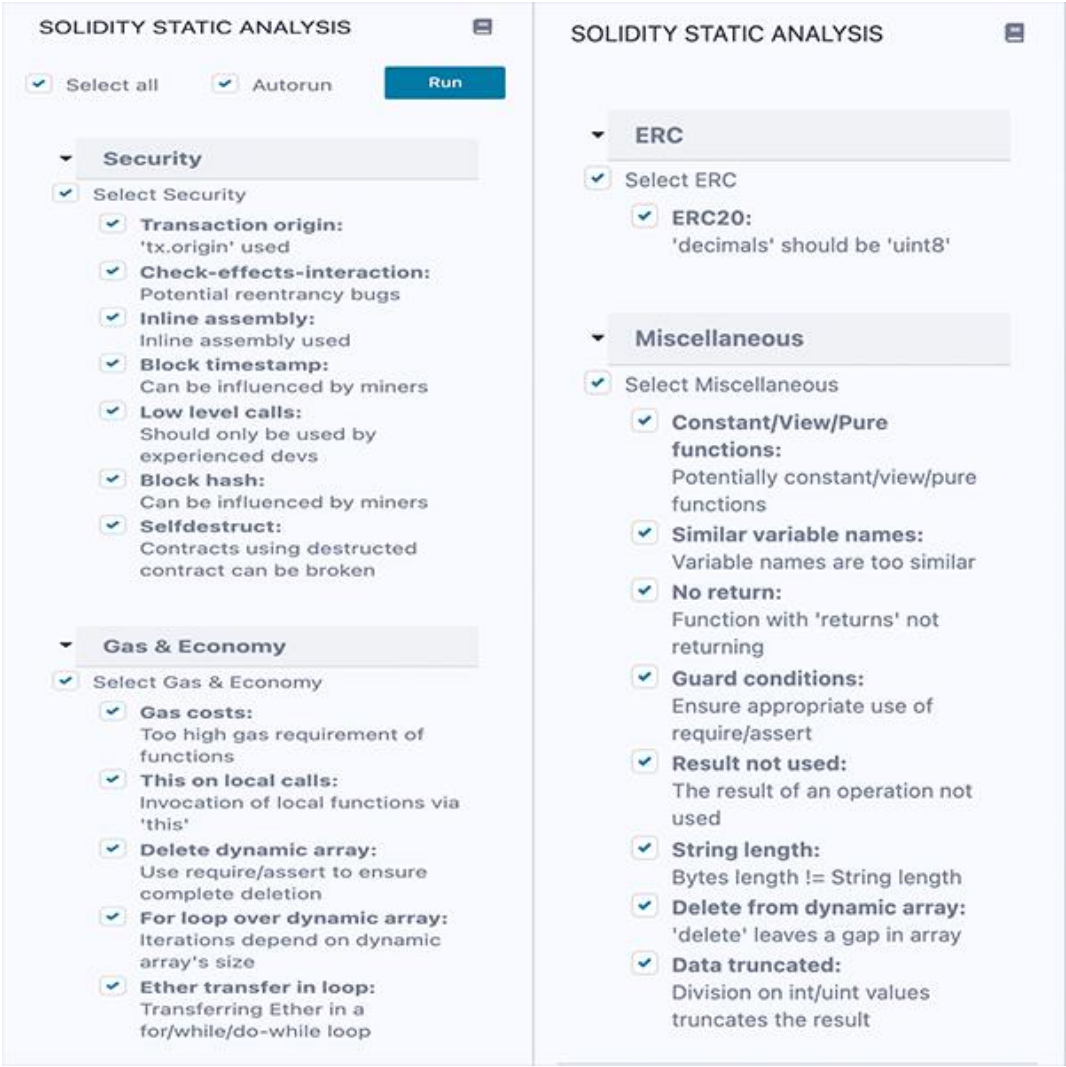
The checkEmergencyStatus function has a parameter bytes4 that is not used within the function body. This adheres to the IEmergencyAware interface, so it's not a functional bug, but it's good to note.

Recommendation

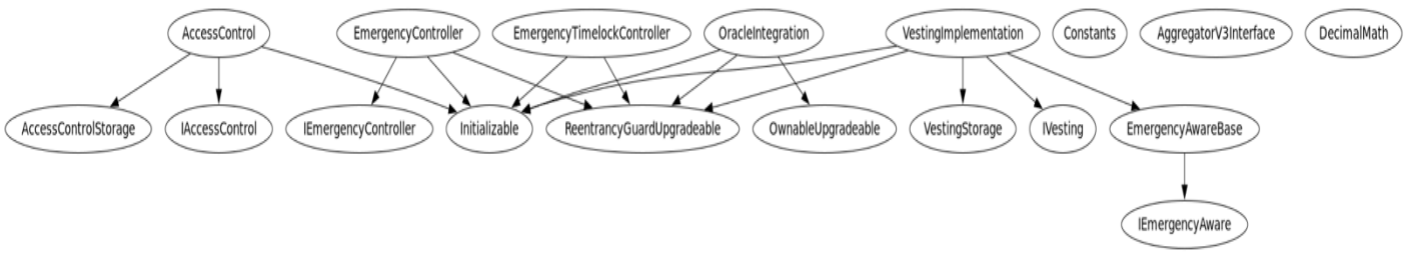
This is an interface requirement, so no change is needed within this contract. It's simply an observation. If you control the interface, you could consider if that parameter is always necessary.

Automatic Testing

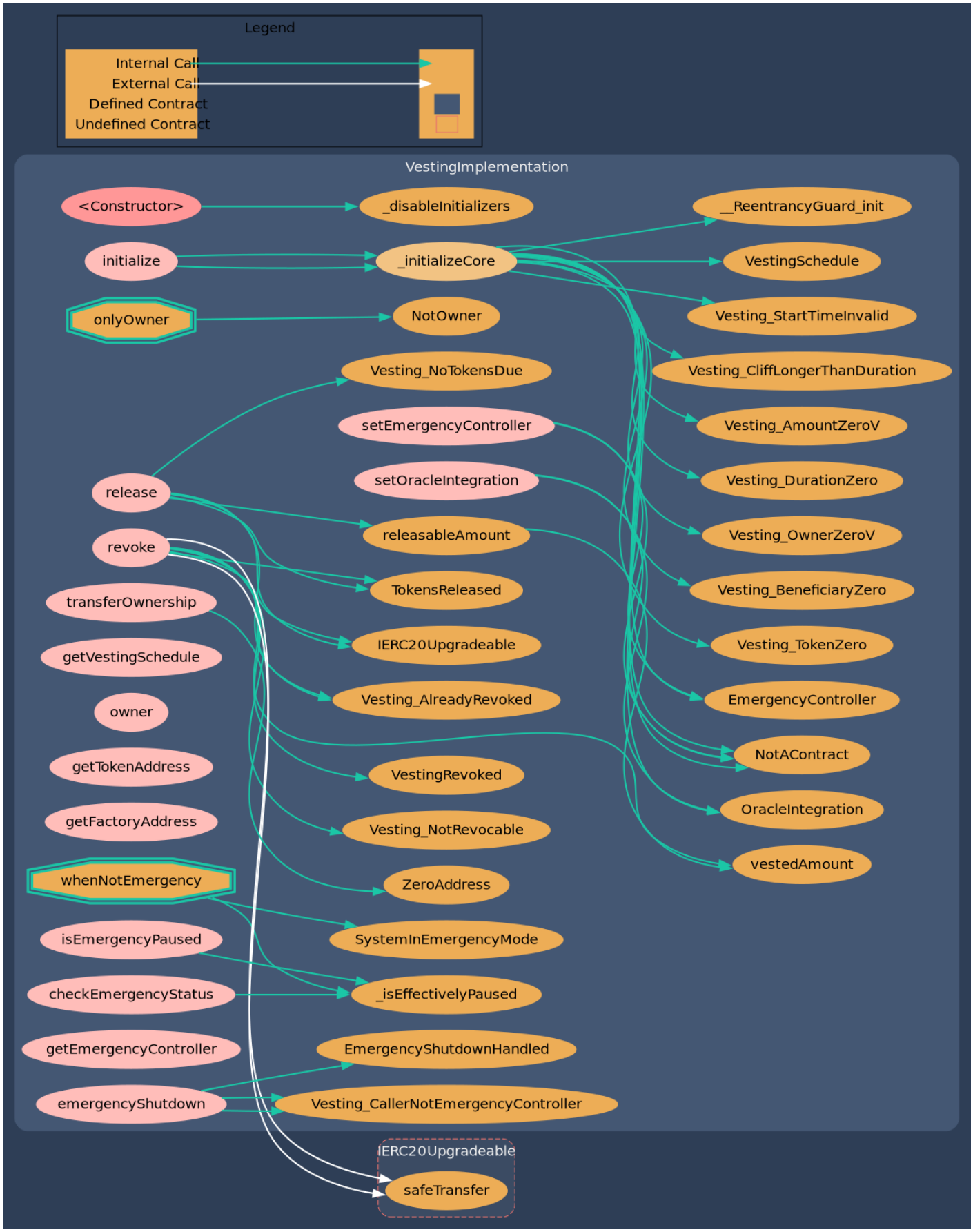
1- SOLIDITY STATIC ANALYSIS



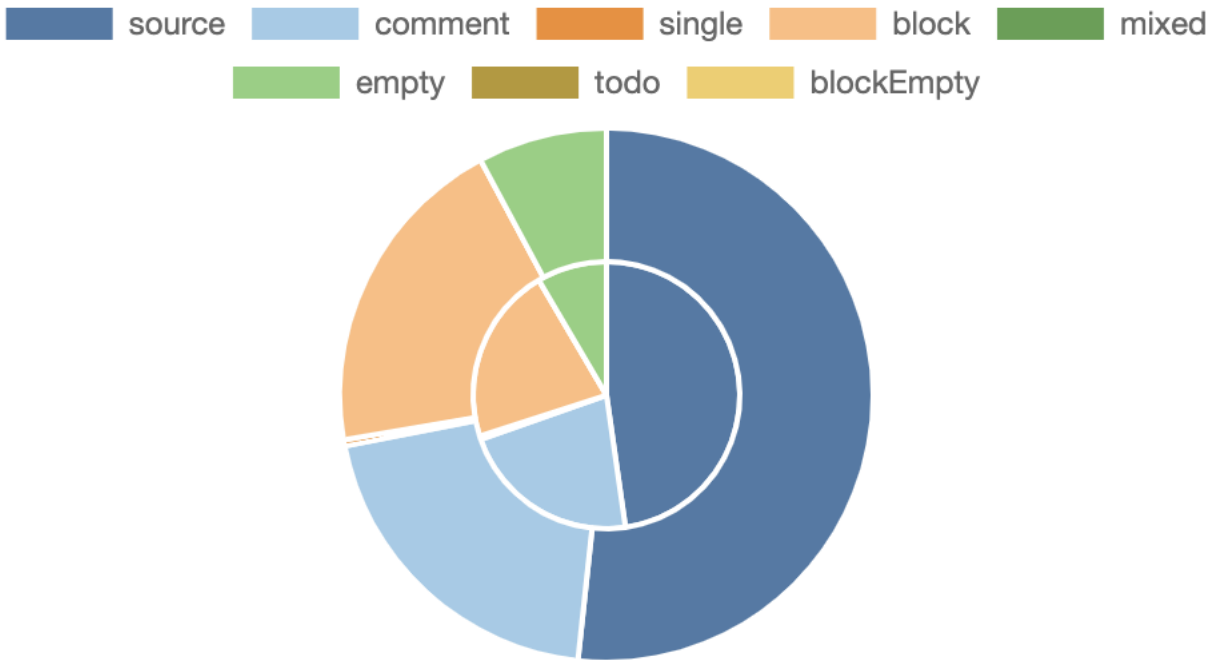
2- Inheritance graph



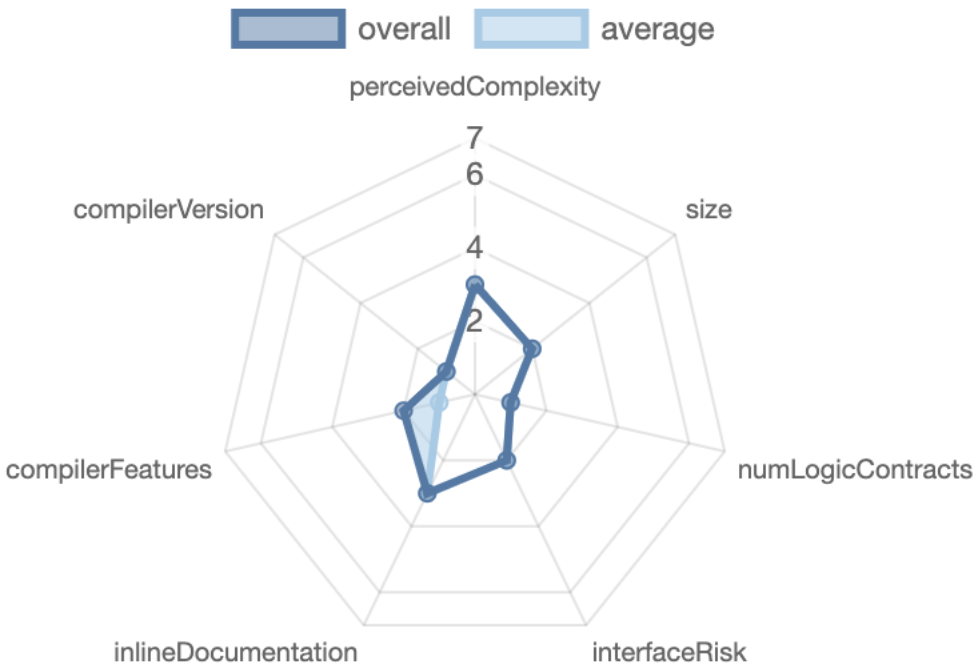
3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
VestingImplementation
<Constructor>
_disableInitializers
__ReentrancyGuard_init
initialize
_initializeCore
VestingSchedule
onlyOwner
NotOwner
Vesting_StartTimeInvalid
Vesting_NoTokensDue
Vesting_CliffLongerThanDuration
setEmergencyController
Vesting_AmountZeroV
setOracleIntegration
Vesting_DurationZero
release
releasableAmount
revoke
Vesting_OwnerZeroV
TokensReleased
transferOwnership
Vesting_BeneficiaryZero
IERC20Upgradeable
getVestingSchedule
Vesting_TokenZero
Vesting_AlreadyRevoked
EmergencyController
owner
NotAContract
getTokenAddress
VestingRevoked
OracleIntegration
getFactoryAddress
Vesting_NotRevocable
vestedAmount
whenNotEmergency
ZeroAddress
isEmergencyPaused
SystemInEmergencyMode
checkEmergencyStatus
_isEffectivelyPaused
getEmergencyController
EmergencyShutdownHandled
emergencyShutdown
Vesting_CallerNotEmergencyController
IERC20Upgradeable
safeTransfer

Source lines



Risk level



Source units in scope

Source Units in Scope

Source Units Analyzed: `1`
Source Units in Scope: `1` **(100%)**

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/vesting/VestingImplementation.sol	1	———	418	378	231	106	178	💻
📝	**Totals**	**1**	———	**418**	**378**	**231**	**106**	**178**	💻

Legend: [➖]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	💊Abstract
1	0	0	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
17	0

External	Internal	Private	Pure	View
15	11	0	0	9

StateVariables

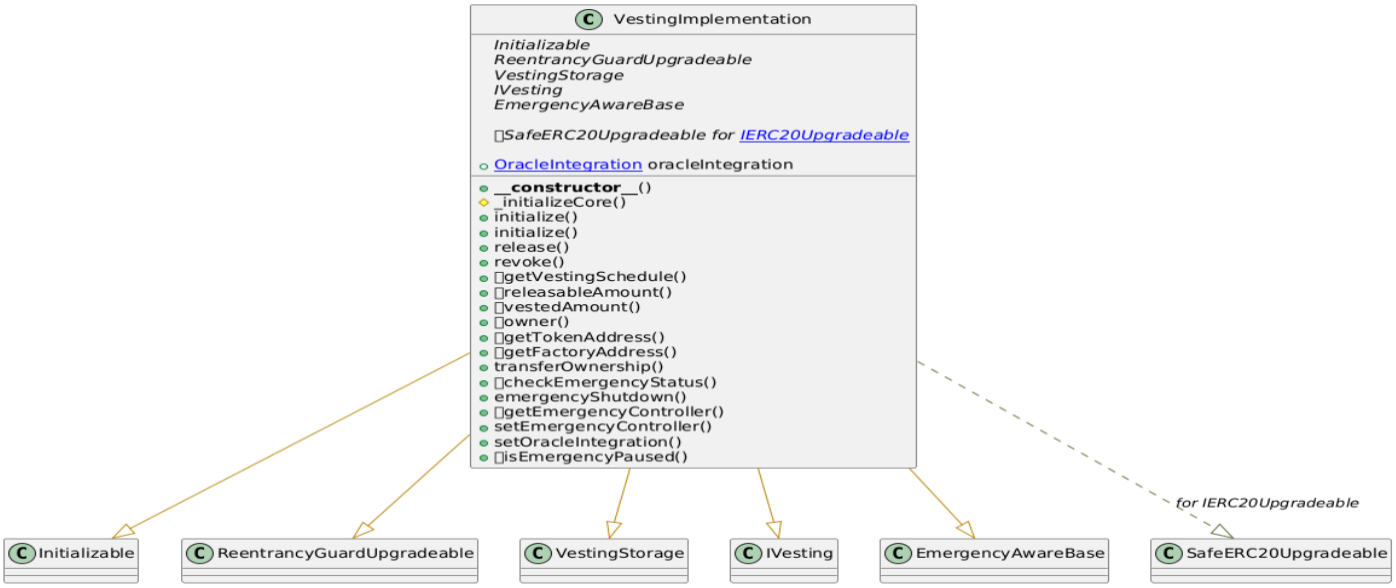
Total	🌐Public
1	1

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.8.28`		———	`yes` (4 asm blocks)	———

📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🦑 ECRecover	🌀 New/Create/Create2
———	———	———	———	———	———

Unified Modeling Language (UML)



Functions signature

Function Name	Sighash	Function Signature
initialize	3811ddea	initialize(address,address,uint256,uint256,uint256,bool,uint256,address,address,address)
initialize	318449be	initialize(address,address,uint256,uint256,uint256,bool,uint256,address)
release	86d1a69f	release()
revoke	b6549f75	revoke()
getVestingSchedule	3a05f0d8	getVestingSchedule()
releasableAmount	5b940081	releasableAmount()
vestedAmount	1bfce853	vestedAmount(uint256)
owner	8da5cb5b	owner()
getTokenAddress	10fe9ae8	getTokenAddress()
getFactoryAddress	a9c2e36c	getFactoryAddress()
transferOwnership	f2fde38b	transferOwnership(address)
checkEmergencyStatus	8aed668a	checkEmergencyStatus(bytes4)
emergencyShutdown	eb1676c1	emergencyShutdown(uint8)
getEmergencyController	c993cc9d	getEmergencyController()
setEmergencyController	6ca7dc89	setEmergencyController(address)
setOracleIntegration	a9361199	setOracleIntegration(address)
isEmergencyPaused	290d10c4	isEmergencyPaused()

Automatic general report

```
Files Description Table
|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/drt-
pREWA/contracts/vesting/VestingImplementation.sol |
ca981ea3841d8d1e6371ba6f4148f0fe72524502 |
| /Users/macbook/Desktop/drt-
pREWA/contracts/vesting/storage/VestingStorage.sol |
9bd80c22ca7b0ab86006be628fa326f59af151a2 |
| /Users/macbook/Desktop/drt-
pREWA/contracts/vesting/interfaces/IVesting.sol |
6c98d2baafb89fa90585b23233948daed42670fc |
| /Users/macbook/Desktop/drt-
pREWA/contracts/interfaces/IEmergencyAware.sol |
be6e0a371b0a5f62d5c7dc06e5c0c4121dcc7ca1 |
| /Users/macbook/Desktop/drt-
pREWA/contracts/controllers/EmergencyController.sol |
8e73765eaa98f95db8f5eb7a40cd13b0dc845266 |
| /Users/macbook/Desktop/drt-pREWA/contracts/access/AccessControl.sol |
40163fa249f10365c03cf1fa9ddd26e2f6eb1005 |
| /Users/macbook/Desktop/drt-
pREWA/contracts/access/storage/AccessControlStorage.sol |
2a1f4c3d6956a89011b090a62b15254b1d720d65 |
| /Users/macbook/Desktop/drt-
pREWA/contracts/access/interfaces/IAccessControl.sol |
540ec54eaade1c05a650af086ebf959654ec6322 |
| /Users/macbook/Desktop/drt-pREWA/contracts/libraries/Errors.sol |
0974f6f49e3b655fa93a2792154f1b506ec03c74 |
| /Users/macbook/Desktop/drt-
pREWA/contracts/interfaces/IEmergencyController.sol |
40bc149ec8a1da85c0e4c6170b06f3a0f3b77f3b |
| /Users/macbook/Desktop/drt-
pREWA/contracts/controllers/EmergencyTimelockController.sol |
94735e88ad7f750ead488b7ad618b1486cbdb483 |
| /Users/macbook/Desktop/drt-pREWA/contracts/libraries/Constants.sol |
62835a8af9d7ab21c0cad9ca03abff6cf4e8e68f |
| /Users/macbook/Desktop/drt-pREWA/contracts/oracle/OracleIntegration.sol
| 56835e91c3d168cbc4e565e742c575c2117bb3bb |
| /Users/macbook/Desktop/drt-
pREWA/contracts/interfaces/AggregatorV3Interface.sol |
008fa0a63980b32bdecef492c7da1ae5b441a5c3 |
| /Users/macbook/Desktop/drt-pREWA/contracts/libraries/DecimalMath.sol |
1f5b0a2a73498dc5cb877c08aea3c971646fb4a5 |
| /Users/macbook/Desktop/drt-pREWA/contracts/utils/EmergencyAwareBase.sol
| 528be04847c5d3cf03edd961c878359eb3cafc6a |

 Contracts Description Table
|  Contract  |         Type        |       Bases      |
|                |
```

|:----------:|:------------------:|:---------------:|:---------------
:|:--------------:|
| └ | **Function Name** | **Visibility** | **Mutability**
| **Modifiers** |
||||||
| **VestingImplementation** | Implementation | Initializable,
ReentrancyGuardUpgradeable, VestingStorage, IVesting, EmergencyAwareBase
└	<Constructor>	Public ❗	🛑	NO❗
└	_initializeCore	Internal 🔒	🛑	
└	initialize	External ❗	🛑	initializer
└	initialize	External ❗	🛑	initializer
└	release	External ❗	🛑	nonReentrant whenNotEmergency
└	revoke	External ❗	🛑	onlyOwner nonReentrant
└	getVestingSchedule	External ❗		NO❗
└	releasableAmount	Public ❗		NO❗
└	vestedAmount	Public ❗		NO❗
└	owner	External ❗		NO❗
└	getTokenAddress	External ❗		NO❗
└	getFactoryAddress	External ❗		NO❗
└	transferOwnership	External ❗	🛑	onlyOwner nonReentrant
└	checkEmergencyStatus	External ❗		NO❗
└	emergencyShutdown	External ❗	🛑	NO❗
└	getEmergencyController	External ❗		NO❗
└	setEmergencyController	External ❗	🛑	onlyOwner
└	setOracleIntegration	External ❗	🛑	onlyOwner
└	isEmergencyPaused	External ❗		NO❗
VestingStorage	Implementation			
IVesting	Interface			
└	initialize	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
└	release	External ❗	🛑	NO❗
└	revoke	External ❗	🛑	NO❗
└	getVestingSchedule	External ❗		NO❗
└	releasableAmount	External ❗		NO❗
└	vestedAmount	External ❗		NO❗
└	owner	External ❗		NO❗
IEmergencyAware	Interface			
└	checkEmergencyStatus	External ❗		NO❗
└	emergencyShutdown	External ❗	🛑	NO❗
└	getEmergencyController	External ❗		NO❗
└	setEmergencyController	External ❗	🛑	NO❗
└	isEmergencyPaused	External ❗		NO❗
EmergencyController	Implementation	Initializable,
ReentrancyGuardUpgradeable, IEmergencyController		
└	<Constructor>	Public ❗	🛑	NO❗
└	initialize	External ❗	🛑	initializer


└	setRequiredApprovals	External ❗	🛑	onlyAdminRole
└	setLevel3TimelockDuration	External ❗	🛑	onlyAdminRole
└	setRecoveryAdminAddress	External ❗	🛑	onlyAdminRole
└	approveLevel3Emergency	External ❗	🛑	onlyEmergencyRole nonReentrant
└	cancelLevel3Emergency	External ❗	🛑	onlyEmergencyRole nonReentrant
└	executeLevel3Emergency	External ❗	🛑	onlyEmergencyRole nonReentrant
└	setEmergencyLevel	External ❗	🛑	onlyEmergencyRole nonReentrant
└	enableEmergencyWithdrawal	External ❗	🛑	onlyEmergencyRole nonReentrant
└	pauseSystem	External ❗	🛑	onlyPauserRole nonReentrant
└	unpauseSystem	External ❗	🛑	onlyPauserRole nonReentrant
└	recoverTokens	External ❗	🛑	onlyEmergencyRole nonReentrant
└	registerEmergencyAwareContract	External ❗	🛑	onlyEmergencyRole
└	removeEmergencyAwareContract	External ❗	🛑	onlyEmergencyRole
└	processEmergencyForContract	External ❗	🛑	nonReentrant
└	updateFunctionRestriction	External ❗	🛑	onlyEmergencyRole
└	getEmergencyLevel	External ❗		NO❗
└	getEmergencyWithdrawalSettings	External ❗		NO❗
└	isSystemPaused	External ❗		NO❗
└	getEmergencyAwareContractsPaginated	External ❗		NO❗
└	getApprovalStatus	External ❗		NO❗
└	isFunctionRestricted	External ❗		NO❗
└	_resetApprovals	Internal 🔒	🛑	
AccessControl	Implementation	Initializable, AccessControlStorage, IAccessControl		
└	<Constructor>	Public ❗	🛑	NO❗
└	initialize	External ❗	🛑	initializer
└	hasRole	External ❗		NO❗
└	getRoleAdmin	External ❗		NO❗
└	grantRole	External ❗	🛑	NO❗
└	revokeRole	External ❗	🛑	NO❗
└	renounceRole	External ❗	🛑	NO❗
└	setRoleAdmin	External ❗	🛑	NO❗
└	getRoleMember	External ❗		NO❗
└	getRoleMemberCount	External ❗		NO❗
└	getRoleMembersPaginated	External ❗		NO❗
└	_grantRole	Internal 🔒	🛑	
└	_revokeRole	Internal 🔒	🛑	
└	_setRoleAdmin	Internal 🔒	🛑	
AccessControlStorage	Implementation			
IAccessControl	Interface			

└	hasRole	External ❗			NO❗
└	getRoleAdmin	External ❗			NO❗
└	grantRole	External ❗	🔘		NO❗
└	revokeRole	External ❗	🔘		NO❗
└	renounceRole	External ❗	🔘		NO❗
└	getRoleMember	External ❗			NO❗
└	getRoleMemberCount	External ❗			NO❗
└	getRoleMembersPaginated	External ❗			NO❗
IEmergencyController	Interface				
└	setEmergencyLevel	External ❗	🔘		NO❗
└	enableEmergencyWithdrawal	External ❗	🔘		NO❗
└	pauseSystem	External ❗	🔘		NO❗
└	unpauseSystem	External ❗	🔘		NO❗
└	recoverTokens	External ❗	🔘		NO❗
└	getEmergencyLevel	External ❗			NO❗
└	getEmergencyWithdrawalSettings	External ❗			NO❗
└	isSystemPaused	External ❗			NO❗
└	getEmergencyAwareContractsPaginated	External ❗			NO❗
EmergencyTimelockController	Implementation	Initializable, ReentrancyGuardUpgradeable		
└	<Constructor>	Public ❗	🔘		NO❗
└	initialize	External ❗	🔘		initializer
└	setAllowedTarget	External ❗	🔘		onlyEmergencyRole
└	setAllowedFunctionSelector	External ❗	🔘		onlyEmergencyRole
└	proposeEmergencyAction	External ❗	🔘		onlyEmergencyRole nonReentrant
└	executeEmergencyAction	External ❗	🔘		onlyEmergencyRole nonReentrant
└	cancelEmergencyAction	External ❗	🔘		onlyEmergencyRole nonReentrant
└	updateTimelockDuration	External ❗	🔘		onlyEmergencyRole
└	getActionStatus	External ❗			NO❗
└	getAllActionIds	External ❗			NO❗
└	getActionDetails	External ❗			NO❗
└	isFunctionSelectorAllowed	External ❗			NO❗
└	isTargetAllowed	External ❗			NO❗
Constants	Library				
OracleIntegration	Implementation	Initializable, OwnableUpgradeable, ReentrancyGuardUpgradeable		
└	<Constructor>	Public ❗	🔘		NO❗
└	initialize	External ❗	🔘		initializer
└	setLiquidityManagerAddress	External ❗	🔘		onlyOwner
└	setPriceFeed	External ❗	🔘		onlyOwner nonReentrant
└	registerLPToken	External ❗	🔘		onlyLiquidityManagerOrOwner nonReentrant
└	setFallbackPrice	External ❗	🔘		onlyOwner nonReentrant

└	setStalenessThreshold	External ❗	🛑	onlyOwner nonReentrant
└	setMinAcceptablePrice	External ❗	🛑	onlyOwner nonReentrant
└	getTokenPrice	Public ❗		NO❗
└	fetchAndReportTokenPrice	External ❗	🛑	nonReentrant
└	getLPTokenValue	External ❗		NO❗
└	getLPTokenValueAlternative	External ❗		NO❗
└	validatePriceAgainstOracle	External ❗		NO❗
└	_getPriceFromOracle	Private 🔐		
└	_fetchAndReportPriceFromOracle	Private 🔐	🛑	
└	_verifyPriceFeed	Private 🔐		
└	_getPriceFeedDecimals	Private 🔐		
└	convertToStandardPrecision	Public ❗		NO❗
└	_getTokenDecimals	Private 🔐		
└	getPriceFeedInfo	External ❗		NO❗
└	getLPTokenInfo	External ❗		NO❗
└	getStalenessThreshold	External ❗		NO❗
└	getMinAcceptablePrice	External ❗		NO❗
AggregatorV3Interface	Interface			
└	decimals	External ❗		NO❗
└	description	External ❗		NO❗
└	version	External ❗		NO❗
└	latestRoundData	External ❗		NO❗
└	getRoundData	External ❗		NO❗
DecimalMath	Library			
└	mulScaled	Internal 🔒		
└	divScaled	Internal 🔒		
└	mulDiv	Internal 🔒		
└	calculatePercentage	Internal 🔒		
└	applySlippage	Internal 🔒		
└	scaleUp	Internal 🔒		
└	scaleDown	Internal 🔒		
└	weightedAverage	Internal 🔒		
└	compound	Internal 🔒		
└	divRounded	Internal 🔒		
EmergencyAwareBase	Implementation	IEmergencyAware		
└	_isEffectivelyPaused	Internal 🔒		

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.