InterFi NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT OF

WOMEN OF MANKIND

WOMEN Of Mankind

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	Women of Mankind
Website	https://womenofmankind.com/
Twitter	https://twitter.com/MankindWomenof/
Instagram	https://www.instagram.com/womenofmankind/
Facebook	https://www.facebook.com/WOMCRYPTO/
YouTube	https://www.youtube.com/channel/UCOqede6YdAe8_2zueFJUe2A/
Report Date	July 27, 2022

About Women of Mankind

Women of Mankind (WOM) is a decentralized non-profit organization designed to assist women all over the world attain financial independence. It is an ecosystem built to help women become financially stable, self-sufficient, attain financial freedom, and able to take care of their children and homes without depending on others for assistance. This project was inspired to offer financial independence/freedom to women all over the world. WOM was created to support women all over the world who have little or no access to finances, banks, markets, good health care, businesses, etc. PURPOSE: A Utility token to assist all women in the world to become financially stable, self-sufficient, independent and able to take care of their children and homes without depending on their significant other or anyone.

Audit Summary

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. According to the smart contract audit:

- Women of Mankind's solidity source code has **LOW RISK SEVERITY**
- Women of Mankind's smart contract has an **ACTIVE OWNERSHIP**
- Important privileges – **SET MAX WALLET, SET MIN & MAX BUY, SET MIN & MAX SELL, SET FEES**

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, functional hack, and audit disclaimer, kindly refer to the audit.

🚨 Token Contract address: **0xD31fa3109e5960E77e57C4862093e4eA1a60d7B6**

⛓️ 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 Women of Mankind to conduct the smart contract security audit of their solidity source code. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

- WomenofMankind.sol

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0xD31fa3109e5960E77e57C4862093e4eA1a60d7B6#code

Contract Name: WOM

Compiler Version: v0.8.0

Optimization Enabled: Yes with 200 runs

Solidity Source Code On InterFi GitHub

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

SHA-1 Hash

Solidity source code is audited at hash # dc4c6bec38e854be5e7a82b86cafce4ab0ab2d91

Audit Methodology

The scope of this report is to audit the smart contract source code of Women of Mankind. 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-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

- ❖ Slither
- ❖ Consensys MythX, Mythril
- ❖ 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

Static Analysis

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

IERC20	Interface			
└	totalSupply	External !		NO !
└	balanceOf	External !		NO !
└	transfer	External !	🛑	NO !
└	allowance	External !		NO !
└	approve	External !	🛑	NO !
└	transferFrom	External !	🛑	NO !
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	

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!
└	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!

└	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❗
IWETH	Interface			
└	deposit	External ❗	💵	NO❗
└	balanceOf	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	withdraw	External ❗	🛑	NO❗
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	External ❗	🛑	onlyOwner
└	transferOwnership	External ❗	🛑	onlyOwner
WOM	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	increaseAllowance	External ❗	🛑	NO❗
└	decreaseAllowance	External ❗	🛑	NO❗
└	isExcludedFromReward	External ❗		NO❗
└	totalFees	External ❗		NO❗
└	deliver	External ❗	🛑	NO❗
└	reflectionFromToken	External ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	External ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	External ❗	🛑	onlyOwner

└	includeInFee	External ❗	🛑	onlyOwner
└	setTaxFeePercent	External ❗	🛑	onlyOwner
└	setLiquidityFeePercent	External ❗	🛑	onlyOwner
└	setOperationsFeePercent	External ❗	🛑	onlyOwner
└	setBurnFeePercent	External ❗	🛑	onlyOwner
└	setMaxAndMinBuy	External ❗	🛑	onlyOwner
└	setMaxAndMinSell	External ❗	🛑	onlyOwner
└	setMaxWallet	External ❗	🛑	onlyOwner
└	setMaxTx	External ❗	🛑	onlyOwner
└	setOperationWallet	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	calculateTaxFee	Private 🔐		
└	calculateBurnFee	Private 🔐		
└	calculateOperationsFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	_takeOperation	Private 🔐	🛑	
└	removeAllFee	Private 🔐	🛑	
└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	buyBackTokens	Private 🔐	🛑	lockTheSwap
└	swapETHForTokens	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	_transferBothExcluded	Private 🔐	🛑	

Software Analysis

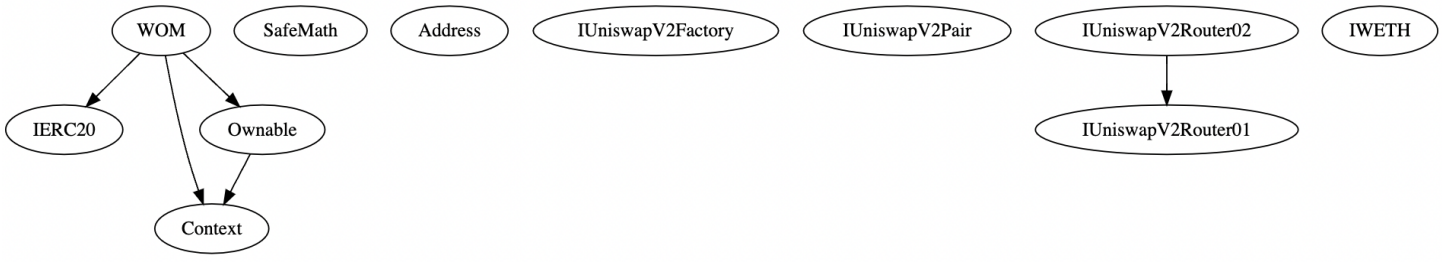
Function Signatures

```
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
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)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
36455e42  =>  _functionCallWithValue(address,bytes,uint256,string)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
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()
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)
d0e30db0  =>  deposit()
2e1a7d4d  =>  withdraw(uint256)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
88f82020  =>  isExcludedFromReward(address)
13114a9d  =>  totalFees()
3bd5d173  =>  deliver(uint256)
4549b039  =>  reflectionFromToken(uint256,bool)
2d838119  =>  tokenFromReflection(uint256)
52390c02  =>  excludeFromReward(address)
```


```
3685d419  =>  includeInReward(address)
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
061c82d0  =>  setTaxFeePercent(uint256)
8ee88c53  =>  setLiquidityFeePercent(uint256)
2e2d7fa6  =>  setOperationsFeePercent(uint256)
cea26958  =>  setBurnFeePercent(uint256)
12545a0c  =>  setMaxAndMinBuy(uint256,uint256)
9c10ea72  =>  setMaxAndMinSell(uint256,uint256)
5d0044ca  =>  setMaxWallet(uint256)
bc337182  =>  setMaxTx(uint256)
21e4b590  =>  setOperationWallet(address)
c49b9a80  =>  setSwapAndLiquifyEnabled(bool)
627f2894  =>  _reflectFee(uint256,uint256,uint256,uint256)
d4780e36  =>  _getValues(uint256)
65c63d72  =>  _getRValues(uint256,uint256,uint256,uint256,uint256)
94e10784  =>  _getRate()
97a9d560  =>  _getCurrentSupply()
6ad88269  =>  calculateBurnFee(uint256)
e50cfa33  =>  calculateOperationsFee(uint256)
cc126a23  =>  calculateLiquidityFee(uint256)
c432df5e  =>  _takeLiquidity(uint256)
45afded9  =>  _takeOperation(address,uint256)
301370af  =>  removeAllFee()
e7e3e3a7  =>  restoreAllFee()
5342acb4  =>  isExcludedFromFee(address)
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
fc155d1d  =>  buyBackTokens(uint256)
2eab2841  =>  swapETHForTokens(uint256)
173865ad  =>  swapAndLiquify(uint256)
b28805f4  =>  swapTokensForEth(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
```

Inheritance Graph



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**
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**
Contract Fees	executes fee collection from swap events and/or transfer events	Yes	**Passed**
Max Transaction	a non-whitelisted wallet can only transfer a specified number of tokens	Yes	**Passed**
Max Wallet	a non-whitelisted wallet can only hold a specified number of tokens	Yes	**Passed**
Transfer Ownership	executes transfer of contract ownership to a specified wallet	Yes	**Passed**

Best Practices ✅

- Owner cannot mint tokens after initial contract creation/deployment.
- The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```
string private _name = "WomenofMankind";
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 ⚠️

- Active smart contract owner: **0x4fe022e6a54ccd42b7e390497f817fdf0217ade6**
- Smart contract owner can **change transaction fees**. Limits are set to allow value change within the set parameters.
- Smart contract owner can **change max transaction limit**. Limits are set to allow value change within the set parameters.
- Smart contract owner can **change max wallet limit**. Limits are set to allow value change within the set parameters.

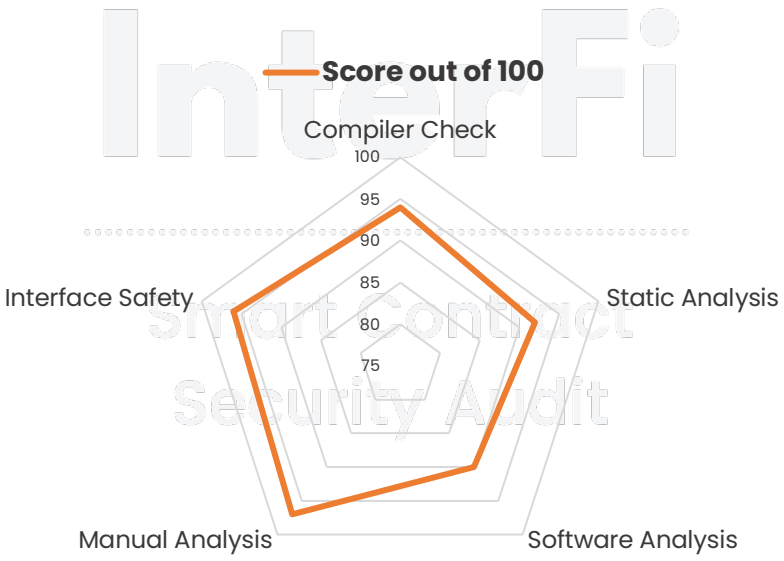
SWC Attacks

SWC ID	Description	Verdict
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	**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 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	1 low severity issue identified
! Informational	1 informational severity issue identified



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.

- Women of Mankind's solidity source code has **LOW RISK SEVERITY**
- Women of Mankind's smart contract has an **ACTIVE OWNERSHIP**

Note for stakeholders

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