Smart Contract Security Audit V1

Presale Chaufr Smart Contract Audit

Jun 10, 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**: PresaleChaufr
- **Language** : Solidity
- **Contract Address**: 0xc554c6DC24F0c462B2acb2fFb0709eD2eadb6741
- **Code Source**: https://testnet.bscscan.com/address/0xc554c6DC24F0c462B2acb2fFb0709eD2eadb6741#code

PresaleChaufr Smart Contract

Crowdfunding Token Sale with Vesting

Overview

A secure, decentralized presale platform for CHUFR tokens with Chainlink price feeds and vesting

- Built on Solidity 0.8.28
- Uses OpenZeppelin's ERC20, Ownable, and ReentrancyGir security
- Intogrates Chainlink for real-time BNB and USDT price feeds

Key Features

Purchase Options

- Buy CHUFR takens with BNB or USDT
- Bonus 15% during presale, 5% post-presale

Vesting

- Tokens vest over 24 months starting and post-pre-presale

Limits

- Min Buy: 10 tokens
- Max Buy: 1M tokens

Presale

How It Works

User

↓

Connfink

↓

Token Calculation

Vesting Assignment

Contract Detalls

Powered by Salidity, Opanzeppelin, and Chalnlink

Security & Admin Controls

Security

- Only owner can update prices, vesting, or admin address
- Only owner can udate prices, vesting

Admin Functions

- Update presale prices, vesting schedule, or buy limits
- Withdraw stuck BNB or tokens
- VestingDurationUpda-ted, PresaleEndTime-stamp Updated
- AdminAddressUpdated BuyLimitUpdated

Events EMITTED

- TokensPurchased tracks buyer, amoun cost and currency
- PriceUpdated presale phase 1 price
- VestingDuration Updated

audited by SaferICO

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 2 note in all solidity files of the contract

The files:

PresaleChaufr.sol

Audit Score:

99% secure

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
PresaleChaufr.sol	198fb3829689cb5375cfebd44dd30b7ebcfb30f0	0xc554c6DC24F0c462B2acb2fFb0709eD2eadb6741

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

Function	Test Result	Type / Return Type	Score
admin	✓	Read / public	**Passed**
chainLinkBnb	✓	Read / public	**Passed**
chainLinkUsdt	✓	Read / public	**Passed**
getBnbPricePerToken	✓	Read / public	**Passed**
getUsdtPricePerToken	✓	Read / public	**Passed**
presaleEndTimestamp	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
maxBuyLimit	✓	Read / public	**Passed**
minBuyLimit	✓	Read / public	**Passed**
presalePhase1UsdPrice	✓	Read / public	**Passed**
presalePhase2UsdPrice	✓	Read / public	**Passed**
presalePurchaseLimit	✓	Read / public	**Passed**
token	✓	Read / public	**Passed**
totalPurchasedToken	✓	Read / public	**Passed**

usdt	✓	Read / public	**Passed**
vesting	✓	Read / public	**Passed**
vestingDuration	✓	Read / public	**Passed**
vestingStartTime	✓	Read / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
buyTokenUSDT	✓	Write / public	**Passed**
buyTokenBNB	✓	Write / payable	**Passed**
updateAdminAddress	✓	Write / public	**Passed**
updateBuyLimit	✓	Write / public	**Passed**
updatePresalePhase2Price	✓	Write / public	**Passed**
updatePresalePhase1Price	✓	Write / public	**Passed**
updatePresaleTimestamp	✓	Write / public	**Passed**
updateVestingSchedule	✓	Write / public	**Passed**
withdrawStuckTokens	✓	Write / public	**Passed**
withdrawStuckBNB	✓	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:

#Admin Address vs. Owner:

Description

The contract uses `admin` for receiving funds but `owner()` (from Ownable) for management functions. This creates a potential for a mismatch if the `admin` address is intended to be the primary control. While `owner()` is the typical pattern for `Ownable`, if the `admin` is meant to be the sole recipient of funds and also have management capabilities, it might be clearer to rely solely on the `owner()` from `Ownable` and transfer ownership if needed, or make `admin` also the `owner()`.

Recommendation:

Option 1 (Preferred): Remove the `admin` state variable and associated functions (`updateAdminAddress`). Funds should be sent to `owner()`. If the intention is to separate roles, consider using OpenZeppelin's `AccessControl` for more granular role management.

Option 2 (If separation is intended): Clearly document the distinction between `admin` and `owner`. Ensure that the `admin` address can only be changed by the `owner()`, which is already implemented. The current setup is technically correct but might lead to confusion.

P.S: This issue is common to the majority of those smart contracts.
Status: Acknowledged.

#Owner privileges (In the period when the owner isn't renounced)

Description

The owner can change the price of the presale.
The owner can change the time of the presale.
The owner can change the limit of buying.

```
function updatePresalePhase1Price(uint256 _usdPresalePhase1Price) external
onlyOwner {
        require(_usdPresalePhase1Price > 0, "Price cannot be zero");

        presalePhase1UsdPrice = _usdPresalePhase1Price;

        emit PriceUpdated(_usdPresalePhase1Price);
    }

    function updatePresalePhase2Price(uint256 _usdPresalePhase2Price) external
onlyOwner {
        require(_usdPresalePhase2Price > 0, "Price cannot be zero");

        presalePhase2UsdPrice = _usdPresalePhase2Price;

        emit PriceUpdatedForAfterPresale(_usdPresalePhase2Price);
    }
function updatePresaleTimestamp(uint256 _presaleEndTimestamp) external onlyOwner {
        require(_presaleEndTimestamp >= block.timestamp, "Timestamp must be in the
future");

        presaleEndTimestamp = _presaleEndTimestamp;

        vesting.setVestingPresaleEndTimestamp(_presaleEndTimestamp);

        emit PresaleEndTimestampUpdated(_presaleEndTimestamp);
    }
function updateBuyLimit(uint256 _minBuyLimit, uint256 _maxBuyLimit) external
onlyOwner {
        require(_minBuyLimit > 0 && _maxBuyLimit > 0, "Amount could not be zero");
        require(_minBuyLimit != minBuyLimit || _maxBuyLimit != maxBuyLimit, "At
least one value must be different");

        minBuyLimit = _minBuyLimit;
        maxBuyLimit = _maxBuyLimit;

        emit BuyLimitUpdated(_minBuyLimit, _maxBuyLimit);
    }
```

Remediation

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

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

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Magic Numbers:

Description

Hardcoded numbers like `1e18`, `1e10`, `15`, `5`, `10`, `1_000_000`, `300_000_000` are used directly in the code without clear explanations.

Recommendation

Define these as named constants (e.g., `_DECIMALS`, `_BONUS_PRESALE`, `_BONUS_AFTER_PRESALE`, `_DEFAULT_MIN_BUY_LIMIT`, `_DEFAULT_MAX_BUY_LIMIT`, `_DEFAULT_PRESALE_PURCHASE_LIMIT`). This improves readability and maintainability.

Code Example:

Solidity

```
uint256 public constant TOKEN_DECIMALS_FACTOR = 1e18;
uint256 public constant CHAINLINK_PRICE_DECIMALS_ADJUSTMENT = 1e10;
uint256 private constant PRESALE_BONUS_PERCENTAGE = 15;
uint256 private constant AFTER_PRESALE_BONUS_PERCENTAGE = 5;

// In constructor
minBuyLimit = 10 * TOKEN_DECIMALS_FACTOR;
uint256 bonusPercentage = block.timestamp <= presaleEndTimestamp ?
PRESALE_BONUS_PERCENTAGE : AFTER_PRESALE_BONUS_PERCENTAGE;
```

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 updateVestingSchedule(uint256 _vestingStartTime, uint256
_vestingDuration) external onlyOwner {
        require(_vestingStartTime > block.timestamp, "Start
timestamp must be in the future");
        require(_vestingDuration > 0, "Vesting duration can't be
zero");
        vestingStartTime = _vestingStartTime;
        vestingDuration = _vestingDuration;
    emitVestingDurationUpdated(_vestingStartTime,_vestingDuration;}
```

Recommendation

Avoid use of block.timestamp.

Automatic Testing

1- SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS

Select all | Autorun | Run

Security

Select Security

- **Transaction origin:** 'tx.origin' used
- **Check-effects-interaction:** Potential reentrancy bugs
- **Inline assembly:** Inline assembly used
- **Block timestamp:** Can be influenced by miners
- **Low level calls:** Should only be used by experienced devs
- **Block hash:** Can be influenced by miners
- **Selfdestruct:** Contracts using destructed contract can be broken

Gas & Economy

Select Gas & Economy

- **Gas costs:** Too high gas requirement of functions
- **This on local calls:** Invocation of local functions via 'this'
- **Delete dynamic array:** Use require/assert to ensure complete deletion
- **For loop over dynamic array:** Iterations depend on dynamic array's size
- **Ether transfer in loop:** Transferring Ether in a for/while/do-while loop

SOLIDITY STATIC ANALYSIS

ERC

Select ERC

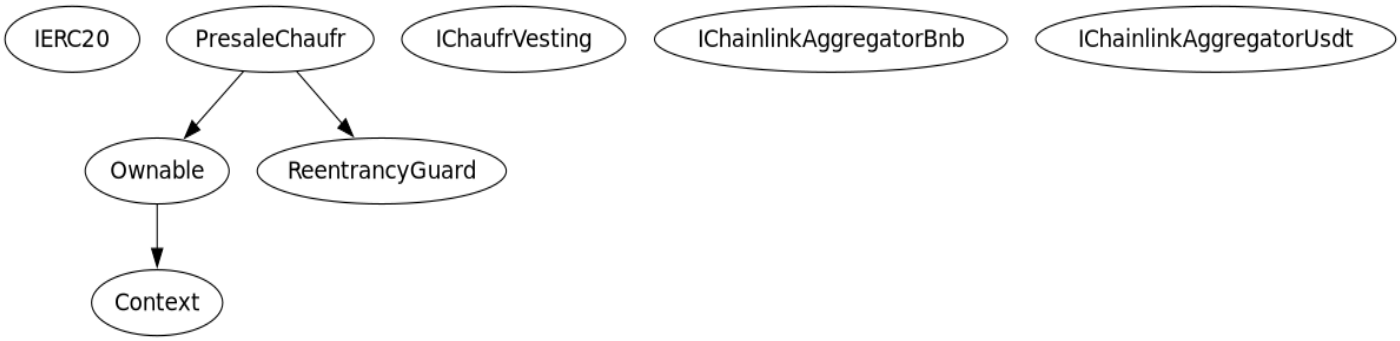
- **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

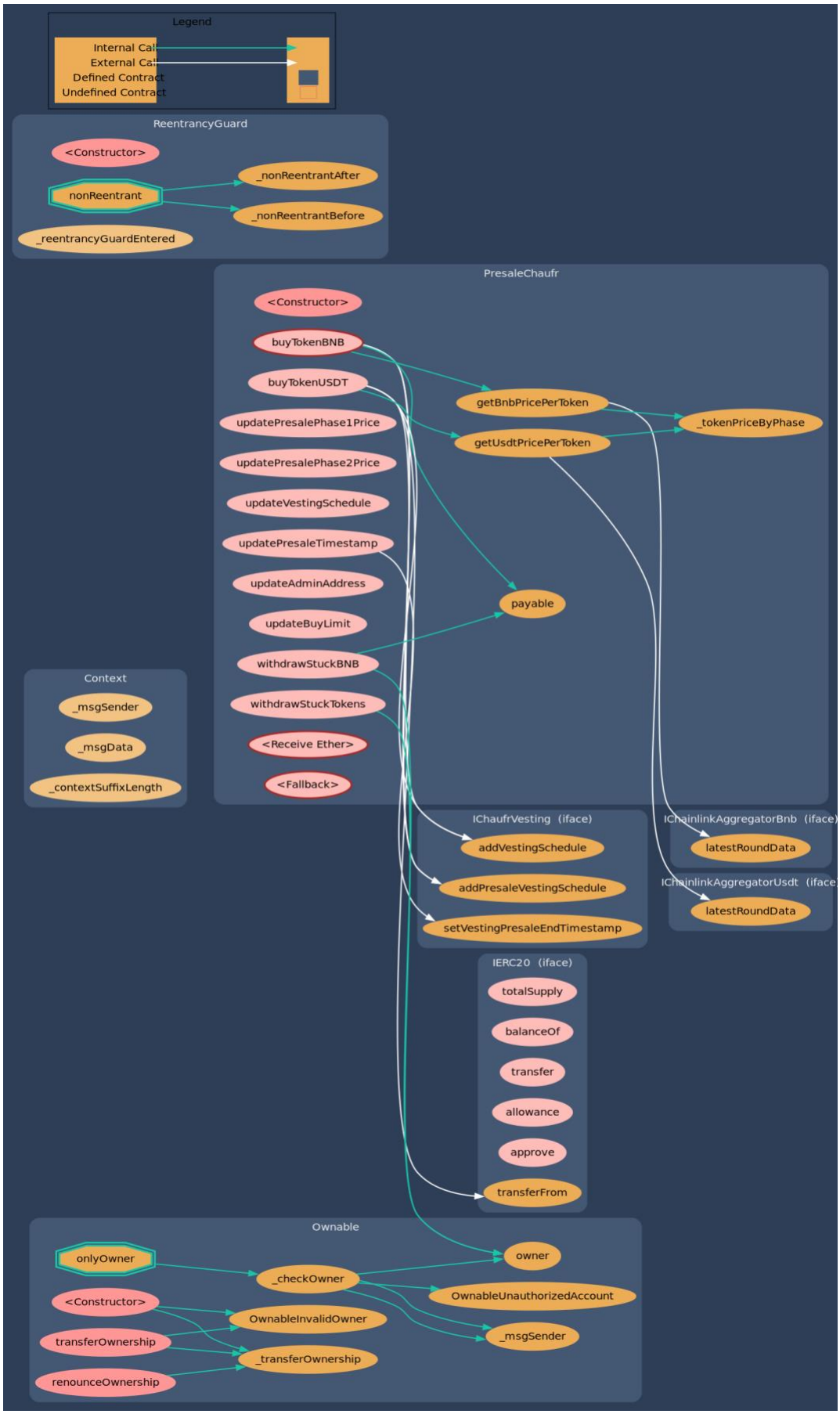
Select Miscellaneous

- **Constant/View/Pure functions:** Potentially constant/view/pure functions
- **Similar variable names:** Variable names are too similar
- **No return:** Function with 'returns' not returning
- **Guard conditions:** Ensure appropriate use of require/assert
- **Result not used:** The result of an operation not used
- **String length:** Bytes length != String length
- **Delete from dynamic array:** 'delete' leaves a gap in array
- **Data truncated:** Division on int/uint values truncates the result

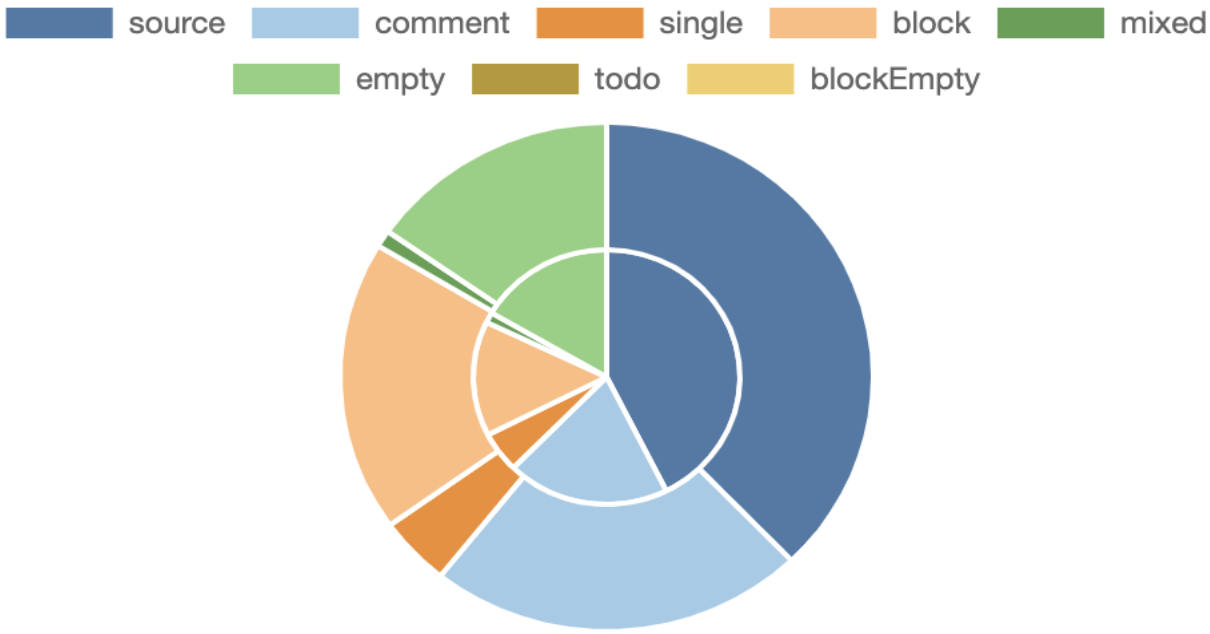
2- Inheritance graph



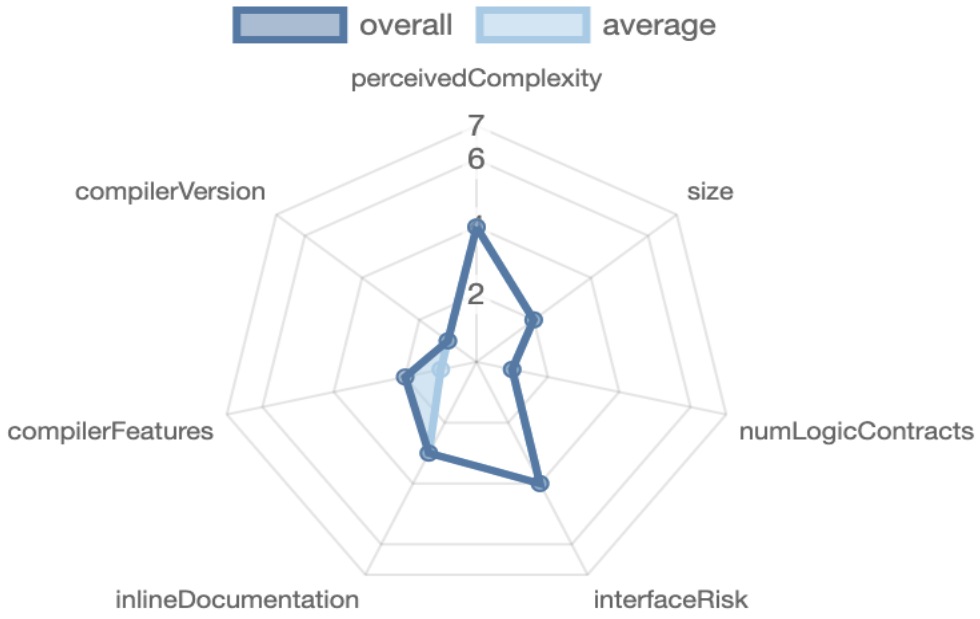
3- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ReentrancyGuard
<Constructor>
nonReentrant
_nonReentrantAfter
_nonReentrantBefore
_reentrancyGuardEntered
PresaleChaufr
<Constructor>
buyTokenBNB
buyTokenUSDT
updatePresalePhase1Price
updatePresalePhase2Price
updateVestingSchedule
updatePresaleTimestamp
updateAdminAddress
updateBuyLimit
withdrawStuckBNB
withdrawStuckTokens
<Receive Ether>
<Fallback>
getBnbPricePerToken
getUsdtPricePerToken
_tokenPriceByPhase
payable
Context
_msgSender
_msgData
_contextSuffixLength
IChaufrVesting (iface)
addVestingSchedule
addPresaleVestingSchedule
setVestingPresaleEndTimestamp
IChainlinkAggregatorBnb (iface)
latestRoundData
IChainlinkAggregatorUsdt (iface)
latestRoundData
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
onlyOwner
<Constructor>
transferOwnership
renounceOwnership
_checkOwner
OwnableInvalidOwner
_transferOwnership
owner
OwnableUnauthorizedAccount
_msgSender

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
📝🔍🎨	PresaleChaufr.sol	4	4	563	468	257	172	217	💰📩🔆
📝🔍🎨	**Totals**	**4**	**4**	**563**	**468**	**257**	**172**	**217**	💰📩🔆

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	4	3

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
28	3

External	Internal	Private	Pure	View
23	30	2	0	14

StateVariables

Total	🌐Public
19	15

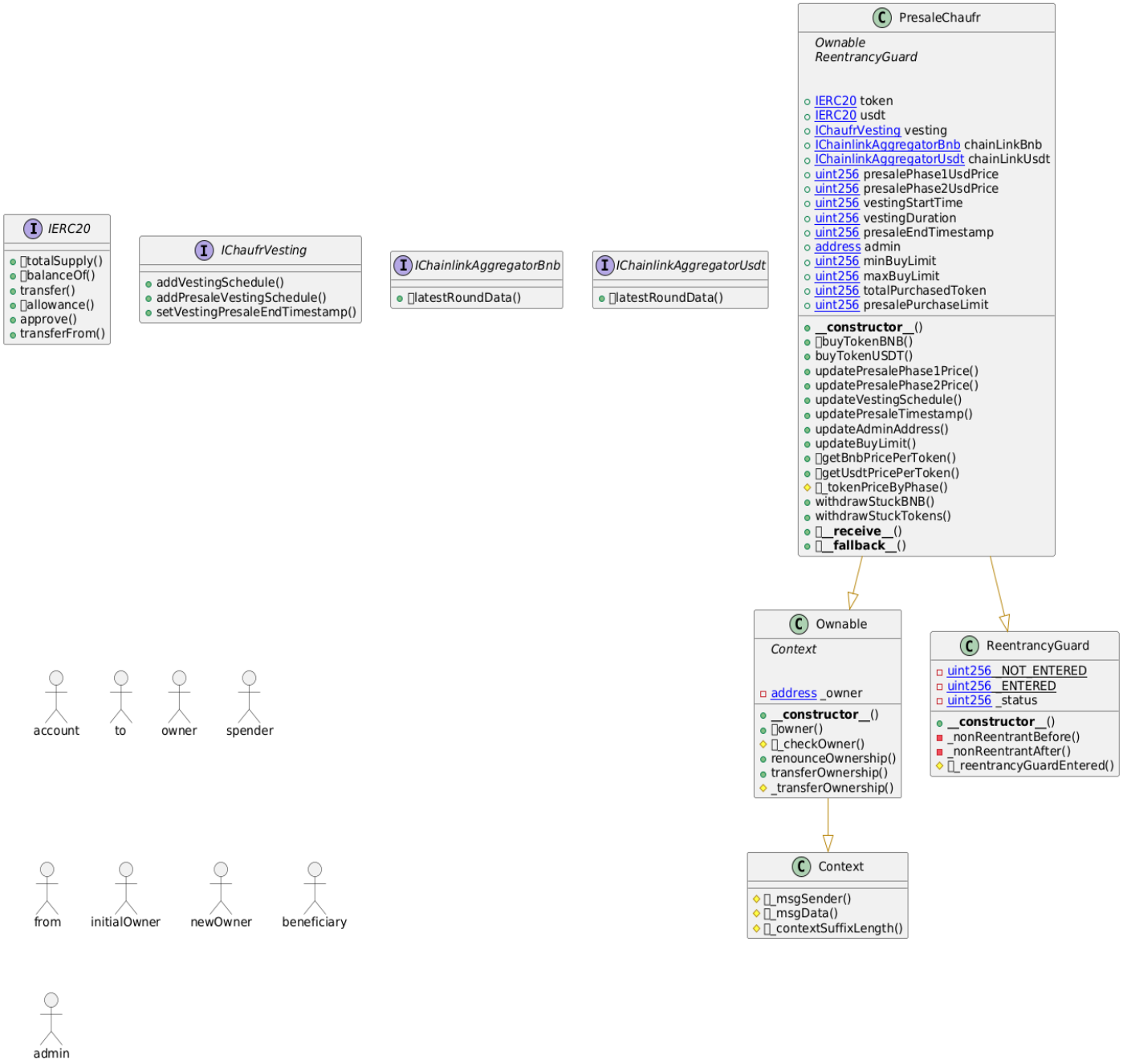
Capabilities

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

`^0.8.20`
`^0.8.0`
`0.8.28`
		`yes`	—	—

📩 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) |
| addVestingSchedule | 24ef8c1b |
addVestingSchedule(address,uint256,uint256,uint256) |
| addPresaleVestingSchedule | ccc65551 |
addPresaleVestingSchedule(address,uint256,uint256,uint256) |
| setVestingPresaleEndTimestamp | 0a3cbdd5 |
setVestingPresaleEndTimestamp(uint256) |
| latestRoundData | feaf968c | latestRoundData() |
| latestRoundData | feaf968c | latestRoundData() |
| buyTokenBNB | f27827b4 | buyTokenBNB() |
| buyTokenUSDT | b5e75e1c | buyTokenUSDT(uint256) |
| updatePresalePhase1Price | 5ccb48c8 | updatePresalePhase1Price(uint256)
|
| updatePresalePhase2Price | 0b7ab158 | updatePresalePhase2Price(uint256)
|
| updateVestingSchedule | 72b320fe |
updateVestingSchedule(uint256,uint256) |
| updatePresaleTimestamp | 227640f9 | updatePresaleTimestamp(uint256) |
| updateAdminAddress | 85e2381c | updateAdminAddress(address) |
| updateBuyLimit | 983168fc | updateBuyLimit(uint256,uint256) |
| getBnbPricePerToken | f5bb4f59 | getBnbPricePerToken() |
| getUsdtPricePerToken | 74508abc | getUsdtPricePerToken() |
| withdrawStuckBNB | 484ed334 | withdrawStuckBNB() |
| withdrawStuckTokens | cb963728 | withdrawStuckTokens(address) |
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/PresaleChaufr.sol |
198fb3829689cb5375cfebd44dd30b7ebcfb30f0 |

 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 🔒 |   | |
||||||
| **IChaufrVesting** | Interface |  |||
| └ | addVestingSchedule | External ❗ | 🛑  |NO❗ |
| └ | addPresaleVestingSchedule | External ❗ | 🛑  |NO❗ |
| └ | setVestingPresaleEndTimestamp | External ❗ | 🛑  |NO❗ |
||||||
```

IChainlinkAggregatorBnb	Interface			
└	latestRoundData	External ❗		NO❗
IChainlinkAggregatorUsdt	Interface			
└	latestRoundData	External ❗		NO❗
PresaleChaufr	Implementation	Ownable, ReentrancyGuard		
└	<Constructor>	Public ❗	🛑	Ownable
└	buyTokenBNB	External ❗	💵	nonReentrant
└	buyTokenUSDT	External ❗	🛑	nonReentrant
└	updatePresalePhase1Price	External ❗	🛑	onlyOwner
└	updatePresalePhase2Price	External ❗	🛑	onlyOwner
└	updateVestingSchedule	External ❗	🛑	onlyOwner
└	updatePresaleTimestamp	External ❗	🛑	onlyOwner
└	updateAdminAddress	External ❗	🛑	onlyOwner
└	updateBuyLimit	External ❗	🛑	onlyOwner
└	getBnbPricePerToken	Public ❗		NO❗
└	getUsdtPricePerToken	Public ❗		NO❗
└	_tokenPriceByPhase	Internal 🔒		
└	withdrawStuckBNB	External ❗	🛑	onlyOwner nonReentrant
└	withdrawStuckTokens	External ❗	🛑	onlyOwner nonReentrant
└	<Receive Ether>	External ❗	💵	NO❗
└	<Fallback>	External ❗	💵	NO❗

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.