InterFi
NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT OF

DIBS MONEY

SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | KYC | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

Summary

Auditing Firm	InterFi Network
Client Firm	Dibs Money
Architecture	InterFi “Echelon” Auditing Standard
Language	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Final Report Date	January 20, 2022

Audit Summary

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

- ❖ Dibs Money’s smart contract source codes have **LOW RISK SEVERITY**
- ❖ Dibs Money’s smart contracts have an **ACTIVE OWNERSHIP**
- ❖ DibsMoney’s ownership is set in a time lock with multi-sig as a proposer, please verify:

https://bscscan.com/tx/0x8e031fdc2c26adf9ab1bf7cd2e3cd6d5584374f3536fd05eb99fb79aca41e235

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

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

Table Of Contents

Project Information

InterFi “Echelon” Audit Standard

Smart Contract Risk Assessment

Report Summary

Legal Advisory

Project Overview

InterFi was consulted by Dibs Money to conduct the smart contract security audit of their solidity source codes.

About Dibs Money

DibsMoney is a multi-token DeFi protocol launched in early 2022 on the Binance Smart Chain. $DIBS is an algorithmic stablecoin pegged to the price of BNB. DibsMoney consists of the tokens $DIBS, $DSHARE, and $DBOND. Earn high yields and APR by farming, staking, and compounding! The protocol's mechanism dynamically adjusts $DIBS's supply, pushing its price up or down relative to the price of $BNB.

Project	Dibs Money
Blockchain	Binance Smart Chain
Language	Solidity
Contract	Multiple Files Under Scope
Website	https://dibs.money
Telegram	https://t.me/dibs.money
Twitter	https://twitter.com/DibsMoney
Discord	https://discord.com/invite/f55JMnRgSF

Project Logo

Solidity Source Codes On GitHub

https://github.com/dibs-money/dibs-contracts/tree/main/contracts

Solidity Source Codes Under Scope

- DBond.sol
- Treasury.sol
- NewZap.sol
- Piggybank.sol
- Oracle.sol
- DShareRewardPool.sol
- DibsRewardPool.sol
- Dibs.sol
- DShare.sol

SHA-1 Hash

Code packages are imported and audited at hash #d2e18f8610706d053aba5fcb02136e346fa298f1

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Dibs Money. InterFi has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. 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	❖ Ownership Takeover
	❖ Gas Limit and Loops
	❖ Deployment Consistency
	❖ Repository Consistency
	❖ Data Consistency
	❖ Token Supply Manipulation
Functional Assessment	❖ Access Control and Authorization
	❖ Operations Trail and Event Generation
	❖ Assets Manipulation
	❖ Liquidity Access

InterFi's Echelon Audit Standard

The aim of InterFi's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
    - Review of the specifications, sources, and instructions provided to InterFi to make sure we understand the size, scope, and functionality of the smart contract.
    - Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.
2. Static, Manual, and Software analysis:
    - Test coverage analysis, which 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, which is analysing a program to determine what inputs causes 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

- Slither
- Consensys MythX, Mythril
- SWC Registry
- Solidity Coverage
- Open Zeppelin Code Analyzer
- Solidity Code Complier

InterFi's 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 which requires to own 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
! Critical	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.
! High	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
! Medium	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.
! Low	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

Smart Contract – Static Analysis

Symbol	Meaning
🛑	Function can be modified
💵	Function is payable
🔒	Function is locked
🔐	Function can be accessed
❗	Important functionality

```
| **DBond** | Implementation | ERC20Burnable, Operator |||
| └ | <Constructor> | Public ❗ | 🛑  | ERC20 |
| └ | mint | Public ❗ | 🛑  | onlyOperator |
| └ | burn | Public ❗ | 🛑  |NO❗ |
| └ | burnFrom | Public ❗ | 🛑  | onlyOperator |
||||||
| **Dibs** | Implementation | ERC20Burnable, Operator |||
| └ | <Constructor> | Public ❗ | 🛑  | ERC20 |
| └ | getTaxTiersTwapsCount | Public ❗ |   |NO❗ |
| └ | getTaxTiersRatesCount | Public ❗ |   |NO❗ |
| └ | isAddressExcluded | Public ❗ |   |NO❗ |
| └ | setTaxTiersTwap | Public ❗ | 🛑  | onlyTaxOffice |
| └ | setTaxTiersRate | Public ❗ | 🛑  | onlyTaxOffice |
| └ | setBurnThreshold | Public ❗ | 🛑  | onlyTaxOffice |
| └ | _getDibsPrice | Internal 🔒 |   | |
| └ | _updateTaxRate | Internal 🔒 | 🛑  | |
| └ | enableAutoCalculateTax | Public ❗ | 🛑  | onlyTaxOffice |
| └ | disableAutoCalculateTax | Public ❗ | 🛑  | onlyTaxOffice |
| └ | setDibsOracle | Public ❗ | 🛑  | onlyOperatorOrTaxOffice |
| └ | setTaxOffice | Public ❗ | 🛑  | onlyOperatorOrTaxOffice |
| └ | setTaxCollectorAddress | Public ❗ | 🛑  | onlyTaxOffice |
| └ | setTaxRate | Public ❗ | 🛑  | onlyTaxOffice |
| └ | excludeAddress | Public ❗ | 🛑  | onlyOperatorOrTaxOffice |
| └ | includeAddress | Public ❗ | 🛑  | onlyOperatorOrTaxOffice |
| └ | mint | Public ❗ | 🛑  | onlyOperator |
| └ | burn | Public ❗ | 🛑  |NO❗ |
| └ | burnFrom | Public ❗ | 🛑  | onlyOperator |
| └ | transferFrom | Public ❗ | 🛑  |NO❗ |
| └ | _transferWithTax | Internal 🔒 | 🛑  | |
| └ | distributeReward | External ❗ | 🛑  | onlyOperator |
| └ | governanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
||||||
| **DShare** | Implementation | ERC20Burnable, Operator |||
| └ | <Constructor> | Public ❗️ | 🛑  | ERC20 |
| └ | setTreasuryFund | External ❗️ | 🛑  |NO❗️ |
| └ | setDevFund | External ❗️ | 🛑  |NO❗️ |
| └ | unclaimedTreasuryFund | Public ❗️ |   |NO❗️ |
| └ | unclaimedDevFund | Public ❗️ |   |NO❗️ |
| └ | claimRewards | External ❗️ | 🛑  |NO❗️ |
| └ | distributeReward | External ❗️ | 🛑  | onlyOperator |
| └ | burn | Public ❗️ | 🛑  |NO❗️ |
| └ | governanceRecoverUnsupported | External ❗️ | 🛑  | onlyOperator |
||||||
| **IHyperswapRouter01** | Interface |  |||
| └ | factory | External ❗️ |   |NO❗️ |
| └ | WFTM | External ❗️ |   |NO❗️ |
| └ | addLiquidity | External ❗️ | 🛑  |NO❗️ |
| └ | addLiquidityFTM | External ❗️ |  💵 |NO❗️ |
| └ | removeLiquidity | External ❗️ | 🛑  |NO❗️ |
| └ | removeLiquidityFTM | External ❗️ | 🛑  |NO❗️ |
| └ | removeLiquidityWithPermit | External ❗️ | 🛑  |NO❗️ |
| └ | removeLiquidityFTMWithPermit | External ❗️ | 🛑  |NO❗️ |
| └ | swapExactTokensForTokens | External ❗️ | 🛑  |NO❗️ |
| └ | swapTokensForExactTokens | External ❗️ | 🛑  |NO❗️ |
| └ | swapExactFTMForTokens | External ❗️ |  💵 |NO❗️ |
| └ | swapTokensForExactFTM | External ❗️ | 🛑  |NO❗️ |
| └ | swapExactTokensForFTM | External ❗️ | 🛑  |NO❗️ |
| └ | swapFTMForExactTokens | External ❗️ |  💵 |NO❗️ |
| └ | quote | External ❗️ |   |NO❗️ |
| └ | getAmountOut | External ❗️ |   |NO❗️ |
| └ | getAmountIn | External ❗️ |   |NO❗️ |
| └ | getAmountsOut | External ❗️ |   |NO❗️ |
| └ | getAmountsIn | External ❗️ |   |NO❗️ |
||||||
| **IUniswapV2Pair** | Interface |  |||
| └ | name | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🛑  |NO❗️ |
| └ | transfer | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
| └ | DOMAIN_SEPARATOR | External ❗️ |   |NO❗️ |
| └ | PERMIT_TYPEHASH | External ❗️ |   |NO❗️ |
| └ | nonces | External ❗️ |   |NO❗️ |
| └ | permit | External ❗️ | 🛑  |NO❗️ |
| └ | MINIMUM_LIQUIDITY | External ❗️ |   |NO❗️ |
| └ | factory | External ❗️ |   |NO❗️ |
| └ | token0 | External ❗️ |   |NO❗️ |
| └ | token1 | External ❗ |   |NO❗ |
| └ | getReserves | External ❗ |   |NO❗ |
| └ | price0CumulativeLast | External ❗ |   |NO❗ |
| └ | price1CumulativeLast | External ❗ |   |NO❗ |
| └ | kLast | External ❗ |   |NO❗ |
| └ | mint | External ❗ | 🛑  |NO❗ |
| └ | burn | External ❗ | 🛑  |NO❗ |
| └ | swap | External ❗ | 🛑  |NO❗ |
| └ | skim | External ❗ | 🛑  |NO❗ |
| └ | sync | External ❗ | 🛑  |NO❗ |
| └ | initialize | External ❗ | 🛑  |NO❗ |
||||||
| **IUniswapV2Router01** | Interface |  |||
| └ | factory | External ❗ |   |NO❗ |
| └ | WETH | External ❗ |   |NO❗ |
| └ | addLiquidity | External ❗ | 🛑  |NO❗ |
| └ | addLiquidityETH | External ❗ |  💵 |NO❗ |
| └ | removeLiquidity | External ❗ | 🛑  |NO❗ |
| └ | removeLiquidityETH | External ❗ | 🛑  |NO❗ |
| └ | removeLiquidityWithPermit | External ❗ | 🛑  |NO❗ |
| └ | removeLiquidityETHWithPermit | External ❗ | 🛑  |NO❗ |
| └ | swapExactTokensForTokens | External ❗ | 🛑  |NO❗ |
| └ | swapTokensForExactTokens | External ❗ | 🛑  |NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🛑  |NO❗ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🛑  |NO❗ |
| └ | swapExactETHForTokens | External ❗ |  💵 |NO❗ |
| └ | swapTokensForExactETH | External ❗ | 🛑  |NO❗ |
| └ | swapExactTokensForETH | External ❗ | 🛑  |NO❗ |
| └ | swapETHForExactTokens | External ❗ |  💵 |NO❗ |
| └ | quote | External ❗ |   |NO❗ |
| └ | getAmountOut | External ❗ |   |NO❗ |
| └ | getAmountIn | External ❗ |   |NO❗ |
| └ | getAmountsOut | External ❗ |   |NO❗ |
| └ | getAmountsIn | External ❗ |   |NO❗ |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **IVault** | Interface | IERC20 |||
| └ | deposit | External ❗ | 🛑  |NO❗ |
| └ | withdraw | External ❗ | 🛑  |NO❗ |
| └ | want | External ❗ |   |NO❗ |
||||||
| **Address** | Library |  |||
| └ | isContract | Internal 🔒 |   | |
| └ | sendValue | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionStaticCall | Internal 🔒 |   | |
| └ | functionStaticCall | Internal 🔒 |   | |
| └ | functionDelegateCall | Internal 🔒 | 🛑  | |
| └ | functionDelegateCall | Internal 🔒 | 🛑  | |
| └ | _verifyCallResult | Private 🔐 |   | |
||||||
| **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 🔒 |   | |
||||||
| **TransferHelper** | Library |  |||
| └ | safeApprove | Internal 🔒 | 🛑  | |
| └ | safeTransfer | Internal 🔒 | 🛑  | |
| └ | safeTransferFrom | Internal 🔒 | 🛑  | |
| └ | safeTransferETH | Internal 🔒 | 🛑  | |
||||||
| **SafeERC20** | Library |  |||
| └ | safeTransfer | Internal 🔒 | 🛑  | |
| └ | safeTransferFrom | Internal 🔒 | 🛑  | |
| └ | safeApprove | Internal 🔒 | 🛑  | |
| └ | safeIncreaseAllowance | Internal 🔒 | 🛑  | |
| └ | safeDecreaseAllowance | Internal 🔒 | 🛑  | |
| └ | _callOptionalReturn | Private 🔐 | 🛑  | |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | owner | Public ❗️ |   |NO❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑  | onlyOwner |
||||||
| **IZap** | Interface |  |||
| └ | estimateZapInToken | External ❗️ |   |NO❗️ |
| └ | swapToken | External ❗️ | 🛑  |NO❗️ |
| └ | swapToNative | External ❗️ | 🛑  |NO❗️ |
| └ | zapIn | External ❗️ |  💵 |NO❗️ |
| └ | zapInToken | External ❗️ | 🛑  |NO❗️ |
| └ | zapAcross | External ❗️ | 🛑  |NO❗️ |
| └ | zapOut | External ❗️ | 🛑  |NO❗️ |
| └ | zapOutToken | External ❗️ | 🛑  |NO❗️ |
||||||
| **Zap** | Implementation | Ownable, IZap |||
| └ | <Constructor> | Public ❗️ | 🛑  | Ownable |
| └ | <Receive Ether> | External ❗️ |  💵 |NO❗️ |
| └ | zapInToken | External ❗️ | 🛑  |NO❗️ |
| └ | estimateZapInToken | Public ❗️ |   |NO❗️ |
| └ | zapIn | External ❗️ |  💵 |NO❗️ |
| └ | zapAcross | External ❗️ | 🛑  |NO❗️ |
| └ | zapOut | External ❗️ | 🛑  |NO❗️ |
| └ | zapOutToken | External ❗️ | 🛑  |NO❗️ |
| └ | swapToken | External ❗️ | 🛑  |NO❗️ |
| └ | swapToNative | External ❗️ | 🛑  |NO❗️ |
| └ | _approveTokenIfNeeded | Private 🔐 | 🛑  | |
| └ | _swapTokenToLP | Private 🔐 | 🛑  | |
| └ | _swapNativeToLP | Private 🔐 | 🛑  | |
| └ | _swapHalfNativeAndProvide | Private 🔐 | 🛑  | |
| └ | _swapNativeToEqualTokensAndProvide | Private 🔐 | 🛑  | |
| └ | _swapNativeForToken | Private 🔐 | 🛑  | |
| └ | _swapTokenForNative | Private 🔐 | 🛑  | |
| └ | _swap | Private 🔐 | 🛑  | |
| └ | _estimateSwap | Private 🔐 |   | |
| └ | setTokenBridgeForRouter | External ❗️ | 🛑  | onlyOwner |
| └ | withdraw | External ❗️ | 🛑  | onlyOwner |
| └ | setUseNativeRouter | External ❗️ | 🛑  | onlyOwner |
| └ | setIsFeeOnTransfer | External ❗️ | 🛑  | onlyOwner |
||||||
| **Treasury** | Implementation | ContractGuard |||
| └ | isInitialized | Public ❗️ |   |NO❗️ |
| └ | nextEpochPoint | Public ❗️ |   |NO❗️ |
| └ | getDibsPrice | Public ❗️ |   |NO❗️ |
| └ | getDibsUpdatedPrice | Public ❗️ |   |NO❗️ |
| └ | getReserve | Public ❗️ |   |NO❗️ |
| └ | getBurnableDibsLeft | Public ❗️ |   |NO❗️ |
| └ | getRedeemableBonds | Public ❗️ |   |NO❗️ |
| └ | getBondDiscountRate | Public ❗️ |   |NO❗️ |
| └ | getBondPremiumRate | Public ❗️ |   |NO❗️ |
| └ | initialize | Public ❗️ | 🛑  | notInitialized |
| └ | setOperator | External ❗️ | 🛑  | onlyOperator |
| └ | setPiggybank | External ❗️ | 🛑  | onlyOperator |
| └ | setDibsOracle | External ❗️ | 🛑  | onlyOperator |
| └ | setDibsPriceCeiling | External ❗️ | 🛑  | onlyOperator |
| └ | setMaxSupplyExpansionPercents | External ❗ | 🛑  | onlyOperator |
| └ | setSupplyTiersEntry | External ❗ | 🛑  | onlyOperator |
| └ | setMaxExpansionTiersEntry | External ❗ | 🛑  | onlyOperator |
| └ | setBondDepletionFloorPercent | External ❗ | 🛑  | onlyOperator |
| └ | setMaxSupplyContractionPercent | External ❗ | 🛑  | onlyOperator |
| └ | setMaxDebtRatioPercent | External ❗ | 🛑  | onlyOperator |
| └ | setBootstrap | External ❗ | 🛑  | onlyOperator |
| └ | setExtraFunds | External ❗ | 🛑  | onlyOperator |
| └ | setMaxDiscountRate | External ❗ | 🛑  | onlyOperator |
| └ | setMaxPremiumRate | External ❗ | 🛑  | onlyOperator |
| └ | setDiscountPercent | External ❗ | 🛑  | onlyOperator |
| └ | setPremiumThreshold | External ❗ | 🛑  | onlyOperator |
| └ | setPremiumPercent | External ❗ | 🛑  | onlyOperator |
| └ | setMintingFactorForPayingDebt | External ❗ | 🛑  | onlyOperator |
| └ | _updateDibsPrice | Internal 🔒 | 🛑  | |
| └ | getDibsCirculatingSupply | Public ❗ |   |NO❗ |
| └ | buyBonds | External ❗ | 🛑  | onlyOneBlock checkCondition checkOperator |
| └ | redeemBonds | External ❗ | 🛑  | onlyOneBlock checkCondition checkOperator |
| └ | _sendToPiggybank | Internal 🔒 | 🛑  | |
| └ | _calculateMaxSupplyExpansionPercent | Internal 🔒 | 🛑  | |
| └ | allocateSeigniorage | External ❗ | 🛑  | onlyOneBlock checkCondition checkEpoch checkOperator
|
| └ | governanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
| └ | piggybankSetOperator | External ❗ | 🛑  | onlyOperator |
| └ | piggybankSetLockUp | External ❗ | 🛑  | onlyOperator |
| └ | piggybankAllocateSeigniorage | External ❗ | 🛑  | onlyOperator |
| └ | piggybankGovernanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
||||||
| **ShareWrapper** | Implementation |  |||
| └ | totalSupply | Public ❗ |   |NO❗ |
| └ | balanceOf | Public ❗ |   |NO❗ |
| └ | stake | Public ❗ | 🛑  |NO❗ |
| └ | withdraw | Public ❗ | 🛑  |NO❗ |
||||||
| **Piggybank** | Implementation | ShareWrapper, ContractGuard |||
| └ | initialize | Public ❗ | 🛑  | notInitialized |
| └ | setOperator | External ❗ | 🛑  | onlyOperator |
| └ | setLockUp | External ❗ | 🛑  | onlyOperator |
| └ | latestSnapshotIndex | Public ❗ |   |NO❗ |
| └ | getLatestSnapshot | Internal 🔒 |   | |
| └ | getLastSnapshotIndexOf | Public ❗ |   |NO❗ |
| └ | getLastSnapshotOf | Internal 🔒 |   | |
| └ | canWithdraw | External ❗ |   |NO❗ |
| └ | canClaimReward | External ❗ |   |NO❗ |
| └ | epoch | External ❗ |   |NO❗ |
| └ | nextEpochPoint | External ❗ |   |NO❗ |
| └ | getDibsPrice | External ❗ |   |NO❗ |
| └ | rewardPerShare | Public ❗ |   |NO❗ |
| └ | earned | Public ❗ |   |NO❗ |
| └ | stake | Public ❗ | 🛑  | onlyOneBlock updateReward |
| └ | withdraw | Public ❗ | 🛑  | onlyOneBlock memberExists updateReward |
| └ | exit | External ❗ | 🛑  |NO❗ |
| └ | claimReward | Public ❗ | 🛑  | updateReward |
| └ | allocateSeigniorage | External ❗ | 🛑  | onlyOneBlock onlyOperator |
| └ | governanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
||||||
| **DShare** | Implementation | ERC20Burnable, Operator |||
| └ | <Constructor> | Public ❗ | 🛑  | ERC20 |
| └ | setTreasuryFund | External ❗ | 🛑  |NO❗ |
| └ | setDevFund | External ❗ | 🛑  |NO❗ |
| └ | unclaimedTreasuryFund | Public ❗ |   |NO❗ |
| └ | unclaimedDevFund | Public ❗ |   |NO❗ |
| └ | claimRewards | External ❗ | 🛑  |NO❗ |
| └ | distributeReward | External ❗ | 🛑  | onlyOperator |
| └ | burn | Public ❗ | 🛑  |NO❗ |
| └ | governanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
||||||
| **DShareRewardPool** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | checkPoolDuplicate | Internal 🔒 |   | |
| └ | add | Public ❗ | 🛑  | onlyOperator |
| └ | set | Public ❗ | 🛑  | onlyOperator |
| └ | getGeneratedReward | Public ❗ |   |NO❗ |
| └ | pendingShare | External ❗ |   |NO❗ |
| └ | massUpdatePools | Public ❗ | 🛑  |NO❗ |
| └ | updatePool | Public ❗ | 🛑  |NO❗ |
| └ | deposit | Public ❗ | 🛑  |NO❗ |
| └ | withdraw | Public ❗ | 🛑  |NO❗ |
| └ | emergencyWithdraw | Public ❗ | 🛑  |NO❗ |
| └ | safeDShareTransfer | Internal 🔒 | 🛑  | |
| └ | setOperator | External ❗ | 🛑  | onlyOperator |
| └ | governanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
||||||
| **DibsRewardPool** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | checkPoolDuplicate | Internal 🔒 |   | |
| └ | add | Public ❗ | 🛑  | onlyOperator |
| └ | set | Public ❗ | 🛑  | onlyOperator |
| └ | getGeneratedReward | Public ❗ |   |NO❗ |
| └ | pendingDIBS | External ❗ |   |NO❗ |
| └ | massUpdatePools | Public ❗ | 🛑  |NO❗ |
| └ | updatePool | Public ❗ | 🛑  |NO❗ |
| └ | deposit | Public ❗ | 🛑  |NO❗ |
| └ | withdraw | Public ❗ | 🛑  |NO❗ |
| └ | emergencyWithdraw | Public ❗ | 🛑  |NO❗ |
| └ | safeDibsTransfer | Internal 🔒 | 🛑  | |
| └ | setOperator | External ❗ | 🛑  | onlyOperator |
| └ | governanceRecoverUnsupported | External ❗ | 🛑  | onlyOperator |
```

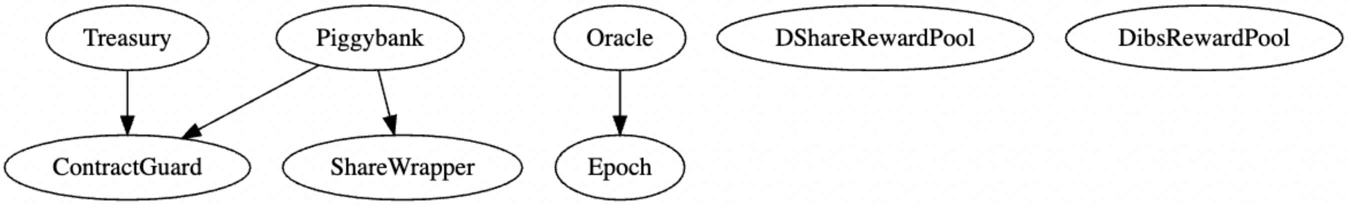
Smart Contract – Software Analysis

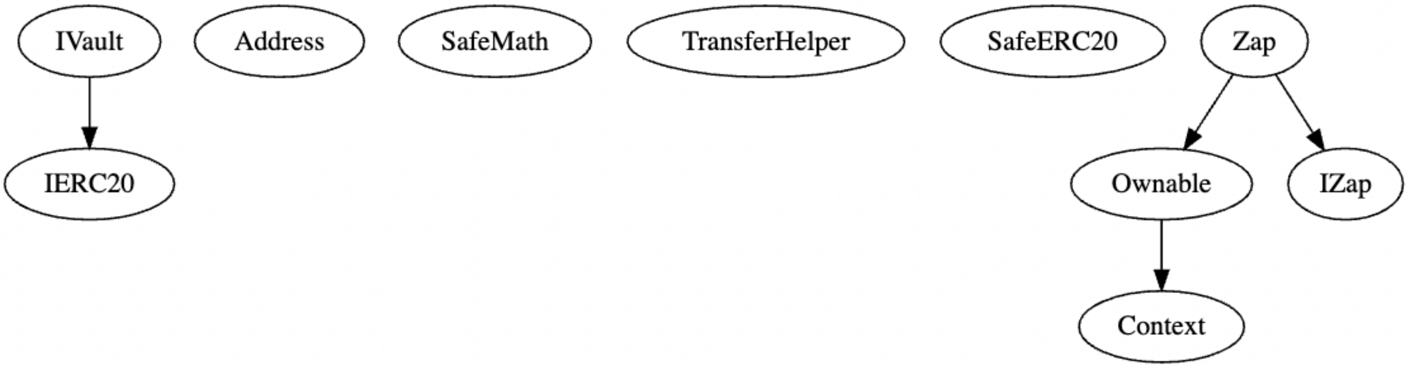
```
16279055  =>  isContract(address)
17764782  =>  unclaimedTreasuryFund()
40c10f19  =>  mint(address,uint256)
42966c68  =>  burn(uint256)
79cc6790  =>  burnFrom(address,uint256)
a6431bba  =>  getTaxTiersTwapsCount()
ee2a9535  =>  getTaxTiersRatesCount()
ebca1bd9  =>  isAddressExcluded(address)
66206ce9  =>  setTaxTiersTwap(uint8,uint256)
b87c5a4a  =>  setTaxTiersRate(uint8,uint256)
9d6b5f21  =>  setBurnThreshold(uint256)
92cfc08e  =>  _getDibsPrice()
9f10267a  =>  _updateTaxRate(uint256)
ff87fc7c  =>  enableAutoCalculateTax()
65bbacd9  =>  disableAutoCalculateTax()
617807b5  =>  setDibsOracle(address)
3f07d76a  =>  setTaxOffice(address)
69356d47  =>  setTaxCollectorAddress(address)
c6d69a30  =>  setTaxRate(uint256)
3758e6ce  =>  excludeAddress(address)
93995d4b  =>  includeAddress(address)
23b872dd  =>  transferFrom(address,address,uint256)
030bed3a  =>  _transferWithTax(address,address,uint256,bool)
8f460a06  =>  distributeReward(address[],address)
7171b047  =>  governanceRecoverUnsupported(IERC20,uint256,address)
f746b718  =>  setTreasuryFund(address)
ae4db919  =>  setDevFund(address)
2c07a624  =>  unclaimedDevFund()
372500ab  =>  claimRewards()
092193ab  =>  distributeReward(address)
c45a0155  =>  factory()
21dbe876  =>  WFTM()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
d427d12c  =>  addLiquidityFTM(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>  removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
900f301e  =>  removeLiquidityFTM(address,uint256,uint256,uint256,address,uint256)
2195995c  =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3
2,bytes32)
e5373ab0  =>
removeLiquidityFTMWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt
es32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
1550f836  =>  swapExactFTMForTokens(uint256,address[],address,uint256)
d6726025  =>  swapTokensForExactFTM(uint256,uint256,address[],address,uint256)
e932f3ac  =>  swapExactTokensForFTM(uint256,uint256,address[],address,uint256)
```

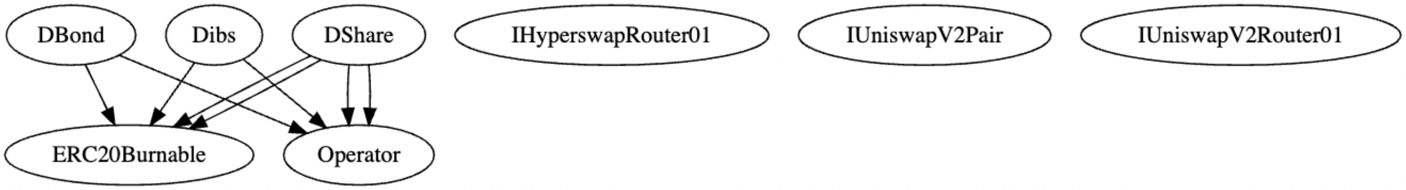
```
5cc59caf  =>  swapFTMForExactTokens(uint256,address[],address,uint256)
ad615dec  =>  quote(uint256,uint256,uint256)
054d50d4  =>  getAmountOut(uint256,uint256,uint256)
85f8c259  =>  getAmountIn(uint256,uint256,uint256)
d06ca61f  =>  getAmountsOut(uint256,address[])
1f00ca74  =>  getAmountsIn(uint256,address[])
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
a9059cbb  =>  transfer(address,uint256)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
6a627842  =>  mint(address)
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
ad5c4648  =>  WETH()
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
02751cec  =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
ded9382a  =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt
es32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
7ff36ab5  =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a  =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5  =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41  =>  swapETHForExactTokens(uint256,address[],address,uint256)
b6b55f25  =>  deposit(uint256)
2e1a7d4d  =>  withdraw(uint256)
1f1fcd51  =>  want()
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)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
18c2c6a2  =>  _verifyCallResult(bool,bytes,string)
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)
eb5625d9  =>  safeApprove(address,address,uint256)
d1660f99  =>  safeTransfer(address,address,uint256)
d9fc4b61  =>  safeTransferFrom(address,address,address,uint256)
7c4368c1  =>  safeTransferETH(address,uint256)
d0c407e1  =>  safeTransfer(IERC20,address,uint256)
5beae096  =>  safeTransferFrom(IERC20,address,address,uint256)
d6dcec8d  =>  safeApprove(IERC20,address,uint256)
390cc046  =>  safeIncreaseAllowance(IERC20,address,uint256)
5164ffed  =>  safeDecreaseAllowance(IERC20,address,uint256)
becc5a20  =>  _callOptionalReturn(IERC20,bytes)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
```

Inheritance Graph





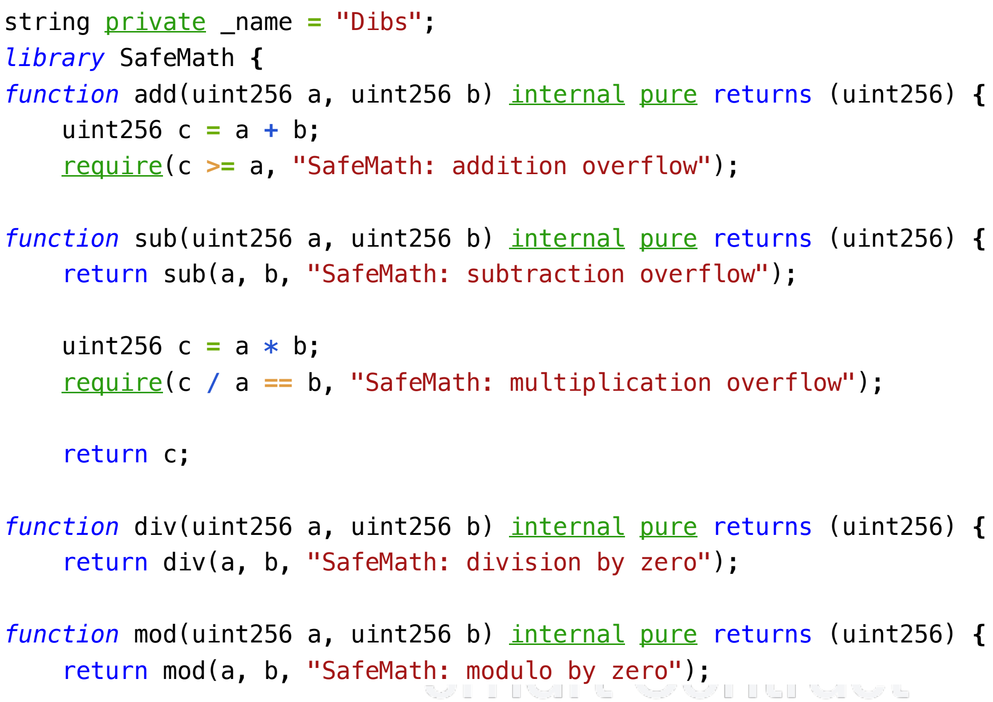


Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
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
Burn	executes transfers of a specified number of tokens to a burn address	Yes	Passed
Mint	executes creation of a specified number of tokens and adds it to the total supply	Yes	! Low
Lock	stops or locks all function modules of the smart contract	Yes	! Low
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

Best Practices ✅

- Owner cannot lock or burn the user assets.
- The smart contract utilizes “SafeMath” function to avoid common smart contract vulnerabilities.



```
string private _name = "Dibs";
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");
```

Note ⚠️

- DibsMoney’s ownership is set in a time lock with multi-sig as a proposer, please verify:

https://bscscan.com/tx/0x8e031fdc2c26adf9ab1bf7cd2e3cd6d5584374f3536fd05eb99fb79aca41e235

- Be aware that active smart contract owner privileges constitute an elevated impact to smart contract’s safety and security.
- Operator can utilize active **minting**. Community consensus is greatly recommended to minimize the centralization risk.
- Dibs Money Ecosystem ***does not utilize “ReentrancyGuard”*** to prevent reentrant calls to a function. Reentrancy Guard is a contract module that helps prevent reentrant calls to a

function. Inheriting from Reentrancy Guard will make the nonReentrant modifier available, which can be applied to functions to make sure there are no nested (reentrant) calls to them.

```
IMPORTANT: because control is transferred to `recipient`, care must be taken to not create
reentrancy vulnerabilities. Consider using {ReentrancyGuard}

https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/security/ReentrancyGuard.sol

or the

https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-
interactions-pattern
```

- The smart contract has ***low severity issue*** which may or may not create any functional vulnerability.

{

"resource": " /Dibs.sol",

"owner": "_generated_diagnostic_collection_name_#0",

"severity": 8, (! Low Severity)

"Expected identifier, got 'LParen",

"source": "solc",

}

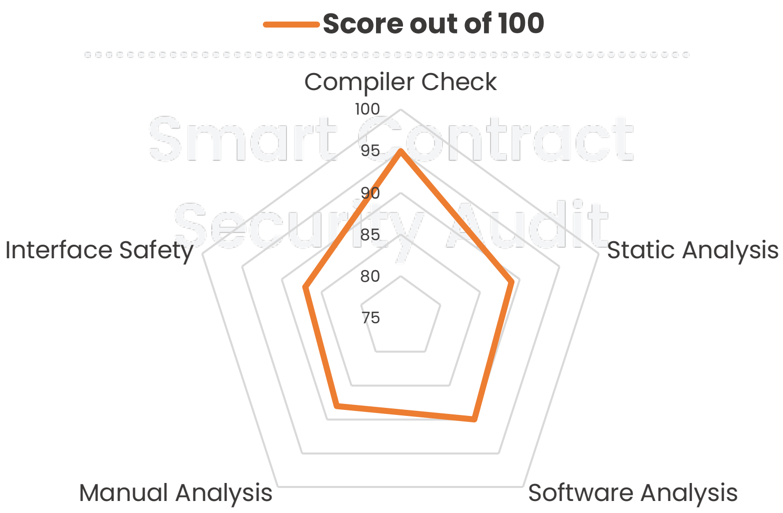
Smart Contract – SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**! Low**
SWC-103	Floating Pragma	**Passed**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELFDESTRUCT 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	**Passed**
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 hardcoded gas amount	**Passed**
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Passed**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	3 low severity issues identified
Verified	54 functions and instances verified and checked



Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

- Dibs Money's smart contract source code has **LOW RISK SEVERITY**
- Dibs Money's smart contract has an **ACTIVE OWNERSHIP**

Note for stakeholders

- Make sure that the project team's KYC/identity is verified by an independent firm, e.g., InterFi.
- Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in 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 of time.
- Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Important Disclaimer

InterFi Network provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes 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 an 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 from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's 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 on 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 own 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 🇨🇦