InterFi NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT OF

Lottery & Reflector Contracts

(Squidoshi Ecosystem)

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 08, 2021
Report Date	November 13, 2021

InterFi

Audit Summary

Smart Contract Security Audit

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:

- **Squidoshi's Lottery & Reflector smart contract source codes have LOW RISK SEVERITY.**
- **Squidoshi's Lottery & Reflector smart contracts have 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**

Project Overview

InterFi was consulted by Squidoshi to conduct the smart contract security audits of their Lottery and Reflector source codes.

About Squidoshi

Introducing Squidoshi, a play to earn DEFI project that was inspired by "Squid Games" and "Shiba Inu" token. With its unique tokenomics and community members, each hodler will get the opportunity to vote on the currency that will pave the way to receive passive income on an ongoing basis. With that being said, Squidoshi uses the power of the lottery system to further reward its community. By "hodling" and fulfilling the lottery requirement, community members get the opportunity to win 100 million Shiba Inu tokens.

InterFi
Smart Contract
Security Audit

Project	Squidoshi
Blockchain	Binance Smart Chain
Language	Solidity
Token Contract	0x3a7e408B7780774843CD7136BEa3EbA55a8A1778
Lottery Contract	0x4E2d982E970f3b7AC2caA92eA5d700cbCD0b0c12
Reflector Contract	0x98073edeADd414fc72F780b0608646c47AE96E3B
Website	https://squidoshi.com/
Telegram	https://t.me/squidtn
Twitter	https://twitter.com/squidoshi

Public logo

Squidoshi's Lottery Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x4E2d982E970f3b7AC2caA92eA5d700cbCD0b0c12#code

Contract Name: SquidoshiSmartLottery

Compiler Version: v0.6.12

Optimization Enabled: Yes with 90 runs

InterFi
Smart Contract
Security Audit

Squidoshi's Reflector Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x98073edeADd414fc72F780b0608646c47AE96E3B#code

Contract Name: SquidoshiReflector

Compiler Version: v0.6.12

Optimization Enabled: Yes with 90 runs

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Squidoshi's Lottery and Reflector smart contracts. 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 Smart Contract Security Audit

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

InterFi
Smart Contract
Security Audit

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.

InterFi
Smart Contract
Security Audit

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

Lottery Smart Contract – Static Analysis

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

IPancakeRouter01	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 !
IPancakeRouter02	Interface	IPancakeRouter01			
└	removeLiquidityETHSupportingFeeOnTransferTokens	External !	🛑	NO !
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !	🛑	NO !
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !	🛑	NO !
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External !	💵	NO !
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	🛑	NO !
IPancakeFactory	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❗
IPancakePair	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❗
IBEP20	Interface	|||
└	totalSupply	External ❗	|NO❗
└	decimals	External ❗	|NO❗
└	symbol	External ❗	|NO❗
└	name	External ❗	|NO❗
└	getOwner	External ❗	|NO❗
└	balanceOf	External ❗	|NO❗
└	transfer	External ❗	🛑 |NO❗
└	allowance	External ❗	|NO❗
└	approve	External ❗	🛑 |NO❗
└	transferFrom	External ❗	🛑 |NO❗
LockableFunction	Implementation	|||

IAuthorizedList	Interface			
└	authorizeCaller	External ❗	🛑	NO❗
IAuthorizedListExt	Interface			
└	authorizeByAuthorized	External ❗	🛑	NO❗
IUserInfoManager	Interface			
IBaseDistributor	Interface			
└	deposit	External ❗	💵	NO❗
└	rewardCurrency	External ❗		NO❗
ISmartLottery	Interface	IBaseDistributor, IAuthorizedListExt, IAuthorizedList		
└	draw	External ❗		NO❗
└	jackpotAmount	External ❗		NO❗
└	isJackpotReady	External ❗		NO❗
└	setJackpot	External ❗	🛑	NO❗
└	checkAndPayJackpot	External ❗	🛑	NO❗
└	excludeFromJackpot	External ❗	🛑	NO❗
└	setMaxAttempts	External ❗	🛑	NO❗
└	setJackpotToCurrency	External ❗	🛑	NO❗
└	setJackpotToToken	External ❗	🛑	NO❗
└	setJackpotEligibilityCriteria	External ❗	🛑	NO❗
└	logTransfer	External ❗	🛑	NO❗
EnumerableSet	Library			
└	_add	Private 🔐	🛑	
└	_remove	Private 🔐	🛑	
└	_contains	Private 🔐		
└	_length	Private 🔐		
└	_at	Private 🔐		
└	add	Internal 🔒	🛑	
└	remove	Internal 🔒	🛑	
└	contains	Internal 🔒		
└	length	Internal 🔒		
└	at	Internal 🔒		
└	add	Internal 🔒	🛑	
└	remove	Internal 🔒	🛑	
└	contains	Internal 🔒		
└	length	Internal 🔒		
└	at	Internal 🔒		
Context	Implementation			
└	<Constructor>	Internal 🔒	🛑	
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗

└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	geUnlockTime	Public ❗		NO❗
└	lock	Public ❗	🛑	onlyOwner
└	unlock	Public ❗	🛑	NO❗
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
AuthorizedList	Implementation	IAuthorizedList, Context, Ownable		
└	<Constructor>	Public ❗	🛑	Ownable
└	authorizeCaller	External ❗	🛑	onlyOwner
└	renounceAuthorization	External ❗	🛑	authorized
AuthorizedListExt	Implementation	IAuthorizedListExt, AuthorizedList		
└	<Constructor>	Public ❗	🛑	AuthorizedList
└	authorizeByAuthorized	External ❗	🛑	authorized
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
└	min	Internal 🔒		
└	sqrt	Internal 🔒		
UserInfoManager	Implementation	Ownable		
└	logTransfer	Public ❗	🛑	onlyOwner
└	_logTransfer	Internal 🔒	🛑	
LPSwapSupport	Implementation	AuthorizedList		
└	_approve	Internal 🔒	🛑	
└	updateRouter	Public ❗	🛑	authorized
└	updateLiquidityReceiver	External ❗	🛑	onlyOwner
└	updateRouterAndPair	Public ❗	🛑	authorized
└	updateLPPair	Public ❗	🛑	authorized
└	setSwapAndLiquifyEnabled	Public ❗	🛑	authorized
└	swapAndLiquify	Internal 🔒	🛑	
└	swapTokensForCurrency	Internal 🔒	🛑	
└	swapTokensForCurrencyAdv	Internal 🔒	🛑	

```
| └ | addLiquidity | Private 🔐 | 🛑 | |
| └ | swapCurrencyForTokens | Internal 🔒 | 🛑 | |
| └ | swapCurrencyForTokensAdv | Internal 🔒 | 🛑 | |
| └ | updateSwapRange | External ❗ | 🛑 | authorized |
||||||
| **SquidoshiSmartLottery** | Implementation | ISmartLottery, LPSwapSupport, LockableFunction,
AuthorizedListExt, UserInfoManager |||
| └ | <Constructor> | Public ❗ | 🛑 | AuthorizedListExt |
| └ | <Receive Ether> | External ❗ | 💵 |NO❗ |
| └ | deposit | External ❗ | 💵 | onlyOwner |
| └ | rewardCurrency | External ❗ | |NO❗ |
| └ | swap | Internal 🔒 | 🛑 | lockTheSwap |
| └ | setJackpotToCurrency | External ❗ | 🛑 | authorized |
| └ | resetToCurrency | Private 🔐 | 🛑 | lockTheSwap |
| └ | setJackpotToToken | External ❗ | 🛑 | authorized |
| └ | resetToToken | Private 🔐 | 🛑 | lockTheSwap |
| └ | lotteryBalance | Public ❗ | |NO❗ |
| └ | _lotteryBalance | Internal 🔒 | | |
| └ | jackpotAmount | Public ❗ | |NO❗ |
| └ | setJackpot | External ❗ | 🛑 | authorized |
| └ | checkAndPayJackpot | Public ❗ | 🛑 |NO❗ |
| └ | isJackpotReady | External ❗ | |NO❗ |
| └ | _selectAndPayWinner | Private 🔐 | 🛑 | lockFunction |
| └ | reward | Private 🔐 | 🛑 | |
| └ | isEligibleForJackpot | External ❗ | |NO❗ |
| └ | _isEligibleForJackpot | Private 🔐 | | |
| └ | pseudoRand | Private 🔐 | | |
| └ | excludeFromJackpot | Public ❗ | 🛑 | authorized |
| └ | logTransfer | Public ❗ | 🛑 | onlyOwner |
| └ | _logTransfer | Internal 🔒 | 🛑 | |
| └ | _approve | Internal 🔒 | 🛑 | |
| └ | setMaxAttempts | External ❗ | 🛑 | authorized |
| └ | setJackpotEligibilityCriteria | External ❗ | 🛑 | authorized |
```

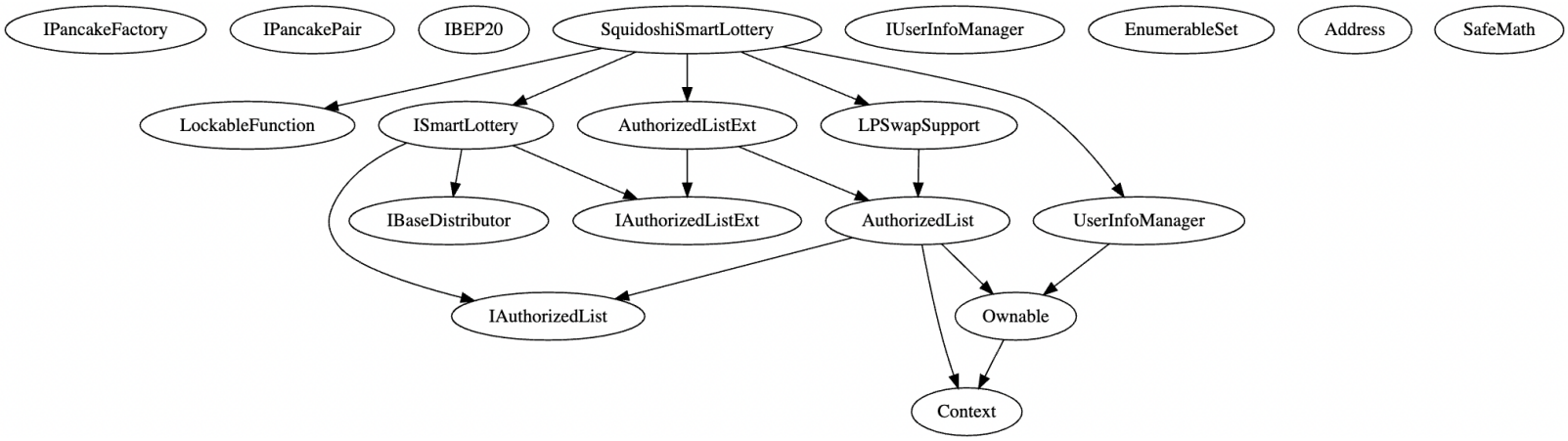
Lottery Smart Contract – Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>  removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec  =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c  =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3
2,bytes32)
ded9382a  =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt
es32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5  =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a  =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5  =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41  =>  swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec  =>  quote(uint256,uint256,uint256)
054d50d4  =>  getAmountOut(uint256,uint256,uint256)
85f8c259  =>  getAmountIn(uint256,uint256,uint256)
d06ca61f  =>  getAmountsOut(uint256,address[])
1f00ca74  =>  getAmountsIn(uint256,address[])
af2979eb  =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984  =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95  =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
```

```
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
a9059cbb  =>  transfer(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
6a627842  =>  mint(address)
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
893d20e8  =>  getOwner()
fcc55e58  =>  authorizeCaller(address,bool)
5528ebd7  =>  authorizeByAuthorized(address)
969cd8fe  =>  logTransfer(address,uint256,address,uint256)
d0e30db0  =>  deposit()
d6c7cc8b  =>  rewardCurrency()
0eecae21  =>  draw()
b1eac37e  =>  jackpotAmount()
d847b8e3  =>  isJackpotReady()
59992cc8  =>  setJackpot(uint256)
3d092c6d  =>  checkAndPayJackpot()
a2340af8  =>  excludeFromJackpot(address,bool)
992bfe27  =>  setMaxAttempts(uint256)
9d24dbd5  =>  setJackpotToCurrency(bool)
688491fc  =>  setJackpotToToken(address,bool)
a55a54e7  =>  setJackpotEligibilityCriteria(uint256,uint256,uint256)
```

Inheritance Graph



InterFi
Smart Contract
Security Audit

Lottery 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	NA	NA
Allowance	returns a set number of tokens from a spender to the owner	NA	NA
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	**Passed**

InterFi Smart Contract Security Audit

Review

- Active smart contract owner: 0x3a7e408b7780774843cd7136bea3eba55a8a1778
- **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.**
- The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```
string private _name = "SquidoshiReflector";

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": " /Squidoshi.sol",

"owner": "_generated_diagnostic_collection_name_#0",

"severity": 8, (! Low Severity)

" Expected token Comma got 'Identifier'

"source": "solc",

}

InterFi

Smart Contract
Security Audit

Lottery 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**

InterFi
Smart Contract
Security Audit

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**

InterFi
Smart Contract
Security Audit

Reflector Smart Contract – Static Analysis

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

```
||||||
| **LockableFunction** | Implementation |  |||
||||||
| **IPancakeRouter01** | 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! |
||||||
| **IPancakeRouter02** | Interface | IPancakeRouter01 |||
| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External ! | 🛑 |NO! |
| └ | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ! | 🛑 |NO! |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! | 🛑 |NO! |
| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ! | 💵 |NO! |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ! | 🛑 |NO! |
||||||
```

IPancakeFactory	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❗
IPancakePair	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❗
IAuthorizedList	Interface			
└	authorizeCaller	External ❗	🛑	NO❗
IAuthorizedListExt	Interface			
└	authorizeByAuthorized	External ❗	🛑	NO❗
IBaseDistributor	Interface			
└	deposit	External ❗	💵	NO❗
└	rewardCurrency	External ❗		NO❗
IBEP20	Interface			

└	totalSupply	External ❗	|NO❗
└	decimals	External ❗	|NO❗
└	symbol	External ❗	|NO❗
└	name	External ❗	|NO❗
└	getOwner	External ❗	|NO❗
└	balanceOf	External ❗	|NO❗
└	transfer	External ❗	🛑 |NO❗
└	allowance	External ❗	|NO❗
└	approve	External ❗	🛑 |NO❗
└	transferFrom	External ❗	🛑 |NO❗
Context	Implementation		
└	<Constructor>	Internal 🔒	🛑	
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context	
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗	|NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	geUnlockTime	Public ❗	|NO❗
└	lock	Public ❗	🛑	onlyOwner
└	unlock	Public ❗	🛑 |NO❗
Address	Library		
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
AuthorizedList	Implementation	IAuthorizedList, Context, Ownable	
└	<Constructor>	Public ❗	🛑	Ownable
└	authorizeCaller	External ❗	🛑	onlyOwner
└	renounceAuthorization	External ❗	🛑	authorized
AuthorizedListExt	Implementation	IAuthorizedListExt, AuthorizedList	
└	<Constructor>	Public ❗	🛑	AuthorizedList
└	authorizeByAuthorized	External ❗	🛑	authorized
ISquidoshiReflector	Interface	IBaseDistributor, IAuthorizedListExt, IAuthorizedList	
└	setDistributionCriteria	External ❗	🛑 |NO❗
└	setShare	External ❗	🛑 |NO❗
└	process	External ❗	🛑 |NO❗
└	setRewardToCurrency	External ❗	🛑 |NO❗
└	setRewardToToken	External ❗	🛑 |NO❗
└	excludeFromReward	External ❗	🛑 |NO❗

└	claimDividendFor	External ❗	🛑	NO❗
SafeMath	Library				
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
LPSwapSupport	Implementation	AuthorizedList			
└	_approve	Internal 🔒	🛑		
└	updateRouter	Public ❗	🛑	authorized
└	updateLiquidityReceiver	External ❗	🛑	onlyOwner
└	updateRouterAndPair	Public ❗	🛑	authorized
└	updateLPPair	Public ❗	🛑	authorized
└	setSwapAndLiquifyEnabled	Public ❗	🛑	authorized
└	swapAndLiquify	Internal 🔒	🛑		
└	swapTokensForCurrency	Internal 🔒	🛑		
└	swapTokensForCurrencyAdv	Internal 🔒	🛑		
└	addLiquidity	Private 🔐	🛑		
└	swapCurrencyForTokens	Internal 🔒	🛑		
└	swapCurrencyForTokensAdv	Internal 🔒	🛑		
└	updateSwapRange	External ❗	🛑	authorized
SquidoshiReflector	Implementation	ISquidoshiReflector, LPSwapSupport, AuthorizedListExt, LockableFunction		
└	<Constructor>	Public ❗	🛑	AuthorizedListExt
└	rewardCurrency	Public ❗		NO❗
└	excludeFromReward	External ❗	🛑	onlyOwner
└	setDistributionCriteria	External ❗	🛑	authorized
└	setShare	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗
└	deposit	External ❗	💵	onlyOwner
└	swap	Private 🔐	🛑	lockTheSwap
└	setRewardToCurrency	External ❗	🛑	authorized
└	resetToCurrency	Private 🔐	🛑	lockTheSwap
└	setRewardToToken	External ❗	🛑	authorized
└	resetToToken	Private 🔐	🛑	lockTheSwap
└	_approve	Internal 🔒	🛑		
└	process	External ❗	🛑	onlyOwner
└	_process	Private 🔐	🛑	lockFunction
└	shouldDistribute	Internal 🔒			
└	distributeDividend	Internal 🔒	🛑		
└	claimDividend	External ❗	🛑	NO❗
└	claimDividendFor	External ❗	🛑	onlyOwner
└	getUnpaidEarnings	Public ❗		NO❗
└	getCumulativeDividends	Internal 🔒			
└	addShareholder	Internal 🔒	🛑		
└	removeShareholder	Internal 🔒	🛑		

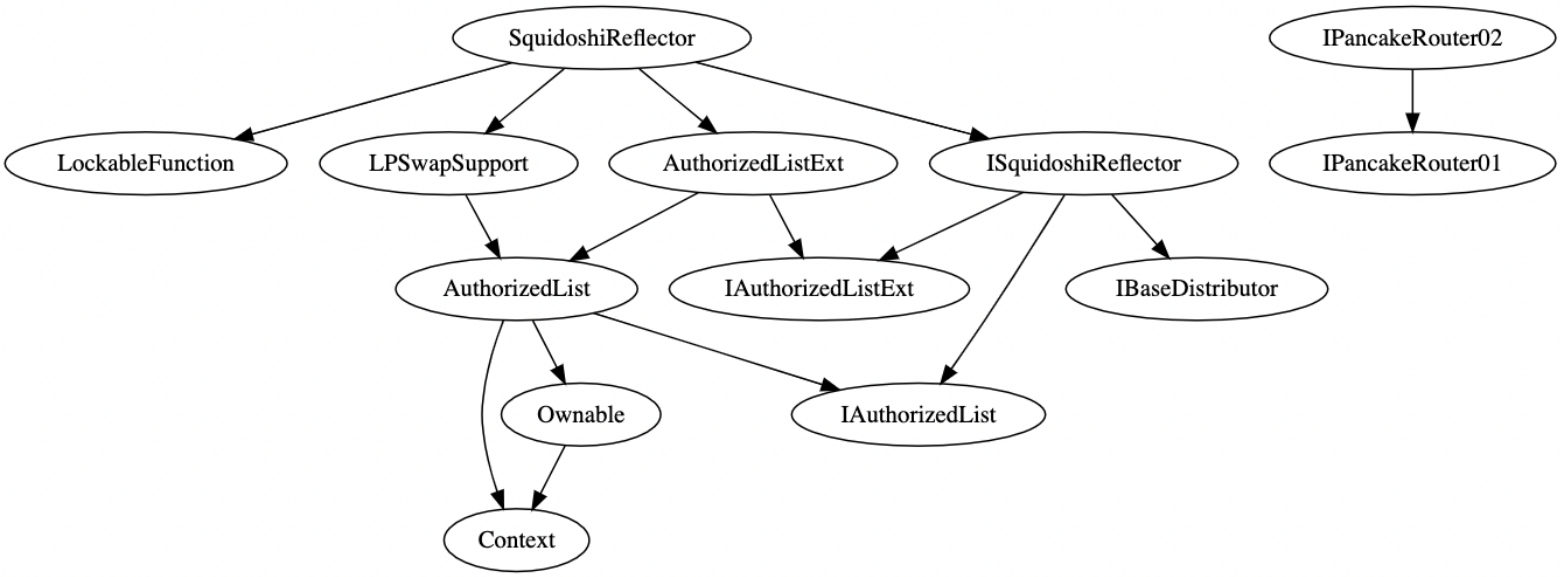
Reflector Smart Contract – Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>  removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec  =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c  =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3
2,bytes32)
ded9382a  =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt
es32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5  =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a  =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5  =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41  =>  swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec  =>  quote(uint256,uint256,uint256)
054d50d4  =>  getAmountOut(uint256,uint256,uint256)
85f8c259  =>  getAmountIn(uint256,uint256,uint256)
d06ca61f  =>  getAmountsOut(uint256,address[])
1f00ca74  =>  getAmountsIn(uint256,address[])
af2979eb  =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984  =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95  =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
```

```
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
a9059cbb  =>  transfer(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
6a627842  =>  mint(address)
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
fcc55e58  =>  authorizeCaller(address,bool)
5528ebd7  =>  authorizeByAuthorized(address)
d0e30db0  =>  deposit()
d6c7cc8b  =>  rewardCurrency()
893d20e8  =>  getOwner()
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
b6c52324  =>  geUnlockTime()
dd467064  =>  lock(uint256)
a69df4b5  =>  unlock()
```

Inheritance Graph



InterFi
Smart Contract
Security Audit

Reflector 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	NA	NA
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	**Passed**

InterFi
Smart Contract
Security Audit

Review

- Active smart contract owner: 0x3a7e408b7780774843cd7136bea3eba55a8a1778
- **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.**
- The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```
string private _name = "SquidoshiReflector";

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": " /Squidoshi.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

Reflector 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	**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**

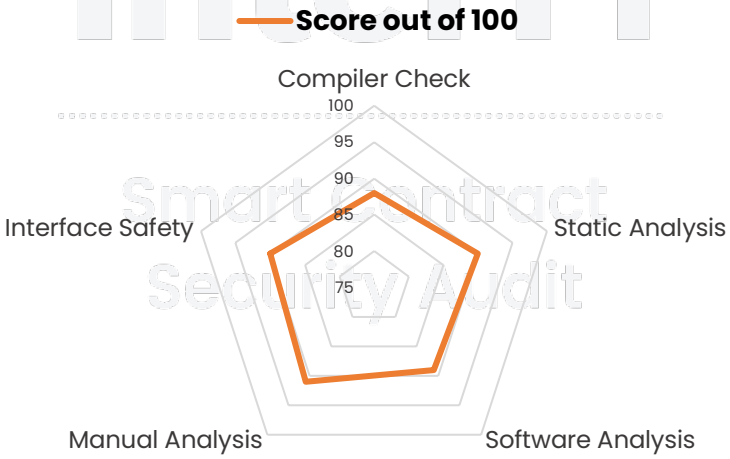
InterFi
Smart Contract
Security Audit

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	**! Low**
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**

InterFi
Smart Contract
Security Audit

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	**2 low severity issue identified**
Passed	**87 functions and instances verified and passed**



Compiler Check	88
Static Analysis	90
Software Analysis	89
Manual Analysis	91
Interface Safety	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.

Squidoshi's Lottery & Reflector smart contract source codes have LOW RISK SEVERITY.

Squidoshi's Lottery & Reflector smart contracts have PASSED the smart contract audit.

InterFi

Smart Contract
Security 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.**

InterFi
Smart Contract
Security Audit

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

InterFi
Smart Contract
Security Audit

@INTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA