RD AUDITORS

Omnia Defi Smart Contract, Code Review and Security Analysis Report

Customer: Omnia Defi
Prepared on: 07th Dec 2021
Platform: Binance Smart Chain
Language: Solidity

rdauditors.com

Table of Contents

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report of Omnia Defi
Platform	BSC / Solidity
File 1	AutoClaim.sol
MD5 hash	172475B769FE3C8429EADFF0CCFB57EF
SHA256 hash	A4801346EDF8DDB05138B45FA9DFEAD063866A00B0E951463CD92D66C874A84F
File 2	AutoClaimDetails.sol
MD5 hash	096FFB8F6BAB3159A4C2489CF88DFDFB
SHA256 hash	55430DF8FC2177AC48CAF9BF08812A1B5AFB309E260A8DE3C468C6449A6D4D5F
File 3	AutoClaimInterval.sol
MD5 hash	BA78F03B00B8E6FB30484BBE37870F53
SHA256 hash	41EB0D65B808919CBD2A412CCDE2F7AFCE4CCEB40E0CD7ED68DD46172DC7AD0C
File 4	AutoClaimPausable.sol

MD5 hash	7069CD67B73309CBD82930A5546BCF19
SHA256 hash	275807D7106AC947D8253A374EC676082BF037FABFBC879C14560FCAF12575CE
File 5	AutoClaimQueueList .sol
MD5 hash	30492D2E78439E0254448B653982F6AD
SHA256 hash	DD057EBB46D348B2E5A70BC42ED4E3E3B1EECAC345AD4F13307897A60253AE44
File 6	ClaimBatchRewards.sol
MD5 hash	5EB8A97A044A03D6F5CB7AF247645561
SHA256 hash	7D5D4E8C5C20A120F56F9C7D671B134C8999DCD67B37A8EDBBE53210114C1ED8
File 7	Claim.sol
MD5 hash	18772866D8A6E3C0346472BE0167E64E
SHA256 hash	A9339588F651CDC64B893100D8E317DD76B9203D4D646C14BD3F53CF53EA91B6
File 8	ClaimCoins.sol
MD5 hash	ADDDC1B4C9C81494EBF605B6C01F0F95

SHA256 hash	0F39A4188238303E08412EBA993C1C1E41F84FCDA681FCBEBB0B37183C2A64F0
File 9	AutoRewardClaimTokensTracker.sol
MD5 hash	27DD1F1762E086C83FC422AFF39DEF6A
SHA256 hash	9E9D3CC85A3F5347C4D6CB5385EC4D8C053DB5644DD5226EA15FBEB935F6F1AE
File 10	CumulativeRewardsCalculator.sol
MD5 hash	28B03380443F5196BC0720A0663C3F99
SHA256 hash	23E885FC99BC30A81EF423907107ABCB99C06D8958A7FE2595665687A56DADDB
File 11	CustomRewardTokenConfig.sol
MD5 hash	1E555EFC72CF1E52BBA8357E15EA3965
SHA256 hash	F44A696BEAF7FED8609E2823AB66D69C86E6F649B13AB0CBBD4598C67E51D1D9
File 12	CustomRewardTokenWhitelist.sol
MD5 hash	1A9E6CDAC0116A2FDDB73743F8373F75

SHA256 hash	741102EFB403F2C2A5F731CF028CA0F8163A5587EE790DD9935AF53BD08C6CF8
File 13	ManualRewardClaimTokensTracker.sol
MD5 hash	A0080F16DAC44C3490242E612D2A1F88
SHA256 hash	08AA1513D91E568994057474FC7697711563BE045B5A4EC2467CE0776AFE0754
File 14	RewardCycle.sol
MD5 hash	3C7BD88320936ABE067C52A2A6FA4D1C
SHA256 hash	40160F02909C1D08FC12FB10B5A7CE43BF1E70B6C60956E5430B458C0A4F7D8C
File 15	RewardExcludedList.sol
MD5 hash	1B35D9B6F4162976F43AA58943DC98C4
SHA256 hash	76015894BCC6E7C0B7DF696CBB8900072B09938BAAFD9BFB458ACC180BE055BA
File 16	StableTokenRewardManager.sol
MD5 hash	85E4249766CEEE3D7C59D3F04A31C213

SHA256 hash	0220939DA32D28F3D3E25D23DDB22B47F83058D98EAC74135D0ED2CA98B20763
File 17	BEP20Swap.sol
MD5 hash	3DFFDB6AA2BC56FC06221095EF8E9968
SHA256 hash	9CD848A293555F9EAB255EC7DE9ECDFE3E9035595883F9709870AE3CD3DEC7CA
File 18	Rewards.sol
MD5 hash	8EE5614B929AC442A3C521EDFE2783B2
SHA256 hash	2A05306156C79AAC79496ED32CA1CB1D85239E9FD9E161AC9C13A8B961F8391D
File 19	RouterManager.sol
MD5 hash	1FB02F713A8CA6F3FE260FFB91B4EA4D
SHA256 hash	21A691BCDF5A474CDA61DC7E9377F4C20F83770D774A1823168704C070634186
File 20	SwapManager.sol
MD5 hash	6332B301F1A5BED6ABCDF3F1CC838E72

SHA256 hash	B802995B9CD7B91FC4B559D3341DAD42CF24B06088B88613776999CDC0BD38CD
File 21	DailySellManager.sol
MD5 hash	CA49E323BD2B5D63EE6277681BD43501
SHA256 hash	1B881EAF10253DBF177E0E8CC7467DEDB8EFA3D2CCC5A87B73D0F66ED708C9EB
File 22	ExternalContracts.sol
MD5 hash	035ACB47C64C941639CE0EBE7C75F0A9
SHA256 hash	320C1A718F158EBD3F0B9982D011C850AF89AD46C3B1C56BB79D10652B49576C
File 23	ERC20.sol
MD5 hash	E2F3CE0755EDAEBB5E5857FB3EDE29EA
SHA256 hash	4D8731097E2B8E004CA68B2B70512F4A653F499ECF56AA4DBDAD7F8E8CF11672
File 24	ERC20Burnable.sol
MD5 hash	8FC0209767F2565CB7923594FAE6CAA5

SHA256 hash	6B96641921F3DC53A1A7961738073B959D93934B150D77EE4BEC374A30A81375
File 25	ERC20Permit.sol
MD5 hash	7EFF8F7F486721F68109BB7C18FD814A
SHA256 hash	2081EF5942EE9F8CF3D663343196BB0D670F4775551FEC1DFACCDE721FBA7C2F
File 26	LiquidityAddingsManager.sol
MD5 hash	B2BBF146C79FB082062E94493F19E15F
SHA256 hash	43CF2D65B250002E10A8368A4F28227E7C04C23310A045F1EB33FDF6E456A0F2
File 27	LiquidityManager.sol
MD5 hash	B81D781FCDF49943F2F7E33A69DC1676
SHA256 hash	8A5410AF411ABB46D46447619E1DA1F7FB62DB6127387106A864BA265656E4D8
File 28	SwapToBNB.sol
MD5 hash	DE54C3CFAFEA0CD9468CB02A7E100E86

SHA256 hash	D45424D1119CEEFE2A3A7C396625AE0C557CD1A31BA87D7143670104248DA218
File 29	TransferFee.sol
MD5 hash	FC6B16E0D6238B1405729CC956765012
SHA256 hash	3E26A9DFEED85BEBB568520EBF1967F8D77AA2096D0524823FBCEBB2DD47C570
File 30	TransferFeeActivation.sol
MD5 hash	D03E0E5FE2B96E0D46BAFC2BE9099B77
SHA256 hash	DA034EFAB6E17336676BC6A96BFB008589A24FF0EA6AC43727BE2A667A581944
File 31	TransferFeeExclusionList.sol
MD5 hash	24E057763AB311AE885AA27EEBDE1014
SHA256 hash	E7AD57B20DAE0C6499D1175A34D90832F2B5F3C2B1BD548B609E4A1196912E9C
File 32	Transfers.sol
MD5 hash	20DF62BBE64721FC64E2BF6D53E9DBAD

SHA256 hash	3B730BB1B7D51404AE1466A3FEDAEFD073ED430015DE5E9C2B273367C6631A39
Date	07/12/2021

Introduction

RD Auditors (Consultant) were contracted by Omnia Defi (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report represents the findings of the security assessment of the customer`s smart contracts and its code review conducted between 29th November - 07th December 2021.

This contract consists of 32 files.

Project Scope

The scope of the project is a smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is **well-secured.**

Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, the 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 AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	0
Critical	0
High	0
Medium	0
Low	0
Very Low	0

Code Quality

Please note that within this report safeMath, IERC20, EnumerableSet Math, SafeERC20, ReentrancyGuard, Pausable, Address, ownable are taken from the popular OpenZeppelin library.

The libraries within this smart contract are part of a logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other contracts.

The Omnia Defi team has provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is commented on. Commenting can provide rich documentation for functions, return variables and more. Use of Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

Documentation

We were given the Omnia code as a link:

https://gitlab.com/createlinx/omnia/omnia-token/-/tree/production/src/contracts

The hash of that file is mentioned in the table. As mentioned above, It's recommended to write comments in the smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

Omnia Defi

File And Function Level Report

File: AutoClaim.sol

Contract: AutoClaim

Import: ClaimBatchRewards,AutoClaimInterval,AutoClaimPausable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	processAutoClaim	write	Passed	All Passed	No Issue	Passed

File: AutoClaimDetails.sol

Contract: AutoClaimDetails

Import: AutoClaim

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setMaxBatchRewardsDistributionGAS	write	Passed	All Passed	No Issue	Passed

2	setAutoClaimMinAmount	write	Passed	All Passed	No Issue	Passed

File: AutoClaimInterval.sol

Contract: AutoClaimInterval

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setAutoClaimInterval	write	Passed	All Passed	No Issue	Passed
2	_updateAutoClaimProcessInterval	write	Passed	All Passed	No Issue	Passed
3	processAutoClaimIntervalReached	read	Passed	All Passed	No Issue	Passed

File: AutoClaimPausable.sol

Contract: AutoClaimPausable

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	autoClaimPaused	read	Passed	All Passed	No Issue	Passed
2	setPauseAutoClaim	write	Passed	All Passed	No Issue	Passed

File: AutoClaimQueueList.sol

Contract: AutoClaimQueueList

Import: RewardExcludedList, EnumerableSet

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	updateAutoClaimQueue	write	Passed	All Passed	No Issue	Passed
2	setExcludedFromAutoClaim	write	Passed	All Passed	No Issue	Passed

File: ClaimBatchRewards.sol

Contract: AutoClaimQueueList

Import: AutoClaimQueueList,CumulativeRewardsCalculator, BEP20Swap

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setMaxBatchRewardClaimLength	write	Passed	All Passed	No Issue	Passed
2	_processRewardsAutoClaim	write	Passed	All Passed	No Issue	Passed
3	_calculateAmountOfRewardsToBeSent	write	Passed	All Passed	No Issue	Passed
4	_swapOMNIAToStablecoinForRewards	write	Passed	All Passed	No Issue	Passed
5	_distributeRewards	write	Passed	All Passed	No Issue	Passed
6	_distributeRewardsOf	write	Passed	All Passed	No Issue	Passed
7	_distributeStablecoinRewardOf	write	Passed	All Passed	No Issue	Passed

File: Claim.sol

Contract: Claim

Import: ClaimCoins

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	claimReward	write	Passed	All Passed	No Issue	Passed
2	_doClaimReward	write	Passed	All Passed	No Issue	Passed

File: ClaimCoins.sol

Contract: ClaimCoins

Import: CumulativeRewardsCalculator, BEP20Swap

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_claimTokens	write	Passed	All Passed	No Issue	Passed
2	_claimCustomToken	write	Passed	All Passed	No Issue	Passed
3	_claimBEP20	write	Passed	All Passed	No Issue	Passed

File: AutoRewardClaimTokensTracker.sol

Contract: AutoRewardClaimTokensTracker

Import: Context

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setAutoRewardClaimTokenConfig	write	Passed	All Passed	No Issue	Passed

2	_calculateAutoClaimRewardTokensOf	read	Passed	All Passed	No Issue	Passed
3	areAutoClaimRewardTokensPercentageSet	read	Passed	All Passed	No Issue	Passed

File: CumulativeRewardsCalculator.sol

Contract: CumulativeRewardsCalculator

Import: RewardCycle,
AutoRewardClaimTokensTracker,
ManualRewardClaimTokensTracker,
RewardExcludedList,
CustomRewardTokenConfig,
StableTokenRewardManager

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_updateRewardsPaid	write	Passed	All Passed	No Issue	Passed
2	updateEligibleSupplyForRewards	write	Passed	All Passed	No Issue	Passed
3	updateRewardPointsOnTransfer	write	Passed	All Passed	No Issue	Passed
4	_updateRewardPointsOf	write	Passed	All Passed	No Issue	Passed

5	_getRewardPointsOf	read	Passed	All Passed	No Issue	Passed
6	calculateAutoClaimRewardTokens	read	Passed	All Passed	No Issue	Passed
7	_calculateOMNIAReward	read	Passed	All Passed	No Issue	Passed

File: CustomRewardTokenConfig.sol

Contract: CustomRewardTokenConfig

Import: CustomRewardTokenWhitelist

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setCustomToken	write	Passed	All Passed	No Issue	Passed
2	customRewardToken	read	Passed	All Passed	No Issue	Passed

File: CustomRewardTokenWhitelist .sol

Contract: CustomRewardTokenWhitelist

Import: Ownable, Initializable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_initializeCustomRewardTokenWhitelist	write	Passed	All Passed	No Issue	Passed
2	addCustomRewardToken	write	Passed	All Passed	No Issue	Passed
3	removeCustomRewardToken	write	Passed	All Passed	No Issue	Passed
4	addBatchCustomRewardTokens	write	Passed	All Passed	No Issue	Passed
5	removeBatchCustomRewardTokens	write	Passed	All Passed	No Issue	Passed

File: ManualRewardClaimTokensTracker.sol

Contract: ManualRewardClaimTokensTracker

Import: Context

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setManualRewardClaimTokenConfig	write	Passed	All Passed	No Issue	Passed
2	_calculateManualClaimRewardTokensOf	read	Passed	All Passed	No Issue	Passed
3	areManualClaimRewardTokensPercentageSet	read	Passed	All Passed	No Issue	Passed

File: RewardCycle .sol

Contract: RewardCycle

Import: Ownable, Initializable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_initializeRewardCycle	write	Passed	All Passed	No Issue	Passed
2	setRewardCycleExtensionThreshold	write	Passed	All Passed	No Issue	Passed
3	updateNextClaimDate	write	Passed	All Passed	No Issue	Passed
4	calculateRewardCycleExtension	read	Passed	All Passed	No Issue	Passed

File: RewardExcludedList .sol

Contract: RewardExcludedList

Import: Ownable, Initializable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_initializeRewardExcludedList	write	Passed	All Passed	No Issue	Passed
2	_setExcludedFromRewards	write	Passed	All Passed	No Issue	Passed
3	excludeContractFromRewards	write	Passed	All Passed	No Issue	Passed
4	includeContractInRewards	write	Passed	All Passed	No Issue	Passed
5	setMinIndividualClaimAmount	write	Passed	All Passed	No Issue	Passed

File: StableTokenRewardManager .sol

Contract: StableTokenRewardManager

Import: Ownable, Initializable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_initializeStableTokenRewardManager	write	Passed	All Passed	No Issue	Passed
2	setRewardStableToken	write	Passed	All Passed	No Issue	Passed

File: BEP20Swap.sol

Contract: BEP20Swap

Import: IRouterManager, ISwapManager,ERC20

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_swapOMNIAForBEP20	write	Passed	All Passed	No Issue	Passed
2	_swapOMNIAForExactStablecoinRewardAmount	write	Passed	All Passed	No Issue	Passed

File: Rewards .sol

Contract: Rewards

Import: Claim, AutoClaimDetails, IRewards

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	_addressesExcludedFromRewardsByDefault	write	Passed	All Passed	No Issue	Passed

3	_tokensWhitelistedByDefault	write	Passed	All Passed	No Issue	Passed
4	transferLostBEP20	write	Passed	All Passed	No Issue	Passed
5	transferLostBNB	write	Passed	All Passed	No Issue	Passed
6	processAutoClaimIntervalReached	read	Passed	All Passed	No Issue	Passed
7	processAutoClaim	write	Passed	All Passed	No Issue	Passed
8	updateAutoClaimQueue	write	Passed	All Passed	No Issue	Passed
9	updateEligibleSupplyForRewards	write	Passed	All Passed	No Issue	Passed
10	updateNextClaimDate	write	Passed	All Passed	No Issue	Passed
11	updateRewardPointsOnTransfer	write	Passed	All Passed	No Issue	Passed
12	excludeContractFromRewards	write	Passed	All Passed	No Issue	Passed

File: RouterManager .sol

Contract: RouterManager

Import: Ownable, Initializable, IRouterManager

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	updateRewards Contract	write	Passed	All Passed	No Issue	Passed
3	setPancakeSwa pRouter	write	Passed	All Passed	No Issue	Passed

File: SwapManager .sol

Contract: SwapManager

Import: ISwapManager, Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setMaxSlippage	write	Passed	All Passed	No Issue	Passed
2	pathAndMaxOu t	read	Passed	All Passed	No Issue	Passed

File: DailySellManager .sol

Contract: DailySellManager

Import: ExternalContracts

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_triggerDailyLimit	write	Passed	All Passed	No Issue	Passed
2	amountAllowedToSell	read	Passed	All Passed	No Issue	Passed
3	calculateCirculatingSupply	read	Passed	All Passed	No Issue	Passed

File: ExternalContracts .sol

Contract: ExternalContracts

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	excludeFarmingContractFromCirculatingSupply	write	Passed	All Passed	No Issue	Passed
2	reincludeFarmingContractInCirculatingSuppy	write	Passed	All Passed	No Issue	Passed
3	renounceExcludingFarmingContract	write	Passed	All Passed	No Issue	Passed
4	setVestingContract	write	Passed	All Passed	No Issue	Passed

File: ERC20.sol

Contract: ERC20

Import: Context, IERC20, IERC20Metadata

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	transfer	write	Passed	All Passed	No Issue	Passed
2	allowance	read	Passed	All Passed	No Issue	Passed
3	approve	write	Passed	All Passed	No Issue	Passed
4	transferFrom	write	Passed	All Passed	No Issue	Passed
5	increaseAllowance	write	Passed	All Passed	No Issue	Passed
6	decreaseAllowance	write	Passed	All Passed	No Issue	Passed
7	_mint	write	Passed	All Passed	No Issue	Passed
8	_burn	write	Passed	All Passed	No Issue	Passed

File: ERC20Burnable.sol

Contract: ERC20Burnable

Import: Context, ERC20Permit

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	burn	write	Passed	All Passed	No Issue	Passed
2	burnFrom	write	Passed	All Passed	No Issue	Passed

File: ERC20Permit .sol

Contract: ERC20Permit

Import: ERC20, IERC20Permit, EIP712

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	permit	write	Passed	All Passed	No Issue	Passed
2	nonces	read	Passed	All Passed	No Issue	Passed
3	_useNonce	write	Passed	All Passed	No Issue	Passed

File: LiquidityAddingsManager.sol

Contract: LiquidityAddingsManager

Import: LiquidityManager,
SwapToBNB,
ERC20Burnable,
TransferFeeActivation

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_addLiquidityIf Needed	write	Passed	All Passed	No Issue	Passed
2	_swapAndAddL iquidity	write	Passed	All Passed	No Issue	Passed

File: LiquidityManager .sol

Contract: LiquidityManager

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setLiquidityAdd ingThreshold	write	Passed	All Passed	No Issue	Passed
2	setAutoLiquidit yWallet	write	Passed	All Passed	No Issue	Passed

File: SwapToBNB.sol

Contract: SwapToBNB

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_swapOMNIAForBNB	write	Passed	All Passed	No Issue	Passed
2	isSwapAddingLiquidity	read	Passed	All Passed	No Issue	Passed

File: TransferFee.sol

Contract: TransferFee

Import: TransferFeeExclusionList, TransferFeeActivation

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_calculateFeeRate	read	Passed	All Passed	No Issue	Passed
2	_canApplyTransferFee	read	Passed	All Passed	No Issue	Passed

File: TransferFeeActivation.sol

Contract: TransferFeeActivation

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	activatePcsListing	read	Passed	All Passed	No Issue	Passed

File: TransferFeeExclusionList.sol

Contract: TransferFeeExclusionList

Import: Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setExcludedFromFees	write	Passed	All Passed	No Issue	Passed

File: Transfers.sol

Contract: Transfers

Import: DailySellManager,

TransferManager, LiquidityAddingsManager

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_mint	write	Passed	All Passed	No Issue	Passed

2	_transfer	write	Passed	All Passed	No Issue	Passed
3	_beforeTokenTransfer	write	Passed	All Passed	No Issue	Passed
4	_processTransfer	write	Passed	All Passed	No Issue	Passed
5	_updateBalances	write	Passed	All Passed	No Issue	Passed
6	_afterTokenTransfer	write	Passed	All Passed	No Issue	Passed
7	_calculateTransferAmounts	write	Passed	All Passed	No Issue	Passed
8	_calculateSpecificFees	read	Passed	All Passed	No Issue	Passed

File: OMNIA.sol

Contract: OMNIA

Import: Transfers

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	_addressesExcludedFromFeesByDefault	write	Passed	All Passed	No Issue	Passed
2	updateRewardsContract	write	Passed	All Passed	No Issue	Passed
3	approveAndCall	write	Passed	All Passed	No Issue	Passed
4	transferLostBEP20	write	Passed	All Passed	No Issue	Passed

5	transferLostBNB	write	Passed	All Passed	No Issue	Passed
6	transferOwnership	write	Passed	All Passed	No Issue	Passed
7	_excludeExternalContractFromFees	write	Passed	All Passed	No Issue	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot 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:

No low severity vulnerabilities were found.

Very Low:

No very low severity vulnerabilities were found.

Conclusion

We were given a contract file and have used all possible tests based on the given object. The contract is written systematically, so it is ready to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) was 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.

The security state of the reviewed contract is now “well secured”

Note For Contract Users

Owner has full control over the smart contract. Thus, technical auditing does not guarantee the project's ethical side.

Please do your due diligence before investing. Our audit report is never an investment advice.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

RD Auditors Disclaimer

The smart contracts given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Email: info@rdauditors.com

Website: www.rdauditors.com