SMART CONTRACT SECURITY AUDIT
Kawakami

SMART CONTRACT AUDIT | TEAM KYC | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 🇨🇦

Summary

Auditing Firm	InterFi Network
Architecture	InterFi "Echelon" Auditing Standard
Smart Contract Audit Approved By	Chris	Blockchain Specialist at InterFi Network
Project Overview Approved BY	Albert	Project Specialist at InterFi Network
Platform	Solidity
Audit Check (Mandatory)	Static, Software, Auto Intelligent & Manual Analysis
Project Check (Optional)	KYC, Website & Socials Analysis (Not Applicable)
Consultation Request Date	September 22, 2021
Report Date	September 27, 2021

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:

- ❖ **Kawakami's smart contract source code has LOW RISK SEVERITY.**
- ❖ **Kawakami has successfully PASSED the smart contract audit.**

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

Table Of Contents

Project Overview

InterFi was consulted by Kawakami on September 22, 2021 to conduct a smart contract security audit of their solidity source code.

Kawakami is a community driven cryptocurrency with a goal to become a decentralized meme token ecosystem. Kawa aims to create an entire ecosystem for meme tokens. Kawa is building a farm, DEX and a venture capital fund! **$KAWA** projects in development are:

- ❖ KawaSeed
- ❖ KawaFarm
- ❖ KawaSwap

Project	KAWAKAMI
Blockchain	Ethereum Mainnet / Ethereum Blockchain Explorer
Language	Solidity
Contract	0x546aed37d202d607f45cbd2b8c0cad0d25fbe339
Buy $KAWA	https://app.uniswap.org/#/swap?outputCurrency=0x546aed37d202d607f45cbd2b8c0cad0d25fbe339
Website	https://kawatoken.io/
Twitter	https://twitter.com/Kawakami_Inu
Telegram	https://t.me/kawatoken
Instagram	https://www.instagram.com/kawakami_inu/
Medium	https://kawatoken.medium.com/

Public logo

Contract Source Code On Blockchain (Etherscan Verified With Exact Match)

Compiler Version: v0.7.6+commit.7338295f

Optimization Enabled: Yes with 200 runs

https://etherscan.io/address/0x546aed37d202d607f45cbd2b8c0cad0d25fbe339#code

Solidity Source Code On InterFi GitHub

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

GitHub Commits

Solidity source code committed at: ba54f7a01f83c5f5605c05f5212413893928fa27

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Kawakami. The source code can be viewed in its entirety on

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

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 (RE)
- ❖ Unhandled Exceptions (UE)
- ❖ Transaction Order Dependency (TO)
- ❖ Integer Overflow (IO)
- ❖ Unrestricted Action (UA)

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 Automated AI 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
- ❖ Consensys Surya
- ❖ 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 manipulation, 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 critical risk severity
! Medium	This level vulnerabilities are 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 – Overview

Contract information

Query	Result
Name	Kawakami
Symbol	KAWA
Decimals	18
Total Max Supply	999,999,999,999
Owner	0xc0e9f051c6ef76d15f777ae8a7cb14c03550e79e
Tax	1
Dev Address	0x93837577c98e01cfde883c23f64a0f608a70b90f
Dev Fee	2
Uniswap V2 Pair	0x689371e9f7833c1d6b33434dd6f121ef5a88a20b
Uniswap V2 Router	0x7a250d5630b4cf539739df2c5dacb4c659f2488d

Smart Contract – 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 🔓 | 🛑 | |
||||||
```

```
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🔴 |NO❗ |
| └ | owner | Public ❗ | |NO❗ |
| └ | renounceOwnership | Public ❗ | 🔴 | onlyOwner |
| └ | transferOwnership | Public ❗ | 🔴 | onlyOwner |
| └ | geUnlockTime | Public ❗ | |NO❗ |
| └ | lock | Public ❗ | 🔴 | onlyOwner |
| └ | unlock | Public ❗ | 🔴 |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❗ |
| └ | 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❗ |
||||||
| **Kawa** | Implementation | Context, IERC20, Ownable |||
| └ | <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❗ |
| └ | isExcludedFromReward | Public ❗ |    |NO❗ |
| └ | totalFees | Public ❗ |    |NO❗ |
| └ | deliver | Public ❗ | 🔴  |NO❗ |
| └ | reflectionFromToken | Public ❗ |    |NO❗ |
| └ | tokenFromReflection | Public ❗ |    |NO❗ |
| └ | excludeFromReward | Public ❗ | 🔴  | onlyOwner |
| └ | includeInReward | External ❗ | 🔴  | onlyOwner |
| └ | _transferBothExcluded | Private 🔒 | 🔴  | |
| └ | excludeFromFee | Public ❗ | 🔴  | onlyOwner |
| └ | includeInFee | Public ❗ | 🔴  | onlyOwner |
| └ | setDevFeePercent | External ❗ | 🔴  | onlyOwner |
```

```
|  └ | setDevAddress | External ❗ | 🔴 | onlyOwner |
|  └ | setTaxFeePercent | External ❗ | 🔴 | onlyOwner |
|  └ | setLiquidityFeePercent | External ❗ | 🔴 | onlyOwner |
|  └ | setSwapAndLiquifyEnabled | Public ❗ | 🔴 | onlyOwner |
|  └ | <Receive Ether> | External ❗ | 💵 |NO❗ |
|  └ | _reflectFee | Private 🔐 | 🔴 | |
|  └ | _getValues | Private 🔐 |    | |
|  └ | _getTValues | Private 🔐 |    | |
|  └ | _getRValues | Private 🔐 |    | |
|  └ | _getRate | Private 🔐 |    | |
|  └ | _getCurrentSupply | Private 🔐 |    | |
|  └ | _takeLiquidity | Private 🔐 | 🔴 | |
|  └ | calculateTaxFee | Private 🔐 |    | |
|  └ | calculateLiquidityFee | Private 🔐 |    | |
|  └ | removeAllFee | Private 🔐 | 🔴 | |
|  └ | restoreAllFee | Private 🔐 | 🔴 | |
|  └ | isExcludedFromFee | Public ❗ |    |NO❗ |
|  └ | _approve | Private 🔐 | 🔴 | |
|  └ | _transfer | Private 🔐 | 🔴 | |
|  └ | swapAndSendToDev | Private 🔐 | 🔴 | lockTheSwap |
|  └ | swapAndLiquify | Private 🔐 | 🔴 | lockTheSwap |
|  └ | swapTokensForEth | Private 🔐 | 🔴 | |
|  └ | addLiquidity | Private 🔐 | 🔴 | |
|  └ | _tokenTransfer | Private 🔐 | 🔴 | |
|  └ | _transferStandard | Private 🔐 | 🔴 | |
|  └ | _transferToExcluded | Private 🔐 | 🔴 | |
|  └ | _transferFromExcluded | Private 🔐 | 🔴 | |
```

Smart Contract
Security Audit

Smart Contract – Software Analysis

Callout functions – Sighash

```
Sighash   |   Function Signature
=========================
11902160  =>  _getTValues(uint256)
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
75128141  =>  calculateTaxFee(uint256)
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)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
b6c52324  =>  geUnlockTime()
dd467064  =>  lock(uint256)
a69df4b5  =>  unlock()
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)
```

```
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)
6ff6cdf4  =>  _transferBothExcluded(address,address,uint256)
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
379e2919  =>  setDevFeePercent(uint256)
d0d41fe1  =>  setDevAddress(address)
061c82d0  =>  setTaxFeePercent(uint256)
8ee88c53  =>  setLiquidityFeePercent(uint256)
c49b9a80  =>  setSwapAndLiquifyEnabled(bool)
184d894e  =>  _reflectFee(uint256,uint256)
d4780e36  =>  _getValues(uint256)
1d5671e4  =>  _getRValues(uint256,uint256,uint256,uint256)
94e10784  =>  _getRate()
97a9d560  =>  _getCurrentSupply()
c432df5e  =>  _takeLiquidity(uint256)
cc126a23  =>  calculateLiquidityFee(uint256)
301370af  =>  removeAllFee()
e7e3e3a7  =>  restoreAllFee()
5342acb4  =>  isExcludedFromFee(address)
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4a6dc6df  =>  swapAndSendToDev(uint256)
173865ad  =>  swapAndLiquify(uint256)
b28805f4  =>  swapTokensForEth(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
b09bbc79  =>  _tokenTransfer(address,address,uint256,bool)
2852df65  =>  _transferStandard(address,address,uint256)
16f1cc83  =>  _transferToExcluded(address,address,uint256)
c7d9be66  =>  _transferFromExcluded(address,address,uint256)
```

Callout functions – Inheritance Graph
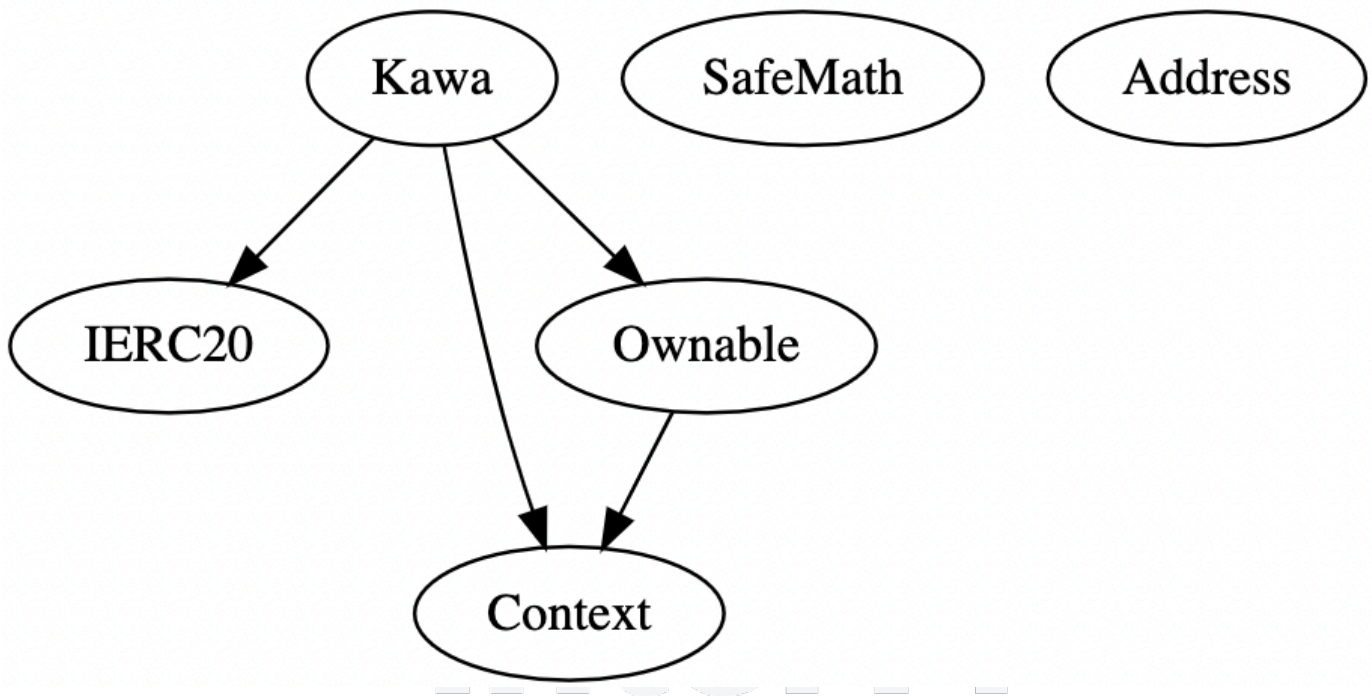
Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
TotalSupply	provides information about the total token supply	Yes	Passed
BalanceOf	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
TransferFrom	executes transfers of a specified number of tokens from 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	NA	NA

Verified

- ❖ Active Owner: [0xc0e9f051c6ef76d15f777ae8a7cb14c03550e79e](0xc0e9f051c6ef76d15f777ae8a7cb14c03550e79e)

- ❖ Owner can mint tokens at token launch.

- ❖ Owner can-not lock or burn user assets.

- ❖ Owner can modify transaction, liquidity taxes, etc.

Important Information

Kawakami smart contract utilizes the "SafeMath" to prevent known vulnerabilities.

```solidity
string private _name = 'Kawakami';
  string private _symbol = 'KAWA';
  uint8 private _decimals = 18;

library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, 'SafeMath: addition overflow');

    return c;
  }
  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 mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: modulo by zero");
        return a % b;
    }
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, 'SafeMath: modulo by zero');
  }
```

Kawakami smart contract has low severity issues which may or may not create any functional vulnerability.

- Expected pragma, import directive or contract/interface/library definition. [234,1]

- Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient. [445,3] [903,3]

- Return value of low-level calls not used. [1309,5]

Smart Contract – SWC Risk Assessment

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	! Low
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT 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	! Low
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

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	**2 Low severity issues identified**
Passed	**40 functions and instances verified and passed**

Score out of 100
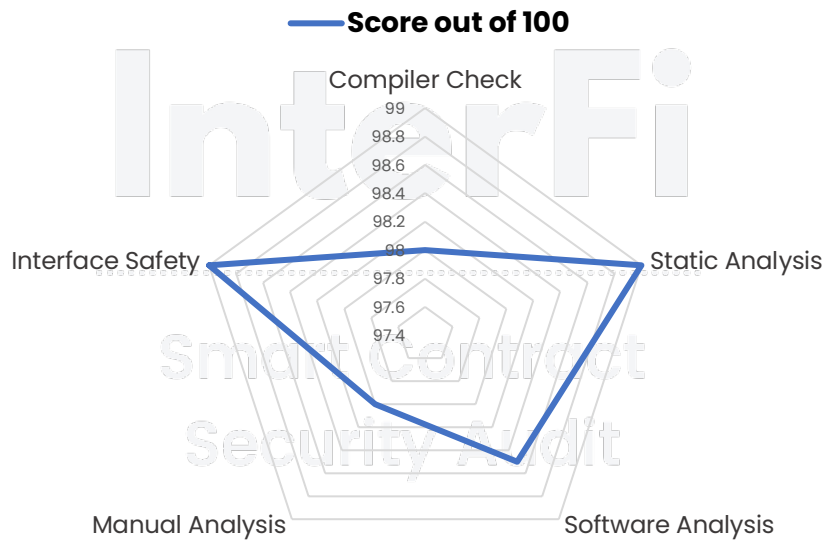
Compiler Check	98
Static Analysis	99
Software Analysis	98.5
Manual Analysis	98
Interface Safety	99

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.

Kawakami smart contract source code has LOW RISK SEVERITY.

Kawakami has successfully PASSED the smart contract audit.

General Note:

❖ Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.

❖ Project's liquidity pair isn't checked and verified due to out of scope.

❖ Project website is not checked due to out of scope. The website hasn't been reviewed for SSL and lighthouse report.

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 🇨🇦