InterFi
NETWORK

SMART CONTRACT SECURITY AUDIT OF
REFLECTO V2

SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

Audit Introduction

Auditing Firm	InterFi Network
Audit Architecture	InterFi Echelon Auditing Standard
Language	Solidity
Client Firm	Reflecto V2
Website	https://reflecto.finance/
Telegram	https://t.me/reflectocoin/
Twitter	https://twitter.com/reflectocoin/
Instagram	https://www.instagram.com/reflectotoken/
Discord	https://discord.gg/jsqp2chuPY/
Reddit	https://www.reddit.com/r/Reflecto/
Linktr.ee	https://linktr.ee/reflectocoin/
Report Date	June 22, 2022

About Reflecto V2

Reflecto V2 (RTO) protects your investment by utilizing an advanced anti-whale system as well as double burn mechanism. It also gives you the ability to select your reward token(s) with a high APY. Reflecto is built with utilities in mind; it has an integrated meta transaction feature that will allow performing gasless transactions, which means users will not need to spend BNB to execute transactions on the blockchain. Reflecto also launched its wallet at reflectowallet.com, so holders can send free transactions and earn passive income from Reflecto token and "Claim and Earn" functionality.

Audit Summary

InterFi team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ Reflecto V2's solidity source code has **LOW RISK SEVERITY**

- ❖ Reflecto V2's smart contract has an **ACTIVE OWNERSHIP**

- ❖ Reflecto V2's centralization risk correlated to the active owner is **MEDIUM**

- ❖ Important privileges – **SET AUTOSTAKING CONTRACT, SET SELL LIMIT, SET MAX SELL CANCEL, UPDATE BUYBACK SETTINGS, MIGRATE V1 TO V2, BURN TOKENS, BLACKLIST REWARD TOKEN**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, exploitability, and audit disclaimer, kindly refer to the audit.

🛑 Contract address: ~~0x5A341DCF49e161CC73591f02e5f8CDE8A29733fb~~ **(Unverified)**

🗝 Blockchain: **Binance Smart Chain**

✅ Verify the authenticity of this report on InterFi's GitHub: **https://github.com/interfinetwork**

Table Of Contents

Audit Scope

InterFi was consulted by Reflecto V2 to conduct the smart contract security audit of their solidity source codes. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

❖ ReflectoV2.sol

Solidity Source Code On InterFi GitHub

https://github.com/interfinetwork/audited-codes/blob/main/ReflectoV2.sol

SHA-1 Hash

Solidity source code is audited at hash #0c0ef9aae7f4f71ad9acfeb961272a004a3ac8c4

Audit Methodology

The scope of this report is to audit the smart contract source code of Reflecto V2. InterFi has scanned contracts and reviewed codes for common vulnerabilities, exploits, hacks, and backdoors. Due to being out of scope, InterFi has not tested contracts on testnet to assess any functional flaws. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities

- ❖ Re-entrancy
- ❖ Unhandled Exceptions
- ❖ Transaction Order Dependency
- ❖ Integer Overflow
- ❖ Unrestricted Action
- ❖ Incorrect Inheritance Order
- ❖ Typographical Errors
- ❖ Requirement Violation

Source Code Review

- ❖ Gas Limit and Loops
- ❖ Deployment Consistency
- ❖ Repository Consistency
- ❖ Data Consistency
- ❖ Token Supply Manipulation
- ❖ Access Control and Authorization
- ❖ Operations Trail and Event Generation
- ❖ Assets Manipulation
- ❖ Ownership Control
- ❖ Liquidity Access

InterFi's Echelon Audit Standard

The aim of InterFi's "Echelon" standard is to analyze smart contracts and identify the vulnerabilities and the hacks. Kindly note, InterFi does not test smart contracts on testnet. It is recommended that smart contracts are thoroughly tested prior to the audit submission. Mentioned are the steps used by InterFi to audit smart contracts:

1. Solidity smart contract source code reviewal:
   - ❖ Review of the specifications, sources, and instructions provided to InterFi to make sure we understand the size, and scope of the smart contract audit.
   - ❖ Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.
2. Static, Manual, and Software analysis:
   - ❖ Test coverage analysis is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - ❖ Symbolic execution is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- ❖ Consensys Tools
- ❖ SWC Registry
- ❖ Solidity Coverage
- ❖ Open Zeppelin Code Analyzer
- ❖ Solidity Code Complier

Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function that requires owning the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! High	This level vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
! Medium	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity
! Low	This level vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Informational	This level vulnerabilities can be ignored. They are code style violations and informational statements in the code. They may not affect the smart contract execution

Centralization Risk

Centralization risk is the most common cause of decentralized finance hacks. When a smart contract has an active contract ownership, the risk related to centralization is elevated. There are some well-intended reasons to be an active contract owner, such as:

❖ Contract owner can be granted the power to `pause()` or `lock()` the contract in case of an external attack.

❖ Contract owner can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale, and to list on an exchange.

Authorizing a full centralized power to a single body can be dangerous. Unfortunately, centralization related risks are higher than common smart contract vulnerabilities. Centralization of ownership creates a risk of rug pull scams, where owners cash out tokens in such quantities that they become valueless. **Most important question to ask here is, how to mitigate centralization risk?** Here's InterFi's recommendation to lower the risks related to centralization hacks:

❖ Smart contract owner's private key must be carefully secured to avoid any potential hack.

❖ Smart contract ownership should be shared by multi-signature (multi-sig) wallets.

❖ Smart contract ownership can be locked in a contract, user voting, or community DAO can be introduced to unlock the ownership.

Reflecto V2's Centralization Status

❖ Reflecto V2's smart contract has an **active ownership.**

❖ Reflecto V2's smart contract is **not verified**.

Static Analysis

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable
🔒	Function is locked
🔓	Function can be accessed
❗	Important functionality

```
| **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❗ |
||||||
| **Auth** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑   |NO❗ |
| └ | authorize | Public ❗ | 🛑   | onlyOwner |
| └ | unauthorize | Public ❗ | 🛑   | onlyOwner |
| └ | isOwner | Public ❗ |    |NO❗ |
| └ | isAuthorized | Public ❗ |    |NO❗ |
| └ | transferOwnership | Public ❗ | 🛑   | onlyOwner |
||||||
| **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 🔒 |    | |
||||||
| **IDEXRouter** | Interface |  |||
|  ∟ | factory | External ❗ |    |NO❗ |
|  ∟ | WETH | External ❗ |    |NO❗ |
|  ∟ | WBNB | External ❗ |    |NO❗ |
|  ∟ | addLiquidity | External ❗ | 🔴   |NO❗ |
|  ∟ | addLiquidityETH | External ❗ | 💵 |NO❗ |
|  ∟ | addLiquidityBNB | External ❗ | 💵 |NO❗ |
|  ∟ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🔴   |NO❗ |
|  ∟ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 |NO❗ |
|  ∟ | swapExactBNBForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 |NO❗ |
|  ∟ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🔴   |NO❗ |
|  ∟ | swapExactTokensForBNBSupportingFeeOnTransferTokens | External ❗ | 🔴   |NO❗ |
||||||
| **IDEXFactory** | Interface |  |||
|  ∟ | createPair | External ❗ | 🔴   |NO❗ |
|  ∟ | getPair | External ❗ |    |NO❗ |
||||||
| **IProviderPair** | Interface |  |||
|  ∟ | getReserves | External ❗ |    |NO❗ |
|  ∟ | sync | External ❗ | 🔴   |NO❗ |
|  ∟ | token0 | External ❗ |    |NO❗ |
|  ∟ | token1 | External ❗ |    |NO❗ |
||||||
| **ICalculator** | Interface |  |||
|  ∟ | getFees | External ❗ |    |NO❗ |
|  ∟ | registerBuySell | External ❗ | 🔴   |NO❗ |
|  ∟ | isCustomFeeReceiverOrSender | External ❗ |    |NO❗ |
|  ∟ | getPressure | External ❗ |    |NO❗ |
|  ∟ | getFundPoolByPressureData | External ❗ |    |NO❗ |
|  ∟ | getSwapThreshold | External ❗ |    |NO❗ |
|  ∟ | getUserFees | External ❗ |    |NO❗ |
|  ∟ | getAlgoBuybackData | External ❗ |    |NO❗ |
|  ∟ | getSellCancelingAmount | External ❗ |    |NO❗ |
|  ∟ | getPrices | External ❗ |    |NO❗ |
||||||
| **IDividendDistributor** | Interface |  |||
|  ∟ | setDistributionCriteria | External ❗ | 🔴   |NO❗ |
|  ∟ | setShare | External ❗ | 🔴   |NO❗ |
|  ∟ | deposit | External ❗ | 💵 |NO❗ |
|  ∟ | process | External ❗ | 🔴   |NO❗ |
||||||
| **ReentrancyGuard** | Implementation |  |||
|  ∟ | <Constructor> | Public ❗ | 🔴   |NO❗ |
||||||
| **DividendDistributor** | Implementation | IDividendDistributor, ReentrancyGuard |||
|  ∟ | <Constructor> | Public ❗ | 🔴   |NO❗ |
|  ∟ | setDistributionCriteria | External ❗ | 🔴   | onlyToken |
|  ∟ | setAddMyTokenFee | External ❗ | 🔴   | onlyOwner |
```

```
| └ | setFreeMyTokenToAdd | External ❗ | 🛑 | onlyOwner |
| └ | rewardsTokensCount | Public ❗ | |NO❗ |
| └ | rewardsTokens | Public ❗ | |NO❗ |
| └ | pendingRewardsTokens | Public ❗ | |NO❗ |
| └ | getUsersRewardsTokens | Public ❗ | |NO❗ |
| └ | setTokenListeningExpirationIn | External ❗ | 🛑 | onlyOwner |
| └ | setOwnRouter | External ❗ | 🛑 | onlyOwner |
| └ | setTokenSupportedByOurDex | External ❗ | 🛑 | onlyOwner |
| └ | setOtherRouter | External ❗ | 🛑 | onlyOwner |
| └ | setRoundRoubinRouter | External ❗ | 🛑 | onlyOwner |
| └ | getRouter | Internal 🔒 | 🛑 | |
| └ | setDefaultRewardTokenForContracts | External ❗ | 🛑 | onlyOwner |
| └ | setDefaultRewardToken | External ❗ | 🛑 | onlyOwner |
| └ | addRewardTokenToList | External ❗ | 🛑 | onlyOwner |
| └ | approveTokenAdding | External ❗ | 🛑 | onlyOwner |
| └ | rejectTokenAdding | External ❗ | 🛑 | onlyOwner |
| └ | blacklistRewardToken | External ❗ | 🛑 | onlyOwner |
| └ | removeRewardTokenFromList | External ❗ | 🛑 | onlyOwner |
| └ | setFeeAddress | External ❗ | 🛑 | onlyOwner |
| └ | setReflectoAddress | External ❗ | 🛑 | onlyOwner |
| └ | setListeningFee | External ❗ | 🛑 | onlyOwner |
| └ | setShare | External ❗ | 🛑 | onlyToken |
| └ | deposit | External ❗ | 💵 | onlyToken |
| └ | process | External ❗ | 🛑 | onlyToken |
| └ | shouldDistribute | Internal 🔒 | | |
| └ | listMyToken | External ❗ | 💵 |NO❗ |
| └ | approveToken | External ❗ | 🛑 | onlyOwner |
| └ | rejectToken | External ❗ | 🛑 | onlyOwner |
| └ | closeExpiredListenings | External ❗ | 🛑 |NO❗ |
| └ | removeItemFromArray | Internal 🔒 | 🛑 | |
| └ | addMyRewardToken | External ❗ | 💵 |NO❗ |
| └ | removeMyRewardToken | External ❗ | 🛑 |NO❗ |
| └ | distributeDividend | Internal 🔒 | 🛑 | |
| └ | isContract | Internal 🔒 | | |
| └ | distributeDividendClaim | Internal 🔒 | 🛑 | |
| └ | sentRewardInBNB | Internal 🔒 | 🛑 | |
| └ | sentRewardInToken | Internal 🔒 | 🛑 | |
| └ | claimDividend | External ❗ | 🛑 | nonReentrant |
| └ | claim | Internal 🔒 | 🛑 | |
| └ | gaslessClaim | External ❗ | 🛑 | onlyOwner nonReentrant |
| └ | getMessageHash | Public ❗ | |NO❗ |
| └ | verify | Internal 🔒 | | |
| └ | recoverSigner | Public ❗ | |NO❗ |
| └ | splitSignature | Public ❗ | |NO❗ |
| └ | setClaimPeriod | External ❗ | 🛑 | onlyOwner |
| └ | getUnpaidEarnings | Public ❗ | |NO❗ |
| └ | getCumulativeDividends | Internal 🔒 | | |
| └ | addShareholder | Internal 🔒 | 🛑 | |
| └ | getShareholders | External ❗ | | onlyOwner |
| └ | getShareholderAmount | External ❗ | |NO❗ |
```

∟	removeShareholder	Internal 🔒	🔴	
CastU256U128	Library			
∟	u128	Internal 🔒		
CastU256U32	Library			
∟	u32	Internal 🔒		
SafeMathInt	Library			
∟	mul	Internal 🔒		
∟	div	Internal 🔒		
∟	sub	Internal 🔒		
∟	add	Internal 🔒		
∟	abs	Internal 🔒		
SafeMath1	Library			
∟	mul	Internal 🔒		
∟	div	Internal 🔒		
∟	sub	Internal 🔒		
∟	add	Internal 🔒		
∟	mod	Internal 🔒		
Initializable	Implementation			
∟	isConstructor	Private 🔐		
IERC20	Interface			
∟	totalSupply	External ❗		NO❗
∟	name	External ❗		NO❗
∟	balanceOf	External ❗		NO❗
∟	allowance	External ❗		NO❗
∟	transfer	External ❗	🔴	NO❗
∟	approve	External ❗	🔴	NO❗
∟	transferFrom	External ❗	🔴	NO❗
Ownable	Implementation	Initializable		
∟	initialize	Public ❗	🔴	initializer
∟	owner	Public ❗		NO❗
∟	isOwner	Public ❗		NO❗
∟	renounceOwnership	Public ❗	🔴	onlyOwner
∟	transferOwnership	Public ❗	🔴	onlyOwner
∟	_transferOwnership	Internal 🔒	🔴	
ReflectoRewards	Implementation	Ownable		
∟	<Constructor>	Public ❗	🔴	NO❗
∟	setMaxRewardPoolLength	External ❗	🔴	onlyOwner
∟	setPoolPrice	External ❗	🔴	onlyOwner
∟	earliest	Internal 🔒		
∟	setDefaultReward	External ❗	🔴	onlyOwner
∟	changeRate	External ❗	🔴	onlyRewardOwner
∟	changeEndDate	External ❗	🔴	onlyRewardOwner

```
| └ | _updateRewardsPerToken | Public ❗ | 🔴  |NO❗ |
| └ | _updateUserRewards | Public ❗ | 🔴  |NO❗ |
| └ | excludeFromRewards | External ❗ | 🔴  | onlyOwner |
| └ | setReflectoContractAddress | External ❗ | 🔴  | onlyOwner |
| └ | updateReflecto | External ❗ | 🔴  | onlyOwner |
| └ | setDeveloperAddress | External ❗ | 🔴  | onlyOwner |
| └ | addReward | External ❗ |  💵 |NO❗ |
| └ | approveReward | Public ❗ | 🔴  | onlyOwner |
| └ | rejectPool | Public ❗ | 🔴  | onlyOwner |
| └ | removeToken | Internal 🔒 | 🔴  | |
| └ | approveMultipleReward | External ❗ | 🔴  |NO❗ |
| └ | approvedTokens | Public ❗ |  |NO❗ |
| └ | pendingTokens | Public ❗ |  |NO❗ |
| └ | setDefaultToken | External ❗ | 🔴  | onlyOwner |
| └ | getMessageHash | Public ❗ |  |NO❗ |
| └ | verify | Internal 🔒 |  | |
| └ | recoverSigner | Public ❗ |  |NO❗ |
| └ | splitSignature | Public ❗ |  |NO❗ |
| └ | setClaimPeriod | External ❗ | 🔴  | onlyOwner |
| └ | gaslessClaim | External ❗ | 🔴  | onlyOwner |
| └ | claim | Public ❗ | 🔴  |NO❗ |
||||||
| **IReflectoRewards** | Interface |  |||
| └ | _updateRewardsPerToken | External ❗ | 🔴  |NO❗ |
| └ | _updateUserRewards | External ❗ | 🔴  |NO❗ |
||||||
| **Reflecto** | Implementation | IBEP20, Auth |||
| └ | <Constructor> | Public ❗ | 🔴  | Auth |
| └ | setProcessing | External ❗ | 🔴  | authorized |
| └ | endMigration | External ❗ | 🔴  | authorized |
| └ | setAutoStakingContract | External ❗ | 🔴  | authorized |
| └ | setSellCancelingEnabled | External ❗ | 🔴  | authorized |
| └ | setMaxSellCancelingAmount | External ❗ | 🔴  | authorized |
| └ | setEnableRoundRobinBuyback | External ❗ | 🔴  | authorized |
| └ | setPoolSendEnabled | External ❗ | 🔴  | authorized |
| └ | addPair | External ❗ | 🔴  | authorized |
| └ | deletePair | External ❗ | 🔴  | authorized |
| └ | setDefaultRouter | External ❗ | 🔴  | authorized |
| └ | setDefaultRouterFromSwapBack | External ❗ | 🔴  | authorized |
| └ | addRouter | External ❗ | 🔴  | authorized |
| └ | removeRouter | External ❗ | 🔴  | authorized |
| └ | getRouter | Internal 🔒 |  | |
| └ | updateRouterIndex | Internal 🔒 | 🔴  | |
| └ | setFeeCalulator | External ❗ | 🔴  | authorized |
| └ | setMinimumStakingLimitAmount | External ❗ | 🔴  | authorized |
| └ | version | Public ❗ |  |NO❗ |
| └ | getChainID | External ❗ |  |NO❗ |
| └ | <Receive Ether> | External ❗ |  💵 |NO❗ |
| └ | donate | External ❗ |  💵 |NO❗ |
| └ | totalSupply | External ❗ |  |NO❗ |
```

```
|  └ | decimals | External ❗ |    |NO❗ |
|  └ | symbol | External ❗ |    |NO❗ |
|  └ | name | External ❗ |    |NO❗ |
|  └ | getOwner | External ❗ |    |NO❗ |
|  └ | balanceOf | Public ❗ |    |NO❗ |
|  └ | allowance | External ❗ |    |NO❗ |
|  └ | approve | Public ❗ | 🔴 |NO❗ |
|  └ | approveMax | External ❗ | 🔴 |NO❗ |
|  └ | transfer | External ❗ | 🔴 |NO❗ |
|  └ | transferFrom | External ❗ | 🔴 |NO❗ |
|  └ | migrateV1ToV2 | External ❗ | 🔴 | authorized |
|  └ | _transferFrom | Internal 🔒 | 🔴 | |
|  └ | setStakes | Internal 🔒 | 🔴 | |
|  └ | _basicTransfer | Internal 🔒 | 🔴 | |
|  └ | checkTxLimit | Internal 🔒 | 🔴 | |
|  └ | shouldTakeFee | Internal 🔒 |    | |
|  └ | takeFee | Internal 🔒 | 🔴 | |
|  └ | shouldSwapBack | Internal 🔒 |    | |
|  └ | sendToThePool | Internal 🔒 | 🔴 | |
|  └ | fundStakingPool | Internal 🔒 | 🔴 | |
|  └ | algoBB | Internal 🔒 | 🔴 | |
|  └ | getPath | Internal 🔒 |    | |
|  └ | swapBack | Internal 🔒 | 🔴 | swapping |
|  └ | shouldAutoBuyback | Internal 🔒 |    | |
|  └ | triggerZeusBuyback | External ❗ | 🔴 | authorized |
|  └ | clearBuybackMultiplier | External ❗ | 🔴 | authorized |
|  └ | buyTokens | Internal 🔒 | 🔴 | swapping |
|  └ | setAutoBuybackSettings | External ❗ | 🔴 | authorized |
|  └ | setDividendDistributer | External ❗ | 🔴 | authorized |
|  └ | setBuybackMultiplierSettings | External ❗ | 🔴 | authorized |
|  └ | launched | Internal 🔒 |    | |
|  └ | launch | Public ❗ | 🔴 | authorized |
|  └ | setSellLimit | External ❗ | 🔴 | authorized |
|  └ | setIsDividendExempt | External ❗ | 🔴 | authorized |
|  └ | setIsFeeExempt | External ❗ | 🔴 | authorized |
|  └ | setIsTxLimitExempt | External ❗ | 🔴 | authorized |
|  └ | setFeeReceivers | External ❗ | 🔴 | authorized |
|  └ | setSwapBackSettings | External ❗ | 🔴 | authorized |
|  └ | setSwapBackOnSell | External ❗ | 🔴 | authorized |
|  └ | setTargetLiquidity | External ❗ | 🔴 | authorized |
|  └ | setDistributionCriteria | External ❗ | 🔴 | authorized |
|  └ | setDistributorSettings | External ❗ | 🔴 | authorized |
|  └ | getCirculatingSupply | Public ❗ |    |NO❗ |
|  └ | getLiquidityBacking | Public ❗ |    |NO❗ |
|  └ | isOverLiquified | Public ❗ |    |NO❗ |
|  └ | _setAllowance | Internal 🔒 | 🔴 | |
|  └ | permit | External ❗ | 🔴 |NO❗ |
|  └ | convertTokensToBuyBack | External ❗ | 🔴 | authorized |
|  └ | burnTokenFromContract | External ❗ | 🔴 | authorized |
```

Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
43509138  =>  div(int256,int256)
48279900  =>  setOwnRouter(address)
67111404  =>  getUserFees(address,address)
72882732  =>  removeRouter(uint256)
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)
b6a5d7de  =>  authorize(address)
f0b37c04  =>  unauthorize(address)
2f54bf6e  =>  isOwner(address)
fe9fbb80  =>  isAuthorized(address)
f2fde38b  =>  transferOwnership(address)
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
8dd95002  =>  WBNB()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
eaaed442  =>  addLiquidityBNB(address,uint256,uint256,uint256,address,uint256)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95  =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
50e27df3  =>  swapExactBNBForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
```

```
d46d2f83  =>
swapExactTokensForBNBSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
c9c65396  =>  createPair(address,address)
e6a43905  =>  getPair(address,address)
0902f1ac  =>  getReserves()
fff6cae9  =>  sync()
0dfe1681  =>  token0()
d21220a7  =>  token1()
db8d55f1  =>  getFees()
f7e4ceb7  =>  registerBuySell(uint256,bool)
bd2e7064  =>  isCustomFeeReceiverOrSender(address,address)
2006f675  =>  getPressure()
62c092a0  =>  getFundPoolByPressureData(uint256,address,uint256)
f005f2ca  =>  getSwapThreshold()
dd7d4e29  =>  getAlgoBuybackData(uint256,bool,address)
ea7597c8  =>  getSellCancelingAmount(address,uint256)
572aae84  =>  getPrices(address)
2d48e896  =>  setDistributionCriteria(uint256,uint256)
14b6ca96  =>  setShare(address,uint256)
d0e30db0  =>  deposit()
ffb2c479  =>  process(uint256)
545b0d90  =>  setAddMyTokenFee(uint256)
d38c7a23  =>  setFreeMyTokenToAdd(address,bool)
2b534804  =>  rewardsTokensCount()
cd00671b  =>  rewardsTokens()
68c67660  =>  pendingRewardsTokens()
3692cb17  =>  getUsersRewardsTokens(address)
76efe514  =>  setTokenListeningExpirationIn(uint256)
e370b1ae  =>  setTokenSupportedByOurDex(address,bool)
3e7e8ec4  =>  setOtherRouter(address)
60a8a57e  =>  setRoundRoubinRouter(bool)
8369166d  =>  getRouter(address)
e7642c39  =>  setDefaultRewardTokenForContracts(address)
71c91527  =>  setDefaultRewardToken(address)
211d17fc  =>  addRewardTokenToList(address,uint256)
d4935925  =>  approveTokenAdding(address)
e77dd0d0  =>  rejectTokenAdding(address)
a561e027  =>  blacklistRewardToken(address,bool)
d2abfb9d  =>  removeRewardTokenFromList(uint256,address)
8705fcd4  =>  setFeeAddress(address)
9a230355  =>  setReflectoAddress(address)
8e7db5d8  =>  setListeningFee(uint256)
8c21cd52  =>  shouldDistribute(address)
aa7a57ad  =>  listMyToken(address)
022fc88b  =>  approveToken(address,uint256)
d0a05131  =>  rejectToken(address,uint256)
a8f26edd  =>  closeExpiredListenings()
36219c28  =>  removeItemFromArray(address[],uint256)
102f5117  =>  addMyRewardToken(address)
f0a4a78a  =>  removeMyRewardToken(address,uint256)
```

```
5319504a  =>  distributeDividend(address)
3f0f4411  =>  distributeDividendClaim(address,address,uint256,uint256)
57bfa9c7  =>  sentRewardInBNB(address,uint256)
eb54232d  =>  sentRewardInToken(address,uint256,address)
85e889ba  =>  claimDividend(address,address[])
8e2eba09  =>  claim(address,address[])
b0fe6d11  =>  gaslessClaim(address,address[],bytes32,bytes)
a35f8a40  =>  getMessageHash(address,uint256,string,uint256)
1a86b550  =>  verify(address,bytes32,bytes)
97aba7f9  =>  recoverSigner(bytes32,bytes)
a7bb5803  =>  splitSignature(bytes)
d2c04f43  =>  setClaimPeriod(uint256)
28fd3198  =>  getUnpaidEarnings(address)
e68af3ac  =>  getCumulativeDividends(uint256)
db29fe12  =>  addShareholder(address)
41ca641e  =>  getShareholders()
abd37753  =>  getShareholderAmount(address)
9babdad6  =>  removeShareholder(address)
f33b0a4f  =>  u128(uint256)
b481ea59  =>  u32(uint256)
bbe93d91  =>  mul(int256,int256)
adefc37b  =>  sub(int256,int256)
a5f3c23b  =>  add(int256,int256)
1b5ac4b5  =>  abs(int256)
c4cbfa44  =>  isConstructor()
c4d66de8  =>  initialize(address)
8da5cb5b  =>  owner()
8f32d59b  =>  isOwner()
715018a6  =>  renounceOwnership()
d29d44ee  =>  _transferOwnership(address)
63bde2eb  =>  setMaxRewardPoolLength(uint256)
fd228051  =>  setPoolPrice(uint256)
d0b30199  =>  earliest(uint256,uint256)
276a46f1  =>  setDefaultReward(uint32,uint32,uint96)
65872c50  =>  changeRate(address,uint96)
91e8075f  =>  changeEndDate(address,uint32)
26c3d7d8  =>  _updateRewardsPerToken()
68d0d54f  =>  _updateUserRewards(address)
2ad4cab7  =>  excludeFromRewards(address,bool)
8a237ee6  =>  setReflectoContractAddress(address)
3da08169  =>  updateReflecto(IERC20)
dcc345f2  =>  setDeveloperAddress(address)
c13249c4  =>  addReward(IERC20,uint32,uint32,uint256)
6f631ca8  =>  approveReward(IERC20)
be06ddc1  =>  rejectPool(IERC20)
bd8aedb0  =>  removeToken(IERC20)
1c29e6b3  =>  approveMultipleReward(IERC20[])
96c705e5  =>  approvedTokens()
c47d7029  =>  pendingTokens()
ae283161  =>  setDefaultToken(IERC20)
```
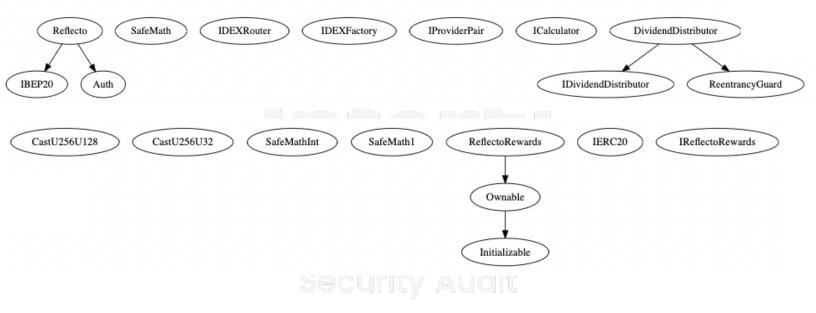
```
12929b13  =>  gaslessClaim(address,IERC20[],bytes32,bytes)
64c15400  =>  claim(address,IERC20[])
e24f4148  =>  setProcessing(bool,bool,bool)
6c525d04  =>  endMigration()
fb6d1c8d  =>  setAutoStakingContract(address)
2c5b8d93  =>  setSellCancelingEnabled(bool)
0d18948b  =>  setMaxSellCancelingAmount(uint256)
3549cf4d  =>  setEnableRoundRobinBuyback(bool)
2f938b46  =>  setPoolSendEnabled(bool)
c2b7bbb6  =>  addPair(address)
20e6a7f0  =>  deletePair(address)
5227767c  =>  setDefaultRouter(address,bool)
a894b045  =>  setDefaultRouterFromSwapBack(bool)
0aed36d0  =>  addRouter(address,bool)
b0f479a1  =>  getRouter()
55b0d487  =>  updateRouterIndex()
dfaf90e6  =>  setFeeCalulator(address)
7793fec1  =>  setMinimumStakingLimitAmount(uint256)
54fd4d50  =>  version()
564b81ef  =>  getChainID()
ed88c68e  =>  donate()
571ac8b0  =>  approveMax(address)
60a0b45d  =>  migrateV1ToV2(address[])
cb712535  =>  _transferFrom(address,address,uint256)
b629cad3  =>  setStakes(address,address)
f0774e71  =>  _basicTransfer(address,address,uint256)
4afa518a  =>  checkTxLimit(address,uint256)
e7c44c69  =>  shouldTakeFee(address)
2d8b8563  =>  takeFee(address,uint256,uint256,uint256)
b91e52c1  =>  shouldSwapBack(address,uint256)
1fa51c41  =>  sendToThePool(uint256)
01a1dcc8  =>  fundStakingPool(uint256)
469982da  =>  algoBB(bool,uint256)
2b544376  =>  getPath(IDEXRouter)
1521321b  =>  swapBack(Swap)
4d4e6fe5  =>  shouldAutoBuyback()
f5cfec0a  =>  triggerZeusBuyback(uint256,bool)
b210b06d  =>  clearBuybackMultiplier()
c625e9b1  =>  buyTokens(uint256,address)
8fe338d9  =>  setAutoBuybackSettings(bool,uint256)
27104a0f  =>  setDividendDistributer(address)
2375ce40  =>  setBuybackMultiplierSettings(uint256,uint256,uint256)
8091f3bf  =>  launched()
01339c21  =>  launch()
a9bf2c09  =>  setSellLimit(uint256)
f708a64f  =>  setIsDividendExempt(address,bool)
658d4b7f  =>  setIsFeeExempt(address,bool)
f84ba65d  =>  setIsTxLimitExempt(address,bool)
d7c01032  =>  setFeeReceivers(address,address,address)
b8c61130  =>  setSwapBackSettings(bool)
```

```
14f05c42  =>  setSwapBackOnSell(bool)
201e7991  =>  setTargetLiquidity(uint256,uint256)
9d1944f5  =>  setDistributorSettings(uint256)
2b112e49  =>  getCirculatingSupply()
266054ad  =>  getLiquidityBacking(uint256,address)
ace69718  =>  isOverLiquified(uint256,uint256,address)
c617fe6e  =>  _setAllowance(address,address,uint256)
8fcbaf0c  =>  permit(address,address,uint256,uint256,bool,uint8,bytes32,bytes32)
42925a06  =>  convertTokensToBuyBack(address)
aa57f9e7  =>  burnTokenFromContract()
```

Inheritance Graph

Manual Analysis

Function	Description	Available	Status
Total Supply	provides information about the total token supply	Yes	Passed
Balance Of	provides account balance of the owner's account	Yes	Passed
Transfer	executes transfers of a specified number of tokens to a specified address	Yes	Passed
Approve	allow a spender to withdraw a set number of tokens from a specified account	Yes	Passed
Allowance	returns a set number of tokens from a spender to the owner	Yes	Passed
Buy Back	is an action in which the project buys back its tokens from the existing holders usually at a market price	Yes	Passed
Burn	executes transfers of a specified number of tokens to a burn address	Yes	Passed
Dividend	executes transfers of a specified dividend token to a specified address	Yes	Passed
Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	Passed
Max Sell	a non-whitelisted wallet can only sell a specified number of tokens	Yes	Passed
Contract Fees	executes fee collection from swap events and/or transfer events	Yes	Passed

Transfer Ownership	executes transfer of contract ownership to a specified wallet	Yes	**Passed**
Renounce Ownership	executes transfer of contract ownership to a dead address	Yes	**Passed**

InterFi

Smart Contract
Security Audit

Notable Information 📝

❖ Smart contract owner can **authorize** a wallet to modify "write contract" parameters. When ownership is transferred, previous owner should not remain authorized.

```
function transferOwnership(address payable adr) public onlyOwner {
    owner = adr;
    authorizations[adr] = true;
```

❖ Smart contract owner can **burn** tokens from contract.

```
function burnTokenFromContract() external authorized {
    require(
        balanceOfContract > swapThreshold,
        "Threshold is gt then blance"
```

❖ Smart contract owner can **change swap back settings**.

```
function setSwapBackSettings(bool _enabled) external authorized {
    swapEnabled = _enabled;
function setSwapBackOnSell(bool _isSwapBackOnSell) external authorized {
    isSwapOnSell = _isSwapBackOnSell;
```

❖ Smart contract owner can **set dividend distributor** gas.

```
function setDistributorSettings(uint256 gas) external authorized {
    require(gas < 999999);
```

❖ Smart contract owner can **add or remove router**.

```
function setDefaultRouter(address router, bool isBNB) external authorized {
    defaultRouter = IDEXRouter(router);
function addRouter(address router, bool isBNB) external authorized {
    IDEXRouter routerNew = IDEXRouter(router);
function removeRouter(uint256 index) external authorized {
    IDEXRouter routerLast = routers[routers.length − 1];
```

❖ Smart contract utilizes **safemath** function to avoid common smart contract vulnerabilities.

```
string private _name = "ReflectoV2";
library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");
    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");
    return c;
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    return div(a, b, "SafeMath: division by zero");
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, "SafeMath: modulo by zero");
```

❖ Smart contract utilizes **re-entrancy guard** to prevent re-entrant calls. However, functions like `deposit(), withdraw(), setShare()` can be protected against re-entrancy.

❖ Smart contract owner can call `receive(),` it is executed on a call to the contract with empty call data. This is the function that is executed on plain ether transfers such as `send(),` and `transfer().` Make sure the contract can receive token through a regular transaction, and does not throw an exception.

```
receive() external payable {}
```

❖ Smart contract utilizes **redundant code** for `transferOwnership().` Ideal transfer ownership code should look be written like:

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
```

❖ Smart contract owner can approve **migration from V1 to V2.** There's an elevated risk of out-of-gas, and potential resource exhaustion errors with multi wallet migration.

```
function migrateV1ToV2(address[] memory v1holders)
    external
    authorized
    require(isMigrationEnded == false, "Migration completed");
    for (uint256 i = 0; i < v1holders.length; i++) {
        uint256 bHolder = IBEP20(0xEA3C823176D2F6feDC682d3cd9C30115448767b3)
```

❖ Smart contract authorized wallet can **set minimum staking limit**.

```
function setMinimumStakingLimitAmount(uint256 amount) external authorized {
    minimumStakingLimit = amount;
```

❖ Smart contract owner can **blacklist** certain reward tokens from interacting with the contract function modules.

```
function blacklistRewardToken(address rewardToken, bool isBlacklisted)
    external
    onlyOwner
```

❖ Smart contract owner can **change token listing fee.**

```
function setListeningFee(uint256 fee) external onlyOwner {
    feeForTokenListening = fee;
```

❖ Smart contract owner can **change max sell limit.** Minimum limit is set to allow the value change within the set parameters.

```
function setSellLimit(uint256 amount) external authorized {
    require(
        amount >= 100000000000000000000,
        "Put amount greater then 100b"
```

❖ Smart contract has a **low severity issue** which may or may not create any functional vulnerability.

"severity": 8, (! Low Severity)

"Typographical Error"

```
Line 770: function setListeningFee
Line 2425: function setFeeCalulator
```

SWC Attacks

SWC ID	Description	Status
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Informational
SWC-103	Floating Pragma	! Low
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	! Low
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	! Low
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with the hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	! Informational
SWC-136	Unencrypted Private Data On-Chain	Passed

Risk Status & Radar Chart

Risk Severity	Status
High	No high severity issues identified
Medium	No medium severity issues identified
Low	3 low severity issues identified
Informational	2 informational severity issues identified
Centralization Risk	Active contract ownership identified

Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

* Reflecto V2's smart contract source code has **LOW RISK SEVERITY**

* Reflecto V2's smart contract has an **ACTIVE OWNERSHIP**

* Reflecto V2's centralization risk correlated to the active owner is **MEDIUM**

Note for stakeholders

* Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security.

* If the smart contract is not deployed on any blockchain at the time of the audit, the contract can be modified or altered before blockchain development. Verify contract's deployment status in the audit report.

* Make sure that the project team's KYC/identity is verified by an independent firm.

* Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.

* Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period.

Important Disclaimer

InterFi Network provides contract development, testing, auditing and project evaluation services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purpose without InterFi's prior written consent.**

InterFi provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, 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 the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren't resistant to external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security. Therefore, InterFi does not guarantee the explicit security of the audited smart contract.**

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.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your due diligence and consult your financial advisor before making any investment decisions.

About InterFi Network

InterFi Network provides intelligent blockchain solutions. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **InterFi's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy to use.**

InterFi is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. **InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit **https://interfi.network**

To view our audit portfolio, visit **https://github.com/interfinetwork**

To book an audit, message **https://t.me/interfiaudits**

@INTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 🇨🇦