SMART CONTRACT SECURITY AUDIT OF

Lisprocoin

Lisprocoin

Audit Introduction

Auditing Firm	Lisprocoin
Audit Architecture	Lisprocoin Auditing Standard
Language	Solidity
Client Firm	Lisprocoin LSP20
Website	https://www.lisprocoin.net
Twitter	https://twitter.com/lisprocoin
Report Date	Dicember 19,2022

About Lisprocoin

Crypto Exchange and swap/ switch Network chain

Audit Summary

Lisprocoin team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ Lisprocoin solidity source code has **LOW RISK SEVERITY** ❖ Lisprocoin smart contract has an **ACTIVE OWNERSHIP** ❖ Important owner privileges –**BLACKLIST, WITHDRAW TO TREASURY**
- ❖ Lisprocoin smart contract owner has multiple "Write Contract" privileges. Centralization risk correlated to the active owner is **MEDIUM**
- ❖ Lisprocoin smart contract utilizes **REBASE.** With rebase, the circulating token supply adjusts (increases or decreases) automatically or manually according to set parameters.

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, exploitability, and audit disclaimer, kindly refer to the audit.

🚨 Contract address: **0x70E546c7a2cA4495cFcbE263a3b6D5ce68B2204C** ⛓ Blockchain: **Polygon Chain**

✅ Verify the authenticity of tisi report on LSP20 GitHub: **https://github.com/15-Lippo/Smart-contract-audit-Lisprocoin**

Table Of Contents

Smart Contract Risk Assessment

Audit Summary

Legal Advisory

Audit Scope

LSP20 was consulted by Lisprocoin to conduct the smart contract security audit of their solidity source codes. The audit scope of work is strictly limited to the mentioned solidity file(s) only:

- LSP20.sol

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

https://polygonscan.com/token/0x70E546c7a2cA4495cFcbE263a3b6D5ce68B2204C

Contract Name: Lisprocoin

Compiler Version: **v0.8.4+commit.c7e474f2**

Optimization Enabled: Yes with 200 runs

Other Settings:

default evmVersion, **GNU GPLv3** license

Solidity Source Code On LSP20 GitHub https://github.com/15-Lippo/smart-contract-audit-lisprocoin

SHA-1 Hash

Solidity source code is audited at hash# `d9db2d3028e7615d9bf7fbbb784e765797ffa9ceb4acf6ec8aadeec2427562c9`

Audit Methodology

The scope of this report is to audit the smart contract source code of Lisprocoin. LSP20 has scanned contracts and reviewed codes for common vulnerabilities, exploits, hacks, and backdoors. Due to being out of scope, LSP20 has not tested contracts on testnet to assess any functional flaws. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	❖ Re-entrancy ❖ Unhandled Exceptions ❖ Transaction Order Dependency ❖ Integer Overflow ❖ Unrestricted Action ❖ Incorrect Inheritance Order ❖ Typographical Errors ❖ Requirement Violation
Source Code Review	❖ Gas Limit and Loops ❖ Deployment Consistency ❖ Repository Consistency ❖ Data Consistency ❖ Token Supply Manipulation ❖ Access Control and Authorization ❖ Operations Trail and Event Generation ❖ Assets Manipulation ❖ Ownership Control ❖ Liquidity Access

LSP20 Echelon Audit Standard

The aim of LSP20" standard is to analyze smart contracts and identify the vulnerabilities and the hacks. Kindly note, LSP20 does not test smart contracts on testnet. It is recommended that smart contracts are thoroughly tested prior to the audit submission. Mentioned are the steps used by LSP20 to audit smart contracts:

1. Solidity smart contract source code reviewal:
   - ❖ Review of the specifications, sources, and instructions provided to LSP20 to make sure we understand the size, and scope of the smart contract audit.
   - ❖ Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.
2. Static, Manual, and Software analysis:
   - ❖ Test coverage analysis is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - ❖ Symbolic execution is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

❖ Consensys Tools ❖ SWC Registry ❖ Solidity Coverage ❖ Open Zeppelin Code Analyzer ❖ Solidity Code Complier

Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/MATIC. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function that requires owning the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! High	This level vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
! Medium	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity
! Low	This level vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Informational	This level vulnerabilities can be ignored. They are code style violations and informational statements in the code. They may not affect the smart contract execution

Centralization Risk

Centralization risk is the most common cause of decentralized finance hacks. When a smart contract has an active contract ownership, the risk related to centralization is elevated. There are some well-intended reasons to be an active contract owner, such as: ❖ Contract owner can be granted the power to `pause()` or `lock()` the contract in case of an external attack.

❖ Contract owner can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale, and to list on an exchange.

Authorizing a full centralized power to a single body can be dangerous. Unfortunately, centralization related risks are higher than common smart contract vulnerabilities. Centralization of ownership creates a risk of rug pull scams, where owners cash out tokens in such quantities that they become valueless. **Most important question to ask here is, how to mitigate centralization risk?** Here's InterFi's recommendation to lower the risks related to centralization hacks:

❖ Smart contract owner's private key must be carefully secured to avoid any potential hack. ❖ Smart contract ownership should be shared by multi-signature (multi-sig) wallets.

❖ Smart contract ownership can be locked in a contract, user voting, or community DAO can be introduced to unlock the ownership.

Lisprocoin Centralization Status ❖ Lisprocoin smart contract has an **active ownership.**

❖ Smart contract ownership is set to.

❖ **0x70E546c7a2cA4495cFcbE263a3b6D5ce68B2204C** at the time of the audit.

Static Analysis

Symbol	Meaning
	Function can modify state
	Function is payable
	Function is locked
	Function can be accessed
	Important functionality

```
| **SafeMathInt** | Library |  |||
| └ | mul | Internal  |   | |
| └ | div | Internal  |   | |
| └ | sub | Internal  |   | |
| └ | add | Internal  |   | |
| └ | abs | Internal  |   | | ||||||
| **SafeMath** | Library |  |||
| └ | add | Internal  |   | |
| └ | sub | Internal  |   | |
| └ | sub | Internal  |   | |
| └ | mul | Internal  |   | |
| └ | div | Internal  |   | |
| └ | div | Internal  |   | |
| └ | mod | Internal  |   | |
||||||
| **ERC20** | Interface |  |||
| └ | totalSupply | External  |   |NO |
| └ | balanceOf | External  |   |NO |
| └ | allowance | External  |   |NO |
| └ | transfer | External  |   |NO |
| └ | approve | External  |   |NO |
| └ | transferFrom | External  |   |NO |
||||||
| **QuickSwapPair** | 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 | ||||||
| **QuickSwapRouter** | 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
└	removeLiquidityETHSupportingFeeOnTransferTokens	External		|NO
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External		|NO
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External		|NO
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External		|NO	
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External		NO
QuickSwapFactory	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
Ownable	Implementation			
└	<Constructor>	Public		|NO
└	owner	Public		NO
└	isOwner	Public		NO
└	renounceOwnership	Public			onlyOwner
└	transferOwnership	Public			onlyOwner
└	_transferOwnership	Internal			
ERC20Detailed	Implementation	IERC20		
└	<Constructor>	Public		|NO
└	name	Public	|NO	
└	symbol	Public	|NO	
└	decimals	Public		NO
ATH	Implementation	ERC20Detailed, Ownable		
└	<Constructor>	Public			ERC20Detailed Ownable
└	startRebase	External			onlyOwner
└	rebase	Internal			
└	transfer	External			validRecipient
└	transferFrom	External			validRecipient
└	_basicTransfer	Internal			
└	_transferFrom	Internal			
└	takeFee	Internal			
└	addLiquidity	Internal			swapping
└	swapBack	Internal			swapping

| └

└	withdrawAllToTreasury	External		swapping onlyOwner
└	shouldTakeFee	Internal		
└	shouldRebase	Internal		
└	shouldAddLiquidity	Internal		
└	shouldSwapBack	Internal		
└setAutoRebase	External		onlyOwner
└	External		
setAutoAddLiquidity onlyOwner			NO
└	allowance	External
└	decreaseAllowance	External		NO
└	increaseAllowance	External		NO
└	approve	External		NO
└	checkFeeExempt	External		NO
└	getCirculatingSupply	Public		NO
└	isNotInSwap	External		NO
└	manualSync	External		NO
└	setFeeReceivers	External		onlyOwner
└	getLiquidityBacking	External		NO
setWhitelist	External		onlyOwner
└	setBotBlacklist	External		onlyOwner
└	setPairAddress	External		onlyOwner
└	setLP	External		onlyOwner
└	totalSupply	External		NO
└	balanceOf	External		NO
└	isContract	Internal		
└	<Receive Ether>	External		NO

| L

| L

Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
43509138  =>  div(int256,int256) bbe93d91
=>  mul(int256,int256) adefc37b  =>
sub(int256,int256) a5f3c23b  =>
add(int256,int256) 1b5ac4b5  =>
abs(int256) 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) 18160ddd  =>
totalSupply() 70a08231  =>
balanceOf(address) dd62ed3e  =>
allowance(address,address) a9059cbb  =>
transfer(address,uint256)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
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)
017e7e58  =>  feeTo() 094b7415
=>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256) 574f2ba3
=>  allPairsLength() c9c65396  =>
createPair(address,address) f46901ed  =>
setFeeTo(address) a2e74af6  =>
setFeeToSetter(address)
8da5cb5b  =>  owner()
8f32d59b  =>  isOwner()
715018a6  =>  renounceOwnership() f2fde38b
=>  transferOwnership(address) d29d44ee
=>  _transferOwnership(address)
706f86e9  =>  startRebase() af14052c
=>  rebase()
f0774e71  =>  _basicTransfer(address,address,uint256)
cb712535  =>  _transferFrom(address,address,uint256)
20cb7bce  =>  takeFee(address,address,uint256)
e8078d94  =>  addLiquidity() 6ac5eeee  =>  swapBack()
bd595581  =>  withdrawAllToTreasury() 332402f8
=>  shouldTakeFee(address,address) 63eab10a
=>  shouldRebase() ee0c53c4  =>
shouldAddLiquidity() 0d5c6cea  =>
shouldSwapBack() e15beb80  =>
setAutoRebase(bool) cfbac92f  =>
setAutoAddLiquidity(bool) a457c2d7  =>
decreaseAllowance(address,uint256) d4399790
=>  checkFeeExempt(address)
2b112e49  =>  getCirculatingSupply()
83b4ac68  =>  isNotInSwap()
753d02a1  =>  manualSync()
3c8e556d  =>  setFeeReceivers(address,address,address,address)
d51ed1c8  =>  getLiquidityBacking(uint256) 854cff2f  =>
setWhitelist(address) 37c9be87  =>
```

```
setBotBlacklist(address,bool) a22d4832  =>
setPairAddress(address)
2f34d282  =>  setLP(address)
```

Manual Analysis

Function	Description	Available	Status
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
Rebase	circulating token supply adjusts (increases or decreases) automatically according to a token's price fluctuations	Yes	Passed

Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	**! Low**
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**

Notable Information 📝

❖ Smart contract utilizes **SafeMath** function to avoid common smart contract vulnerabilities.

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

❖ Smart contract owner can **blacklist** certain wallets from interacting with the contract function modules.

```
    function setBotBlacklist(address _botAddress, bool _flag)
require( isContract(_botAddress),
        "only contract address, not allowed exteranlly owned account"
```

- Lisprocoin smart contract utilizes **rebase.** With rebase, the circulating token supply adjusts (increases or decreases) automatically or manually according to set parameters. ❖ Smart contract owner can **withdraw $LSP20 tokens** from the token contract to treasury.

```
    function withdrawAllToTreasury() external swapping onlyOwner {
uint256 amountToSwap = _gonBalances[address(this)].div(
```

- Smart contract has a **low severity issue** which may or may not create any functional vulnerability.

  "severity": 8, (! Low Severity)

  "Expected token Comma got 'Identifier'"

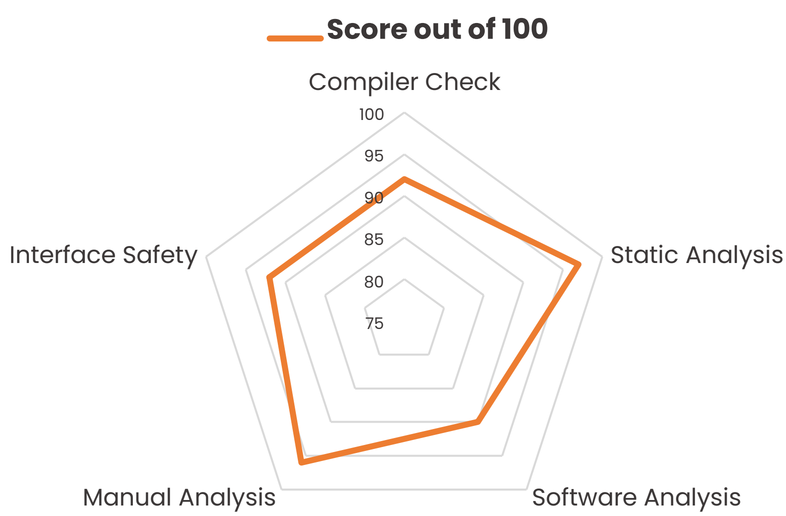
SWC Attacks

SWC ID	Description	Status
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Informational
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	! Low
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

SWC-119	Shadowing State Variables	**Passed**
SWC-120	Weak Sources of Randomness from Chain Attributes	**Passed**
SWC-121	Missing Protection against Signature Replay Attacks	**Passed**
SWC-122	Lack of Proper Signature Verification	**Passed**
SWC-123	Requirement Violation	**Passed**
SWC-124	Write to Arbitrary Storage Location	**Passed**
SWC-125	Incorrect Inheritance Order	**Passed**
SWC-126	Insufficient Gas Griefing	**Passed**
SWC-127	Arbitrary Jump with Function Type Variable	**Passed**
SWC-128	DoS With Block Gas Limit	**Passed**
SWC-129	Typographical Error	**Passed**
SWC-130	Right-To-Left-Override control character (U+202E)	**Passed**
SWC-131	Presence of unused variables	**Passed**
SWC-132	Unexpected Ether balance	**Passed**
SWC-133	Hash Collisions With Multiple Variable Length Arguments	**Passed**
SWC-134	Message call with the hardcoded gas amount	**Passed**
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Passed**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

Risk Status & Radar Chart

Risk Severity	Status
High	No high severity issues identified
Medium	No medium severity issues identified
Low	2 low severity issues identified
Informational	1 informational severity issue identified
Centralization Risk	Active contract ownership identified



Auditor's Verdict

LSP20 team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

- ❖ Lisprocoin smart contract source code has **LOW RISK SEVERITY** ❖ Lisprocoin smart contract has an **ACTIVE OWNERSHIP**
- ❖ Lisprocoin centralization risk correlated to the active owner is **MEDIUM**

Note for stakeholders

- ❖ Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security.
- ❖ If the smart contract is not deployed on any blockchain at the time of the audit, the contract can be modified or altered before blockchain development. Verify contract's deployment status in the audit report.
- ❖ Make sure that the project team's KYC/identity is verified by an independent firm.
- ❖ Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers. ❖ Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the

project’s tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period.

Important Disclaimer

LSP20 provides contract development, testing, auditing and project evaluation services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purpose without InterFi’s prior written consent.**

LSP20 provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren’t resistant to external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on smart contract safety and security. Therefore, LSP20 not guarantee the explicit security of the audited smart contract.**

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report’s content as such. Do conduct your due diligence and consult your financial advisor before making any investment decisions.

About LSP20

LSP20 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. **LSP20 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.**

LSP20 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. **LSP20 provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**