SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

Ponke

PONKE

BEP 20

0xE81c932388783B2521EfB7C7045B1b6A328a5801

Table of Contents

Table of Contents	1
Disclaimer	2
Overview	3
Creation/Audit Date	3
Verified Socials	3
Contract Functions Analysis	4
Contract Safety and Weakness	7
Detected Vulnerability Description	11
Contract Flow Graph	12
Contract Interaction Graph	13
Inheritance Graph	14
Contract Desciptions	15
Audit Scope	22

Global Disclaimer

This document serves as a disclaimer for the crypto smart contract audit conducted by Skeleton Ecosystem. The purpose of the audit was to review the codebase of the smart contracts for potential vulnerabilities and issues. It is important to note the following:

Limited Scope: The audit is based on the code and information available up to the audit completion date. It does not cover external factors, system interactions, or changes made after the audit. The audit itself can not guarantee 100% safaty and can not detect common scam methods like farming and developer sell-out.

No Guarantee of Security: While we have taken reasonable steps to identify vulnerabilities, it is impossible to guarantee the complete absence of security risks or issues. The audit report provides an assessment of the contract’s security as of the audit date.

Continued Development: Smart contracts and blockchain technology are evolving fields. Updates, forks, or changes to the contract post-audit may introduce new risks that were not present during the audit.

Third-party Code: If the smart contract relies on third-party libraries or code, those components were not thoroughly audited unless explicitly stated. Security of these dependencies is the responsibility of their respective developers.

Non-Exhaustive Testing: The audit involved automated analysis, manual review, and testing under controlled conditions. It is possible that certain vulnerabilities or issues may not have been identified.

Risk Evaluation: The audit report includes a risk assessment for identified vulnerabilities. It is recommended that the development team carefully reviews and addresses these risks to mitigate potential exploits.

Not Financial Advice: This audit report is not intended as financial or investment advice. Decisions regarding the use, deployment, or investment in the smart contract should be made based on a comprehensive assessment of the associated risks.

By accessing and using this audit report, you acknowledge and agree to the limitations outlined above. Skeleton Ecosystem and its auditors shall not be held liable for any direct or indirect damages resulting from the use of the audit report or the smart contract itself.

Please consult with legal, technical, and financial professionals before making any decisions related to the smart contract.

Overview

Contract Name	Ponke
Ticker/Simbol	PONKE
Blockchain	Binance Smart Chain BEP20
Contract Address	0xE81c932388783B2521EfB7C7045B1b6A328a5801
Creator Address	0x835991881384b26D584988A3e7a14b42E897BDC1
Current Owner Address	0x0000000000000000000000000000000000000000
Contract Explorer	https://bscscan.com/address/0xe81c932388783b2521efb7c7045b1b6a328a5801#code
Compiler Version	v0.8.10+commit.fc410830
License	MIT
Optimisation	Yes with 200 Runs
Total Supply	554,555,555 **PONKE**
Decimals	9

Creation/Audit

Contract Deployed	25 Dec 2023
Audit Created	28 Dec 2023
Audit Update	V 1.0

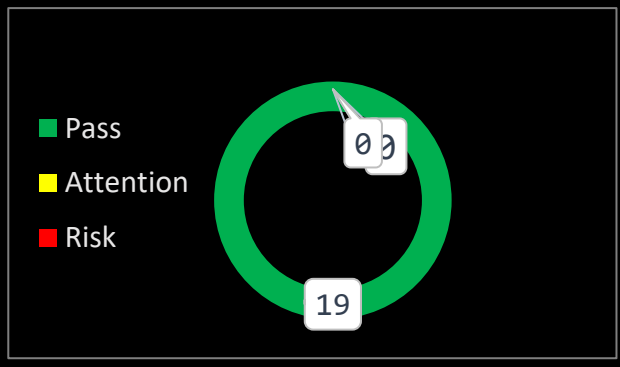
Verified Socials

Website	https://ponkebnb.tech
Telegram	https://t.me/Ponkebnb
Twitter (X)	https://x.com/ponkbnb

PONKE BEP20

Contract Function Analysis

Pass Attention Item Risky Item



Contract Verified	✅	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0x0000000000000000000000000000000000000000 Sometimes referred to as the "zero address" or “dead address” and is not owned by anyone, is commonly used for token burning or to renounce ownership of a smart contract. By setting a contract's ownership to the null address, control over the contract is further decentralized since it is not controlled by any individual.
Buy Tax	5 %	Shows the taxes for purchase transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Sell Tax	5 %	Shows the taxes for sell transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Honeypot Analyse	✅	Holder is able to buy and sell. If honeypot: The contract blocks sell transfer from holder wallet. Multiple events may cause honeypot. Trading disabled, extremely high tax
Liqudity Status	✅	LP Lock Status on 28.12.2023: Lp Locked: 83.47% Pinklock for 3641 days. Lp Burned: 10.30%
Trading Disable Functions	✅	No Trading suspendable function found. If a suspendable code is included, the token maybe neither be bought or sold (honeypot risk). If contract is renounced this function can't be used
Set Fees function	✅	No Fee Setting function found The contract owner may contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded (honeypot risk).
Proxy Contract	✅	Not a proxy contract!
Mint Function	✅	No Mint Function detected Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token. Owner can mint new tokens and sell.

Balance Modifier Function	✅	No Balance Modifier function found. If there is a function for this, the contract owner can have the authority to modify the balance of tokens at other addresses. For example revoke the bought tokens from the holders wallet. Common form of scam: You buy the token, but it's disappearing from your wallet.
Blacklist Function	✅	No Blacklist Setting function found. If there is a blacklist, some addresses may not be able to trade normally. Example: you buy the token and right after your Wallet getting blacklisted. Like so you will be unable to sell. Honeypot Risk.
Whitelist Function	✅	No Whitelist Setting function found If there is a function for this Developer can set zero fee or no max wallet size for adresses (for example team wallets can trade without fee. Can cause farming)
Hidden Owner Analysis	✅	No Hidden or multi owner with authorisation For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.
Retrieve Ownership Function	✅	No Functions found which can retrieve ownership of the contract. If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.
Self Destruct Function	✅	No Self Destruct function found. If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.
Specific Tax Changing Function	✅	No Specific Tax Changing Functions found. If it exists, the contract owner may set a very outrageous tax rate for assigned address to block it from trading. Can assign all wallets at once!
Trading Cooldown Function	✅	No Trading Cooldown Function found. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying. Like a temporary honeypot.
Max Transaction and Holding Modify Function	✅	No Max Transaction and Holding Modify function found. If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot
Transaction Limiting Function	✅	No Transaction Limiter Function Found. The number of overall token transactions may be limited (honeypot risk)

Details of Risk - Attention Items

Removing Risk of contract function based on renounced ownership

Transaction Receipt Event Logs

676

Address 0xe81c932388783b2521efb7c7045b1b6a328a5801

Name OwnershipTransferred (index_topic_1 address previousOwner, index_topic_2 address newOwner) View Source

Topics
0 0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0
1: previousOwner Dec → 0x835991881384b26D584988A3e7a14b42E897BDC1
2: newOwner Dec → 0x0000000000000000000000000000000000000000

Data 0x

Contract Security

Total Findings: 5

- **High 0**
- **Medium 0**
- **Low 3**
- **Info 2**

High Severity Issues: High possibility to cause problems, need to be resolved.

Medium Severity Issue: Will likely cause problems, recommended to resolve.

Low Severity Issues: Won’t cause problems, but for improvement purposes could be adjusted.

Informational Severity Issues: Not harmful in any way, information for the developer team.

Contract Security

List of Found Issues

High severity Issues: (0)

Medium severity issues: (0)

Low severity issues: (2)

- Missing Events
- Outdated Compiler version
- Reentrancy (reduced risk)

Informational severity issues: (2)

- Public Functions Should be Declared External
- Custom errors to save gas

Contract Weakness Classisication

THE SMART CONTRACT WEAKNESS CLASSIFICATION REGISTRY (SWC REGISTRY) IS AN IMPLEMENTATION OF THE WEAKNESS CLASSIFICATION SCHEME PROPOSED IN EIP-1470. IT IS LOOSELY ALIGNED TO THE TERMINOLOGIES AND STRUCTURE USED IN THE COMMON WEAKNESS ENUMERATION (CWE) WHILE OVERLAYING A WIDE RANGE OF WEAKNESS VARIANTS THAT ARE SPECIFIC TO SMART CONTRACTS.

ID	Description	AI	Manual	Result
SWC-100	Function Default Visibility	Passed	Passed	Passed
SWC-101	Integer Overflow and Underflow	Passed	Passed	Passed
SWC-102	Outdated Compiler Version	Low	Low	Low
SWC-103	Floating Pragma	Passed	Passed	Passed
SWC-104	Unchecked Call Return Value	Passed	Passed	Passed
SWC-105	Unprotected Ether Withdrawal	Passed	Passed	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed	Passed	Passed
SWC-107	Reentrancy	High	Low	Low
SWC-108	State Variable Default Visibility	Passed	Passed	Passed
SWC-109	Uninitialized Storage Pointer	Passed	Passed	Passed
SWC-110	Assert Violation	Passed	Passed	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed	Passed	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed	Passed	Passed
SWC-113	DoS with Failed Call	Passed	Passed	Passed
SWC-114	Transaction Order Dependence	Passed	Passed	Passed
SWC-115	Authorization through tx.origin	Passed	Passed	Passed
SWC-116	Block values as a proxy for time	Passed	Passed	Passed
SWC-117	Signature Malleability	Passed	Passed	Passed
SWC-118	Incorrect Constructor Name	Passed	Passed	Passed
SWC-119	Shadowing State Variables	Passed	Passed	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed	Passed	Passed

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

Detected High and Medium Severity Vulnerability Description.

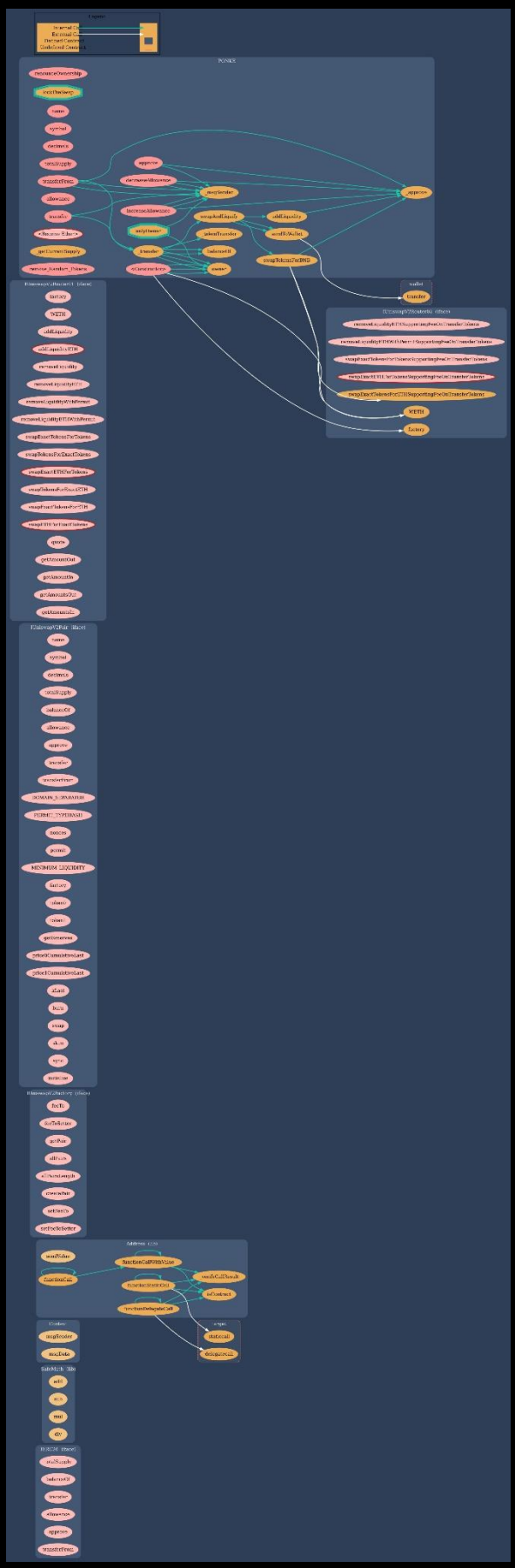
⚠ Reentrancy (1 Item) (Risk reduced on function)

Item: 1	Location:	Line 613-619	Severity:	Low

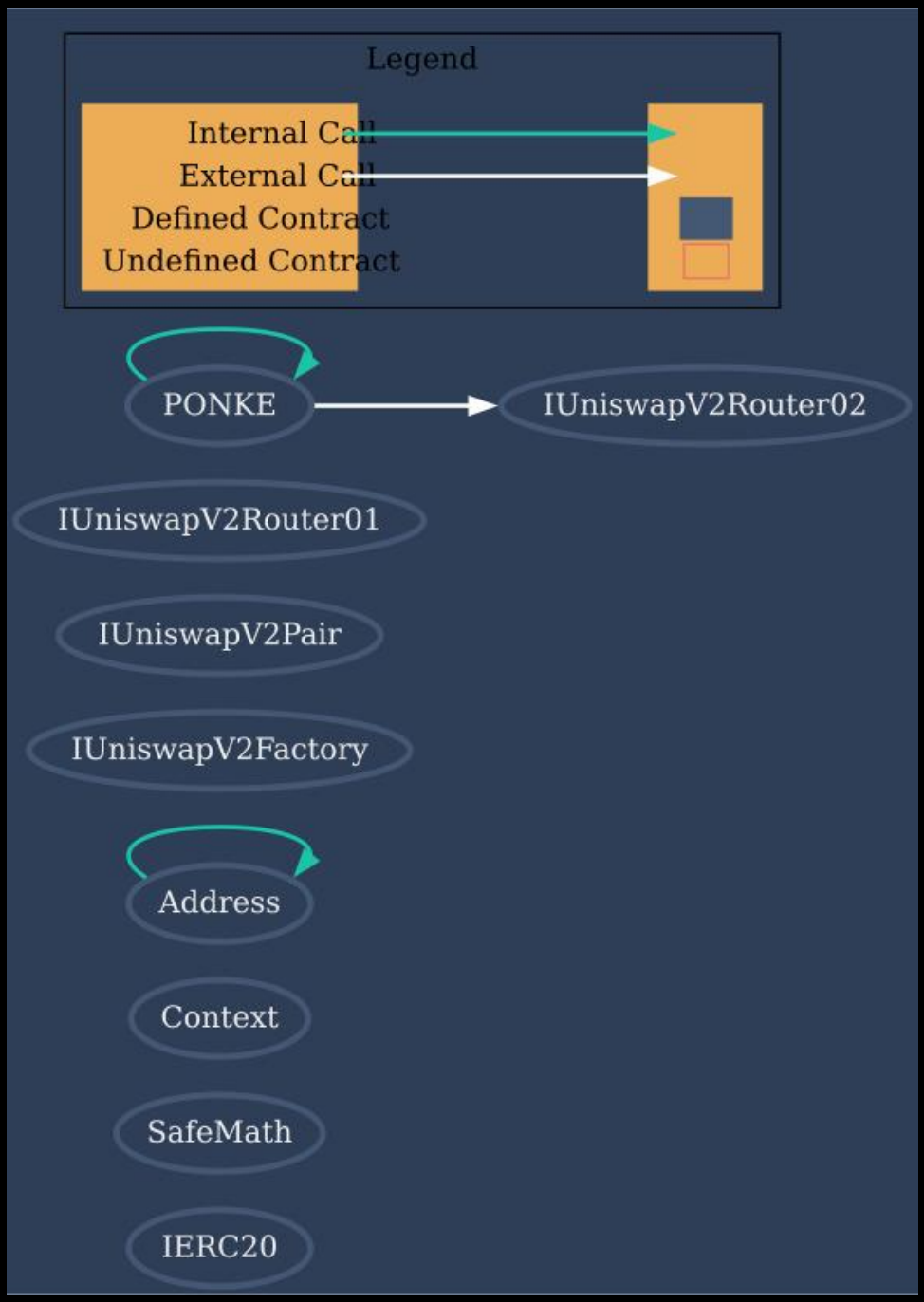
Function	In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls. This may lead to loss of funds, improper value updates, token loss, etc.
Remedation	It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

```
612
        ftrace | funcSig
613     function remove_Random_Tokens(address random_Token_Address1, uint256 percent_of_Tokens1) public returns(bool _sent){
614         require(random_Token_Address1 != address(this), "Can not remove native token");
615         uint256 totalRandom = IERC20(random_Token_Address1).balanceOf(address(this));
616         uint256 removeRandom = totalRandom*percent_of_Tokens1/100;
617         _sent = IERC20(random_Token_Address1).transfer(Wallet_Dev, removeRandom);
618
619     }
620
621
```

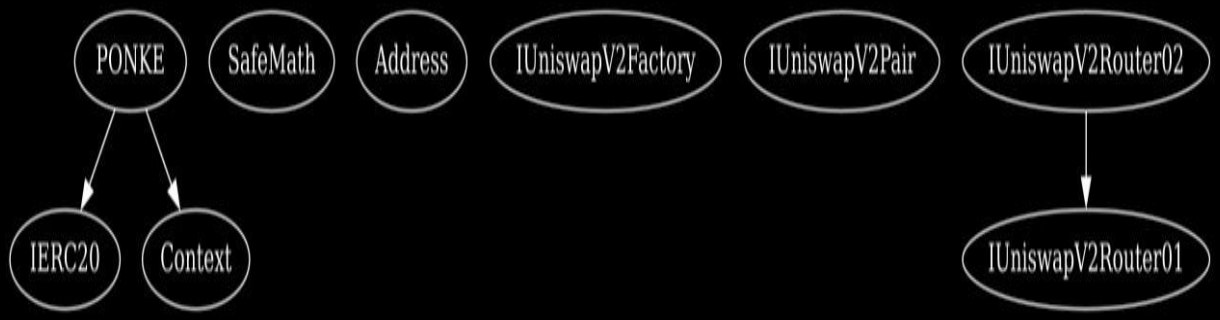
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
L	totalSupply	External !		NO!
L	balanceOf	External !		NO!
L	transfer	External !		NO!
L	allowance	External !		NO!
L	approve	External !		NO!
L	transferFrom	External !		NO!
SafeMath	Library			
L	add	Internal		
L	sub	Internal		
L	mul	Internal		
L	div	Internal		
L	sub	Internal		
L	div	Internal		
Context	Implementation			
L	_msgSender	Internal		
L	_msgData	Internal		
Address	Library			
L	isContract	Internal		

L	sendValue	Internal		
L	functionCall	Internal		
L	functionCall	Internal		
L	functionCallWithValue	Internal		
L	functionCallWithValue	Internal		
L	functionStaticCall	Internal		
L	functionStaticCall	Internal		
L	functionDelegateCall	Internal		
L	functionDelegateCall	Internal		
L	_verifyCallResult	Private		
IUniswapV2Factory	Interface			
L	feeTo	External		NO!
L	feeToSetter	External		NO!
L	getPair	External		NO!
L	allPairs	External		NO!
L	allPairsLength	External		NO!
L	createPair	External		NO!
L	setFeeTo	External		NO!
L	setFeeToSetter	External		NO!

IUniswapV2Pair	Interface			
L	name	External !		NO!
L	symbol	External !		NO!
L	decimals	External !		NO!
L	totalSupply	External !		NO!
L	balanceOf	External !		NO!
L	allowance	External !		NO!
L	approve	External !		NO!
L	transfer	External !		NO!
L	transferFrom	External !		NO!
L	DOMAIN_SEPARATOR	External !		NO!
L	PERMIT_TYPEHASH	External !		NO!
L	nonces	External !		NO!
L	permit	External !		NO!
L	MINIMUM_LIQUIDITY	External !		NO!
L	factory	External !		NO!
L	token0	External !		NO!
L	token1	External !		NO!
L	getReserves	External !		NO!
L	price0CumulativeLast	External !		NO!

L	price1CumulativeLast	External !		NO!
L	kLast	External !		NO!
L	burn	External !		NO!
L	swap	External !		NO!
L	skim	External !		NO!
L	sync	External !		NO!
L	initialize	External !		NO!
IUniswapV2Router01	Interface			
L	factory	External !		NO!
L	WETH	External !		NO!
L	addLiquidity	External !		NO!
L	addLiquidityETH	External !		NO!
L	removeLiquidity	External !		NO!
L	removeLiquidityETH	External !		NO!
L	removeLiquidityWithPermit	External !		NO!
L	removeLiquidityETHWithPermit	External !		NO!
L	swapExactTokensForTokens	External !		NO!
L	swapTokensForExactTokens	External !		NO!
L	swapExactETHForTokens	External !		NO!

L	swapTokensForExactETH	External !		NO!
L	swapExactTokensForETH	External !		NO!
L	swapETHForExactTokens	External !		NO!
L	quote	External !		NO!
L	getAmountOut	External !		NO!
L	getAmountIn	External !		NO!
L	getAmountsOut	External !		NO!
L	getAmountsIn	External !		NO!
IUniswapV2Router02	Interface	IUniswapV2Router01		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External !		NO!
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !		NO!
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !		NO!
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO!
L	swapExactTokensForETHSuppo	External !		NO!

	rtingFeeOnTransferTokens			
PONKE	Implementation	Context, IERC20		
L	owner	Public !		NO!
L	renounceOwnership	Public !		NO!
L		Public !		NO!
L	name	Public !		NO!
L	symbol	Public !		NO!
L	decimals	Public !		NO!
L	totalSupply	Public !		NO!
L	balanceOf	Public !		NO!
L	transfer	Public !		NO!
L	allowance	Public !		NO!
L	approve	Public !		NO!
L	transferFrom	Public !		NO!
L	increaseAllowance	Public !		NO!
L	decreaseAllowance	Public !		NO!
L		External !		NO!
L	_getCurrentSupply	Private		
L	_approve	Private		
L	_transfer	Private		

L	sendToWallet	Private		
L	swapAndLiquify	Private		lockTheSwap
L	swapTokensFor BNB	Private		
L	addLiquidity	Private		
L	remove_Random_Tokens	Public !		NO!
L	_tokenTransfer	Private		

Function can modify state

Function is payable

Audit Scope

Audit Method.

Our smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. Goal: discover errors, issues and security vulnaribilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review
We are using automated tools to scan functions and weeknesses of the contract. Transfers, integer over-undeflow checks such as all CWE events.

Tools we use:
Visual Studio Code
CWE
SWC
Solidity Scan
SVD

In manual code review our auditor looking at source code and performing line by line examination. This method helps to clarify developer's coding decisions and business logic.

Skeleton Ecosystem

https://skeletonecosystem.com

https://github.com/SkeletonEcosystem/Audits