SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

Netron Protocol [NETRON]

BEP 20

0x3b41aC371d58a559A0A43876F4f6DFe237587140

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	Netron Protocol
Ticker/Simbol	NETRON
Blockchain	Binance Smart Chain BEP20
Contract Address	0x3b41aC371d58a559A0A43876F4f6DFe237587140
Creator Address	0x1ccFA07D71762E6aD3462dec63E7c6E4E19972Ca
Current Owner Address	0x1ccFA07D71762E6aD3462dec63E7c6E4E19972Ca
Contract Explorer	https://bscscan.com/token/0x3b41ac371d58a559a0a43876f4f6dfe237587140
Compiler Version	v0.8.18+commit.87f61d96
License	Unlicense
Optimisation	No with 200 Runs
Total Supply	100,000,000 **NETRON**
Decimals	9

Creation/Audit

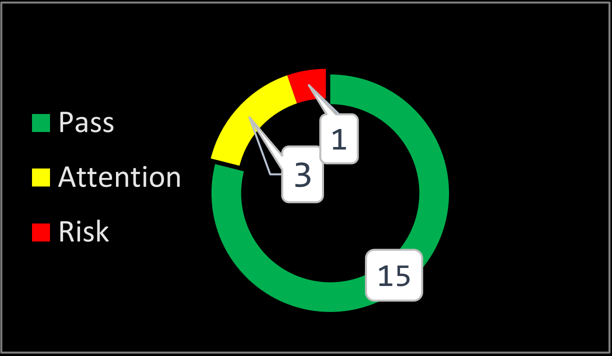
Contract Deployed	15 Nov 2023
Audit Created	21 Nov 2023
Audit Update	V 1.0

Verified Socials

Website	https://netronprotocol.org/
Telegram	https://t.me/Netron_Protocol
Twitter (X)	https://twitter.com/netronprotocol/

Contract Function Analysis

✓ Pass ⚠ Attention Item ❗ Risky Item



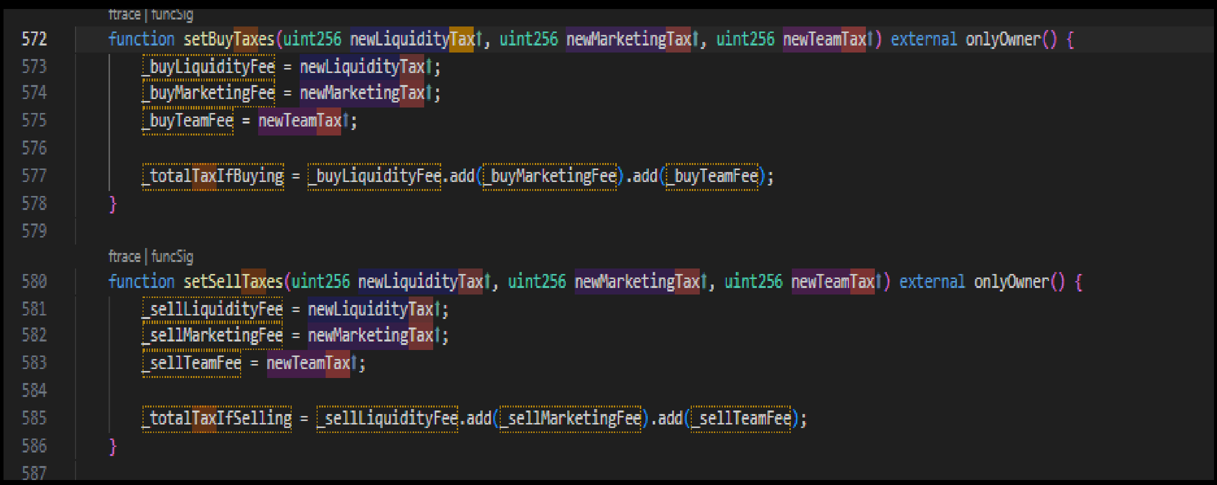
Contract Verified	✓	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0x1ccFA07D71762E6aD3462dec63E7c6E4E19972Ca
Buy Tax	9 %	Shows the taxes for purchase transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Sell Tax	9 %	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 21.11.2023: 100.00% Pinklock for 191 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 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	⚠	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	❗	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	⚠	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

⚠ Set Fee

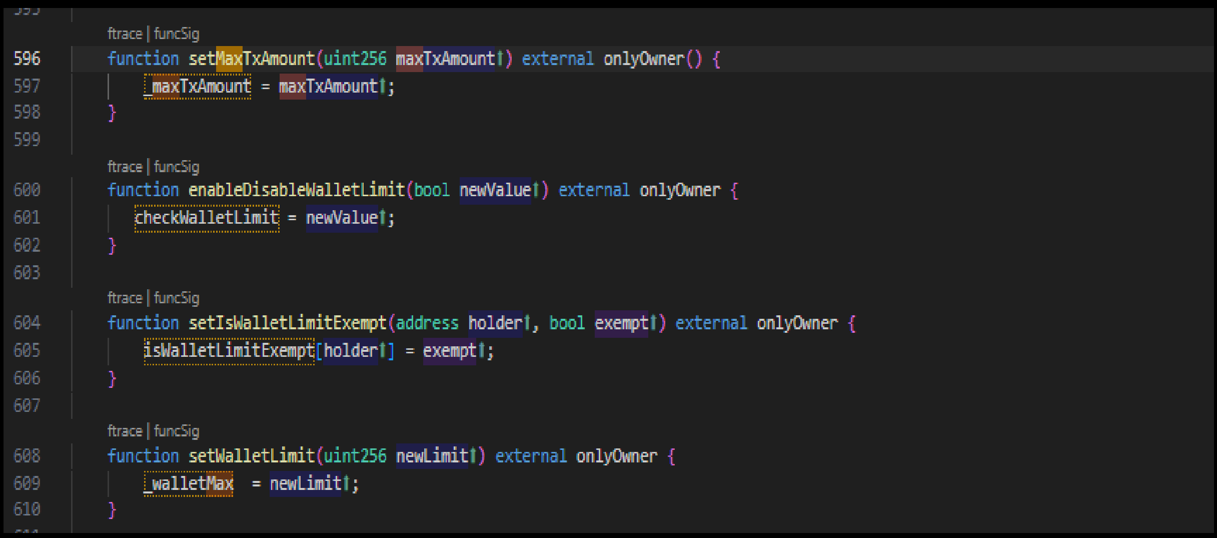
The contract owner may contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded



```
function setBuyTaxes(uint256 newLiquidityTax, uint256 newMarketingTax, uint256 newTeamTax) external onlyOwner() {
    _buyLiquidityFee = newLiquidityTax;
    _buyMarketingFee = newMarketingTax;
    _buyTeamFee = newTeamTax;

    _totalTaxIfBuying = _buyLiquidityFee.add(_buyMarketingFee).add(_buyTeamFee);
}

function setSellTaxes(uint256 newLiquidityTax, uint256 newMarketingTax, uint256 newTeamTax) external onlyOwner() {
    _sellLiquidityFee = newLiquidityTax;
    _sellMarketingFee = newMarketingTax;
    _sellTeamFee = newTeamTax;

    _totalTaxIfSelling = _sellLiquidityFee.add(_sellMarketingFee).add(_sellTeamFee);
}
```

⚠ Max Transaction and Holding Modify Function

If there is a function for this, the maximum trading amount or maximum position can be modified.

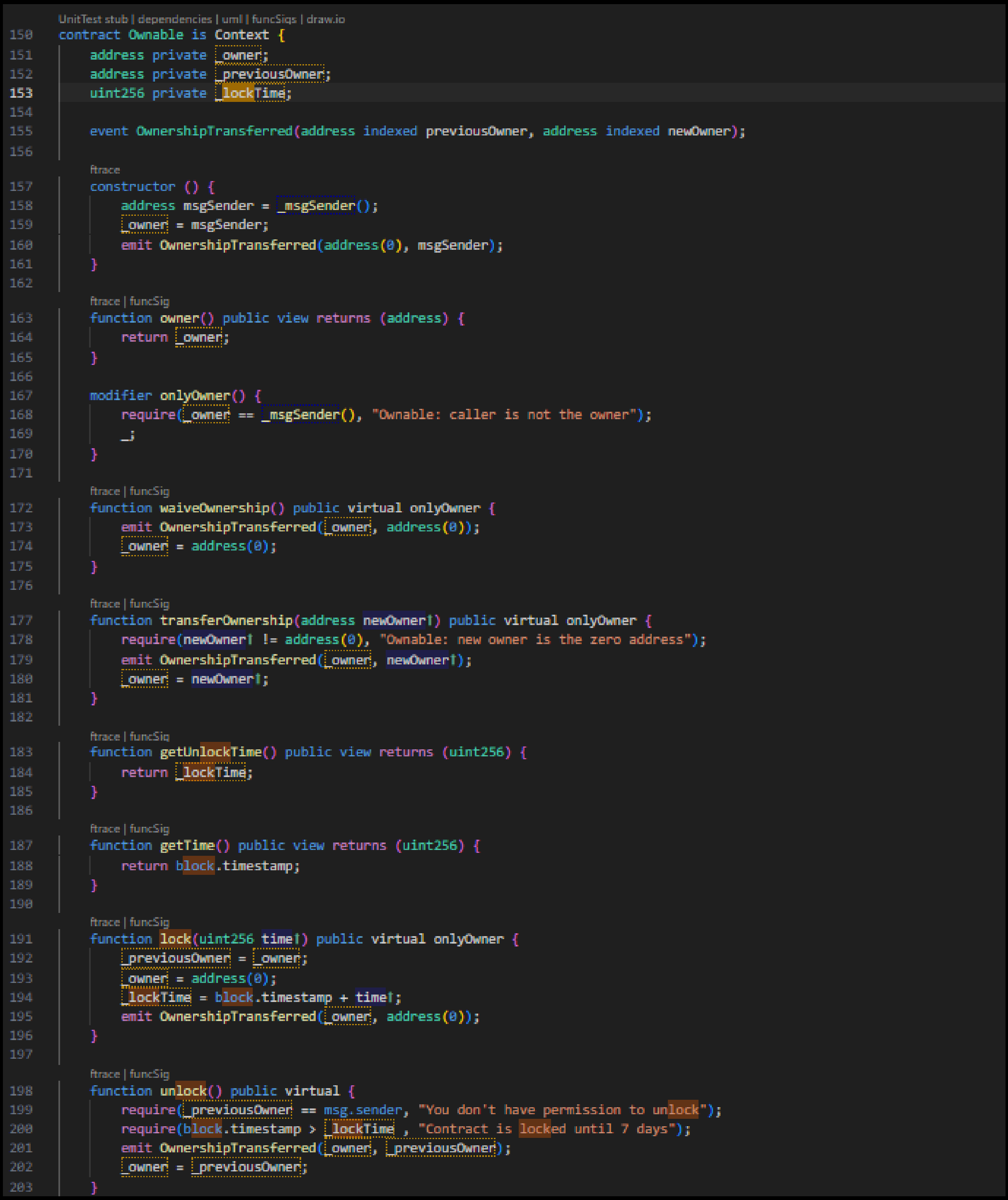


```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    _maxTxAmount = maxTxAmount;
}

function enableDisableWalletLimit(bool newValue) external onlyOwner {
   checkWalletLimit = newValue;
}

function setIsWalletLimitExempt(address holder, bool exempt) external onlyOwner {
    isWalletLimitExempt[holder] = exempt;
}

function setWalletLimit(uint256 newLimit) external onlyOwner {
    _walletMax  = newLimit;
}
```

Retrieve ownership of the contract

Ownership transferring to virtual owner with time lock, after ends deployer may regain ownership

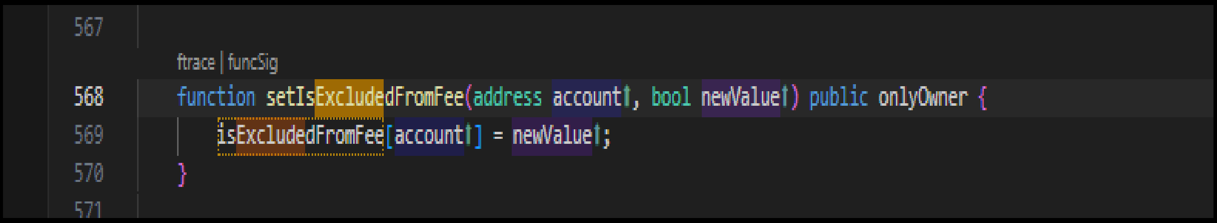
If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.



```
UnitTest stub | dependencies | uml | funcSigs | draw.io
contract Ownable is Context {
    address private _owner;
    address private _previousOwner;
    uint256 private _lockTime;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    ftrace
    constructor () {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSender);
    }

    ftrace | funcSig
    function owner() public view returns (address) {
        return _owner;
    }

    modifier onlyOwner() {
        require(_owner == _msgSender(), "Ownable: caller is not the owner");
        _;
    }

    ftrace | funcSig
    function waiveOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    ftrace | funcSig
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }

    ftrace | funcSig
    function getUnlockTime() public view returns (uint256) {
        return _lockTime;
    }

    ftrace | funcSig
    function getTime() public view returns (uint256) {
        return block.timestamp;
    }

    ftrace | funcSig
    function lock(uint256 time) public virtual onlyOwner {
        _previousOwner = _owner;
        _owner = address(0);
        _lockTime = block.timestamp + time;
        emit OwnershipTransferred(_owner, address(0));
    }

    ftrace | funcSig
    function unlock() public virtual {
        require(_previousOwner == msg.sender, "You don't have permission to unlock");
        require(block.timestamp > _lockTime , "Contract is locked until 7 days");
        emit OwnershipTransferred(_owner, _previousOwner);
        _owner = _previousOwner;
    }
```

⚠ Whitelist Function

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)



```
ftrace | funcSig
function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
    isExcludedFromFee[account] = newValue;
}
```

Contract Security

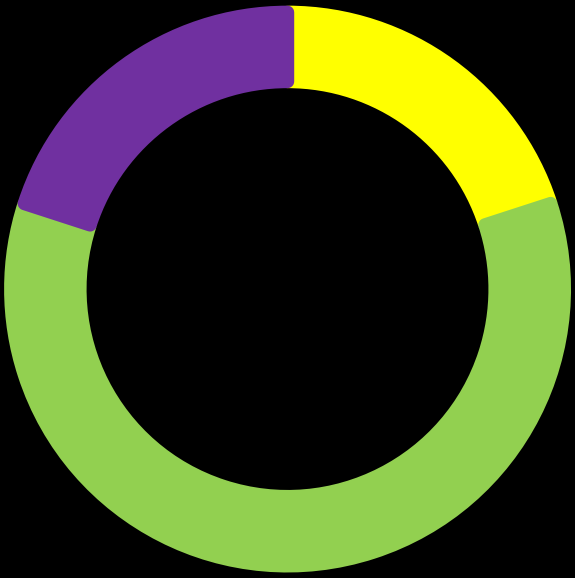
Total Findings: 5

🟥 **High 0**

🟨 **Medium 1**

🟩 **Low 3**

🟪 **Info 1**



🟥 **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: (3)**

- Missing Events
- Long Number Literals
- Floating Pragma

🟪 **Informational severity issues: (1)**

- 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

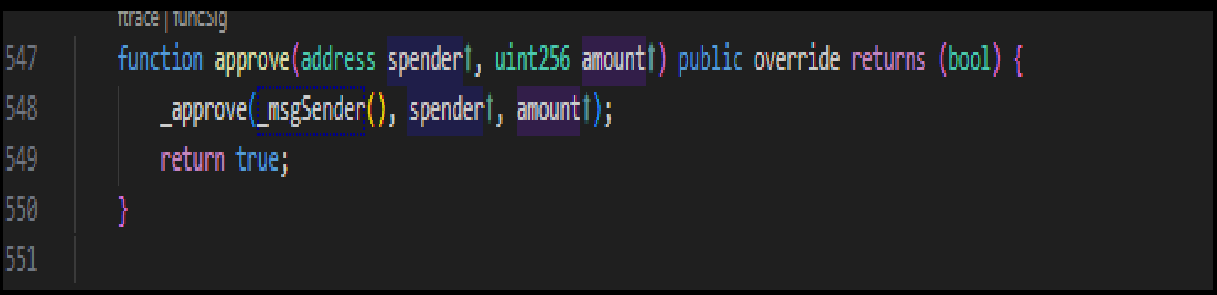
SWC-121	Missing Protection against Signature Replay Attacks	**Passed**	**Passed**	**Passed**
SWC-122	Lack of Proper Signature Verification	**Passed**	**Passed**	**Passed**
SWC-123	Requirement Violation	**Passed**	**Passed**	**Passed**
SWC-124	Write to Arbitrary Storage Location	**Passed**	**Passed**	**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 Acces Control (3 Item)

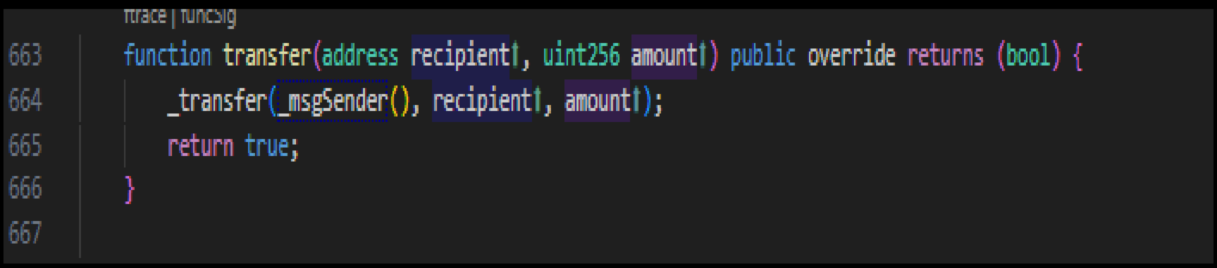
Item: 1	Location:	Line 546-551	Severity:	🟨 Medium

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 NetronProtocol 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


```
ftrace | funcsig
function approve(address spender, uint256 amount) public override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}
```

Item: 2	Location:	Line 662-667	Severity:	🟨 Medium

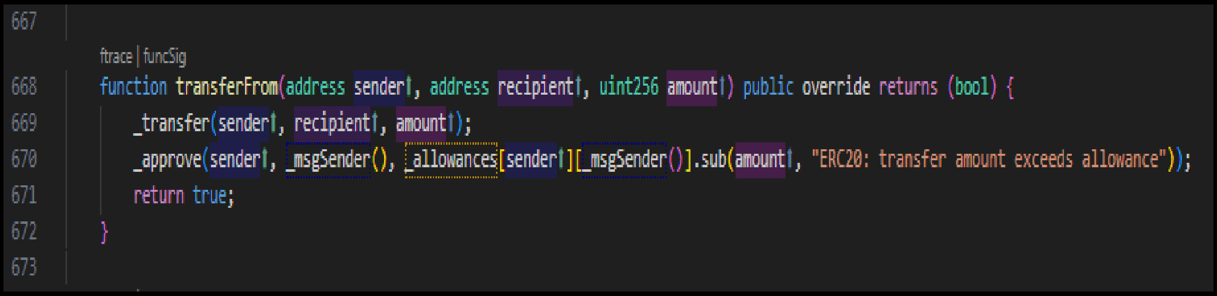
Function	The contract NetronProtocol is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function transfer is missing the modifier onlyOwner.
Remedation	It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same



```
    ftrace | funcSig
663     function transfer(address recipient, uint256 amount) public override returns (bool) {
664         _transfer(_msgSender(), recipient, amount);
665         return true;
666     }
667
```

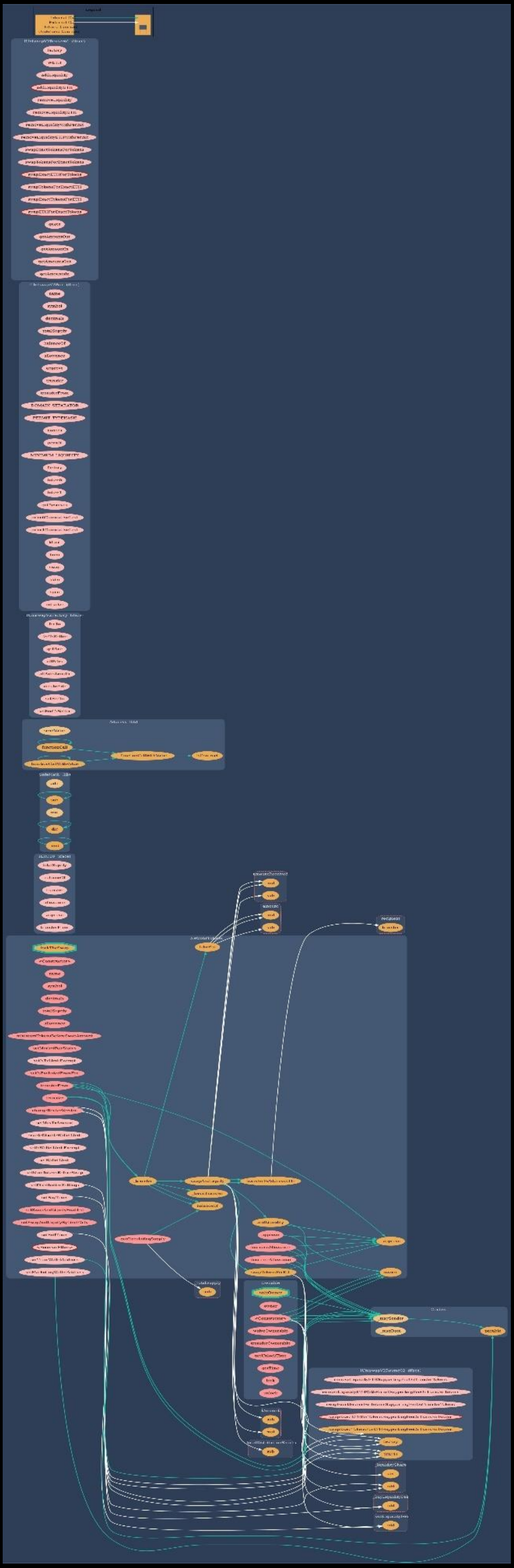
Item: 3	Location:	Line 667-673	Severity:	🟨 Medium

Function	The contract NetronProtocol is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function transferFrom 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

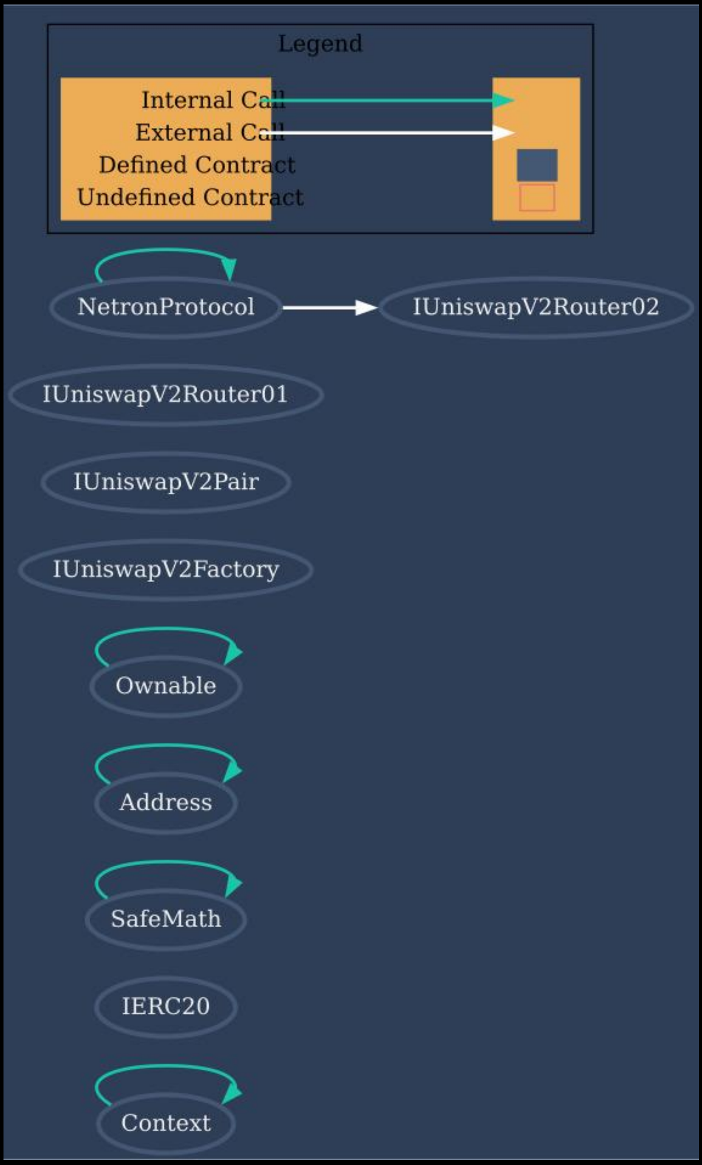


```
667
        ftrace | funcSig
668     function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
669         _transfer(sender, recipient, amount);
670         _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
671         return true;
672     }
673
```

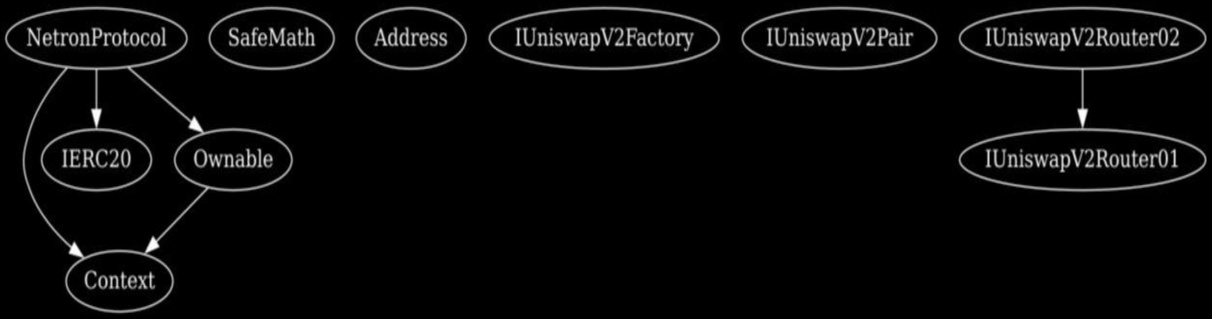
Contract Flow Graph



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			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCall	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	functionCallWithValue	Internal 🔒	🛑	
L	_functionCallWithValue	Private 🔐	🛑	
Ownable	Implementation	Context		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	waiveOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	getUnlockTime	Public ❗		NO❗
L	getTime	Public ❗		NO❗
L	lock	Public ❗	🛑	onlyOwner
L	unlock	Public ❗	🛑	NO❗
IUniswapV2Factory	Interface			
L	feeTo	External ❗		NO❗
L	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❗️

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	swapExactTokensForTokensSu	External ❗	🛑	NO❗

	pportingFeeOnTransferTokens			
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
NetronProtocol	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❗
└	minimumTokensBeforeSwapAmount	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	setMarketPairStatus	Public ❗	🛑	onlyOwner

L	setIsTxLimitExempt	External ❗	🛑	onlyOwner
L	setIsExcludedFromFee	Public ❗	🛑	onlyOwner
L	setBuyTaxes	External ❗	🛑	onlyOwner
L	setSellTaxes	External ❗	🛑	onlyOwner
L	setDistributionSettings	External ❗	🛑	onlyOwner
L	setMaxTxAmount	External ❗	🛑	onlyOwner
L	enableDisableWalletLimit	External ❗	🛑	onlyOwner
L	setIsWalletLimitExempt	External ❗	🛑	onlyOwner
L	setWalletLimit	External ❗	🛑	onlyOwner
L	setNumTokensBeforeSwap	External ❗	🛑	onlyOwner
L	setMarketingWalletAddress	External ❗	🛑	onlyOwner
L	setTeamWalletAddress	External ❗	🛑	onlyOwner
L	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
L	setSwapAndLiquifyByLimitOnly	Public ❗	🛑	onlyOwner
L	getCirculatingSupply	Public ❗		NO❗
L	transferToAddressETH	Private 🔐	🛑	

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