

# SMART CONTRACT SECURITY AUDIT OF



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 | Marketing Specialist at InterFi Network

**Platform** Solidity

Mandatory Audit Check Static, Software, Auto Intelligent & Manual Analysis

Consultation Request Date November 13, 2021

Report Date November 15, 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:

- FlokiSis' smart contract source code has LOW RISK SEVERITY.
- FlokiSis has PASSED the smart contract audit.

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

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



# **Table Of Contents**

### **Project Information**

Overview	4
InterFi "Echelon" Audit Standard	
Audit Scope & Methodology	6
InterFi's Risk Classification	8
Smart Contract Risk Assessment	
Static Analysis	9
Software Analysis	13
Manual Analysis	16
SWC AttacksSmall Continued	19
Risk Status & Radar Chart	
Report Summary	
Auditor's Verdict	22
<u>Legal Advisory</u>	
Important Disclaimer	23
About InterFi Network	24



# **Project Overview**

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

### **About FlokiSis**

A deflationary token which is centered by the community and driven by its internal force

Project	FlokiSis
Blockchain	Binance Smart Chain
Language	Solidity
Contract	0xf39aEe2FF2869BA7D212b119DDFE73Ef869D333C
Website	https://flokisis.io/
Telegram	https://t.me/FlokiSis
Twitter	https://twitter.com/FlokiSis

### <u>Public logo</u>





### Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0xf39aEe2FF2869BA7D2I2bII9DDFE73Ef869D333C#code

Contract Name: FlokiSis (SATOSHI)

Symbol: FSIS

Compiler Version: v0.6.12

Optimization Enabled: No with 200 runs

### Solidity Source Code On InterFi GitHub

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

#### SHA-1 Hash

Solidity source code is audited at hash #b81b7c898ba5fb88d510dc7ed1a7f25664bf4bce



# **Audit Scope & Methodology**

The scope of this report is to audit the smart contract source code of FlokiSis. 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

▼ Re-entrumely	•	-entrancy
----------------	---	-----------

- Unhandled Exceptions
- Transaction Order Dependency
- Integer Overflow
- Unrestricted Action
- Incorrect Inheritance Order
- Typographical Errors
- Requirement Violation
- Ownership Takeover
- Gas Limit and Loops
- Deployment Consistency
- Repository Consistency
- Data Consistency
- Token Supply Manipulation
- Access Control and Authorization
- Operations Trail and Event Generation
- Assets Manipulation
- Liquidity Access

#### **Smart Contract Vulnerabilities**

#### **Source Code Review**

#### **Functional Assessment**



#### 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
- 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.

		SHIGH COHINGEL
Risk severity	Meaning	Security Audit
! Critical	This level vulner	abilities could be exploited easily, and can lead to asset loss, data
	loss, asset mani	pulation, or data manipulation. They should be fixed right away.
! High	This level vulner	abilities are hard to exploit but very important to fix, they carry an
	elevated risk of s	smart contract manipulation, which can lead to critical risk severity
! Medium	This level vulner	abilities are should be fixed, as they carry an inherent risk of future
	exploits, and had	cks which may or may not impact the smart contract execution.
	This level vulne	erabilities can be ignored. They are code style violations, and
! Low	informational st	catements 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

```
<mark>**IERC20**</mark> | Interface |  |||
L | totalSupply | External |
                                   |N0
L | balanceOf | External | |
                                 |NO |
👢 | transfer | External 📒 | 🥌
L | allowance | External ! |
  | approve | External | |
  | transferFrom | External | | 🛑 |NO! |
<mark>**SafeMath**</mark> | Library |
  | add | Internal 🖴 |
L | sub | Internal 🗎
    sub | Internal 🖴
    mul | Internal 🗎
    div | Internal 🖴
    div | Internal 🔒
    mod | Internal 🖴
    mod | Internal 🖴
**Context** | Implementation | |||
^{\mathsf{L}} | _msgSender | Internal ^{\mathsf{L}} | | |
└ | _msgData | Internal 🛍 |
**Address** | Library | |||
└ | isContract | Internal 🗎 | | |
👢 | sendValue | Internal 🛍 | 🥮 | |
L | functionCall | Internal 🗎 | 🥌
👢 | functionCall | Internal 🛍 | 🥌
└ | functionCallWithValue | Internal 🗎 |
└ | functionCallWithValue | Internal 🗎
    _functionCallWithValue | Private 💕
```



```
| Implementation | Context |||
     <Constructor> | Internal 🗎 | 🥮
     owner | Public | | NO! |
     renounceOwnership | Public
                                      | onlyOwner |
     transferOwnership | Public 📒 | 🥌
                                    | onlyOwner |
     geUnlockTime | Public | | |NO! |
         <mark>unlock</mark> | Public 🏮 | 🥮 |NO 🎙 |
 **IUniswapV2Factory** | Interface | |||
 L | feeTo | External | | NO! |
 L | feeToSetter | External | |
                                |NO |
    getPair | External | | |NO
    allPairs | External 📒 |
                             allPairsLength | External | | |NO! |
 L | createPair | External | | 🛑 |NO! |
                              |N0 |
     setFeeTo | External 📒 | 🥌
     setFeeToSetter | External | | 🛑 |NO! |
\square
 **IUniswapV2Pair** | Interface | |||
 L | name | External | | NO | |
    decimals | External 📒 |
                             |N0 |
    totalSupply | External ! |
                                |N0 |
     balanceOf | External | |
                               |N0 | |
     allowance | External
     approve | External 📒 | 🥌
                              NO !
     transfer | External 📘 📗
     transferFrom | External 📘 | 🥌
     DOMAIN_SEPARATOR | External | |
     PERMIT_TYPEHASH | External !
                                    |NO |
     nonces | External | |
                            |N0 |
     permit | External 📒 | 🥌
                             |N0
     MINIMUM_LIQUIDITY | External | |
                                      |NO |
     factory | External ! |
                             |N0 | |
     token0 | External
                            INO
    token1 | External
     getReserves | External | |
                                |N0 |
     price0CumulativeLast | External !
                                         |N0
     price1CumulativeLast | External
                                         |N0 |
    kLast | External | |
                           |N0 !
     mint | External ! |
                            |N0 |
                            |N0
     burn | External
     swap | External
                            |N0
                            |N0
     skim | External
     sync | External 📒 | 🥌
                            IN0
     initialize | External 📒 | 🥮 |NO 📙 |
**IUniswapV2Router01** | Interface | |||
 L | factory | External | | | NO!
```



```
WETH | External | |
                            INO !
     addLiquidity | External | |
                                      |N0
     addLiquidityETH | External
                                   | 🕮 |N0 !
     removeLiquidity | External
                                        INO !
     removeLiquidityETH | External 👎
                                           INO !
     removeLiquidityWithPermit | External
     removeLiquidityETHWithPermit | External
     swapExactTokensForTokens | External
                                                 IN0
     swapTokensForExactTokens | External
                                                 |N0
     swapExactETHForTokens | External
     swapTokensForExactETH | External
                                               IN0
     swapExactTokensForETH | External
     swapETHForExactTokens | External
     quote | External ! |
                             |N0
     getAmountOut | External | |
                                    |NO | |
     getAmountIn | External | |
                                   |NO |
     getAmountsOut | External | | | NO!
     getAmountsIn | External ! |
                                    |NO | |
\Pi\Pi\Pi\Pi
 **IUniswapV2Router02** | Interface | IUniswapV2Router01 |||
     removeLiquidityETHSupportingFeeOnTransferTokens | External 📒 | 🥌 |NO
     removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External
     swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! |
                                                                              |N0
     swapExactETHForTokensSupportingFeeOnTransferTokens | External
     swapExactTokensForETHSupportingFeeOnTransferTokens | External 🕴 | 🥌
    ATOSHI** | Implementation | Context, IERC20, Ownable |||
     <Constructor> | Public
     name | Public | |
                          |NO | |
     symbol | Public | | NO
     decimals | Public | |
                              |N0
     totalSupply | Public | | NO | |
     balanceOf | Public | |
                               |N0 |
     transfer | Public | |
                               |N0
     allowance | Public ! |
                               |NO
     approve | Public | |
                               |N0
     transferFrom | Public |
     increaseAllowance | Public
     decreaseAllowance | Public
     isExcludedFromReward | Public
                                          |NO | |
     totalFees | Public | |
                               |N0 |
     deliver | Public 📒 | 🥌
                              |N0
     reflectionFromToken | Public
                                         |N0 |
     tokenFromReflection | Public
                                         |N0 |
     excludeFromReward | Public
                                         | onlyOwner |
     includeInReward | External
     _transferBothExcluded | Private 🔐
     excludeFromFee | Public | | 🛑
                                    | onlyOwner |
     includeInFee | Public | |
                                   | onlyOwner |
      setTaxFeePercent
                      | External 📒 | 🥌
                                        | onlyOwner |
```



```
setLiquidityFeePercent | External 📘 | 🥮 | onlyOwner |
   setMaxTxPercent | External ! | 🔴 | onlyOwner |
  setSwapAndLiquifyEnabled | Public | | 🛑 | onlyOwner |
   _reflectFee | Private 😭 | 🥮
  _getValues | Private 聲 |  | |
  _getTValues | Private 😭 | | |
  _getRValues | Private 聲 | | |
  _getRate | Private 🔐 | | |
  _getCurrentSupply | Private 聲 | | | |
L | _takeLiquidity | Private 🔐 | 🥮 | |
L | calculateTaxFee | Private ≝ |  |  |
└ | calculateLiquidityFee | Private 🔐 | | |
  removeAllFee | Private 🗳 | 🥮 | |
  restoreAllFee | Private 🛍 | 🥮
L | isExcludedFromFee | Public | | NO! |
  | _approve | Private 🔐 | 🥮 | |
L | _transfer | Private ≅ | ● | |
L | swapAndLiquify | Private 🔐 | 🥌 | lockTheSwap |
└ | swapTokensForEth | Private 🛍 | 🔎 | |
└ | addLiquidity | Private 🔐 | 🛑 | |
👢 | _transferStandard | Private 😭 | 🥌
L | _transferToExcluded | Private 🗳 | 🥌 | |
📙 | _transferFromExcluded | Private 😭 | 🥌 | |
```

Smart Contract Security Audit



# **Smart Contract - Software Analysis**

### **Function Signatures**

```
1902160 => 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 => transfer0wnership(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()
```



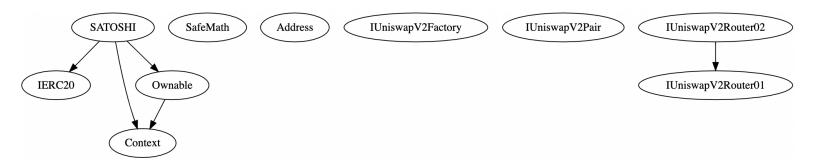
```
7ecebe00 => nonces(address)
d505accf => permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56 => MINIMUM LIQUIDITY()
c45a0155 => factory()
0dfe1681 => token0()
d21220a7 => token1()
0902f1ac => getReserves()
5909c0d5 => price0CumulativeLast()
5a3d5493 => price1CumulativeLast()
7464fc3d => kLast()
6a627842 \Rightarrow 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)
061c82d0 => setTaxFeePercent(uint256)
8ee88c53 => setLiquidityFeePercent(uint256)
d543dbeb => setMaxTxPercent(uint256)
c49b9a80 => setSwapAndLiquifyEnabled(bool)
184d894e => _reflectFee(uint256,uint256)
d4780e36 => getValues(uint256)
1d5671e4 => _getRValues(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)
173865ad => swapAndLiquify(uint256)
b28805f4 => swapTokensForEth(uint256)
9cd441da => addLiquidity(uint256,uint256)
```

# Security Audit

### **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		
Building Of	account	Yes	Passed
Transfer	executes transfers of a specified number of	Vaa	Dansond
Hullstei	tokens to a specified address	Yes	Passed
Approve	allow a spender to withdraw a set number of	V	Passed
Approve	tokens from a specified account	Yes	
Allowance	returns a set number of tokens from a spender to the owner	Yes	Passed
	is an action in which the project buys back its		
Buy Back	tokens from the existing holders usually at a	NA	NA
	market price nort Contract		
Burn	executes transfers of a specified number of tokens to a burn address	NA	NA
	executes creation of a specified number of		
Mint	tokens and adds it to the total supply	NA	NA
	circulating token supply adjusts (increases or		
Rebase	decreases) automatically according to a token's	NA	NA
	price fluctuations		
Blacklist	stops specified wallets from interacting with the	N.1.6	
BidCKiiSt	smart contract function modules	NA	NA
Lock	stops or locks all function modules of the smart	V	Deces
Look	<mark>contract</mark>	Yes	Passed



#### **Review**

- Active smart contract owner: 0xa65f9b6a6dfa9121816605c3f757b9f7d8ff751c
- Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.
- Smart contract can be locked by the owner. This stops or locks all function modules of the smart contract.
- Owner can-not lock or burn user assets.
- Owner can-not stop or pause the smart contract.
- Owner can-not mint tokens after launch.
- The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

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



The smart contract has 1 low severity issue which may or may not create any functional vulnerability.

```
"resource": " /FlokiSis.sol",
"owner": "_generated_diagnostic_collection_name_#0",
"severity": 8, (! Low Severity)
```

"Expected pragma, import directive or contract/interface/library definition",

"source": "solc",

}

Interfi

Smart Contract Security Audit



# **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	! 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 Smart Contract	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	1 low severity issue identified
Passed	41 functions and instances verified and passed Score out of 100
	Interface Safety  Static Analysis  Manual Analysis  Software Analysis
	Compiler Check 88
	Static Analysis 90

Software Analysis

Manual Analysis

Interface Safety

92

91

90



### **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.

FlokiSis' smart contract source code has LOW RISK SEVERITY.

FlokiSis 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 <a href="https://interfi.network">https://interfi.network</a>

To view our audit portfolio, visit <a href="https://github.com/interfinetwork">https://github.com/interfinetwork</a>

To book an audit, message <a href="https://t.me/interfiaudits">https://t.me/interfiaudits</a>





