smart state

Web3 security easier than ever

Cellini Art Fund Tarocchi e Racconti

Smart contract audit report

Aug 23 2023

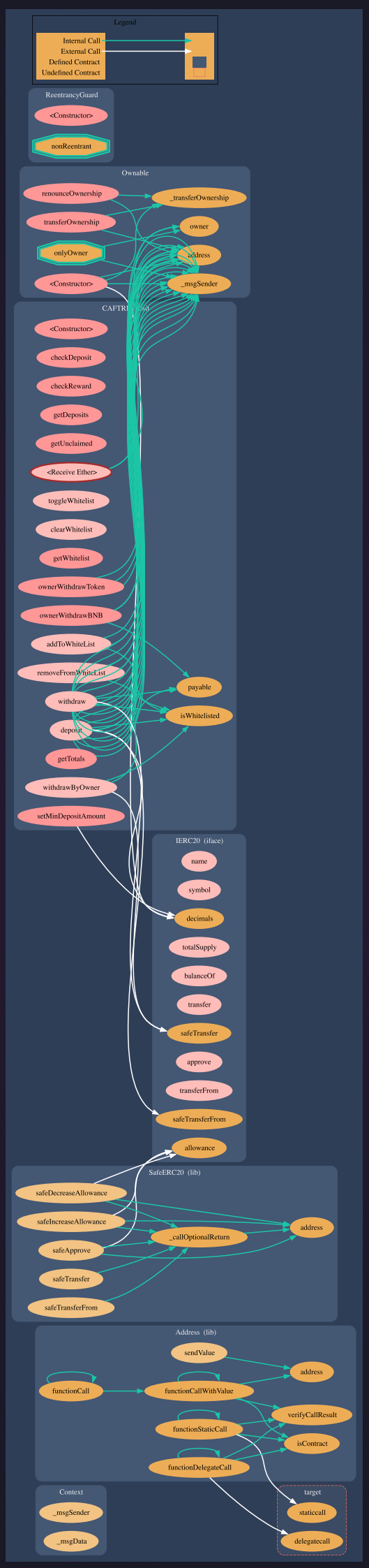
Table of contents

Methodology

- Best code practices
- ERC20/BEP20 compliance (if applicable)
- FA2 compliance (if applicable)
- Logical bugs
- General Denial Of Service(DOS)
- Locked ether
- Private data leaks
- Using components with known vulns
- Weak PRNG
- Unused vars
- Unchecked call return method
- Code with no effects
- Pool Asset Security (backdoors in the underlying ERC-20)
- Function visibility
- Use of deprecated functions
- Authorization issues
- Re-entrancy
- Arithmetic Over/Under Flows
- Hidden Malicious Code
- External Contract Referencing
- Short Address/Parameter Attack
- Race Conditions/Front Running
- Uninitialized Storage Pointers
- Floating Points and Precision
- Signatures Replay

Structure of contract

CAFTRDeposit.sol



pic.1.1 CAFTRDeposit.sol

Contract check summary:

It is an implementation of a deposit contract that allows users to deposit a specific ERC20 token and receive rewards in BNB (Binance Coin) based on their deposited amount.

Here is a breakdown of the contract's functionality:

1. The contract uses the Ownable and ReentrancyGuard libraries.
2. It defines a struct called DepositData to store deposit information.
3. It declares private state variables including whitelist, depositors, token, minDepositAmount, whitelistIdx, depositorsIdx, depositAmounts, whitelistActive, totalDeposited, totalClaimed, and BNBReward.
4. It emits events for Deposit, Withdrawal, and Reward.
5. The constructor initializes the contract with an initial owner and sets the ERC20 token address and minimum deposit amount.
6. The contract provides a function to check the deposit amount for a given account.
7. It provides a function to check the reward amount for a given account based on the deposited amount and the total deposited amount.
8. The contract provides functions to get a list of deposits and unclaimed rewards within a specified range.
9. It allows the owner to set the minimum deposit amount and toggle the whitelist feature.
10. The contract provides functions to manage the whitelist, including adding and removing addresses.
11. Users can deposit tokens by calling the deposit function, which checks if the caller is whitelisted and the deposit amount meets the minimum requirement. It transfers the tokens to the contract and updates the deposit information.
12. Users can withdraw their deposited tokens and claim their rewards by calling the withdraw function. It transfers the tokens and rewards to the caller's address and updates the deposit information.
13. The contract allows the owner to withdraw tokens and BNB from the contract.
14. The receive function is used to receive BNB rewards from the owner and update the reward amount.
15. The contract provides a function to get the total deposited amount, total BNB reward, total claimed amount, and remaining BNB reward.
16. The owner can withdraw tokens and BNB from the contract using the ownerWithdrawToken and ownerWithdrawBNB functions.

Contract methods analysis:

checkDeposit()
Vulnerabilities not detected

checkReward()
Vulnerabilities not detected

getDeposits()
Vulnerabilities not detected

getUnclaimed()
Vulnerabilities not detected

INFO
setMinDepositAmount()
Function should emit an event

toggleWhitelist()
Vulnerabilities not detected

INFO
clearWhitelist()
Function should emit an event

getWhitelist()
Vulnerabilities not detected

isWhitelisted()
Vulnerabilities not detected

addToWhiteList()
Vulnerabilities not detected

INFO
removeFromWhiteList()
Function should emit an event

deposit()
Vulnerabilities not detected

withdraw()
Vulnerabilities not detected

INFO
withdrawByOwner()
Function should emit an event

INFO

`receive()`

Function should emit an event

`getTotals()`

Vulnerabilities not detected

`ownerWithdrawToken()`

Vulnerabilities not detected

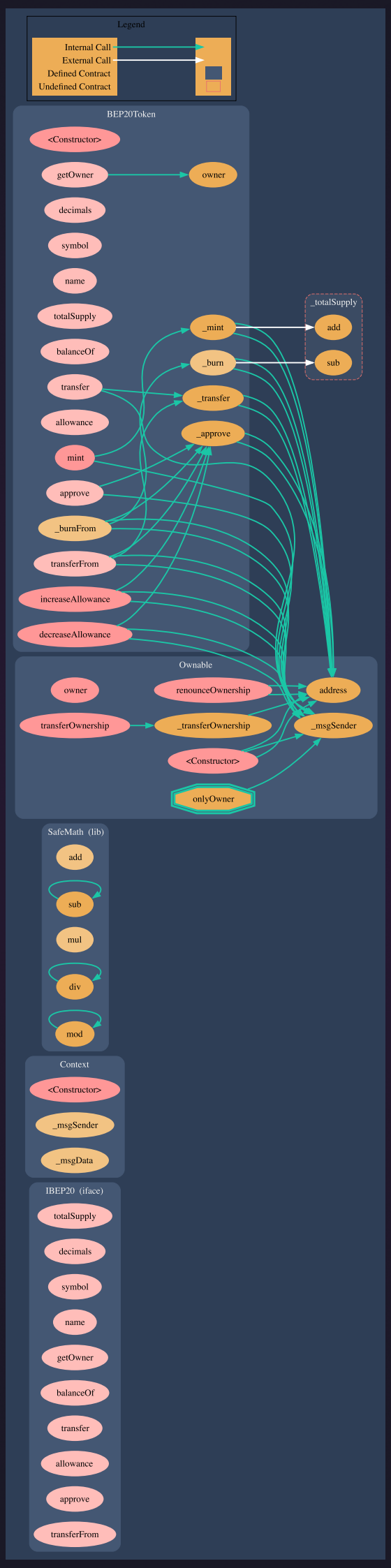
INFO

`ownerWithdrawBNB()`

Function should emit an event

Structure of contract

BEP20Token.sol



pic.1.2 BEP20Token.sol

Contract check summary:

0x143Dd8D84B66bfCBdbdFCd1D407Be582bdE1Ed1 code is a Solidity smart contract that implements a BEP20 token. The contract defines an interface called IBEP20 which includes functions for retrieving the total supply, decimals, symbol, and name of the token, as well as functions for getting the owner's address, checking the balance of an account, transferring tokens, and managing allowances.
The contract also includes a library called SafeMath which provides safe arithmetic operations to prevent overflow and underflow vulnerabilities.

Additionally, the contract inherits from the Context contract, which provides internal functions for retrieving the sender's address and message data.

The BEP20Token contract is the main implementation of the BEP20 token. It includes functions for transferring tokens, managing allowances, minting new tokens, and burning tokens. The contract also includes an Ownable contract, which provides ownership functionality, allowing the contract owner to transfer ownership to another address.

The contract initializes with an initial supply of 0 tokens and the token's name, symbol, and decimals are set. The contract owner is initially set to the deployer of the contract.
Overall, this code represents a basic implementation of a BEP20 token with ownership functionality and safe arithmetic operations.

Contract methods analysis:

totalSupply()
Vulnerabilities not detected

decimals()
Vulnerabilities not detected

symbol()
Vulnerabilities not detected

name()
Vulnerabilities not detected

getOwner()
Vulnerabilities not detected

balanceOf()
Vulnerabilities not detected

transfer()
Vulnerabilities not detected

allowance()
Vulnerabilities not detected

approve()
Vulnerabilities not detected

transferFrom()
Vulnerabilities not detected

increaseAllowance()
Vulnerabilities not detected

decreaseAllowance()
Vulnerabilities not detected

mint()
Vulnerabilities not detected

_transfer()
Vulnerabilities not detected

_mint()
Vulnerabilities not detected

_burn()
Vulnerabilities not detected

_approve()
Vulnerabilities not detected

_burnFrom()
Vulnerabilities not detected

`_msgSender()`
Vulnerabilities not detected

`_msgData()`
Vulnerabilities not detected

`owner()`
Vulnerabilities not detected

`renounceOwnership()`
Vulnerabilities not detected

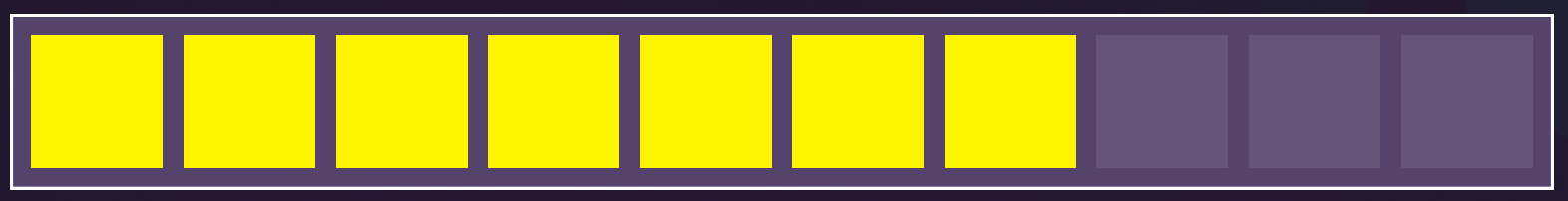
`transferOwnership()`
Vulnerabilities not detected

`_transferOwnership`
Vulnerabilities not detected

Verification check sums

Contract name	Bytecode hash(SHA 256)
CAFTRDeposit.sol	6955c083f1d21e0b5c1fd0da75059eebe14642c8c9a6a a9f74d30ed63c865de6
BEP20Token.sol	a5a23b04f401345ff2219695411ad3dfb7fad3cc890da e44374e275d81fb1f43
Link to source code:	https://bscscan.com/ address/0x143Dd8D84B66bfCBdbdFCd1D407Be582bdE1Ed1F#writeContract

Project evaluation



7/10

Get in touch 👋

@smartstatetech

@smartstate

@SmartStateAudit

@smartstatetech

@smartstate.tech

info@smartstate.tech

smartstate.tech