Smart Contract Security Audit V1

Chaufr Vesting Smart Contract Audit

Jun 11, 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**: Chaufr Vesting
- **Language** : Solidity
- **Contract Address**: 0x93632cd07750689D79243C492934A5a8Cd5977e4
- **Code Source**: https://testnet.bscscan.com/address/0x93632cd07750689D79243C492934A5a8Cd5977e4#code

ChaufrVesting

TOKEN VESTING FOR CHAUFRCOIN (CHUFR)

SECURE ERC20 VESTING WITH PRESALE UNLOCK | Audi t by SaferICO

KEY INFORMATION

Token: ChaufrCoin (CHUFR ERC20)

Purpose: Locks tokens for gradual release to beneficiaries

Presale Unlock: 15% of presale tokens released post-presale

Default Presale End: Jan 1, 2026 (configurable)

Presale Mehanism
15 % of presale tokens unlocked after presal.e ends

Remaining 85% vested over time

Admin Controls
Add/update vesting schedules
Grant/revoke vesting/release permissions

Withdraw unvested tokens
Update presale end timestamp & release Interval

HOW VESTING WORKS

Admin adsesced → Add Schedule

Step 1: Admin adds vesting schedule (beneficiary, start time duration, amount)

For presale. 15% marked for unlock 85% vested

Admin adds vesting schedules
Grant/revoke vesting/release permissions

Withdraw unvested tokens
Update presale end timestamp & release inr val

Schedule deleted after full release

SECURITY & INTEGRATIONS

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

The files:

ChaufrVesting.sol

Audit Score:

99% secure

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
ChaufrVesting.sol	676690f7cc75ca941ac8a149cd73acec77ebc9f6	0x93632cd07750689D79243C492934A5a8Cd5977e4

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

Function	Test Result	Type / Return Type	Score
addressList	✓	Read / public	**Passed**
allowedToReleaseToken	✓	Read / public	**Passed**
allowedToVest	✓	Read / public	**Passed**
getAllAddressesForReleaseToken	✓	Read / public	**Passed**
getPresaleUnlockRecord	✓	Read / public	**Passed**
getReleasedAmount	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
getReleaseHistory	✓	Read / public	**Passed**
getRemainingAmount	✓	Read / public	**Passed**
getTotalVestDetails	✓	Read / public	**Passed**
getVestingBeneficiaries	✓	Read / public	**Passed**
getVestingSchedule	✓	Read / public	**Passed**
isAddressAllowedToRelease	✓	Read / public	**Passed**
isVestingBeneficiary	✓	Read / public	**Passed**

lockedTokensForRelease	✓	Read / public	**Passed**
presaleEndTimestamp	✓	Read / public	**Passed**
presaleUnlocks	✓	Read / public	**Passed**
purchasedDetails	✓	Read / public	**Passed**
releasableAmountView	✓	Read / public	**Passed**
releaseHistory	✓	Read / public	**Passed**
releaseInterval	✓	Read / public	**Passed**
token	✓	Read / public	**Passed**
totalTokensToUnlockForPresale	✓	Read / public	**Passed**
vestingBeneficiaries	✓	Read / public	**Passed**
vestingSchedules	✓	Read / public	**Passed**
vestingStartTime	✓	Read / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
addPresaleVestingSchedule	✓	Write / public	**Passed**
addVestingSchedule	✓	Write / public	**Passed**
distributePresaleTokens	✓	Write / public	**Passed**
grantTokenReleaser	✓	Write / public	**Passed**
grantVestingPermission	✓	Write / public	**Passed**
releaseTokens	✓	Write / public	**Passed**
revokeTokenReleaser	✓	Write / public	**Passed**
revokeVestingPermission	✓	Write / public	**Passed**
setVestingPresaleEndTimestamp	✓	Write / public	**Passed**
updateReleaseInterval	✓	Write / public	**Passed**
withdrawTokens	✓	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	**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:

#Inconsistent Vesting Duration Calculation for Monthly Release

Description

The `releaseTokens` function calculates `monthlyRelease` based on `schedule.duration / releaseInterval`. However, `schedule.duration` can be any length, and `releaseInterval` is fixed (e.g., 30 days). If `schedule.duration` is not a perfect multiple of `releaseInterval`, this division will truncate, leading to an incorrect `monthlyRelease` amount, and potentially leaving some tokens permanently stuck in the contract or causing over-release in edge cases.

- Example: If `duration` is 60 days and `releaseInterval` is 30 days, `60/30 = 2` months. `monthlyRelease = totalAmount / 2`.
- If `duration` is 70 days and `releaseInterval` is 30 days, `70/30 = 2` months (truncates). `monthlyRelease = totalAmount / 2`. The remaining 10 days of vesting are not accounted for, meaning the last "month" of tokens will never be claimable via the monthly release logic, leaving tokens stuck.

Recommendation:

- Option A (Preferred): Ensure `_duration` provided when adding a vesting schedule is always a multiple of `releaseInterval`. Add a `require` statement in `addPresaleVestingSchedule` and `addVestingSchedule` to enforce this: `require(_duration % releaseInterval == 0, "Duration must be a multiple of release interval");`.
- Option B**:** Redesign the `monthlyRelease` calculation to handle non-multiples more robustly. Instead of `monthlyRelease`, calculate the amount released *per second* (`schedule.totalAmount / schedule.duration`) and then multiply by the elapsed time. This is often more precise.P.S: This issue is common to the majority of those smart contracts.

Status: Acknowledged.

#Unnecessary SafeMath for Solidity 0.8.x

Description

Solidity 0.8.0 and later automatically include checked arithmetic, meaning overflow/underflow will revert by default. Using SafeMath is redundant and adds unnecessary gas cost.

Remediation

Remove using SafeMath for uint256.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

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.

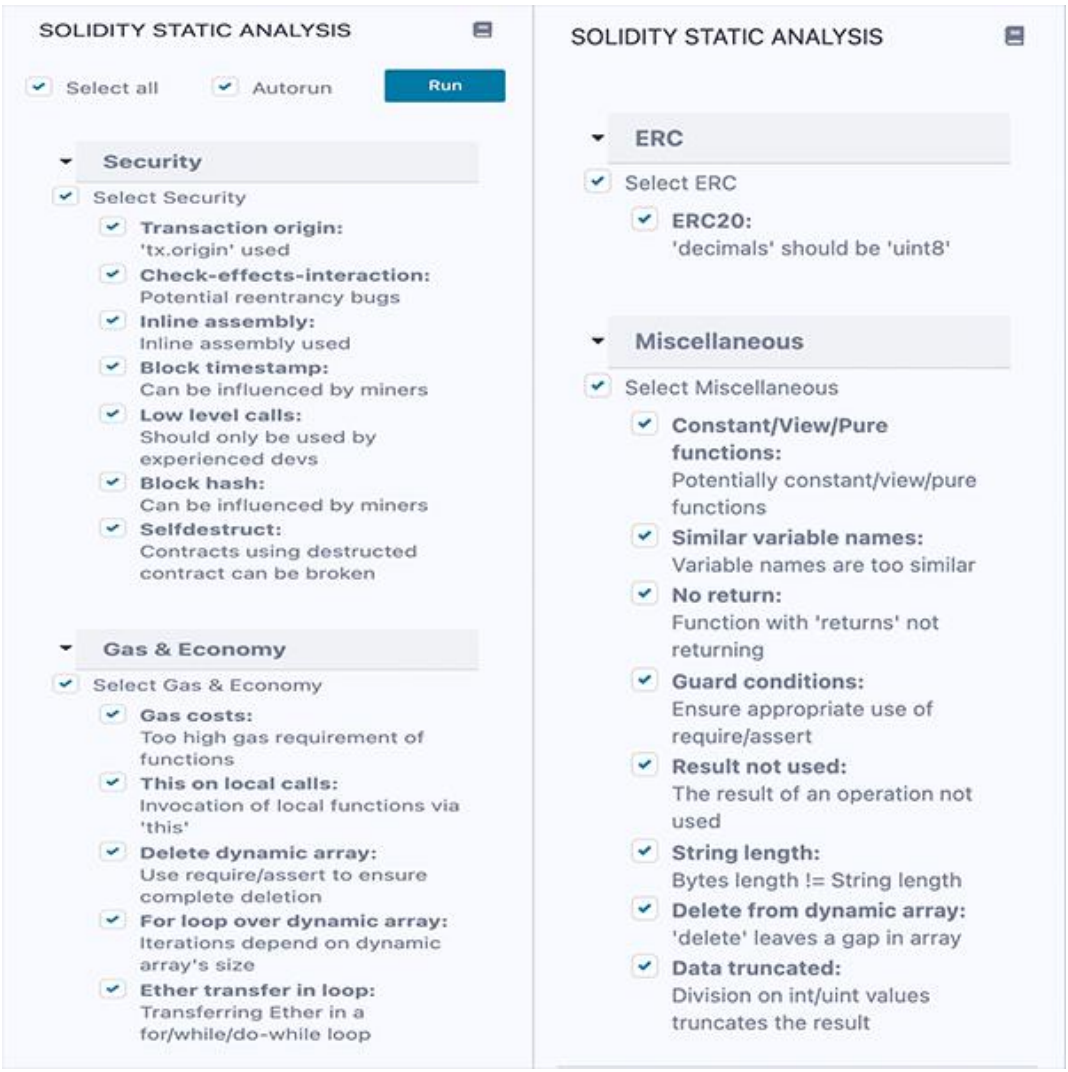
```
function addPresaleVestingSchedule(
        address _beneficiary,
        uint256 _startTime,
        uint256 _duration,
        uint256 _totalAmount
    ) external {
        require(allowedToVest[msg.sender], "Sender is not allowed
to add vesting");
        require(_beneficiary != address(0), "Beneficiary address
cannot be zero");
        require(_startTime >= block.timestamp, "Start time must be
in the future");
        require(_totalAmount > 0, "Total amount must be greater
than zero");
        require(token.balanceOf(address(this)) >= _totalAmount,
"Not enough tokens in contract for vesting.");

        uint256 unlockedTokenAmount = (_totalAmount * 15) / 100; //
15% token unlock after presale
        uint256 vestAmount = _totalAmount - unlockedTokenAmount; //
85% for vesting
```

Recommendation

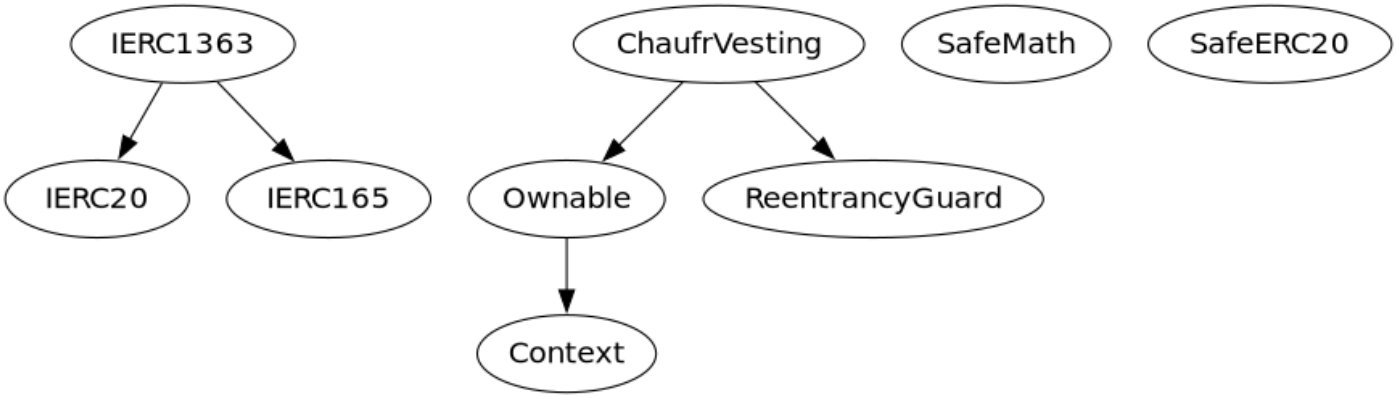
Avoid use of block.timestamp.

Automatic Testing

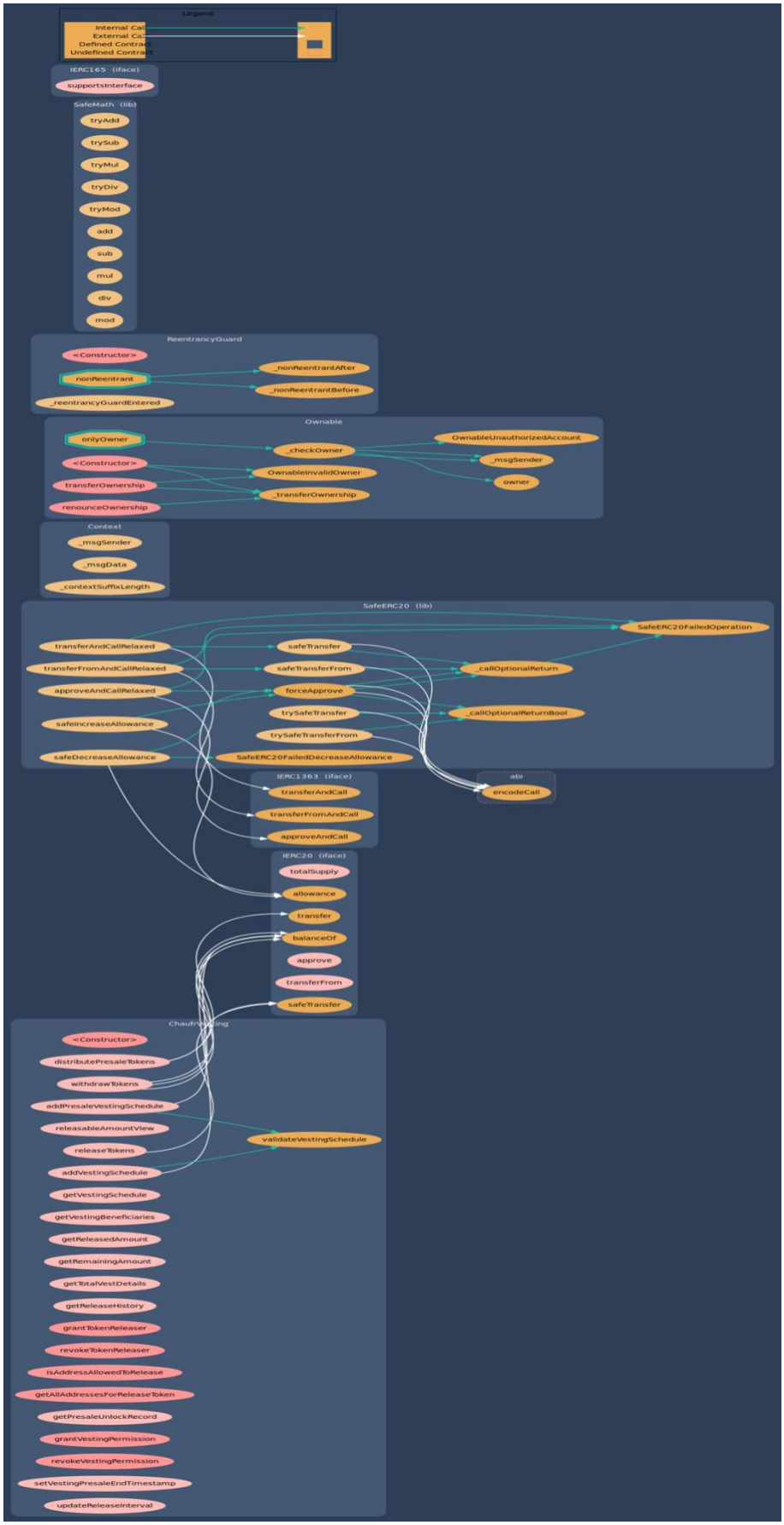
1- SOLIDITY STATIC ANALYSIS



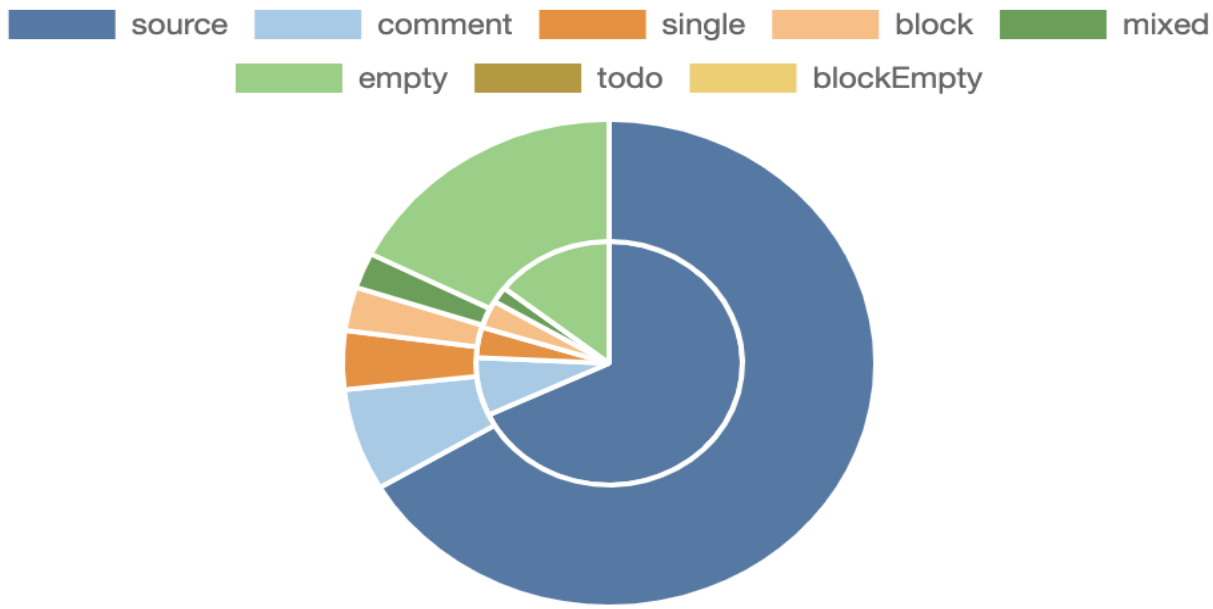
2- Inheritance graph



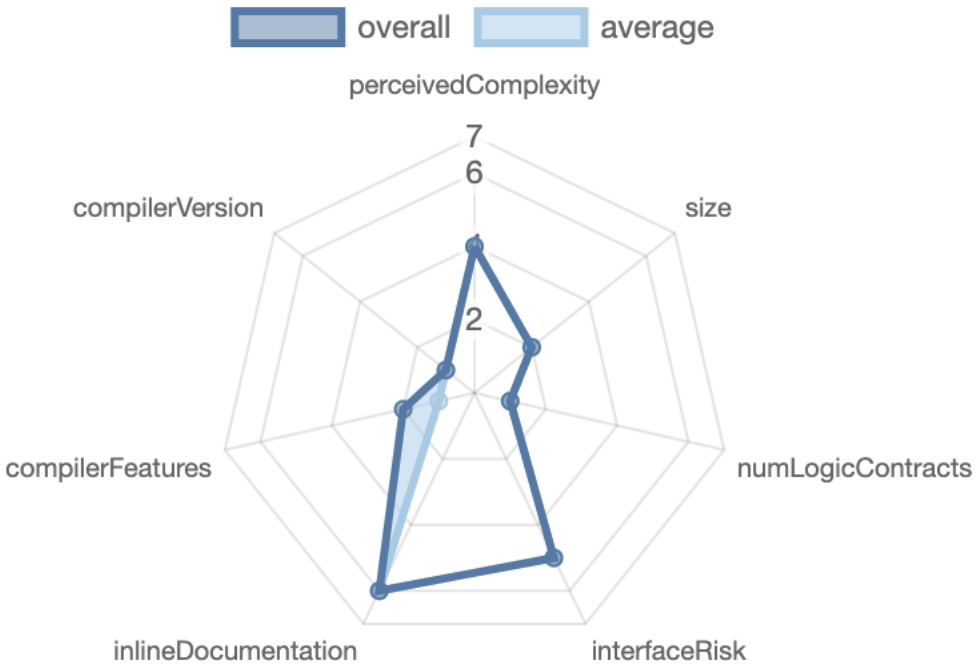
3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC165 (iface)
supportsInterface
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
ReentrancyGuard
<Constructor>
nonReentrant
_reentrancyGuardEntered
_nonReentrantAfter
_nonReentrantBefore
Ownable
onlyOwner
<Constructor>
transferOwnership
renounceOwnership
_checkOwner
OwnableInvalidOwner
_transferOwnership
OwnableUnauthorizedAccount
_msgSender
owner
Context
_msgSender
_msgData
_contextSuffixLength
SafeERC20 (lib)
transferAndCallRelaxed
transferFromAndCallRelaxed
approveAndCallRelaxed
safeIncreaseAllowance
safeDecreaseAllowance
safeTransfer
safeTransferFrom
forceApprove
trySafeTransfer
trySafeTransferFrom
SafeERC20FailedDecreaseAllowance
_callOptionalReturn
_callOptionalReturnBool
SafeERC20FailedOperation
IERC1363 (iface)
transferAndCall
transferFromAndCall
approveAndCall
abi
encodeCall
IERC20 (iface)
totalSupply
allowance
transfer
balanceOf
approve
transferFrom
safeTransfer
ChaufVesting
<Constructor>
distributePresaleTokens
withdrawTokens
addPresaleVestingSchedule
releasableAmountView
releaseTokens
addVestingSchedule
getVestingSchedule
getVestingBeneficiaries
getReleasedAmount
getRemainingAmount
getTotalVestDetails
getReleaseHistory
grantTokenReleaser
revokeTokenReleaser
isAddressAllowedToRelease
getAllAddressesForReleaseToken
getPresaleUnlockRecord
grantVestingPermission
revokeVestingPermission
setVestingPresaleEndTimestamp
updateReleaseInterval
validateVestingSchedule

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
📝🔍💊	Coinbird.sol	2	1	639	501	387	49	365	🧱
📝🔍💊	**Totals**	**2**	**1**	**639**	**501**	**387**	**49**	**365**	🧱

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	1	1

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
86	0

External	Internal	Private	Pure	View
43	66	3	0	25

StateVariables

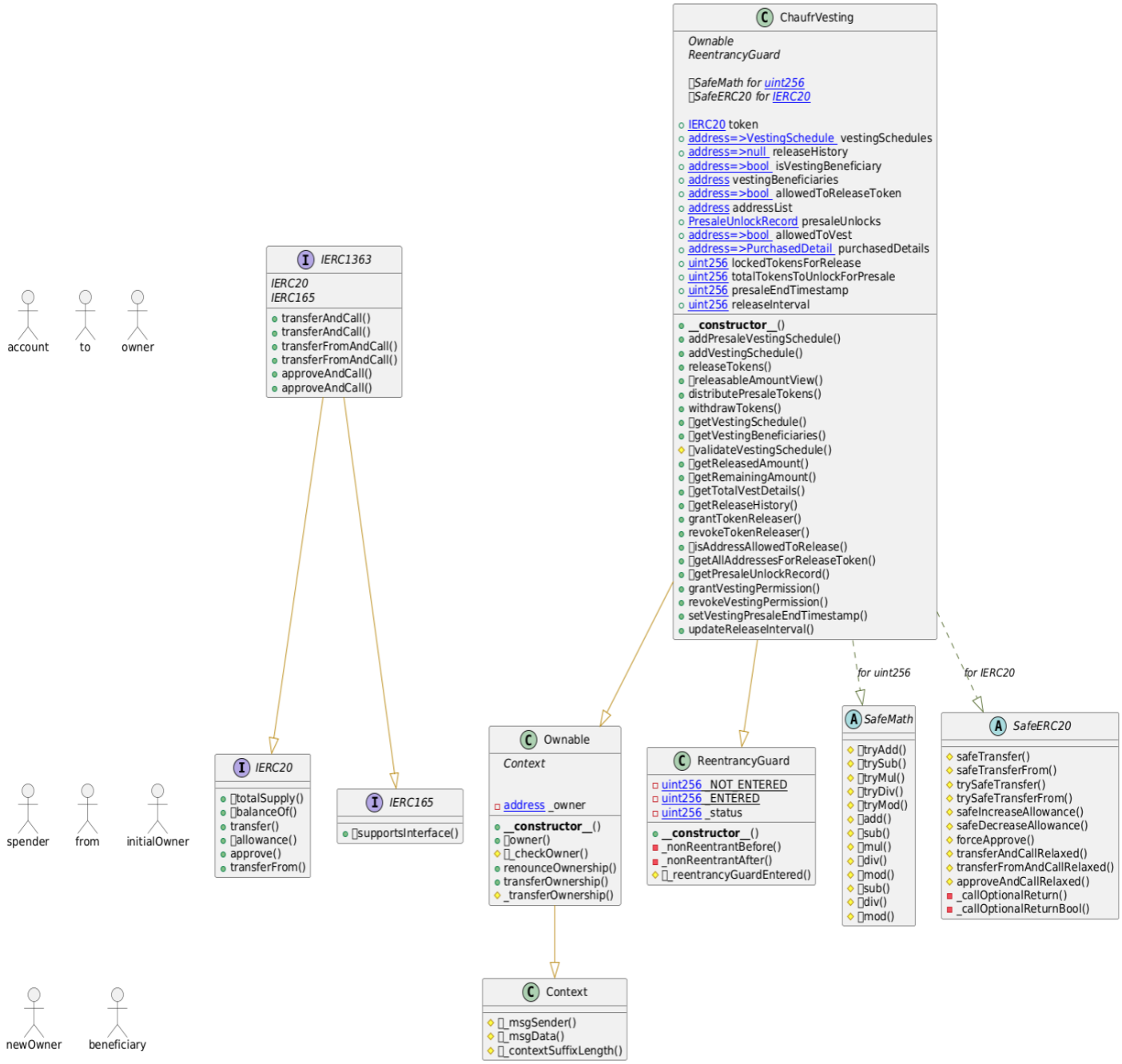
Total	🌐Public
47	36

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`0.8.14`				

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

Unified Modeling Language (UML)



Functions signature

Function Name	Sighash	Function Signature
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
transfer	a9059cbb	transfer(address,uint256)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
owner	8da5cb5b	owner()
renounceOwnership	715018a6	renounceOwnership()
transferOwnership	f2fde38b	transferOwnership(address)
supportsInterface	01ffc9a7	supportsInterface(bytes4)
transferAndCall	1296ee62	transferAndCall(address,uint256)
transferAndCall	4000aea0	transferAndCall(address,uint256,bytes)
transferFromAndCall	d8fbe994	transferFromAndCall(address,address,uint256)
transferFromAndCall	c1d34b89	transferFromAndCall(address,address,uint256,bytes)
approveAndCall	3177029f	approveAndCall(address,uint256)
approveAndCall	cae9ca51	approveAndCall(address,uint256,bytes)
addPresaleVestingSchedule	ccc65551	addPresaleVestingSchedule(address,uint256,uint256,uint256)
addVestingSchedule	24ef8c1b	addVestingSchedule(address,uint256,uint256,uint256)
releaseTokens	87b0be48	releaseTokens(address)
releasableAmountView	5db6ada6	releasableAmountView(address)
distributePresaleTokens	c5b07a24	distributePresaleTokens()
withdrawTokens	8d8f2adb	withdrawTokens()
getVestingSchedule	9f829063	getVestingSchedule(address)
getVestingBeneficiaries	49355e60	getVestingBeneficiaries()
getReleasedAmount	86197781	getReleasedAmount(address)
getRemainingAmount	faf7eba6	getRemainingAmount(address)
getTotalVestDetails	2886495a	getTotalVestDetails()
getReleaseHistory	8c6e250c	getReleaseHistory(address)
grantTokenReleaser	fae8ef36	grantTokenReleaser(address)
revokeTokenReleaser	a589255a	revokeTokenReleaser(address)
isAddressAllowedToRelease	2939f064	isAddressAllowedToRelease(address)
getAllAddressesForReleaseToken	e597d9d9	getAllAddressesForReleaseToken()
getPresaleUnlockRecord	fabbc0a0	getPresaleUnlockRecord()
grantVestingPermission	e127cc77	grantVestingPermission(address)
revokeVestingPermission	9e3609f7	revokeVestingPermission(address)
setVestingPresaleEndTimestamp	0a3cbdd5	setVestingPresaleEndTimestamp(uint256)
updateReleaseInterval	7a9ac6a3	updateReleaseInterval(uint256)

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/ChaufrVesting.sol	676690f7cc75ca941ac8a149cd73acec77ebc9f6

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
└	_contextSuffixLength	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
ReentrancyGuard	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	_nonReentrantBefore	Private 🔐	🛑	
└	_nonReentrantAfter	Private 🔐	🛑	
└	_reentrancyGuardEntered	Internal 🔒		
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		

└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
IERC165	Interface			
└	supportsInterface	External ❗		NO❗
IERC1363	Interface	IERC20, IERC165		
└	transferAndCall	External ❗	🛑	NO❗
└	transferAndCall	External ❗	🛑	NO❗
└	transferFromAndCall	External ❗	🛑	NO❗
└	transferFromAndCall	External ❗	🛑	NO❗
└	approveAndCall	External ❗	🛑	NO❗
└	approveAndCall	External ❗	🛑	NO❗
SafeERC20	Library			
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	trySafeTransfer	Internal 🔒	🛑	
└	trySafeTransferFrom	Internal 🔒	🛑	
└	safeIncreaseAllowance	Internal 🔒	🛑	
└	safeDecreaseAllowance	Internal 🔒	🛑	
└	forceApprove	Internal 🔒	🛑	
└	transferAndCallRelaxed	Internal 🔒	🛑	
└	transferFromAndCallRelaxed	Internal 🔒	🛑	
└	approveAndCallRelaxed	Internal 🔒	🛑	
└	_callOptionalReturn	Private 🔐	🛑	
└	_callOptionalReturnBool	Private 🔐	🛑	
ChaufrVesting	Implementation	Ownable, ReentrancyGuard		
└	<Constructor>	Public ❗	🛑	Ownable
└	addPresaleVestingSchedule	External ❗	🛑	NO❗
└	addVestingSchedule	External ❗	🛑	NO❗
└	releaseTokens	External ❗	🛑	nonReentrant
└	releasableAmountView	External ❗		NO❗
└	distributePresaleTokens	External ❗	🛑	onlyOwner
└	withdrawTokens	External ❗	🛑	onlyOwner nonReentrant
└	getVestingSchedule	External ❗		NO❗
└	getVestingBeneficiaries	External ❗		NO❗
└	validateVestingSchedule	Internal 🔒		
└	getReleasedAmount	External ❗		NO❗
└	getRemainingAmount	External ❗		NO❗
└	getTotalVestDetails	External ❗		NO❗
└	getReleaseHistory	External ❗		NO❗
└	grantTokenReleaser	Public ❗	🛑	onlyOwner

└	revokeTokenReleaser	Public ❗	🛑	onlyOwner
└	isAddressAllowedToRelease	Public ❗		NO❗
└	getAllAddressesForReleaseToken	Public ❗		NO❗
└	getPresaleUnlockRecord	External ❗		NO❗
└	grantVestingPermission	Public ❗	🛑	onlyOwner
└	revokeVestingPermission	Public ❗	🛑	onlyOwner
└	setVestingPresaleEndTimestamp	External ❗	🛑	NO❗
└	updateReleaseInterval	External ❗	🛑	onlyOwner

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.