Smart Contract Security Audit V1

Infinity Presale Smart Contract

26/12/2021

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
- **Contract Address**: 0xe227406b51507dB631894DA7d8627652c9547E29

Presale Smart Contract Information

- Name: Presale
- Hard cap: 560
- Token per BNB: 2.5 BNB
- End Time: Wed Jan 12 2022 20:58:10 GMT

Contracts address deployed to test net (BSC)

Presale Smart contract on testnet.bsc (BSC Test Net)
https://testnet.bscscan.com/address/0xdb89c81d3e27e6e3d119bcea8702f6874b42887a

Executive Summary

According to our assessment, the customer`s solidity smart contract is **InSecured**.

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 4 critical, 1 high, 2 medium, 0 low, 0 very low-level issues and 0 note in all solidity files.

The files:

Presale.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
presale.sol	**fb0ce094825403b8ac444dccbe09a579968368854fb819ddb491b164747ce386**	0xe227406b51507dB631894DA7d8627652c9547E29

- Contract: Presale
- Inherit: Ownable
- Observation: Not passed including security check
- Test Report: Not passed
- Score: Not passed
- Conclusion: Not passed

Function	Test Result	Type / Return Type	Score
claimedTokens	✓	Read / public	**Passed**
contributions	✓	Read / public	**Passed**
endTime	✓	Read / public	**Passed**
finalized	✓	Read / public	**Passed**
hardCap	✓	Read / public	**Passed**
softCap	✓	Read / public	**Passed**
isOpen	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
maxContribution	✓	Read / public	**Passed**
minContribution	✓	Read / public	**Passed**
softCapReached	✓	Read / public	**Passed**
refunds	✓	Read / public	**Passed**

startTime	✓	Read / public	Passed
token	✓	Read / public	Passed
tokensPerBnb	✓	Read / public	Passed
weiRaised	✓	Read / public	Passed
withdrawAddr	✓	Read / public	Passed
renounceOwnership	✓	Write / public	Not Passed
transferOwnership	✓	Write / public	Not Passed
claimRefund	✓	Write / public	Not Passed
claimToken	✓	Write / public	Not Passed
buyToken	✓	Write / payable	Passed
endPresale	✓	Write / public	Passed
setHardCap	✓	Write / public	Passed
setStartTime	✓	Write / public	Passed
setEndTime	✓	Write / public	Passed
setSoftCap	✓	Write / public	Passed
setWithdrawAddr	✓	Write / public	Passed
withdrawTokens	✓	Write / public	Not Passed

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Front running.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**
18	Design Logic.	**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:

# Issue 1 Inactive buyToken function

Description

The most important function in the contract which allows to the investors buy token is not active which make all the contract useless

```
function _buyTokens(address beneficiary) internal{
        uint256 weiToHardcap = hardCap.sub(weiRaised);
        uint256 weiAmount = weiToHardcap < msg.value ? weiToHardcap :
msg.value;
        _buyTokens(beneficiary, weiAmount);

        uint256 refund = msg.value.sub(weiAmount);
        if (refund > 0) {
            payable(beneficiary).transfer(refund);
        }
    }
```

# Issue 2 &3 claimTokens and claimRefunds functions not work

Description

These functions don't work even all conditions is enabled.

```
function claimTokens() external {
        require(hasEnded(), "Presale: presale is not over");
        require(softCapReached(), "Presale: soft cap not reached, refund is
available");
        require(contributions[msg.sender] > 0, "Presale: nothing to claim");
        uint256 tokens = _getTokenAmount(contributions[msg.sender]);
        contributions[msg.sender] = 0;
        claimedTokens[msg.sender] = tokens;
        token.safeTransfer(msg.sender, tokens);
        emit TokenClaim(msg.sender, tokens);
    }

    function claimRefund() external {
        require(hasEnded(), "Presale: presale is not over");
        require(!softCapReached(), "Presale: soft cap not reached");
        require(contributions[msg.sender] > 0, "Presale: nothing to claim");
        uint256 refundAmount = contributions[msg.sender];
        contributions[msg.sender] = 0;
        refunds[msg.sender] = refundAmount;
        payable(msg.sender).transfer(refundAmount);
        emit Refund(msg.sender, refundAmount);
    }
```

# Issue 4 withdrawTokens isn't working

Description

After the presale had ended the owner use this function everyone get 0 token not the amount you should have

```
function withdrawTokens() public onlyOwner {
        require(hasEnded(), "Presale: presale is not over");
        uint256 tokens = token.balanceOf(address(this));
        token.transfer(owner(), tokens);
    }
```

High:

# Issue 1 can’t change the owner even the owner uses the transfer ownership

Description

Initializes the contract setting the deployer as the initial owner if he wants change the owner using transfer ownership he will still the controller of the owner even if he use renounceOwnership he still the controller and the new owner can’t controller anything of the contract.

Medium:

# Issue 1 The deploy address can change withdraw address

Description

The owner has the ability to change withdraw address anytime he wants.

```
function setWithDrawAddr(address _withdrawAddr) public onlyOwner {
        withdrawAddr = _withdrawAddr;
    }
```

# Issue 2 The owner can change almost everything in the contract even after deployed it

Description

The owner has the ability to set new start time , can start presale again anytime he want , change all of these end time of presale, softCap and hardCap.

```
function setStartTime(uint256 _startTime) public onlyOwner {
        startTime = _startTime;
    }

    function setEndTime(uint256 _endTime) public onlyOwner {
        endTime = _endTime;
    }

    function setHardCap(uint256 _hardCap) public onlyOwner {
        hardCap = _hardCap;
    }

    function setSoftCap(uint256 _softCap) public onlyOwner {
        softCap = _softCap;
    }
```

Low:

No Very Low severity vulnerabilities were found.

Very Low:

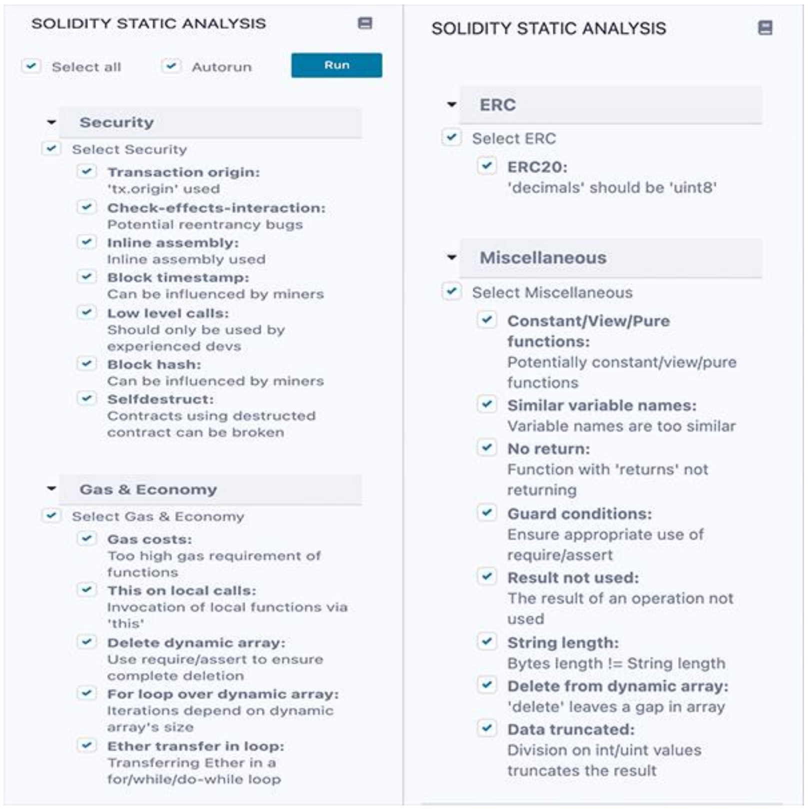
No Very Low severity vulnerabilities were found.

Notes:

No Notes were found.

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

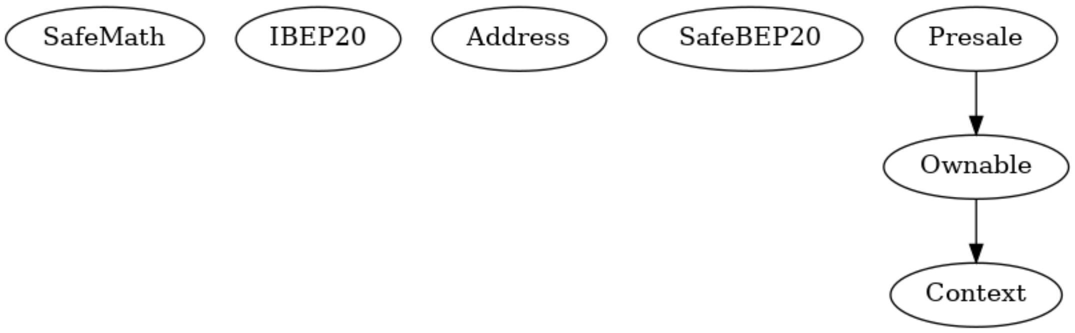
- ✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

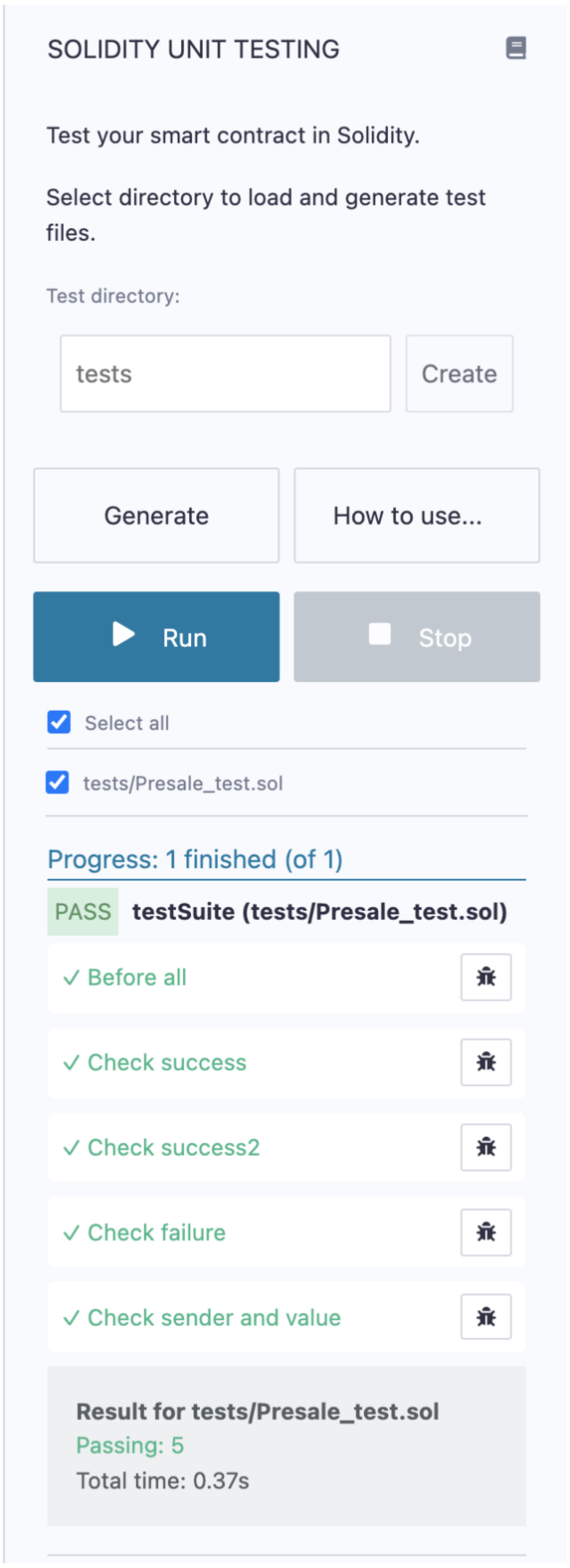
✓ Select Miscellaneous

- ✓ **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- SOLIDITY UNIT TESTING


SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory: tests

☑ Select all

☑ tests/Presale_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/Presale_test.sol)**

✓ Before all

✓ Check success

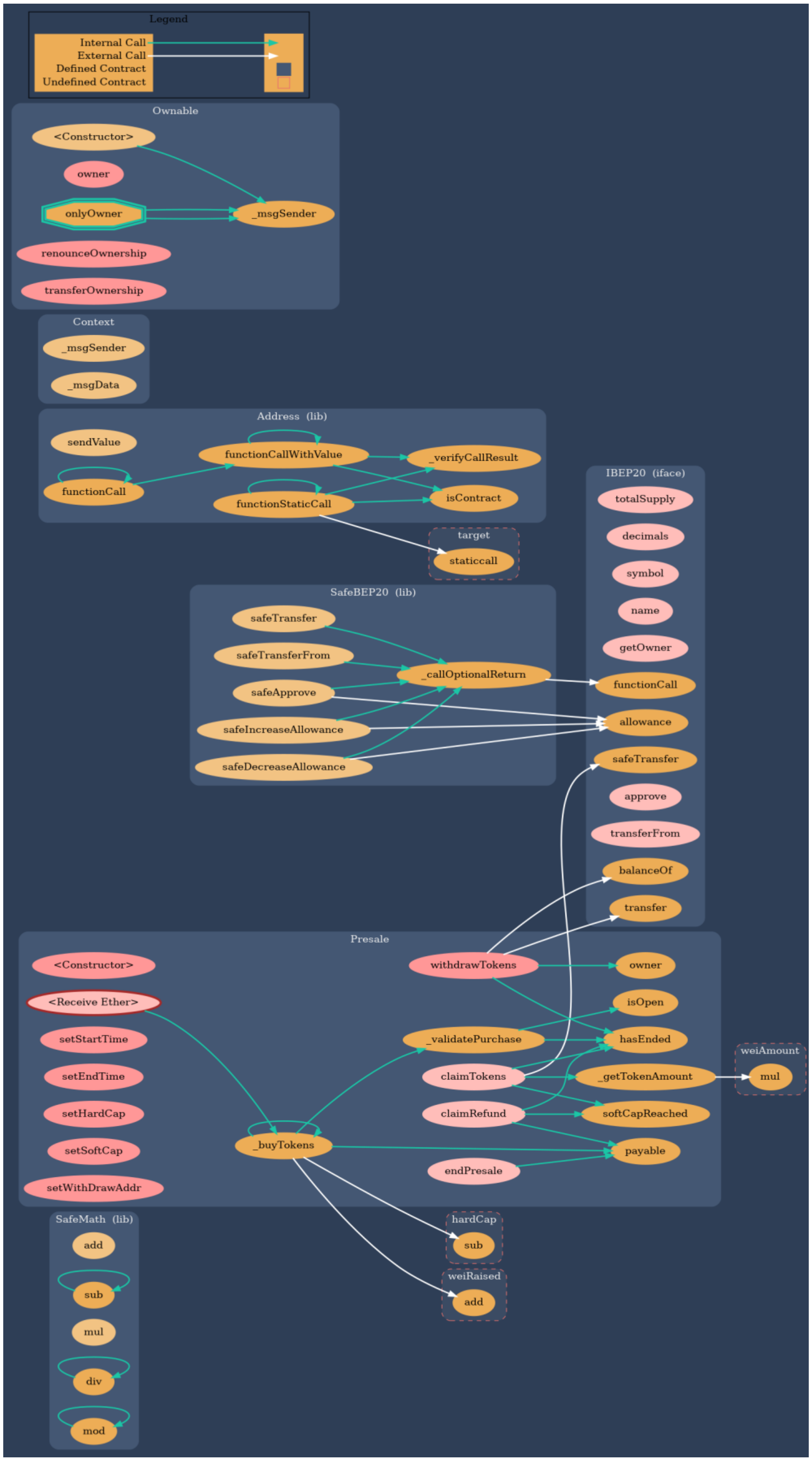
✓ Check success2

✓ Check failure

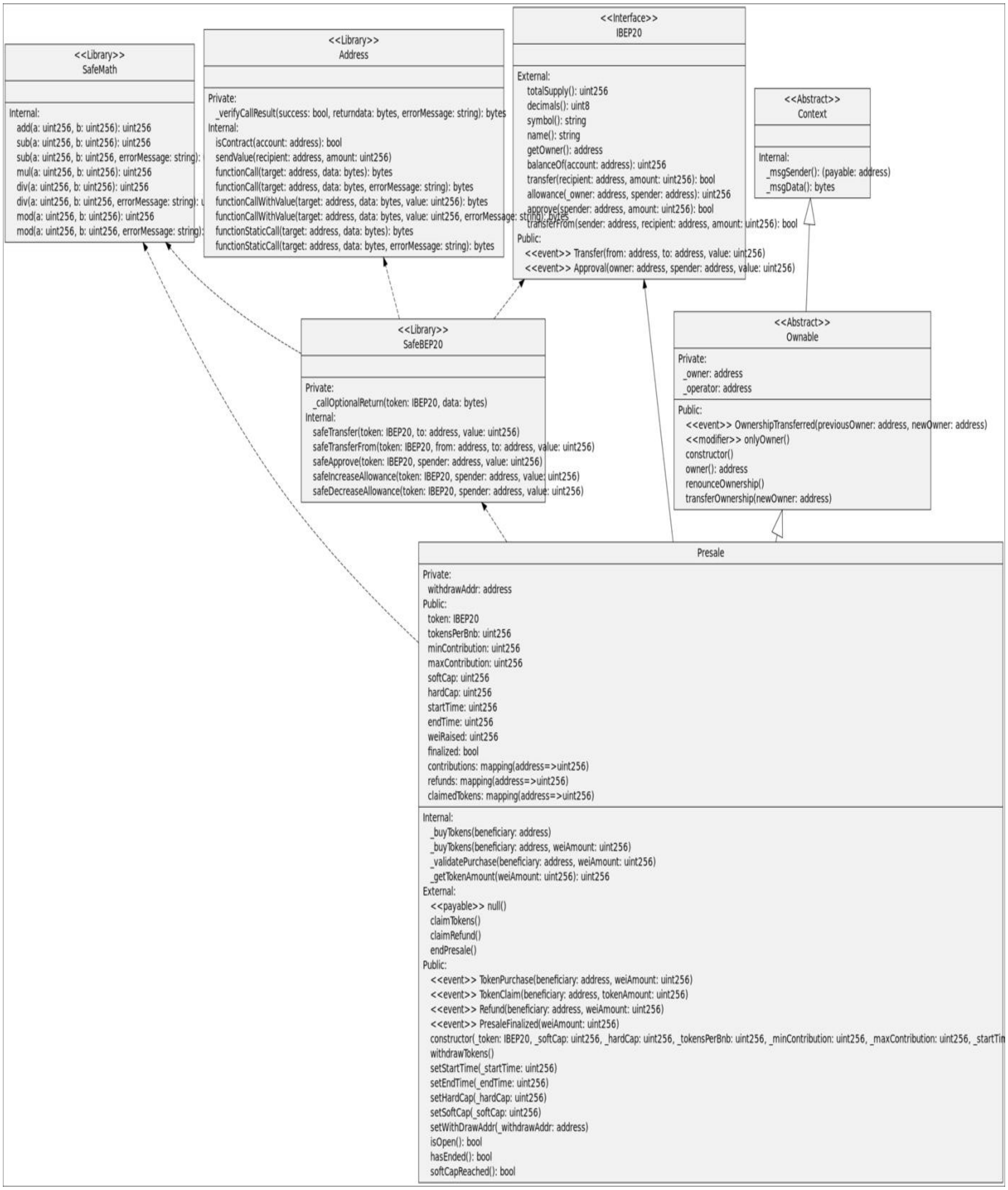
✓ Check sender and value

Result for tests/Presale_test.sol
Passing: 5
Total time: 0.37s

4- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
<Constructor>
owner
onlyOwner
_msgSender
renounceOwnership
transferOwnership
Context
_msgSender
_msgData
Address (lib)
sendValue
functionCallWithValue
_verifyCallResult
functionCall
functionStaticCall
isContract
target
staticcall
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
functionCall
allowance
safeTransfer
approve
transferFrom
balanceOf
transfer
SafeBEP20 (lib)
safeTransfer
safeTransferFrom
_callOptionalReturn
safeApprove
safeIncreaseAllowance
safeDecreaseAllowance
Presale
<Constructor>
withdrawTokens
owner
<Receive Ether>
isOpen
setStartTime
_validatePurchase
hasEnded
setEndTime
claimTokens
_getTokenAmount
setHardCap
claimRefund
softCapReached
setSoftCap
_buyTokens
payable
setWithDrawAddr
endPresale
weiAmount
mul
SafeMath (lib)
add
sub
mul
div
mod
hardCap
sub
weiRaised
add

Unified Modeling Language (UML)


<<Library>>
SafeMath
Internal:
add(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256, errorMessage: string):
mul(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256, errorMessage: string): u
mod(a: uint256, b: uint256): uint256
mod(a: uint256, b: uint256, errorMessage: string):
<<Library>>
Address
Private:
_verifyCallResult(success: bool, returndata: bytes, errorMessage: string): bytes
Internal:
isContract(account: address): bool
sendValue(recipient: address, amount: uint256)
functionCall(target: address, data: bytes): bytes
functionCall(target: address, data: bytes, errorMessage: string): bytes
functionCallWithValue(target: address, data: bytes, value: uint256): bytes
functionCallWithValue(target: address, data: bytes, value: uint256, errorMessage: string): bytes
functionStaticCall(target: address, data: bytes): bytes
functionStaticCall(target: address, data: bytes, errorMessage: string): bytes
<<Interface>>
IBEP20
External:
totalSupply(): uint256
decimals(): uint8
symbol(): string
name(): string
getOwner(): address
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(_owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
<<event>> Transfer(from: address, to: address, value: uint256)
<<event>> Approval(owner: address, spender: address, value: uint256)
<<Abstract>>
Context
Internal:
_msgSender(): (payable: address)
_msgData(): bytes
<<Library>>
SafeBEP20
Private:
_callOptionalReturn(token: IBEP20, data: bytes)
Internal:
safeTransfer(token: IBEP20, to: address, value: uint256)
safeTransferFrom(token: IBEP20, from: address, to: address, value: uint256)
safeApprove(token: IBEP20, spender: address, value: uint256)
safeIncreaseAllowance(token: IBEP20, spender: address, value: uint256)
safeDecreaseAllowance(token: IBEP20, spender: address, value: uint256)
<<Abstract>>
Ownable
Private:
_owner: address
_operator: address
Public:
<<event>> OwnershipTransferred(previousOwner: address, newOwner: address)
<<modifier>> onlyOwner()
constructor()
owner(): address
renounceOwnership()
transferOwnership(newOwner: address)
Presale
Private:
withdrawAddr: address
Public:
token: IBEP20
tokensPerBnb: uint256
minContribution: uint256
maxContribution: uint256
softCap: uint256
hardCap: uint256
startTime: uint256
endTime: uint256
weiRaised: uint256
finalized: bool
contributions: mapping(address=>uint256)
refunds: mapping(address=>uint256)
claimedTokens: mapping(address=>uint256)
Internal:
_buyTokens(beneficiary: address)
_buyTokens(beneficiary: address, weiAmount: uint256)
_validatePurchase(beneficiary: address, weiAmount: uint256)
_getTokenAmount(weiAmount: uint256): uint256
External:
<<payable>> null()
claimTokens()
claimRefund()
endPresale()
Public:
<<event>> TokenPurchase(beneficiary: address, weiAmount: uint256)
<<event>> TokenClaim(beneficiary: address, tokenAmount: uint256)
<<event>> Refund(beneficiary: address, weiAmount: uint256)
<<event>> PresaleFinalized(weiAmount: uint256)
constructor(_token: IBEP20, _softCap: uint256, _hardCap: uint256, _tokensPerBnb: uint256, _minContribution: uint256, _maxContribution: uint256, _startTin
withdrawTokens()
setStartTime(_startTime: uint256)
setEndTime(_endTime: uint256)
setHardCap(_hardCap: uint256)
setSoftCap(_softCap: uint256)
setWithDrawAddr(_withdrawAddr: address)
isOpen(): bool
hasEnded(): bool
softCapReached(): bool

Functions signature

```
16279055  =>  isContract(address)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
18c2c6a2  =>  _verifyCallResult(bool,bytes,string)
71ef09a2  =>  safeTransfer(IBEP20,address,uint256)
fee2e02e  =>  safeTransferFrom(IBEP20,address,address,uint256)
a93f2c02  =>  safeApprove(IBEP20,address,uint256)
e6ae193b  =>  safeIncreaseAllowance(IBEP20,address,uint256)
4988652c  =>  safeDecreaseAllowance(IBEP20,address,uint256)
a096fbe2  =>  _callOptionalReturn(IBEP20,bytes)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
1f5e881f  =>  _buyTokens(address)
781db835  =>  _buyTokens(address,uint256)
f07da436  =>  _validatePurchase(address,uint256)
48c54b9d  =>  claimTokens()
b5545a3c  =>  claimRefund()
a43be57b  =>  endPresale()
8d8f2adb  =>  withdrawTokens()
3e0a322d  =>  setStartTime(uint256)
ccb98ffc  =>  setEndTime(uint256)
d18d944b  =>  setHardCap(uint256)
d5cf5c72  =>  setSoftCap(uint256)
37ad4fc4  =>  setWithDrawAddr(address)
7a99bb0a  =>  _getTokenAmount(uint256)
47535d7b  =>  isOpen()
ecb70fb7  =>  hasEnded()
2b9edee9  =>  softCapReached()
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/Presale.sol |
7eb7b1a1f227faba3251b85189056773a5ae7a68 |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **SafeMath** | Library |  |||
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
||||||
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **Address** | Library |  |||
| └ | isContract | Internal 🔒 |   | |
| └ | sendValue | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionStaticCall | Internal 🔒 |   | |
| └ | functionStaticCall | Internal 🔒 |   | |
| └ | _verifyCallResult | Private 🔐 |   | |
||||||
| **SafeBEP20** | Library |  |||
| └ | safeTransfer | Internal 🔒 | 🛑  | |
| └ | safeTransferFrom | Internal 🔒 | 🛑  | |
| └ | safeApprove | Internal 🔒 | 🛑  | |
| └ | safeIncreaseAllowance | Internal 🔒 | 🛑  | |
| └ | safeDecreaseAllowance | Internal 🔒 | 🛑  | |
| └ | _callOptionalReturn | Private 🔐 | 🛑  | |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Internal 🔒 | 🛑  | |
| └ | owner | Public ❗️ |   |NO❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑  | onlyOwner |
||||||
| **Presale** | Implementation | Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | <Receive Ether> | External ❗️ |  💵 |NO❗️ |
| └ | _buyTokens | Internal 🔒 | 🛑  | |
| └ | _buyTokens | Internal 🔒 | 🛑  | |
| └ | _validatePurchase | Internal 🔒 |   | |
| └ | claimTokens | External ❗️ | 🛑  |NO❗️ |
| └ | claimRefund | External ❗️ | 🛑  |NO❗️ |
| └ | endPresale | External ❗️ | 🛑  | onlyOwner |
| └ | withdrawTokens | Public ❗️ | 🛑  | onlyOwner |
| └ | setStartTime | Public ❗️ | 🛑  | onlyOwner |
| └ | setEndTime | Public ❗️ | 🛑  | onlyOwner |
| └ | setHardCap | Public ❗️ | 🛑  | onlyOwner |
| └ | setSoftCap | Public ❗️ | 🛑  | onlyOwner |
| └ | setWithDrawAddr | Public ❗️ | 🛑  | onlyOwner |
| └ | _getTokenAmount | Internal 🔒 |   | |
| └ | isOpen | Public ❗️ |   |NO❗️ |
| └ | hasEnded | Public ❗️ |   |NO❗️ |
| └ | softCapReached | Public ❗️ |   |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 “IN secured”.

X No volatile code.
X Not many high severity issues were found.
X Contract Ownership Renounced.

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.