SMART CONTRACT SECURITY AUDIT OF

HAZEL COIN

Summary

Auditing Firm	InterFi Network
Client Firm	Hazel Coin
Architecture	InterFi "Echelon" Auditing Standard
Platform	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Final Report Date	December 20, 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:

- ❖ Hazel Coin's smart contract source code has **LOW RISK SEVERITY**
- ❖ Hazel Coin has **PASSED** the smart contract audit
- ❖ Hazel Coin's smart contract has an **ACTIVE OWNERSHIP**
- ❖ Owner privileges to watch – **LOCK – UNLOCK, MAX TX LIMIT, SET TX FEES**

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

🔴 Contract address: **0x969F03345Ee3E1867bE0bDd3a090bae1c189596d**

⚌ Blockchain: **Binance Smart Chain**

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

Table Of Contents

Project Overview

InterFi was consulted by Hazel Coin to conduct the smart contract security audit of their solidity source code.

About Hazel Coin

HΛZΞL COIN is a social cryptocurrency to help communities and members exchange, tip, share and donate to various content creators, promotion, tickets, clubs, courses, sporting events and E-Gaming. Allowing donations to social causes and humanitarian programs on many major social integrated networks around the world.

Hazel is the main token of HΛZΞL COIN a native token of the Binance Smart Chain (BSC) network, which follows the BEP-20 protocol. Thus, it allows users to get the high transaction capacity of Blockchain and the smart contract functionality of BSC, ensuring low fees and fast transaction speed, but what is even more special is its compatibility with Ethereum ERC20 VM.

Project	Hazel Coin
Blockchain	Binance Smart Chain
Language	Solidity
Contract	0x969F03345Ee3E1867bE0bDd3a090bae1c189596d
Website	https://www.hazelcoin.net/
Reddit	https://www.reddit.com/user/Hazelcoins
LinkedIn	https://www.linkedin.com/in/hazel-coin-000905214

Project Logo

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x969F03345Ee3E1867bE0bDd3a090bae1c189596d#code

Contract Name: Hazel Coin

Compiler Version: v0.6.2

Optimization Enabled: Yes with 200 runs

Solidity Source Code On InterFi GitHub

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

SHA-1 Hash

Solidity source code is audited at hash #05842b3000950ff8de56e4a8a6113b534561cf57

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Hazel Coin. 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 ❖ Ownership Takeover ❖ Gas Limit and Loops
Source Code Review	❖ 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 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 – Static Analysis

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

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

Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	getUnlockTime	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❗ |
||||||
| **HAZELCOIN** | 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 |
| └ | setLiquidityFeePercent | External ❗ | 🔴   | onlyOwner |
| └ | setMaxTxPercent | 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❗ |
| └ | swapTokensForEth | Private 🔒 | 🛑 | |
| └ | _transferToExcluded | Private 🔒 | 🛑 | |
| └ | _approve | Private 🔒 | 🛑 | |
| └ | _transferFromExcluded | Private 🔒 | 🛑 | |
| └ | addLiquidity | Private 🔒 | 🛑 | |
| └ | _tokenTransfer | Private 🔒 | 🛑 | |
| └ | _transferStandard | Private 🔒 | 🛑 | |
| └ | _transfer | Private 🔒 | 🛑 | |
| └ | swapAndLiquify | Private 🔒 | 🛑 | lockTheSwap |
| └ | includeInFee | Public ❗ | 🛑 | onlyOwner |
| └ | setTaxFeePercent | External ❗ | 🛑 | onlyOwner |
```

Smart Contract
Security Audit

Smart Contract – Software Analysis

Function Signatures

```
11902160  =>  _getTValues(uint256)
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
75128141  =>  calculateTaxFee(uint256)
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)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
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)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
602bc62b  =>  getUnlockTime()
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)
8ee88c53  =>  setLiquidityFeePercent(uint256)
d543dbeb  =>  setMaxTxPercent(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)
b28805f4  =>  swapTokensForEth(uint256)
16f1cc83  =>  _transferToExcluded(address,address,uint256)
104e81ff  =>  _approve(address,address,uint256)
c7d9be66  =>  _transferFromExcluded(address,address,uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
b09bbc79  =>  _tokenTransfer(address,address,uint256,bool)
2852df65  =>  _transferStandard(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
173865ad  =>  swapAndLiquify(uint256)
ea2f0b37  =>  includeInFee(address)
061c82d0  =>  setTaxFeePercent(uint256)
```

Inheritance Graph
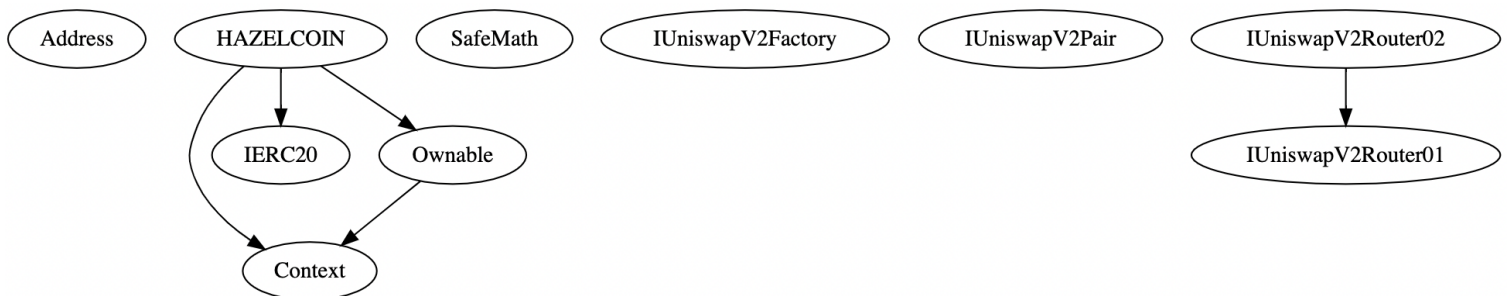
Smart Contract – Manual Analysis

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**
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 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
Blacklist	stops specified wallets from interacting with the smart contract function modules	NA	NA
Lock	stops or locks all function modules of the smart contract	Yes	**! Low**

Function	Description	Tested	Verdict
Dividend	executes transfers of a specified dividend token to a specified address	NA	NA
Airdrop	executes transfers of a specified number of tokens to a specified address	NA	NA
Max Transaction	a non-whitelisted wallet can only transfer a specified number of tokens	Yes	! Low
Max Wallet	a non-whitelisted wallet can only hold a specified number of tokens	NA	NA
Cooldown Timer	functionality to limit the number of transactions that a wallet can make within a 24-hour span	NA	NA
Anti Bot	stops some or all bot wallets from interacting with the smart contract	NA	NA
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 stop or pause the smart contract.

❖ Owner cannot lock or burn the user assets.

❖ Owner cannot mint tokens after initial contract creation/deployment.

❖ The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```solidity
string private _name = "HazelCoin";
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: 0xa831763b37584973c9a8c630746570c449615d4d

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

❖ Smart contract owner can **_lock or unlock_** the smart contract function modules.

❖ Smart contract owner can **_change the buy and sell fees_**. This function module can be used to impose extraordinary transaction fees.

❖ Smart contract owner can **change max transaction %**. The smart contract owner can change the value to "zero".

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

{

    "resource": " /HazelCoin.sol",

    "owner": "_generated_diagnostic_collection_name_#0",

    "severity": 8, (! Low Severity)

    "Expected token Comma got 'Payable",

    "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	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 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
Safety Score	89 out of 100

Score out of 100

Compiler Check
Static Analysis
Software Analysis
Manual Analysis
Interface Safety
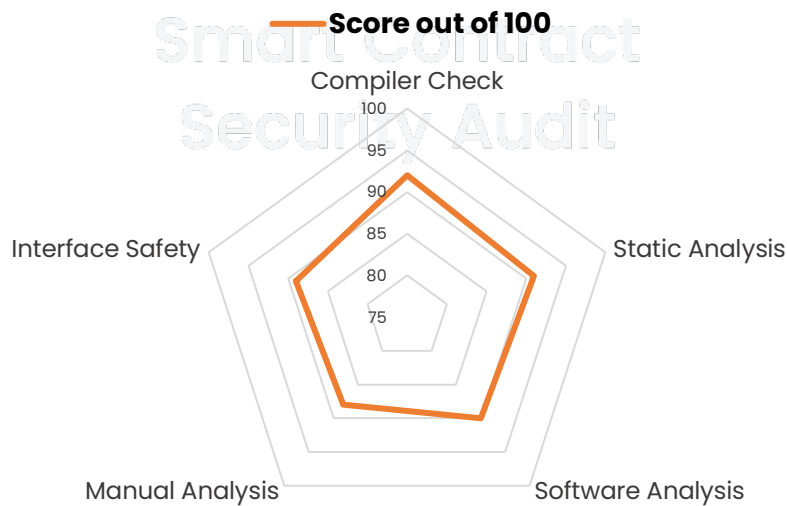
Auditor's Verdict

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.

Hazel Coin's smart contract source code has LOW RISK SEVERITY.

Hazel Coin has PASSED the smart contract audit.

Note for stakeholders

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

- ❖ 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 🇨🇦