SKELETON ECOSYSTEM
SMART CONTRACT AUDIT

ProSoft Chain [$PRCH]

BEP 20

0x301074950069483df2729effd5e7f5b70ae27cf0

Table of Contents

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	ProSoft Chain
Ticker/Simbol	$PRCH
Blockchain	Binance Smart Chain BEP20
Contract Address	0x301074950069483df2729effd5e7f5b70ae27cf0
Creator Address	0xF150038AB6ACE05Fff7bB1265276ae1A63255c3e
Current Owner Address	0xF150038AB6ACE05Fff7bB1265276ae1A63255c3e
Contract Explorer	https://bscscan.com/token/0x301074950069483df2729effd5e7f5b70ae27cf0
Compiler Version	v0.8.17+commit.8df45f5f
License	MIT
Optimisation	Yes with 200 Runs
Total Supply	1,000,000,000 **$PRCH**
Decimals	9

Creation/Audit

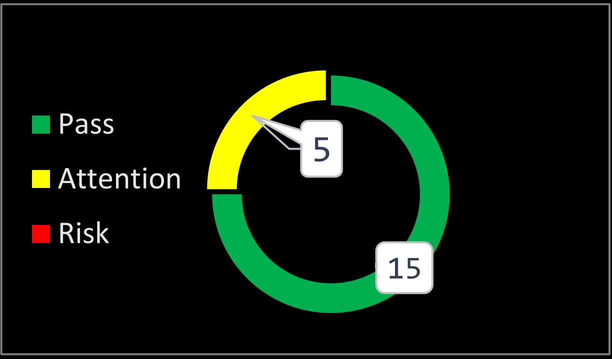
Contract Deployed	28 Sept 2023
Audit Created	29 Sept 2023
Audit Update	V 1.0

Verified Socials

Website	https://prosoftchain.org
Telegram	https://t.me/ProSoftChain
Twitter (X)	https://x.com/prosoftchain

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	⚠	0xF150038AB6ACE05Fff7bB1265276ae1A63255c3e
Buy Tax	10 %	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	10 %	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	✓	Liquidity lock status on 29.09.2023: 84.12% Pinklock for *121 days*.
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	⚠	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 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	⚠	Whitelist 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 authorised hidden owner found. For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.
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	⚠	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	⚠	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)

Contract Security

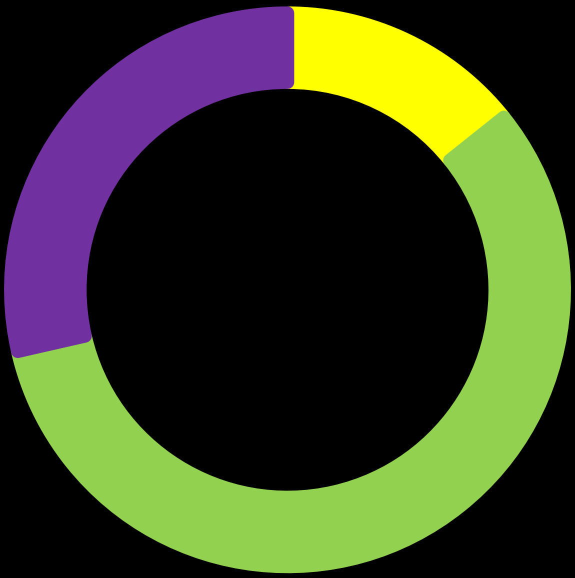
Total Findings: 9

🟥 **High 0**

🟨 **Medium 1**

🟩 **Low 4**

🟪 **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: (1)**

- Incorrect Acces Control

🟩 **Low severity issues: (4)**

- Numeric Notation Best Practices
- Use of Floating Pragma
- Low level Calls
- Missing Events

🟪 **Informational severity issues: (2)**

- Hard Coded Address
- Public Functions Should be Declared External

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	Passed	Passed	Passed
SWC-103	Floating Pragma	Low	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	Passed	Passed	Passed
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

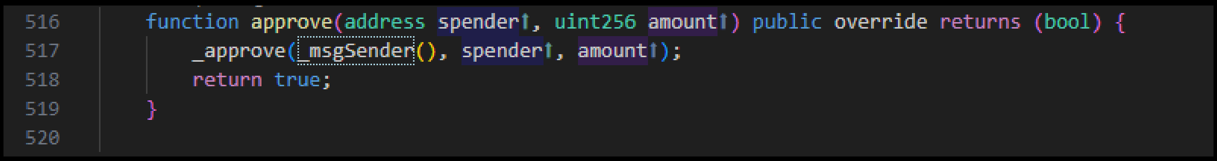
SWC-121	Missing Protection against Signature Replay Attacks	**Passed**	**Passed**	**Passed**
SWC-122	Lack of Proper Signature Verification	**Passed**	**Passed**	**Passed**
SWC-123	Requirement Violation	**Passed**	**Passed**	**Passed**
SWC-124	Write to Arbitrary Storage Location	**Passed**	**Passed**	**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.

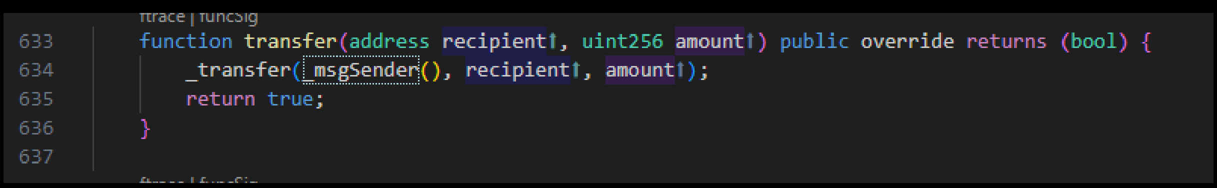
⚠ Incorrect Access Control (3 Items)

Item: 1	Location:	Line 516-519	Severity:	🟨 Medium
Item: 2	Location:	Line 633-636	Severity:	🟨 Medium
Item: 3	Location:	Line 638-642	Severity:	🟨 Medium

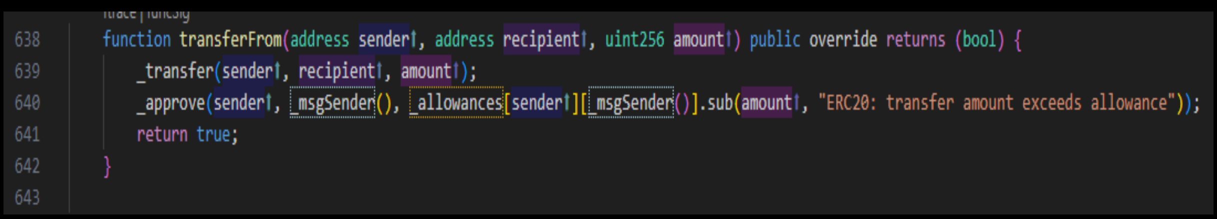
Function	Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract. The contract ProSoftChain is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function approve is missing the modifier onlyOwner.
Remedation	It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same



```
516      function approve(address spender, uint256 amount) public override returns (bool) {
517          _approve(_msgSender(), spender, amount);
518          return true;
519      }
520
```

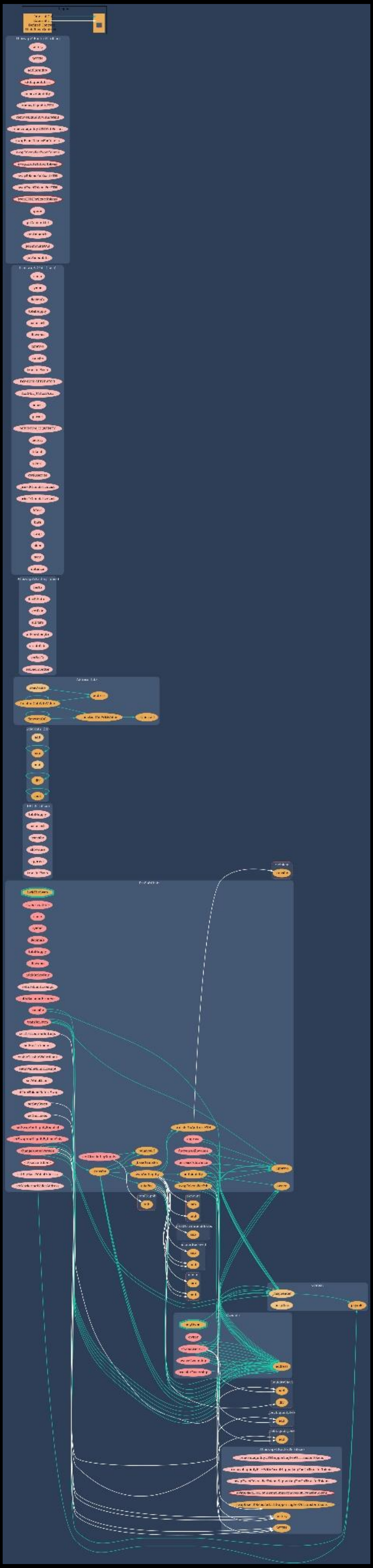


```
         ftrace | funcSig
633      function transfer(address recipient, uint256 amount) public override returns (bool) {
634          _transfer(_msgSender(), recipient, amount);
635          return true;
636      }
637
```

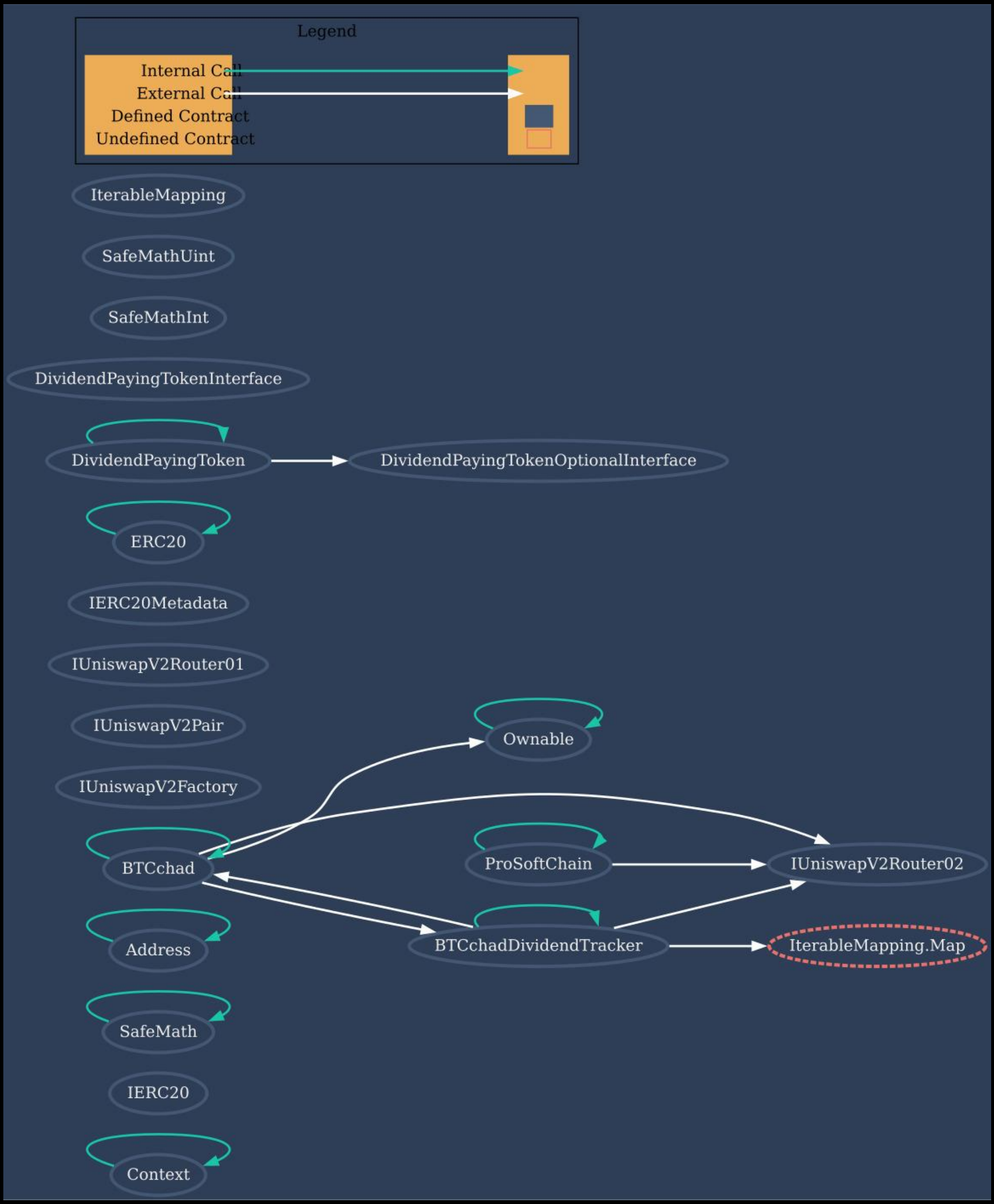


```
         ftrace | funcSig
638      function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
639          _transfer(sender, recipient, amount);
640          _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
641          return true;
642      }
643
```

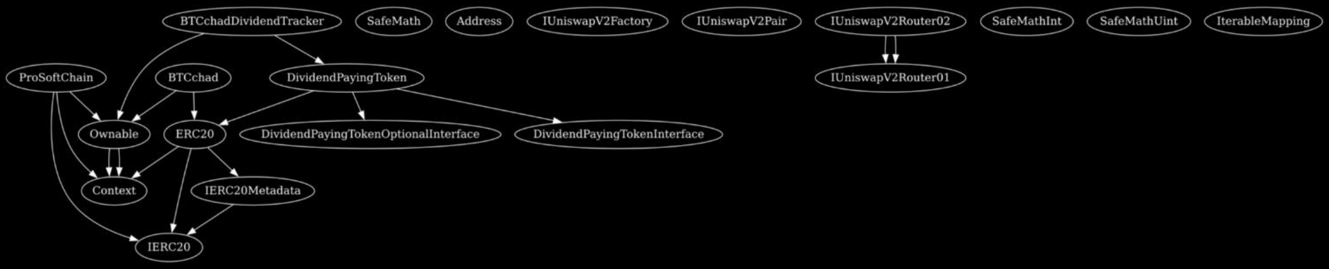
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
	_msgSender	Internal 🔒		
	_msgData	Internal 🔒		
IERC20	Interface			
	totalSupply	External ❗		NO❗
	balanceOf	External ❗		NO❗
	transfer	External ❗	🛑	NO❗
	allowance	External ❗		NO❗
	approve	External ❗	🛑	NO❗
	transferFrom	External ❗	🛑	NO❗
SafeMath	Library			
	add	Internal 🔒		
	sub	Internal 🔒		
	sub	Internal 🔒		
	mul	Internal 🔒		
	div	Internal 🔒		
	div	Internal 🔒		
	mod	Internal 🔒		
	mod	Internal 🔒		
Address	Library			
	isContract	Internal 🔒		
	sendValue	Internal 🔒	🛑	
	functionCall	Internal 🔒	🛑	
	functionCall	Internal 🔒	🛑	
	functionCallWithValue	Internal 🔒	🛑	
	functionCallWithValue	Internal 🔒	🛑	
	_functionCallWithValue	Private 🔐	🛑	
Ownable	Implementation	Context		
		Public ❗	🛑	NO❗
	owner	Public ❗		NO❗
	waiveOwnership	Public ❗	🛑	onlyOwner

	transferOwnership	Public !	🛑	onlyOwner
IUniswapV2Factory	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!
IUniswapV2Pair	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!
	burn	External !	🛑	NO!
	swap	External !	🛑	NO!
	skim	External !	🛑	NO!
	sync	External !	🛑	NO!
	initialize	External !	🛑	NO!

IUniswapV2Router01	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❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗

	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
ProSoftChain	Implementation	Context, IERC20, Ownable		
		Public ❗	🛑	NO❗
	name	Public ❗		NO❗
	symbol	Public ❗		NO❗
	decimals	Public ❗		NO❗
	totalSupply	Public ❗		NO❗
	balanceOf	Public ❗		NO❗
	allowance	Public ❗		NO❗
	increaseAllowance	Public ❗	🛑	NO❗
	decreaseAllowance	Public ❗	🛑	NO❗
	approve	Public ❗	🛑	NO❗
	_approve	Private 🔐	🛑	
	addMarketPair	Public ❗	🛑	onlyOwner
	setIsTxLimitExempt	External ❗	🛑	onlyOwner
	setIsExcludedFromFee	Public ❗	🛑	onlyOwner
	setBuyTaxes	External ❗	🛑	onlyOwner
	setSellTaxes	External ❗	🛑	onlyOwner
	setDistributionSettings	External ❗	🛑	onlyOwner
	setMaxTxAmount	External ❗	🛑	onlyOwner
	enableDisableWalletLimit	External ❗	🛑	onlyOwner
	setIsWalletLimitExempt	External ❗	🛑	onlyOwner
	setWalletLimit	External ❗	🛑	onlyOwner
	setNumTokensBeforeSwap	External ❗	🛑	onlyOwner
	setMarketingWalletAddress	External ❗	🛑	onlyOwner
	setBuyBackWalletAddress	External ❗	🛑	onlyOwner
	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
	setSwapAndLiquifyByLimitOnly	Public ❗	🛑	onlyOwner
	getCirculatingSupply	Public ❗		NO❗

	transferToAddress ETH	Private 🔐	🛑	
	changeRouterVersion	Public ❗	🛑	onlyOwner
		External ❗	💵	NO❗
	transfer	Public ❗	🛑	NO❗
	transferFrom	Public ❗	🛑	NO❗
	_transfer	Private 🔐	🛑	
	_basicTransfer	Internal 🔒	🛑	
	swapAndLiquify	Private 🔐	🛑	lockTheSwap
	swapTokensForEth	Private 🔐	🛑	
	addLiquidity	Private 🔐	🛑	
	takeFee	Internal 🔒	🛑	

🛑	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