Full Audit Report

Ajira Pay Security Assessment

Audit Report

SECURI LAB contact@securi-lab.com

FULL AUDIT REPORT

Table of Contents

FULL AUDIT REPORT

Report Information

About Report	**Ajira Pay Security Assessment**
Version	**v1.1**
Client	**Ajira Pay**
Language	**Solidity**
Confidentiality	**Public**
Contract Address	0xC55b03dC07EC7Bb8B891100E927E982540f0d181
Audit Method	**Whitebox**
Security Assessment Author	**Auditor** SECURI LAB **Mark K. [Security Researcher | Redteam]** **Kevin N. [Security Researcher | Web3 Dev]** **Yusheng T. [Security Researcher | Incident Response]** **Approve Document** **Ronny C. CTO & Head of Security Researcher**

*Audit Method

Whitebox: Securi Team receives all source code from the client to provide the assessment.
Blackbox: Securi Team receives only bytecode from the client to provide the assessment.

Digital Sign (Only Full Audit Report)

Digitally signed by Chinnakit J.
DN: cn=Chinnakit J., o=SECURI LAB (Thailand), ou=CEO & Founder, email=contact@securi-lab.com, c=TH
Date: 2023.02.27 14:42:46 +07'00'

FULL AUDIT REPORT

Disclaimer

Regarding this security assessment, there are no guarantees about the security of the program instruction received from the client is hereinafter referred to as “**Source code**”.
And **SECURI Lab** hereinafter referred to as “**Service Provider**”, the **Service Provider** will not be held liable for any legal liability arising from errors in the security assessment. The responsibility will be the responsibility of the **Client**, hereinafter referred to as “**Service User**” and the **Service User** agrees not to be held liable to the **service provider** in any case. By contract **Service Provider** to conduct security assessments with integrity with professional ethics, and transparency to deliver security assessments to users The **Service Provider** has the right to postpone the delivery of the security assessment. If the security assessment is delayed whether caused by any reason and is not responsible for any delayed security assessments.
If **the service provider** finds a vulnerability The **service provider** will notify the **service user** via the Preliminary Report, which will be kept confidential for security. The **service provider** disclaims responsibility in the event of any attacks occurring whether before conducting a security assessment. Or happened later All responsibility shall be sole with the **service user**.

Security Assessment Not Financial/Investment Advice Any loss arising from any investment in any project is the responsibility of the investor.

SECURI disclaims any liability incurred. Whether it's Rugpull, Abandonment, Soft Rugpull

The SECURI LAB team has conducted a comprehensive security assessment of the vulnerabilities.
This assessment is tested with an expert assessment. Using the following test requirements

1. Smart Contract Testing with Expert Analysis By testing the most common and uncommon vulnerabilities.
2. Automated program testing It includes a sample vulnerability test and a sample of the potential vulnerabilities being used for the most frequent attacks.
3. Visibility, Mutability, Modifier function testing, such as whether a function can be seen in general, or whether a function can be changed and if so, who can change it.
4. Function association test It will be displayed through the association graph.
5. This safety assessment is cross-checked prior to the delivery of the assessment results.

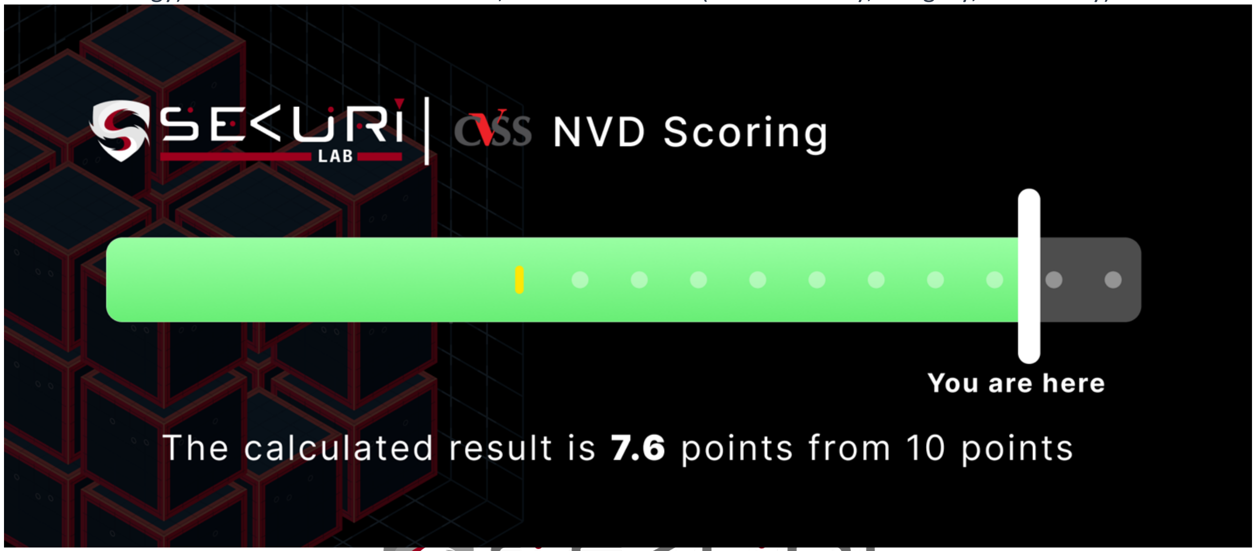
FULL AUDIT REPORT

Executive Summary

For this security assessment, SECURI LAB received a request from Ajira Pay on Saturday, February 25, 2023.

NVD CVSS Scoring

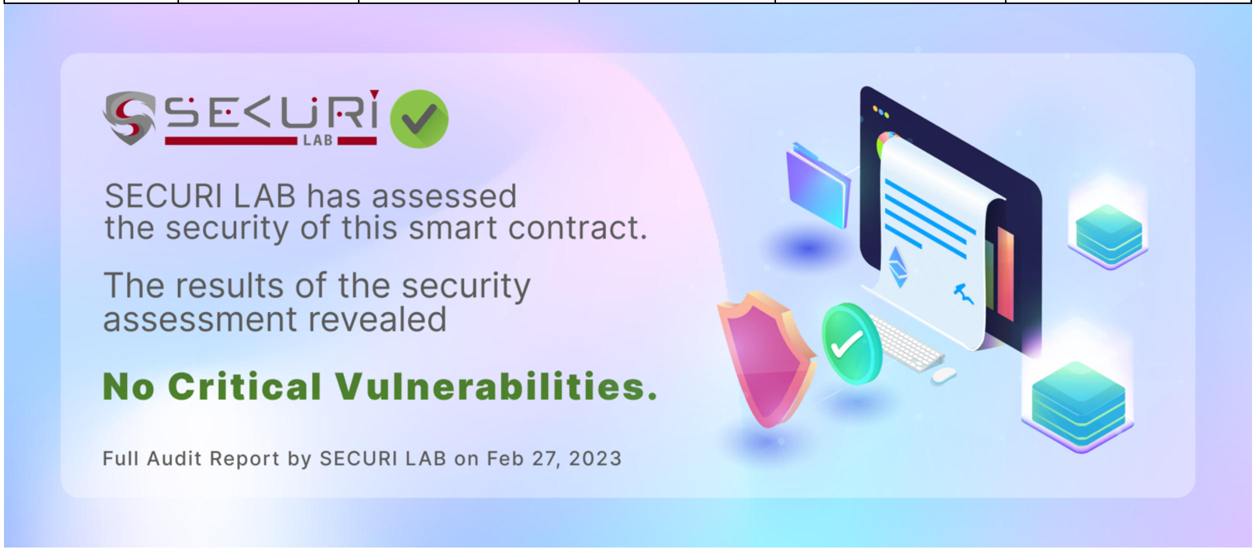
The score was calculated using the NVD (National Vulnerability Database) of NIST (National Institute of Standards and Technology) under the CVSS 3.1 standard, based on the CIA (Confidentiality, Integrity, Availability).



Audit Result

SECURI LAB evaluated the smart contract security of the project and found: [Total : 6]

Critical	High	Medium	Low	Very Low	Informational
0	1	1	0	0	4


SECURI LAB
SECURI LAB has assessed the security of this smart contract.
The results of the security assessment revealed
No Critical Vulnerabilities.
Full Audit Report by SECURI LAB on Feb 27, 2023

FULL AUDIT REPORT

Project Introduction

Scope Information:

Project Name	**Ajira Pay**
Website	**https://ajirapay.finance/**
Chain	**BNB Chain (Previously Binance Smart Chain)**
Language	**Solidity**

Audit Information:

Request Date	**Saturday, February 25, 2023**
Audit Date	**Sunday, February 26, 2023**
Re-assessment Date	-

Audit Version History:

Version	Date	Description
1.0	**Monday, February 27, 2023**	**Preliminary Report**
1.1	**Monday, February 27, 2023**	**Full Audit Report**

FULL AUDIT REPORT

Initial Audit Scope:

Smart Contract	0xC55b03dC07EC7Bb8B891100E927E982540f0d181
Compiler Version	**v0.8.4+commit.c7e474f2**

FULL AUDIT REPORT

Security Assessment Procedure

Securi has the following procedures and regulations for conducting security assessments:

1.Request Audit Client submits a form request through the Securi channel. After receiving the request, Securi will discuss a security assessment. And drafting a contract and agreeing to sign a contract together with the Client

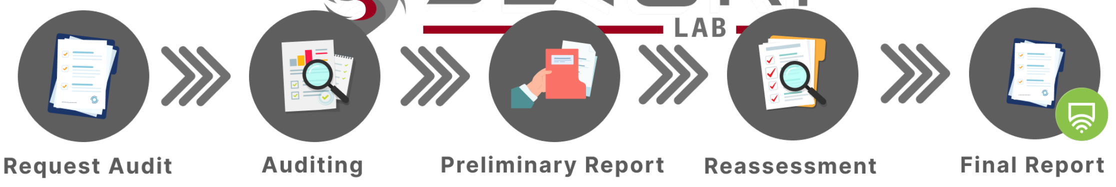
2.Auditing Securi performs security assessments of smart contracts obtained through automated analysis and expert manual audits.

3.Preliminary Report At this stage, Securi will deliver an initial security assessment. To report on vulnerabilities and errors found under Audit Scope <u>will not publish preliminary reports for safety</u>.

4.Reassessment After Securi has delivered the Preliminary Report to the Client, Securi will track the status of the vulnerability or error, which will be published to the Final Report at a later date with the following statuses:

- **a.Acknowledge** The client has been informed about errors or vulnerabilities from the security assessment.
- **b.Resolved** The client has resolved the error or vulnerability. Resolved is probably just a commit, and Securi is unable to verify that the resolved has been implemented or not.
- **c.Decline** Client has rejected the results of the security assessment on the issue.

5.Final Report Securi providing full security assessment report and public



Risk Rating

Risk rating using this commonly defined: $Risk\ rating = impact * confidence$

Impact	The severity and potential impact of an attacker attack
Confidence	Ensuring that attackers expose and use this vulnerability

Both have a total of 3 levels: **High**, **Medium**, **Low**. By *Informational* will not be classified as a level

Confidence / Impact [Likelihood]	Low	Medium	High
Low	**Very Low**	**Low**	**Medium**
Medium	**Low**	**Medium**	**High**
High	**Medium**	**High**	**Critical**

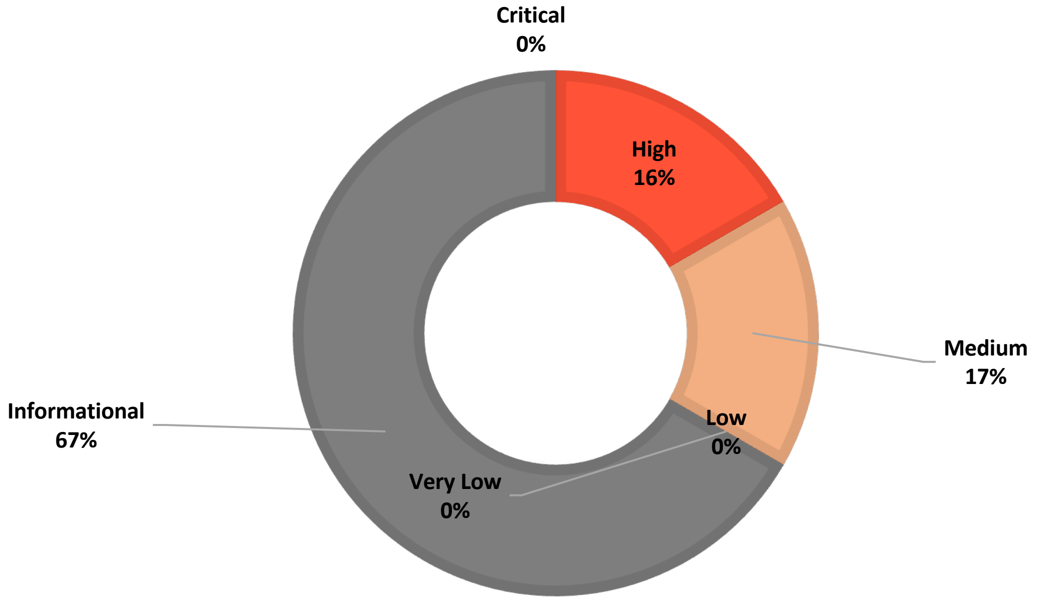
Severity is a risk assessment It is calculated from the Impact and Confidence values using the following calculation methods, $Risk\ rating = impact * confidence$ It is categorized into **5 categories based** on the **lowest severity**: Very Low, Low, Medium, **High**, Critical.
For **Informational** will not be counted as **severity**

FULL AUDIT REPORT

Vulnerability Severity Summary

Vulnerability Severity Level	Total
Critical	**0**
High	**1**
Medium	**1**
Low	**0**
Very Low	**0**
Informational (Non severity level)	**4**

VULNERABILITY SERVERITY PIE CHART



FULL AUDIT REPORT

Vulnerability Findings

ID	Title	Severity	Status
SEC-01	Owner can change user balance by using burn function	High	**Acknowledge**
SEC-02	Imprecise arithmetic operations order (divide-before-multiply)	Medium	**Acknowledge**
SEC-03	Unused state variables (unused-state)	Informational	**Acknowledge**
SEC-04	Comparison to boolean constant (boolean-equal)	Informational	**Acknowledge**
SEC-05	If different pragma directives are used (pragma)	Informational	**Acknowledge**
SEC-06	Conformity to Solidity naming conventions (naming-convention)	Informational	**Acknowledge**

FULL AUDIT REPORT

SEC-01: Owner can change user balance by using burn function

Type	Severity	Location	Status
Owner can change user balance by using burn function	High	Check on finding	**Acknowledge**

Finding:

❌ (AjiraPayFinanceToken.sol#373-382)

Exploit Scenario:

The contract owner has the authority to modify the balance of tokens at other addresses, which may result in a loss of assets.

Alleviation:

Ajira Pay team has Acknowledge this issue.

SEC-02: Imprecise arithmetic operations order (divide-before-multiply)

Type	Severity	Location	Status
Imprecise arithmetic operations order (divide-before-multiply)	Medium	Check on finding	**Acknowledge**

Finding:

❌ AjiraPayFinanceToken._swapAndLiquify(uint256) (AjiraPayFinanceToken.sol:432-459) performs a multiplication on the result of a division:

- buyBackTreasuryAmount = leftOverBnb / totalTreasury * buyBackTreasuryPercent (AjiraPayFinanceToken.sol#447)

❌ AjiraPayFinanceToken._swapAndLiquify(uint256) (AjiraPayFinanceToken.sol:432-459) performs a multiplication on the result of a division:

- liquidityTreasuryAmount = leftOverBnb / totalTreasury * liquidityTreasuryPercent (AjiraPayFinanceToken.sol#448)

Recommendation:

Consider ordering multiplication before division.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

FULL AUDIT REPORT

Exploit Scenario:

Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.

```
contract A {
          function f(uint n) public {
      coins = (oldSupply / n) * interest;
   }
}
```

If n is greater than oldSupply, coins will be zero. For example, with oldSupply = 5; n = 10, interest = 2, coins will be zero.
If (oldSupply * interest / n) was used, coins would have been 1.
In general, it's usually a good idea to re-arrange arithmetic to perform multiplication before division, unless the limit of a smaller type makes this dangerous.

Alleviation:

Ajira Pay team has Acknowledge this issue.

SEC-03: Unused state variables (unused-state)

Type	Severity	Location	Status
Unused state variables (unused-state)	Informational	Check on finding	**Acknowledge**

Finding:

❌ AjiraPayFinanceToken._allowances (AjiraPayFinanceToken.sol:219) is never used in AjiraPayFinanceToken (AjiraPayFinanceToken.sol#199-538)
❌ AjiraPayFinanceToken.devTreasuryPercent (AjiraPayFinanceToken.sol:229) is never used in AjiraPayFinanceToken (AjiraPayFinanceToken.sol#199-538)

Recommendation:

Remove unused state variables.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Exploit Scenario:

-

Alleviation:

Ajira Pay team has Acknowledge this issue.

SEC-04: Comparison to boolean constant (boolean-equal)

Type	Severity	Location	Status
Comparison to boolean constant (boolean-equal)	Informational	Check on finding	**Acknowledge**

Finding:

❌ AjiraPayFinanceToken._transfer(address,address,uint256) (AjiraPayFinanceToken.sol:387-419) compares to a boolean constant:
- _isExcludedFromFee[_sender] == true (AjiraPayFinanceToken.sol#415)

❌ AjiraPayFinanceToken._transfer(address,address,uint256) (AjiraPayFinanceToken.sol:387-419) compares to a boolean constant:
- isInTaxHoliday == true (AjiraPayFinanceToken.sol#416)

Recommendation:

Remove the equality to the boolean constant.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

FULL AUDIT REPORT

Exploit Scenario:

Detects the comparison to boolean constants.

```
contract A {
          function f(bool x) public {
                    // ...
          if (x == true) { // bad!
        // ...
      }
                    // ...
      }
}
```

Boolean constants can be used directly and do not need to be compare to true or false.

Alleviation:

Ajira Pay team has Acknowledge this issue.

SEC-05: If different pragma directives are used (pragma)

Type	Severity	Location	Status
If different pragma directives are used (pragma)	Informational	Check on finding	**Acknowledge**

Finding:

❌ Different versions of Solidity are used:

- Version used: ['=0.8.4', '^0.8.0', '^0.8.1']
- =0.8.4 (AjiraPayFinanceToken.sol:2)
- ^0.8.0 (@openzeppelin/contracts/access/AccessControl.sol#4)
- ^0.8.0 (@openzeppelin/contracts/access/IAccessControl.sol#4)
- ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4)
- ^0.8.0 (@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
- ^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
- ^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- ^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
- ^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/Strings.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/math/Math.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/math/SignedMath.sol#4)
- ^0.8.0 (erc-payable-token/contracts/token/ERC1363/ERC1363.sol#3)
- ^0.8.0 (erc-payable-token/contracts/token/ERC1363/IERC1363.sol#3)
- ^0.8.0 (erc-payable-token/contracts/token/ERC1363/IERC1363Receiver.sol#3)
- ^0.8.0 (erc-payable-token/contracts/token/ERC1363/IERC1363Spender.sol#3)
- ^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4)

Recommendation:

Use one Solidity version.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Exploit Scenario:

-

Alleviation:

Ajira Pay team has Acknowledge this issue.

SEC-06: Conformity to Solidity naming conventions (naming-convention)

Type	Severity	Location	Status
Conformity to Solidity naming conventions (naming-convention)	Informational	Check on finding	**Acknowledge**

Finding:

❌ Parameter AjiraPayFinanceToken.burn(address,uint256)._account (AjiraPayFinanceToken.sol:373) is not in mixedCase
Finding:
❌ Parameter AjiraPayFinanceToken.burn(address,uint256)._amount (AjiraPayFinanceToken.sol:373) is not in mixedCase
❌ Parameter AjiraPayFinanceToken.setBuyBackEnabled(bool)._enabled (AjiraPayFinanceToken.sol:364) is not in mixedCase
❌ Variable AjiraPayFinanceToken.DEAD (AjiraPayFinanceToken.sol:209) is not in mixedCase
❌ Variable AjiraPayFinanceToken._isExcludedFromFee (AjiraPayFinanceToken.sol:217) is not in mixedCase

Recommendation:

Follow the Solidity [naming convention](https://solidity.readthedocs.io/en/v0.4.25/style-guide.html#naming-conventions).

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Exploit Scenario:

Solidity defines a naming convention that should be followed.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (ERC20).
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

Follow the Solidity naming convention.

Alleviation:

Ajira Pay team has Acknowledge this issue.

SWC Findings

ID	Title	Scanning	Result
SWC-100	Function Default Visibility	Complete	No risk
SWC-101	Integer Overflow and Underflow	Complete	No risk
SWC-102	Outdated Compiler Version	Complete	No risk
SWC-103	Floating Pragma	Complete	No risk
SWC-104	Unchecked Call Return Value	Complete	No risk
SWC-105	Unprotected Ether Withdrawal	Complete	No risk
SWC-106	Unprotected SELFDESTRUCT Instruction	Complete	No risk
SWC-107	Reentrancy	Complete	No risk
SWC-108	State Variable Default Visibility	Complete	No risk
SWC-109	Uninitialized Storage Pointer	Complete	No risk
SWC-110	Assert Violation	Complete	No risk
SWC-111	Use of Deprecated Solidity Functions	Complete	No risk
SWC-112	Delegatecall to Untrusted Callee	Complete	No risk
SWC-113	DoS with Failed Call	Complete	No risk

FULL AUDIT REPORT

SWC-114	Transaction Order Dependence	Complete	No risk
SWC-115	Authorization through tx.origin	Complete	No risk
SWC-116	Block values as a proxy for time	Complete	No risk
SWC-117	Signature Malleability	Complete	No risk
SWC-118	Incorrect Constructor Name	Complete	No risk
SWC-119	Shadowing State Variables	Complete	No risk
SWC-120	Weak Sources of Randomness from Chain Attributes	Complete	No risk
SWC-121	Missing Protection against Signature Replay Attacks	Complete	No risk
SWC-122	Lack of Proper Signature Verification	Complete	No risk
SWC-123	Requirement Violation	Complete	No risk
SWC-124	Write to Arbitrary Storage Location	Complete	No risk
SWC-125	Incorrect Inheritance Order	Complete	No risk
SWC-126	Insufficient Gas Griefing	Complete	No risk
SWC-127	Arbitrary Jump with Function Type Variable	Complete	No risk
SWC-128	DoS With Block Gas Limit	Complete	No risk

FULL AUDIT REPORT

SWC-129	Typographical Error	Complete	No risk
SWC-130	Right-To-Left-Override control character (U+202E)	Complete	No risk
SWC-131	Presence of unused variables	Complete	No risk
SWC-132	Unexpected Ether balance	Complete	No risk
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Complete	No risk
SWC-134	Message call with hardcoded gas amount	Complete	No risk
SWC-135	Code With No Effects	Complete	No risk
SWC-136	Unencrypted Private Data On-Chain	Complete	No risk

Visibility, Mutability, Modifier function testing

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
1	0	4	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
78	5

External	Internal	Private	Pure	View
63	51	10	10	25

StateVariables

Total	🌐Public
25	15

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`=0.8.4`		`yes`		

FULL AUDIT REPORT

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
yes			yes		

♻️ TryCatch	Σ Unchecked
	yes

FULL AUDIT REPORT

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IPancakeswapV2Factory	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❗
IPancakeSwapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗

FULL AUDIT REPORT

Contract	Type	Bases		
└	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 ❗

FULL AUDIT REPORT

Contract	Type	Bases		
└	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❗
IPancakeRouter01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗

FULL AUDIT REPORT

Contract	Type	Bases		
└	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❗

FULL AUDIT REPORT

Contract	Type	Bases		
└	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❗
AjiraPayFinanceToken	Implementation	Ownable, ERC1363, AccessControl, ReentrancyGuard		
└		Public ❗	🛑	ERC20
└	balanceOf	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	supportsInterface	Public ❗		NO❗

FULL AUDIT REPORT

Contract	Type	Bases		
└	recoverBNB	Public ❗️	🛑	onlyRole nonReentrant
└	recoverLostTokensForInvestor	Public ❗️	🛑	onlyRole
└	updateTreasury	Public ❗️	🛑	onlyRole
└	updateRouterAddress	External ❗️	🛑	onlyRole
└	setDeductionFeePercentages	Public ❗️	🛑	onlyRole
└	setTreasuryPercentages	Public ❗️	🛑	onlyRole
└	setSwapAndLiquifyEnabled	External ❗️	🛑	onlyRole
└	excludeFromFee	Public ❗️	🛑	onlyRole
└	includeInFee	Public ❗️	🛑	onlyRole
└	setMaxTransactionAmount	External ❗️	🛑	onlyRole
└	setTaxHolidayEnabled	Public ❗️	🛑	onlyRole
└	setTransferFeeEnabled	Public ❗️	🛑	onlyRole
└	setBuyBackEnabled	Public ❗️	🛑	onlyRole
└	updateMinTokensToLiquify	Public ❗️	🛑	onlyRole

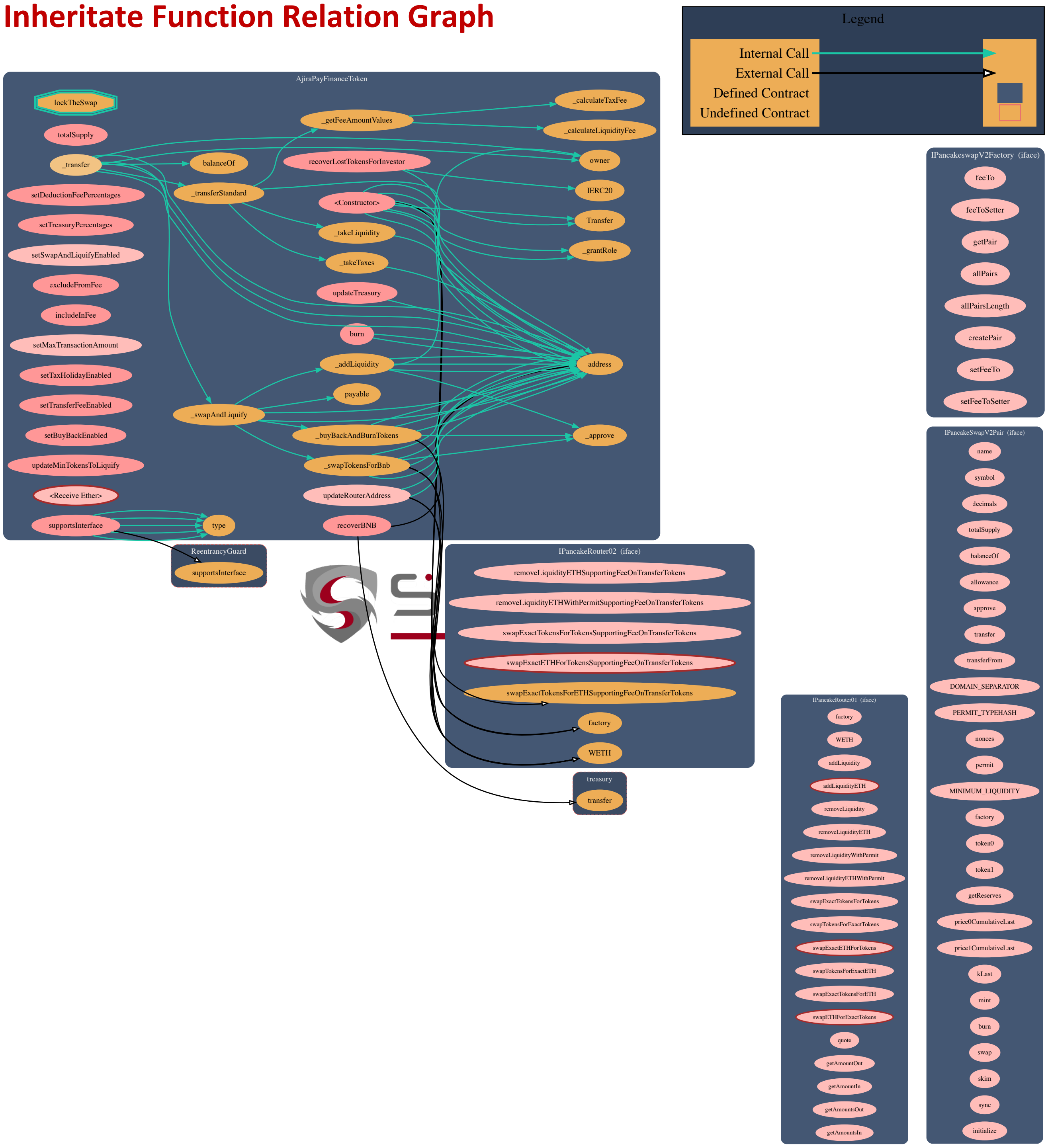
FULL AUDIT REPORT

Contract	Type	Bases		
				nonReentrant
└	burn	Public ❗	🛑	onlyRole
└		External ❗	💵	NO❗
└	_transfer	Internal 🔒	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	_swapTokensForBnb	Private 🔐	🛑	
└	_addLiquidity	Private 🔐	🛑	
└	_buyBackAndBurnTokens	Private 🔐	🛑	
└	_calculateLiquidityFee	Private 🔐		
└	_calculateTaxFee	Private 🔐		
└	_getFeeAmountValues	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	_takeTaxes	Private 🔐	🛑	

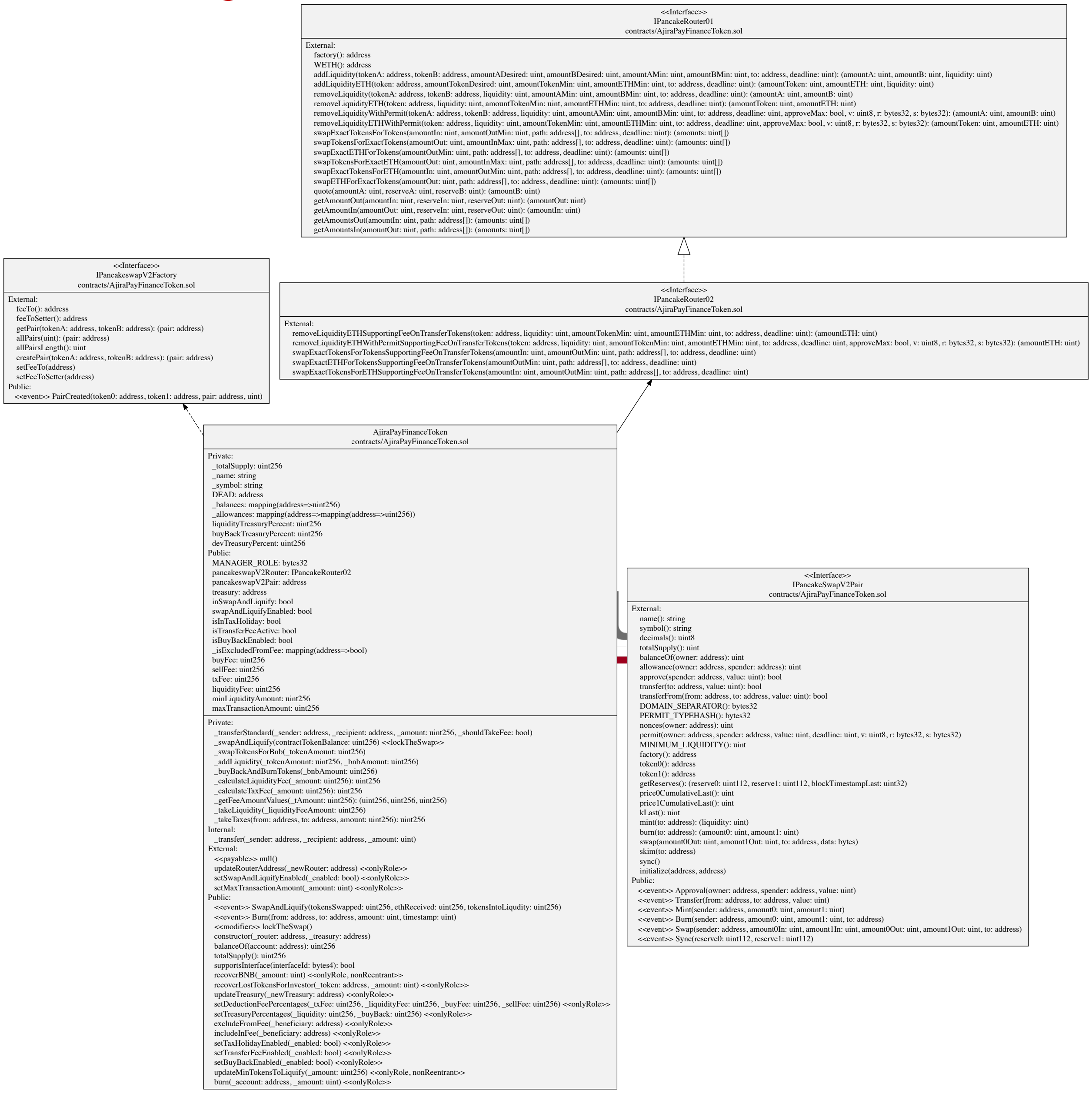
Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritate Function Relation Graph



UML Class Diagram



```
<<Interface>>
IPancakeRouter01
contracts/AjiraPayFinanceToken.sol

External:
   factory(): address
   WETH(): address
   addLiquidity(tokenA: address, tokenB: address, amountADesired: uint, amountBDesired: uint, amountAMin: uint, amountBMin: uint, to: address, deadline: uint): (amountA: uint, amountB: uint, liquidity: uint)
   addLiquidityETH(token: address, amountTokenDesired: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint): (amountToken: uint, amountETH: uint, liquidity: uint)
   removeLiquidity(tokenA: address, tokenB: address, liquidity: uint, amountAMin: uint, amountBMin: uint, to: address, deadline: uint): (amountA: uint, amountB: uint)
   removeLiquidityETH(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint): (amountToken: uint, amountETH: uint)
   removeLiquidityWithPermit(tokenA: address, tokenB: address, liquidity: uint, amountAMin: uint, amountBMin: uint, to: address, deadline: uint, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountA: uint, amountB: uint)
   removeLiquidityETHWithPermit(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountToken: uint, amountETH: uint)
   swapExactTokensForTokens(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
   swapTokensForExactTokens(amountOut: uint, amountInMax: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
   swapExactETHForTokens(amountOutMin: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
   swapTokensForExactETH(amountOut: uint, amountInMax: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
   swapExactTokensForETH(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
   swapETHForExactTokens(amountOut: uint, path: address[], to: address, deadline: uint): (amounts: uint[])
   quote(amountA: uint, reserveA: uint, reserveB: uint): (amountB: uint)
   getAmountOut(amountIn: uint, reserveIn: uint, reserveOut: uint): (amountOut: uint)
   getAmountIn(amountOut: uint, reserveIn: uint, reserveOut: uint): (amountIn: uint)
   getAmountsOut(amountIn: uint, path: address[]): (amounts: uint[])
   getAmountsIn(amountOut: uint, path: address[]): (amounts: uint[])
```

```
<<Interface>>
IPancakeswapV2Factory
contracts/AjiraPayFinanceToken.sol

External:
   feeTo(): address
   feeToSetter(): address
   getPair(tokenA: address, tokenB: address): (pair: address)
   allPairs(uint): (pair: address)
   allPairsLength(): uint
   createPair(tokenA: address, tokenB: address): (pair: address)
   setFeeTo(address)
   setFeeToSetter(address)
Public:
  <<event>> PairCreated(token0: address, token1: address, pair: address, uint)
```

```
<<Interface>>
IPancakeRouter02
contracts/AjiraPayFinanceToken.sol

External:
   removeLiquidityETHSupportingFeeOnTransferTokens(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint): (amountETH: uint)
   removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(token: address, liquidity: uint, amountTokenMin: uint, amountETHMin: uint, to: address, deadline: uint, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountETH: uint)
   swapExactTokensForTokensSupportingFeeOnTransferTokens(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint)
   swapExactETHForTokensSupportingFeeOnTransferTokens(amountOutMin: uint, path: address[], to: address, deadline: uint)
   swapExactTokensForETHSupportingFeeOnTransferTokens(amountIn: uint, amountOutMin: uint, path: address[], to: address, deadline: uint)
```

```
AjiraPayFinanceToken
contracts/AjiraPayFinanceToken.sol

Private:
  _totalSupply: uint256
  _name: string
  _symbol: string
  DEAD: address
  _balances: mapping(address=>uint256)
  _allowances: mapping(address=>mapping(address=>uint256))
  liquidityTreasuryPercent: uint256
  buyBackTreasuryPercent: uint256
  devTreasuryPercent: uint256
Public:
  MANAGER_ROLE: bytes32
  pancakeswapV2Router: IPancakeRouter02
  pancakeswapV2Pair: address
  treasury: address
  inSwapAndLiquify: bool
  swapAndLiquifyEnabled: bool
  isInTaxHoliday: bool
  isTransferFeeActive: bool
  isBuyBackEnabled: bool
  _isExcludedFromFee: mapping(address=>bool)
  buyFee: uint256
  sellFee: uint256
  txFee: uint256
  liquidityFee: uint256
  minLiquidityAmount: uint256
  maxTransactionAmount: uint256

Private:
   _transferStandard(_sender: address, _recipient: address, _amount: uint256, _shouldTakeFee: bool)
   _swapAndLiquify(contractTokenBalance: uint256) <<lockTheSwap>>
   _swapTokensForBnb(_tokenAmount: uint256)
   _addLiquidity(_tokenAmount: uint256, _bnbAmount: uint256)
   _buyBackAndBurnTokens(_bnbAmount: uint256)
   _calculateLiquidityFee(_amount: uint256): uint256
   _calculateTaxFee(_amount: uint256): uint256
   _getFeeAmountValues(_tAmount: uint256): (uint256, uint256, uint256)
   _takeLiquidity(_liquidityFeeAmount: uint256)
   _takeTaxes(from: address, to: address, amount: uint256): uint256
Internal:
   _transfer(_sender: address, _recipient: address, _amount: uint)
External:
   <<payable>> null()
   updateRouterAddress(_newRouter: address) <<onlyRole>>
   setSwapAndLiquifyEnabled(_enabled: bool) <<onlyRole>>
   setMaxTransactionAmount(_amount: uint) <<onlyRole>>
Public:
   <<event>> SwapAndLiquify(tokensSwapped: uint256, ethReceived: uint256, tokensIntoLiqudity: uint256)
   <<event>> Burn(from: address, to: address, amount: uint, timestamp: uint)
   <<modifier>> lockTheSwap()
   constructor(_router: address, _treasury: address)
   balanceOf(account: address): uint256
   totalSupply(): uint256
   supportsInterface(interfaceId: bytes4): bool
   recoverBNB(_amount: uint) <<onlyRole, nonReentrant>>
   recoverLostTokensForInvestor(_token: address, _amount: uint) <<onlyRole>>
   updateTreasury(_newTreasury: address) <<onlyRole>>
   setDeductionFeePercentages(_txFee: uint256, _liquidityFee: uint256, _buyFee: uint256, _sellFee: uint256) <<onlyRole>>
   setTreasuryPercentages(_liquidity: uint256, _buyBack: uint256) <<onlyRole>>
   excludeFromFee(_beneficiary: address) <<onlyRole>>
   includeInFee(_beneficiary: address) <<onlyRole>>
   setTaxHolidayEnabled(_enabled: bool) <<onlyRole>>
   setTransferFeeEnabled(_enabled: bool) <<onlyRole>>
   setBuyBackEnabled(_enabled: bool) <<onlyRole>>
   updateMinTokensToLiquify(_amount: uint256) <<onlyRole, nonReentrant>>
   burn(_account: address, _amount: uint) <<onlyRole>>
```

```
<<Interface>>
IPancakeSwapV2Pair
contracts/AjiraPayFinanceToken.sol

External:
   name(): string
   symbol(): string
   decimals(): uint8
   totalSupply(): uint
   balanceOf(owner: address): uint
   allowance(owner: address, spender: address): uint
   approve(spender: address, value: uint): bool
   transfer(to: address, value: uint): bool
   transferFrom(from: address, to: address, value: uint): bool
   DOMAIN_SEPARATOR(): bytes32
   PERMIT_TYPEHASH(): bytes32
   nonces(owner: address): uint
   permit(owner: address, spender: address, value: uint, deadline: uint, v: uint8, r: bytes32, s: bytes32)
   MINIMUM_LIQUIDITY(): uint
   factory(): address
   token0(): address
   token1(): address
   getReserves(): (reserve0: uint112, reserve1: uint112, blockTimestampLast: uint32)
   price0CumulativeLast(): uint
   price1CumulativeLast(): uint
   kLast(): uint
   mint(to: address): (liquidity: uint)
   burn(to: address): (amount0: uint, amount1: uint)
   swap(amount0Out: uint, amount1Out: uint, to: address, data: bytes)
   skim(to: address)
   sync()
   initialize(address, address)
Public:
  <<event>> Approval(owner: address, spender: address, value: uint)
  <<event>> Transfer(from: address, to: address, value: uint)
  <<event>> Mint(sender: address, amount0: uint, amount1: uint)
  <<event>> Burn(sender: address, amount0: uint, amount1: uint, to: address)
  <<event>> Swap(sender: address, amount0In: uint, amount1In: uint, amount0Out: uint, amount1Out: uint, to: address)
  <<event>> Sync(reserve0: uint112, reserve1: uint112)
```

FULL AUDIT REPORT

About SECURI LAB

SECURI LAB is a group of cyber security experts providing cyber security consulting, smart contract security audits, and KYC services.

Follow Us On:

Website	https://securi-lab.com/
Twitter	https://twitter.com/SECURI_LAB
Telegram	https://t.me/securi_lab
Medium	https://medium.com/@securi