SMART CONTRACT SECURITY AUDIT OF

MOCEANS

(MINER CONTRACT OF OCEANS FINANCE)

Audit Introduction

Auditing Firm	InterFi Network
Audit Architecture	InterFi Echelon Auditing Standard
Language	Solidity
Client Firm	Oceans Finance
Website	https://oceans.finance/
Telegram	https://t.me/finance_oceans/
Twitter	https://twitter.com/oceans_finance/
Discord	https://discord.gg/m8An2vHZ6A/
Medium	https://oceansfi.medium.com/
Report Date	May 18, 2022

About MOCEANS

$MOCEANS Token Is WORLD'S FIRST Auto Compounding MINER with an APR Up to 13% DAILY! This Miner will Work 24/7 365! This Miner will be the FIRST To Launch on any Blockchain.

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:

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

- ❖ MOCEANS's smart contract has an **ACTIVE OWNERSHIP**

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

- ❖ Important owner privileges – **BOT BLACKLIST, SET SELL TX FEE, SET BUY AND SELL FEES, SET TWENTY-FOUR HOURS**

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: **0x6c4D9bB6Be7dbB88cE3e0e7b6848E306e03fEC55**

⚙ 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 Oceans Finance to conduct the smart contract security audit of MOCEANS source code. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

❖ MOCEANS.sol

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x6c4d9bb6be7dbb88ce3e0e7b6848e306e03fec55#code

Contract Name: MOCEANS

Compiler Version: v0.8.7

Optimization Enabled: Yes with 5000 runs

Solidity Source Code On InterFi GitHub

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

Payment Verification

Network: Binance Smart Chain

Hash: 0xd4295e0a7434a09dd7cf5fbff15c0adfc3137ff883ca4a028d89aebd78ff6116

SHA-1 Hash

Solidity source code is audited at hash #27f9a5ba8213eba9bc51208e33d92b96f90b45d6

Audit Methodology

The scope of this report is to audit the smart contract source code of MOCEANS. InterFi has scanned contracts and reviewed codes for common vulnerabilities, exploits, hacks, and back-doors. 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

- ❖ Gas Limit and Loops
- ❖ Deployment Consistency
- ❖ Repository Consistency
- ❖ Data Consistency
- ❖ Token Supply Manipulation

Source Code Review

- ❖ 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.

MOCEANS's Centralization Status

- ❖ MOCEANS's smart contract has an **active ownership.**
- ❖ Smart contract ownership is set to **0x16a12c6485459539b58bf7e52acbdf63f951e7cb** at the time of the audit.

Static Analysis

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

```
| **IPinkAntiBot** | Interface |   |||
| └ | setTokenOwner | External ❗ | 🛑  |NO❗ |
| └ | onPreTransferCheck | External ❗ | 🛑  |NO❗ |
||||||
| **Context** | Implementation |   |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **IERC20** | Interface |   |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
||||||
| **SafeMath** | Library |   |||
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
```

```
| └ | sub | Internal 🔒 |    | |
| └ | mul | Internal 🔒 |    | |
| └ | div | Internal 🔒 |    | |
| └ | div | Internal 🔒 |    | |
| └ | mod | Internal 🔒 |    | |
| └ | mod | Internal 🔒 |    | |
||||||
| **SafeMathInt** | Library |  |||
| └ | mul | Internal 🔒 |    | |
| └ | div | Internal 🔒 |    | |
| └ | sub | Internal 🔒 |    | |
| └ | add | Internal 🔒 |    | |
| └ | abs | Internal 🔒 |    | |
| └ | toUint256Safe | Internal 🔒 |    | |
||||||
| **SafeMathUint** | Library |  |||
| └ | toInt256Safe | Internal 🔒 |    | |
||||||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | name | Public ❗ |    |NO❗ |
| └ | symbol | Public ❗ |    |NO❗ |
| └ | decimals | Public ❗ |    |NO❗ |
| └ | totalSupply | Public ❗ |    |NO❗ |
| └ | balanceOf | Public ❗ |    |NO❗ |
| └ | transfer | Public ❗ | 🔴  |NO❗ |
| └ | allowance | Public ❗ |    |NO❗ |
| └ | approve | Public ❗ | 🔴  |NO❗ |
| └ | transferFrom | Public ❗ | 🔴  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🔴  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🔴  |NO❗ |
| └ | _transfer | Internal 🔒 | 🔴  | |
| └ | _cast | Internal 🔒 | 🔴  | |
| └ | _burn | Internal 🔒 | 🔴  | |
| └ | _approve | Internal 🔒 | 🔴  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🔴  | |
||||||
| **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❗ |
| └ | 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❗ |
||||||
| **IUniswapV2Router02** | Interface | IUniswapV2Router01 |||
|  └  | removeLiquidityETHSupportingFeeOnTransferTokens | External ❗ | 🔴   |NO❗ |
|  └  | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ❗ | 🔴  |NO❗ |
|  └  | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🔴   |NO❗ |
|  └  | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 |NO❗ |
|  └  | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🔴   |NO❗ |
||||||
| **IUniswapV2Factory** | Interface |  |||
|  └  | feeTo | External ❗ |   |NO❗ |
|  └  | feeToSetter | External ❗ |   |NO❗ |
|  └  | getPair | External ❗ |   |NO❗ |
|  └  | allPairs | External ❗ |   |NO❗ |
|  └  | allPairsLength | External ❗ |   |NO❗ |
|  └  | createPair | External ❗ | 🔴  |NO❗ |
|  └  | setFeeTo | External ❗ | 🔴  |NO❗ |
|  └  | setFeeToSetter | 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❗ |
|  └  | burn | External ❗ | 🔴  |NO❗ |
|  └  | swap | External ❗ | 🔴  |NO❗ |
|  └  | skim | External ❗ | 🔴  |NO❗ |
```

└	sync	External ❗	🔴	NO❗
└	initialize	External ❗	🔴	NO❗
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗		NO❗
└	withdrawDividend	External ❗	🔴	NO❗
DividendPayingTokenOptionalInterface	Interface			
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
DividendPayingToken	Implementation	ERC20, Ownable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
└	<Constructor>	Public ❗	🔴	ERC20
└	distributeReflectionDividends	Public ❗	🔴	onlyOwner
└	withdrawDividend	Public ❗	🔴	NO❗
└	_withdrawDividendOfUser	Internal 🔒	🔴	
└	dividendOf	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	withdrawnDividendOf	Public ❗		NO❗
└	accumulativeDividendOf	Public ❗		NO❗
└	_transfer	Internal 🔒	🔴	
└	_cast	Internal 🔒	🔴	
└	_burn	Internal 🔒	🔴	
└	_setBalance	Internal 🔒	🔴	
MoceansDividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🔴	DividendPayingToken
└	_transfer	Internal 🔒		
└	withdrawDividend	Public ❗		NO❗
└	setMinimumTokenBalanceForDividends	External ❗	🔴	onlyOwner
└	excludeFromDividends	External ❗	🔴	onlyOwner
└	updateClaimWait	External ❗	🔴	onlyOwner
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfTokenHolders	External ❗		NO❗
└	isExcludedFromDividends	Public ❗		NO❗
└	getAccount	Public ❗		NO❗
└	getAccountAtIndex	Public ❗		NO❗
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🔴	onlyOwner
└	process	Public ❗	🔴	NO❗
└	processAccount	Public ❗	🔴	onlyOwner
└	MAPGet	Public ❗		NO❗
└	MAPGetIndexOfKey	Public ❗		NO❗
└	MAPGetKeyAtIndex	Public ❗		NO❗
└	MAPSize	Public ❗		NO❗
└	MAPSet	Public ❗	🔴	NO❗
└	MAPRemove	Public ❗	🔴	NO❗

MOCEANS		Implementation		ERC20, Ownable		
L		<Constructor>	Public ❗	🛑	ERC20
L		<Receive Ether>	External ❗	💵	NO❗
L		updateMinimumTokenBalanceForDividends	Public ❗	🛑	onlyOwner
L		updateUniswapV2Router	Public ❗	🛑	onlyOwner
L		excludeFromFees	Public ❗	🛑	onlyOwner
L		excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
L		setMarketingWallet	External ❗	🛑	onlyOwner
L		setGasFeeCollector	External ❗	🛑	onlyOwner
L		setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
L		setBotBlacklist	External ❗	🛑	onlyOwner
L		isContract	Internal 🔒		
L		_setAutomatedMarketMakerPair	Private 🔑	🛑	
L		updateGasForProcessing	Public ❗	🛑	onlyOwner
L		updateClaimWait	External ❗	🛑	onlyOwner
L		excludeFromDividends	External ❗	🛑	onlyOwner
L		processDividendTracker	External ❗	🛑	NO❗
L		setSwapTokensAtAmount	Public ❗	🛑	onlyOwner
L		setIsSellLimitExempt	Public ❗	🛑	onlyOwner
L		setSellTxFee	External ❗	🛑	onlyOwner
L		setTwentyFourhours	External ❗	🛑	onlyOwner
L		enableDisableWhale	External ❗	🛑	onlyOwner
L		clearStuckBalance	External ❗	🛑	onlyOwner
L		rescueToken	External ❗	🛑	onlyOwner
L		setEnableAntiBot	External ❗	🛑	onlyOwner
L		setBuyTaxes	External ❗	🛑	onlyOwner
L		setSellTaxes	External ❗	🛑	onlyOwner
L		getClaimWait	External ❗		NO❗
L		getTotalDividendsDistributed	External ❗		NO❗
L		isExcludedFromFees	Public ❗		NO❗
L		withdrawableDividendOf	Public ❗		NO❗
L		dividendTokenBalanceOf	Public ❗		NO❗
L		isExcludedFromDividends	Public ❗		NO❗
L		getAccountDividendsInfo	External ❗		NO❗
L		getAccountDividendsInfoAtIndex	External ❗		NO❗
L		claim	External ❗	🛑	NO❗
L		getLastProcessedIndex	External ❗		NO❗
L		getNumberOfDividendTokenHolders	External ❗		NO❗
L		swapManual	Public ❗	🛑	onlyOwner
L		_transfer	Internal 🔒	🛑	
L		ActivateAntiWhale	Private 🔑	🛑	
L		getCirculatingSupply	Public ❗		NO❗
L		swapAndGasFee	Private 🔑	🛑	
L		swapAndSendToFee	Private 🔑	🛑	
L		swapAndLiquify	Private 🔑	🛑	
L		swapTokensForEth	Private 🔑	🛑	
L		swapTokensForReflection	Private 🔑	🛑	
L		addLiquidity	Private 🔑	🛑	
L		swapAndSendDividends	Private 🔑	🛑	

Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
43509138  =>  div(int256,int256)
48760858  =>  onPreTransferCheck(address,address,uint256)
18e02bd9  =>  setTokenOwner(address)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
bbe93d91  =>  mul(int256,int256)
adefc37b  =>  sub(int256,int256)
a5f3c23b  =>  add(int256,int256)
1b5ac4b5  =>  abs(int256)
744f7c7d  =>  toUint256Safe(int256)
e823b9bf  =>  toInt256Safe(uint256)
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
3fff10b9  =>  _cast(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>  removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
```

```
02751cec => removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3
2,bytes32)
ded9382a =>
removeLiquidityETHWithPermit(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)
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)
ad615dec => quote(uint256,uint256,uint256)
054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
af2979eb =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
5c11d795 =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
017e7e58 => feeTo()
094b7415 => feeToSetter()
e6a43905 => getPair(address,address)
1e3dd18b => allPairs(uint256)
574f2ba3 => allPairsLength()
c9c65396 => createPair(address,address)
f46901ed => setFeeTo(address)
a2e74af6 => setFeeToSetter(address)
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()
89afcb44 => burn(address)
022c0d9f => swap(uint256,uint256,address,bytes)
bc25cf77 => skim(address)
```

```
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
91b89fba  =>  dividendOf(address)
6a474002  =>  withdrawDividend()
a8b9d240  =>  withdrawableDividendOf(address)
aafd847a  =>  withdrawnDividendOf(address)
27ce0147  =>  accumulativeDividendOf(address)
47977cd9  =>  distributeReflectionDividends(uint256)
373de4aa  =>  _withdrawDividendOfUser(address)
ab86e0a6  =>  _setBalance(address,uint256)
5ebf4db9  =>  setMinimumTokenBalanceForDividends(uint256)
0483f7a0  =>  excludeFromDividends(address,bool)
e98030c7  =>  updateClaimWait(uint256)
e7841ec0  =>  getLastProcessedIndex()
09bbedde  =>  getNumberOfTokenHolders()
c705c569  =>  isExcludedFromDividends(address)
fbcbc0f1  =>  getAccount(address)
5183d6fd  =>  getAccountAtIndex(uint256)
77fdb837  =>  canAutoClaim(uint256)
e30443bc  =>  setBalance(address,uint256)
ffb2c479  =>  process(uint256)
bc4c4b37  =>  processAccount(address,bool)
556cafb0  =>  MAPGet(address)
a3395cb4  =>  MAPGetIndexOfKey(address)
10a8c46b  =>  MAPGetKeyAtIndex(uint256)
b1181e55  =>  MAPSize()
1105e740  =>  MAPSet(address,uint256)
0716c611  =>  MAPRemove(address)
0dcb2e89  =>  updateMinimumTokenBalanceForDividends(uint256)
65b8dbc0  =>  updateUniswapV2Router(address)
c0246668  =>  excludeFromFees(address,bool)
c492f046  =>  excludeMultipleAccountsFromFees(address[],bool)
5d098b38  =>  setMarketingWallet(address)
393569f6  =>  setGasFeeCollector(address)
9a7a23d6  =>  setAutomatedMarketMakerPair(address,bool)
37c9be87  =>  setBotBlacklist(address,bool)
a7f7b36f  =>  _setAutomatedMarketMakerPair(address,bool)
871c128d  =>  updateGasForProcessing(uint256)
700bb191  =>  processDividendTracker(uint256)
afa4f3b2  =>  setSwapTokensAtAmount(uint256)
b44a1a8f  =>  setIsSellLimitExempt(address,bool)
86700cb4  =>  setSellLimitByAmount(uint256)
6d3da479  =>  setSellLimit(uint256,uint256)
1f46b1c6  =>  setEnableAntiBot(bool)
e4bf1bed  =>  setBuyTaxes(uint256,uint256,uint256,uint256)
f480fec2  =>  setSellTaxes(uint256,uint256,uint256,uint256)
a26579ad  =>  getClaimWait()
30bb4cff  =>  getTotalDividendsDistributed()
4fbee193  =>  isExcludedFromFees(address)
6843cd84  =>  dividendTokenBalanceOf(address)
```

```
ad56c13c  =>  getAccountDividendsInfo(address)
f27fd254  =>  getAccountDividendsInfoAtIndex(uint256)
4e71d92d  =>  claim()
64b0f653  =>  getNumberOfDividendTokenHolders()
d2a8b440  =>  swapManual()
2b112e49  =>  getCirculatingSupply()
234fb5a3  =>  swapAndGasFee(uint256)
a210621e  =>  swapAndSendToFee(uint256)
173865ad  =>  swapAndLiquify(uint256)
b28805f4  =>  swapTokensForEth(uint256)
13e7c945  =>  swapTokensForReflection(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
818c19dc  =>  swapAndSendDividends(uint256)
```
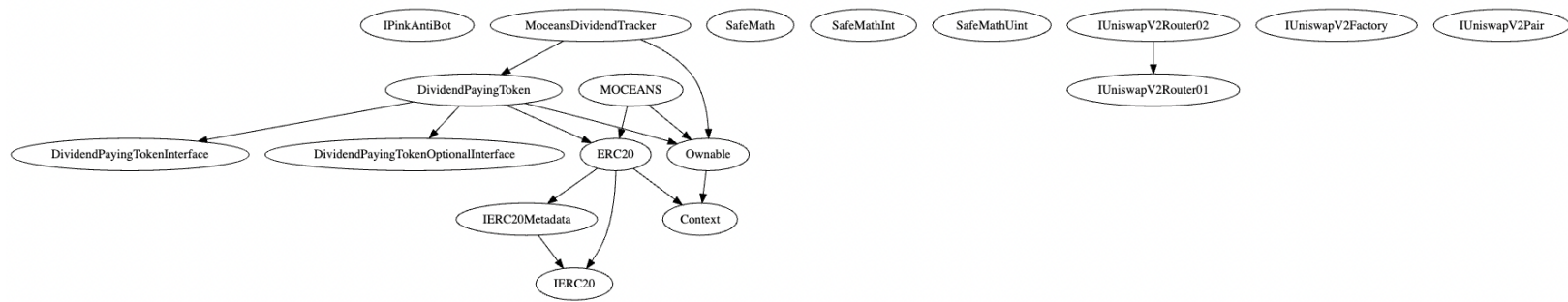
InterFi

Smart Contract
Security Audit

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	NA	NA
Burn	executes transfers of a specified number of tokens to a burn address	NA	NA
Mint	executes the creation of a specified number of tokens and adds it to the total supply	NA	NA
Rebase	circulating token supply adjusts (increases or decreases) automatically according to a token's price fluctuations	NA	NA
Dividend	executes transfers of a specified dividend token to a specified address	Yes	Passed
Lock / Pause	locks or pauses all or some function modules of the smart contract	NA	NA

Function	Description	Tested	Verdict
Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	**Passed**
Airdrop	executes transfers of a specified number of tokens to a specified address	NA	NA
Anti Whale	a non-whitelisted wallet can only transfer a specified number of tokens	Yes	**! Low**
Contract Fees	executes fee collection from swap events and/or transfer events	Yes	**Passed**
Cooldown Timer	functionality to limit the number of transactions that a wallet can make within 24-hours	NA	NA
Anti Bot	stops some or all bot wallets from interacting with the smart contract	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**

Notable Information 📝

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

```solidity
string private _name = "MOCEANS";
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 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.

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

❖ Smart contract utilizes `pinkAntibot()` function module to allow limited transactions after a number of blocks from launch-block.

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

```solidity
function setBotBlacklist(address _botAddress, bool _flag) external onlyOwner {
    require(isContract(_botAddress), "only contract address, not allowed externally owned
account");
```

❖ Smart contract owner can **change transaction fees**. Maximum fee limits are set to allow the value change within the set parameters.

❖ Smart contract owner can **change sell tx fee.** The smart contract owner can change the value to "zero". No arbitrary limit set.

```
function setSellTxFee(uint _value) external onlyOwner {
    txfee = _value;
function setTwentyFourhours(uint256 _time) external onlyOwner {
    TwentyFourhours = _time;
function enableDisableWhale(bool _value) external onlyOwner {
    enableDisableAntiWhale = _value;
```

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

"severity": 8, (! Low Severity)

"Redundant Codelines"

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

"severity": 8, (! Low Severity)

"Authorization through tx.origin"

SWC Attacks

| SWC ID	Description	Status
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Informational
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
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	! Low
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 the hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	! Low
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	1 informational severity issue identified
Centralization Risk	Active contract ownership identified
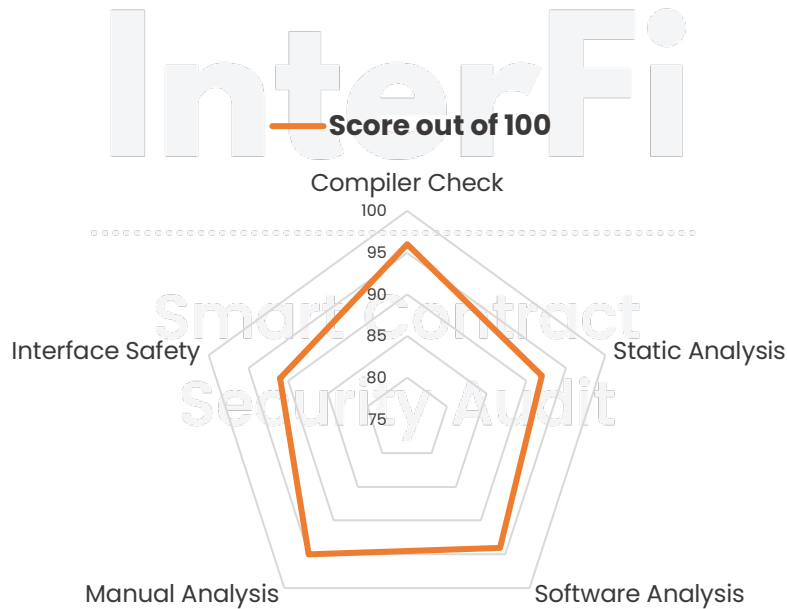
Auditor's Verdict

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:

❖ MOCEANS's smart contract source code has **LOW RISK SEVERITY**

❖ MOCEANS's smart contract has an **ACTIVE OWNERSHIP**

❖ MOCEANS'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