SCRL

Full Audit Report

Mobile Wallet Token Security Assessment

Mobile Wallet Token Security Assessment

FULL AUDIT REPORT

Security Assessment by SCRL on **Monday, July 29, 2024**

SCRL is deliver a security solution for Web3 projects by expert security researchers.

Executive Summary

For this security assessment, SCRL received a request on Thursday, July 25, 2024

Client	Language	Audit Method	Confidential	Network Chain	Contract
Mobile Wallet Token	**Solidity**	**Whitebox**	**Public**	**BNB Chain**	**0x81e9A4b6dc7379D0dd6dCdD279236b6C8618070A**

Report Version	Twitter	Telegram	Website
1.0	**https://x.com/mobilewalletinc**	**https://t.me/MW_Token**	**https://mobile-wallet.app/**

Scoring:

Vulnerability Summary

9	9	0	0	9	0
Total Findings	Unresolved	Resolved	Mitigate	Acknowledge	Decline

▪	0 **Critical**		Critical severity is assigned to security vulnerabilities that pose a severe threat to the smart contract and the entire blockchain ecosystem.
▪	0 **High**		High-severity issues should be addressed quickly to reduce the risk of exploitation and protect users' funds and data.
▪	1 **Medium**	1 Unresolved	It's essential to fix medium-severity issues in a reasonable timeframe to enhance the overall security of the smart contract.
▪	0 **Low**		While low-severity issues can be less urgent, it's still advisable to address them to improve the overall security posture of the smart contract.
▪	0 **Very Low**		Very Low severity is used for minor security concerns that have minimal impact and are generally of low risk.
▪	4 **Informational**	4 Unresolved	Used to categorize security findings that do not pose a direct security threat to the smart contract or its users. Instead, these findings provide additional information, recommendations
▪	4 **Gas-optimization**	4 Unresolved	Suggestions for more efficient algorithms or improvements in gas usage, even if the current code is already secure.

Audit Scope:

File	SHA-1 Hash
WDIStandardToken.sol	e42a9f5bf042263ec5ee5029c723d3c63fa6281f

Audit Version History:

Version	Date	Description
1.0	Friday, July 26, 2024	Preliminary Report
1.1	Monday, July 29, 2024	Full Audit Report

Audit information:

Request Date	Audit Date	Re-assessment Date
Thursday, July 25, 2024	Friday, July 26, 2024	-

Smart Contract Audit Summary

Security Assessment Author

Auditor:	**Mark K.**	[Security Researcher | Redteam]
	Kevin N.	[Security Researcher | Web3 Dev]
	Yusheng T.	[Security Researcher | Incident Response]
Document Approval:	**Ronny C.**	CTO & Head of Security Researcher
	Chinnakit J.	CEO & Founder

Digital Sign

Digitally signed by Chinnakit J.
DN: cn=Chinnakit J., o=SCRL, ou=CEO & Founder, email=contact@scrl.io, c=TH
Date: 2024.07.29 19:41:00 +07'00'

Disclaimer

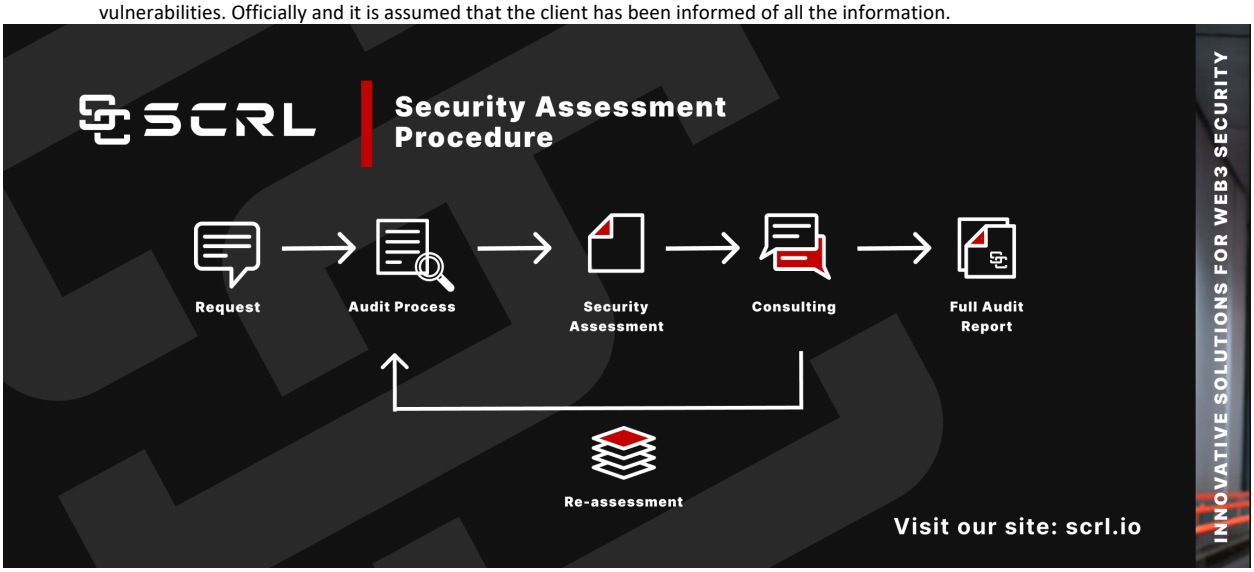
Regarding this security assessment, there are no guarantees about the security of the program instruction received from the client is hereinafter referred to as "**Source code**".
And **SCRL** hereinafter referred to as "**Service Provider**", the **Service Provider** will not be held liable for any 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 Is Not Financial/Investment Advice Any loss arising from any investment in any project is the responsibility of the investor.

SCRL disclaims any liability incurred. Whether it's Rugpull, Abandonment, Soft Rugpull, Exploit, Exit Scam.

Security Assessment Procedure

1. **Request** The client must submit a formal request and follow the procedure. By submitting the source code and agreeing to the terms of service.
2. **Audit Process** Check for vulnerabilities and vulnerabilities from source code obtained by experts using formal verification methods, including using powerful tools such as Static Analysis, SWC Registry, Dynamic Security Analysis, Automated Security Tools, CWE, Syntax & Parameter Check with AI ,WAS (Warning Avoidance System a python script tools powered by SCRL).
3. **Security Assessment** Deliver Preliminary Security Assessment to clients to acknowledge the risks and vulnerabilities.
4. **Consulting** Discuss on risks and vulnerabilities encountered by clients to apply to their source code to mitigate risks.
    a. **Re-assessment** Reassess the security when the client implements the source code improvements and if the client is satisfied with the results of the audit. We will proceed to the next step.
5. **Full Audit Report** SCRL provides clients with official security assessment reports informing them of risks and vulnerabilities. Officially and it is assumed that the client has been informed of all the information.



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

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

7 categories severity based

For **Informational** & **Non-class/Optimization/Best-practices will** not be counted as **severity**

Category

Centralization

Centralization Risk is The risk incurred by a sole proprietor, such as the Owner being able to change something without permission

Economics Risk

Economics Risk is Risks that may affect the economic mechanism system, such as the ability to increase Mint token

Logical Issue

Logical Issue is that can cause errors to core processing, such as any prior operations that cause background processes to crash.

Authorization

Authorization is Possible pitfalls from weak coding allows unrelated people to take any action to modify the values.

Mathematical

Mathematical Any erroneous arithmetic operations affect the operation of the system or lead to erroneous values.

Naming Conventions

Naming Conventions naming variables that may affect code understanding or naming inconsistencies

Security Risk

Security Risk of loss or damage if it's no mitigate

Coding Style

Coding Style is Tips coding for efficiency performance

Best Practices

Best Practices is suggestions for improvement

Optimization

Optimization is performance improvement

Gas Optimization

Gas Optimization is increase performance to avoid expensive gas

Dead Code

Dead Code having unused code This may result in wasted resources and gas fees.

Table Of Content

Source Units in Scope

Source Units Analyzed: `1`
Source Units in Scope: `1` (**100%**)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝 🔍 🎨	src/WDIStandardToken.sol	4	5	767	533	413	6	383	💰🔽 🔆Σ
📝 🔍 🎨	**Totals**	**4**	**5**	**767**	**533**	**413**	**6**	**383**	💰🔽 🔆Σ

Legend: [📝]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Visibility, Mutability, Modifier function testing

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
2	0	5	2

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
79	5

External	Internal	Private	Pure	View
58	65	0	5	33

StateVariables

Total	🌐Public
21	8

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
^0.8.0		yes		

📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🪚 ECRecover	🌀 New/Create/Create2
yes					

TryCatch	Σ Unchecked
	yes

Dependencies / External Imports

Dependency / Import Path	Count

Vulnerability Findings

ID	Vulnerability Detail	Severity	Category	Status
CEN-01	Centralization Risk	Medium	Centralization	**Acknowledge**
SEC-01	Empty Function Body - Consider commenting why	Informational	Best Practices	**Acknowledge**
SEC-02	`require()` / `revert()` statements should have descriptive reason strings	Informational	Best Practices	**Acknowledge**
SEC-03	Array indices should be referenced via `enum` rather than via numeric literals	Informational	Best Practices	**Acknowledge**
SEC-04	Functions not used internally could be marked external	Informational	Optimization	**Acknowledge**
GAS-01	Use `selfbalance()` instead of `address(this).balance`	Gas-optimization	Gas Optimization	**Acknowledge**
GAS-02	Use Custom Errors	Gas-optimization	Gas Optimization	**Acknowledge**
GAS-03	Long revert strings	Gas-optimization	Gas Optimization	**Acknowledge**
GAS-04	Use != 0 instead of > 0 for unsigned integer comparison	Gas-optimization	Gas Optimization	**Acknowledge**

CEN-01: Centralization Risk

Vulnerability Detail	Severity	Location	Category	Status
Centralization Risk	Medium	Check on finding	Centralization	**Acknowledge**

Finding:

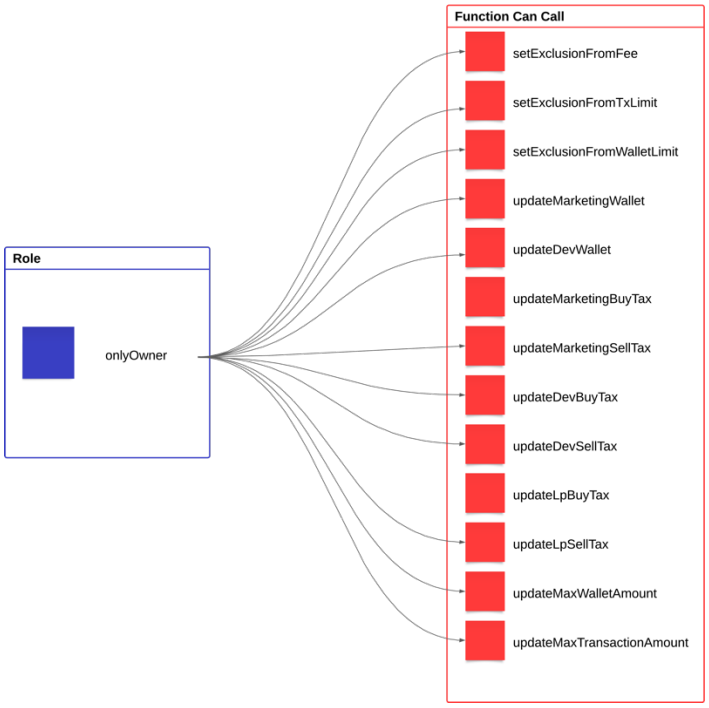
```
File: WDIStandardToken.sol

451: contract WDIStandardToken is ERC20, Ownable {

593:     function setExclusionFromFee(address account, bool value) public onlyOwner {

597:     function setExclusionFromTxLimit(address account, bool value) public onlyOwner
{

601:     function setExclusionFromWalletLimit(address account, bool value) public
onlyOwner {

605:     function updateMarketingWallet(address newWallet) external onlyOwner {

612:     function updateDevWallet(address newWallet) external onlyOwner {

619:     function updateMarketingBuyTax(uint256 tax) external onlyOwner {

624:     function updateMarketingSellTax(uint256 tax) external onlyOwner {

629:     function updateDevBuyTax(uint256 tax) external onlyOwner {

634:     function updateDevSellTax(uint256 tax) external onlyOwner {

639:     function updateLpBuyTax(uint256 tax) external onlyOwner {

644:     function updateLpSellTax(uint256 tax) external onlyOwner {

649:     function updateMaxWalletAmount(uint256 maxWallet) external onlyOwner {

655:     function updateMaxTransactionAmount(uint256 maxTx) external onlyOwner {

```

Explain Function Capability:

The contract provides several functions:

1. setExclusionFromFee(address account, bool value):
   If value is true, the account will not be charged fees on transactions.
2. setExclusionFromTxLimit(address account, bool value):
   If value is true, the account will not be subject to transaction limits imposed on other accounts.
3. setExclusionFromWalletLimit(address account, bool value):
   If value is true, the account will not be subject to wallet size limits.
4. updateMarketingWallet(address newWallet):
   Changes the wallet address where marketing funds are sent.
5. updateDevWallet(address newWallet):
   Changes the wallet address where development funds are sent.
6. updateMarketingBuyTax(uint256 tax):
   Changes the tax rate applied to marketing-related purchases.
7. updateMarketingSellTax(uint256 tax):
   Changes the tax rate applied to marketing-related sales.
8. updateDevBuyTax(uint256 tax):
   Changes the tax rate applied to development-related purchases.
9. updateDevSellTax(uint256 tax):
   Changes the tax rate applied to development-related sales.
10. updateLpBuyTax(uint256 tax):
    Changes the tax rate applied to liquidity pool-related purchases.
11. updateLpSellTax(uint256 tax):
    Changes the tax rate applied to liquidity pool-related sales.
12. updateMaxWalletAmount(uint256 maxWallet):
    Changes the maximum amount of tokens a wallet can hold.
13. updateMaxTransactionAmount(uint256 maxTx):
    Changes the maximum amount of tokens that can be transferred in a single transaction.

Centralization Risk



Recommendation:

In terms of timeframes, there are three categories: short-term, long-term, and permanent.

For short-term solutions, a combination of timelock and multi-signature (2/3 or 3/5) can be used to mitigate risk by delaying sensitive operations and avoiding a single point of failure in key management. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; assigning privileged roles to multi-signature wallets to prevent private key compromise; and sharing the timelock contract and multi-signer addresses with the public via a medium/blog link.

For long-term solutions, a combination of timelock and DAO can be used to apply decentralization and transparency to the system. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; introducing a DAO/governance/voting module to increase transparency and user involvement; and sharing the timelock contract, multi-signer addresses, and DAO information with the public via a medium/blog link.

Finally, permanent solutions should be implemented to ensure the ongoing security and protection of the system.

Alleviation:

-

SEC-01: Empty Function Body - Consider commenting why

Vulnerability Detail	Severity	Location	Category	Status
Empty Function Body - Consider commenting why	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

199:    function _beforeTokenTransfer(address from, address to, uint256 amount)
internal virtual {}

201:    function _afterTokenTransfer(address from, address to, uint256 amount)
internal virtual {}

766:    receive() external payable {}

```

Recommendation:

1. Commenting Empty Functions: Provide comments for empty function bodies to explain their intended purpose or future plans. This helps maintain clarity for developers who may work with the code in the future and prevents misunderstanding about why the function body is empty.
2. Review Design: Ensure that the empty functions are intentionally left empty and align with the intended contract design. If the functions are meant to be implemented later, consider providing comments about their intended functionality or the reason they are currently empty.

Alleviation:

-

SEC-02: `require()` / `revert()` statements should have descriptive reason strings

Vulnerability Detail	Severity	Location	Category	Status
`require()` / `revert()` statements should have descriptive reason strings	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

664:        require(totalFees > 0);
```

Recommendation:
Add Descriptive Reason Strings: Include a clear and descriptive reason string in require and revert statements to explain why the condition must be true. This improves the contract's usability and facilitates easier debugging.

Reference: https://docs.soliditylang.org/en/v0.8.19/control-structures.html#require-statements

Alleviation:
-

SEC-03: Array indices should be referenced via `enum` rather than via numeric literals

Vulnerability Detail	Severity	Location	Category	Status
Array indices should be referenced via `enum` rather than via numeric literals	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

671:        path[0] = address(this);

672:        path[1] = weth;
```

Recommendation:

1. Introduce enums for Array Indices: Define an enum to represent the indices of the array. This will improve the readability of the code by providing meaningful names for each index.
2. Update Array Index References: Replace numeric literals with enum values to enhance code clarity.

Reference: https://docs.soliditylang.org/en/v0.8.19/types.html#enums

Alleviation:

-

SEC-04: Functions not used internally could be marked external

Vulnerability Detail	Severity	Location	Category	Status
Functions not used internally could be marked external	Informational	Check on finding	Optimization	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

553:    function getTokenInfo() public view returns (TokenInfo memory _tokenInfo) {

565:    function totalTaxFees() public view returns (uint256) {

593:    function setExclusionFromFee(address account, bool value) public onlyOwner {

597:    function setExclusionFromTxLimit(address account, bool value) public onlyOwner
{

601:    function setExclusionFromWalletLimit(address account, bool value) public
onlyOwner {
```

Recommendation:
Change Visibility to external: Update the visibility of these functions to external where applicable. This will optimize gas usage and clarify that these functions are meant to be called from outside the contract.

Reference: https://docs.soliditylang.org/en/v0.8.19/contracts.html#function-visibility

Alleviation:
-

GAS-01: Use `selfbalance()` instead of `address(this).balance`

Vulnerability Detail	Severity	Location	Category	Status
Use `selfbalance()` instead of `address(this).balance`	-	Check on finding	Gas Optimization	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

674:        uint256 beforeEthBalance = address(this).balance;

684:        uint256 ethBalance = address(this).balance - beforeEthBalance;
```

Recommendation:

Using address(this).balance can be less gas-efficient compared to using selfbalance() in Solidity. selfbalance() is an assembly function that can be used to retrieve the balance of the current contract with slightly lower gas costs. For external contracts, using balance(address) is more efficient than address.balance().

Alleviation:

-

GAS-02: Use Custom Errors

Vulnerability Detail	Severity	Location	Category	Status
Use Custom Errors	-	Check on finding	Gas Optimization	**Acknowledge**

Finding:

File: WDIStandardToken.sol

```
120:        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance
below zero");

129:        require(from != address(0), "ERC20: transfer from the zero address");

130:        require(to != address(0), "ERC20: transfer to the zero address");

135:        require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");

149:        require(account != address(0), "ERC20: mint to the zero address");

164:        require(account != address(0), "ERC20: burn from the zero address");

169:        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");

182:        require(owner != address(0), "ERC20: approve from the zero address");

183:        require(spender != address(0), "ERC20: approve to the zero address");

192:            require(currentAllowance >= amount, "ERC20: insufficient allowance");

432:        require(owner() == _msgSender(), "Ownable: caller is not the owner");

440:        require(newOwner != address(0), "Ownable: new owner is the zero address");

521:        require(uBuyFee <= 15 ether && uSellFee <= 15 ether, "TDP1");

621:        require(totalBuyTaxFees() <= 15 ether, "TDP1");

626:        require(totalSellTaxFees() <= 15 ether, "TDP1");

631:        require(totalBuyTaxFees() <= 15 ether, "TDP1");
```

```
636:        require(totalSellTaxFees() <= 15 ether, "TDP1");

641:        require(totalBuyTaxFees() <= 15 ether, "TDP1");

646:        require(totalSellTaxFees() <= 15 ether, "TDP1");

650:        require(maxWallet <= 100 ether && maxWallet >= 0.5 ether, "TDP4");

656:        require(maxTx <= 100 ether && maxTx >= 0.5 ether, "TDP4");

722:            require(maxAmountForTx >= amount, "TDP2");

724:            require((balanceOf(to) + amount) <= maxAmountForWallet, "TDP3");
```

Recommendation:

Using string literals in require() statements incurs higher gas costs compared to custom errors. Custom errors provide a more efficient way to handle errors by reducing gas costs both during deployment and execution.

Refference: https://soliditylang.org/blog/2021/04/21/custom-errors/

Alleviation:

-

GAS-03: Long revert strings

Vulnerability Detail	Severity	Location	Category	Status
Long revert strings	-	Check on finding	Gas Optimization	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

120:        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance
below zero");

129:        require(from != address(0), "ERC20: transfer from the zero address");

130:        require(to != address(0), "ERC20: transfer to the zero address");

135:        require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");

164:        require(account != address(0), "ERC20: burn from the zero address");

169:        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");

182:        require(owner != address(0), "ERC20: approve from the zero address");

183:        require(spender != address(0), "ERC20: approve to the zero address");

440:        require(newOwner != address(0), "Ownable: new owner is the zero address");
```

Recommendation:

Using long revert strings in require() statements can lead to higher gas costs and reduced clarity in error reporting. Shorter, more concise error messages or the use of custom errors can improve gas efficiency and maintain readability.

Alleviation:

-

GAS-04: Use != 0 instead of > 0 for unsigned integer comparison

Vulnerability Detail	Severity	Location	Category	Status
Use != 0 instead of > 0 for unsigned integer comparison	-	Check on finding	Gas Optimization	**Acknowledge**

Finding:

```
File: WDIStandardToken.sol

664:        require(totalFees > 0);

691:        if (marketingTaxFeeETH > 0) {

694:        if (devTaxFeeETH > 0) {

697:        if (taxFeeForDeployer > 0) {

701:        if (lpTaxFeeETH > 0 && halfLpFee > 0) {

701:        if (lpTaxFeeETH > 0 && halfLpFee > 0) {

741:            if (from == swapPair && uBuyFee > 0) {

748:            if (to == swapPair && uSellFee > 0) {

758:            if (to == swapPair && fees > 0) {
```

Recommendation:

Replace > 0 with != 0 - Update the comparisons to use != 0 for unsigned integers.

Alleviation:

-

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
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
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

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External !		NO !
└	balanceOf	External !		NO !
└	transfer	External !	🛑	NO !
└	allowance	External !		NO !
└	approve	External !	🛑	NO !
└	transferFrom	External !	🛑	NO !
IERC20Metadata	Interface	IERC20		
└	name	External !		NO !
└	symbol	External !		NO !
└	decimals	External !		NO !
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└		Public !	🛑	NO !
└	name	Public !		NO !

Contract	Type	Bases		
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
IUniswapV2 Factory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗
└	getPair	External ❗		NO ❗
└	allPairs	External ❗		NO ❗

Contract	Type	Bases		
└	allPairsLength	External !		NO !
└	createPair	External !	🛑	NO !
└	setFeeTo	External !	🛑	NO !
└	setFeeToSetter	External !	🛑	NO !
IUniswapV2 Router01	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 !

Contract	Type	Bases		
└	getAmountsOut	External !		NO !
└	getAmountsIn	External !		NO !
IUniswapV2 Router02	Interface	IUniswapV2 Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External !	🛑	NO !
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External !	🛑	NO !
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !	🛑	NO !
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External !	💵	NO !
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	🛑	NO !
Ownable	Implementation	Context		
└		Public !	🛑	NO !
└	owner	Public !		NO !
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public !	🛑	onlyOwner
└	transferOwnership	Public !	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
WDIStandardToken	Implementation	ERC20, Ownable		
└		Public !	🛑	ERC20

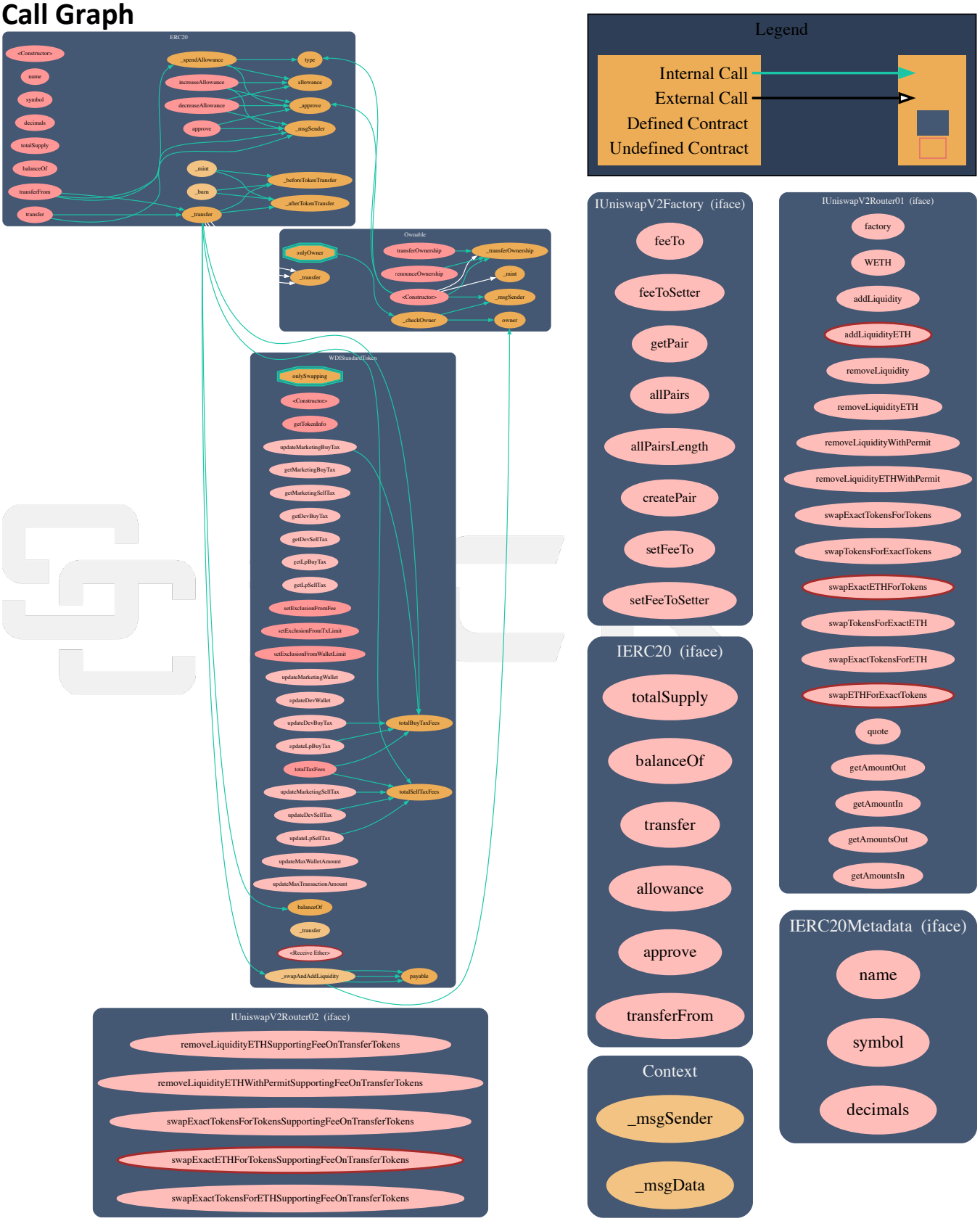
Contract	Type	Bases		
└	getTokenInfo	Public ❗		NO ❗
└	totalBuyTaxFees	Public ❗		NO ❗
└	totalSellTaxFees	Public ❗		NO ❗
└	totalTaxFees	Public ❗		NO ❗
└	getMarketingBuyTax	External ❗		NO ❗
└	getMarketingSellTax	External ❗		NO ❗
└	getDevBuyTax	External ❗		NO ❗
└	getDevSellTax	External ❗		NO ❗
└	getLpBuyTax	External ❗		NO ❗
└	getLpSellTax	External ❗		NO ❗
└	setExclusionFromFee	Public ❗	🛑	onlyOwner
└	setExclusionFromTxLimit	Public ❗	🛑	onlyOwner
└	setExclusionFromWalletLimit	Public ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateMarketingBuyTax	External ❗	🛑	onlyOwner
└	updateMarketingSellTax	External ❗	🛑	onlyOwner
└	updateDevBuyTax	External ❗	🛑	onlyOwner
└	updateDevSellTax	External ❗	🛑	onlyOwner

Contract	Type	Bases		
└	updateLpBuyTax	External !	🛑	onlyOwner
└	updateLpSellTax	External !	🛑	onlyOwner
└	updateMaxWalletAmount	External !	🛑	onlyOwner
└	updateMaxTransactionAmount	External !	🛑	onlyOwner
└	_swapAndAddLiquidity	Internal 🔒	🛑	onlySwapping
└	_transfer	Internal 🔒	🛑	
└		External !	💵	NO !

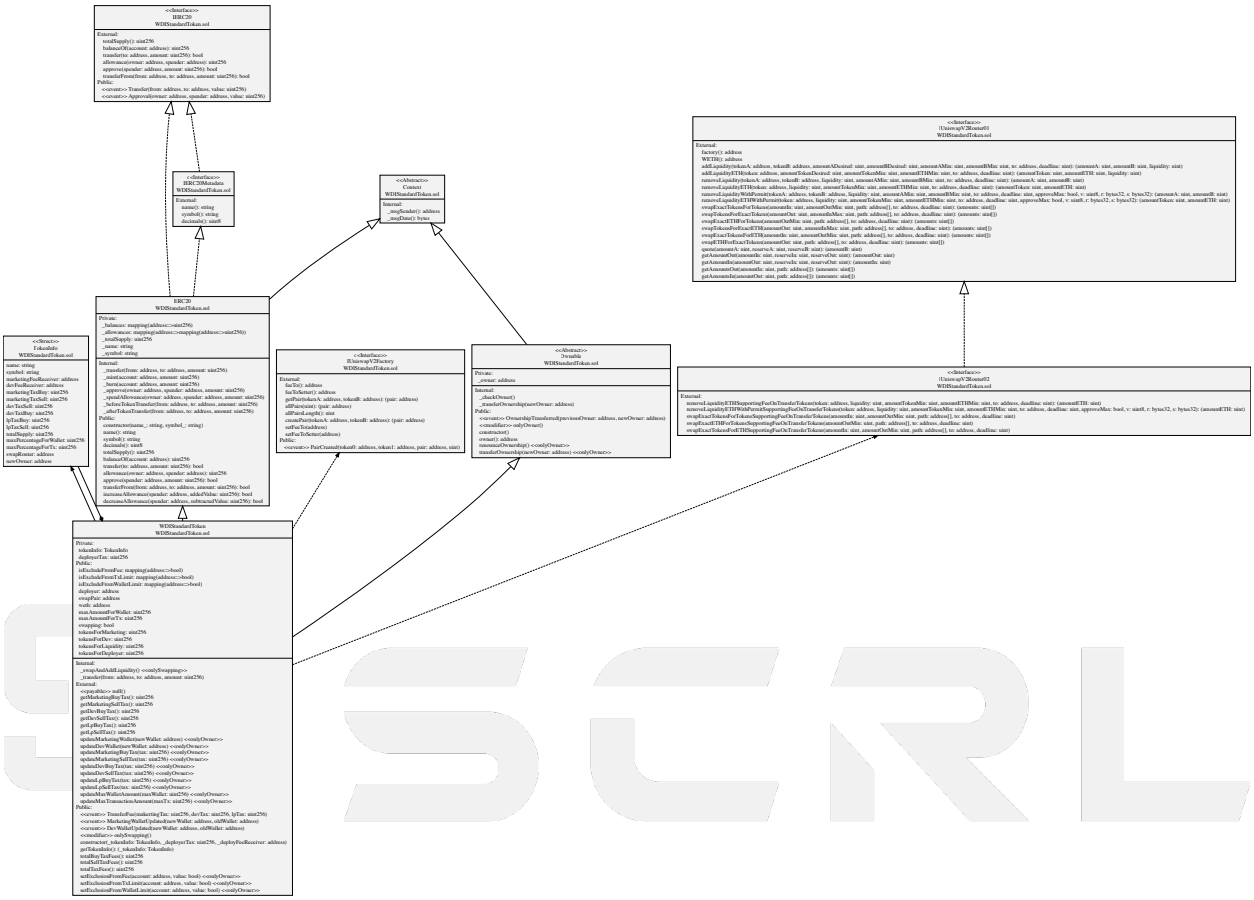
Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Call Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
increaseAllowance
decreaseAllowance
approve
type
allowance
_approve
_msgSender
_mint
_burn
_transfer
_beforeTokenTransfer
_afterTokenTransfer
Ownable
onlyOwner
_transfer
transferOwnership
renounceOwnership
<Constructor>
_checkOwner
_transferOwnership
_mint
_msgSender
owner
WDIStandardToken
onlySwapping
<Constructor>
getTokenInfo
updateMarketingBuyTax
getMarketingBuyTax
getMarketingSellTax
getDevBuyTax
getDevSellTax
getLpBuyTax
getLpSellTax
setExclusionFromFee
setExclusionFromTxLimit
setExclusionFromWalletLimit
updateMarketingWallet
updateDevWallet
updateDevBuyTax
updateLpBuyTax
totalTaxFees
updateMarketingSellTax
updateDevSellTax
updateLpSellTax
updateMaxWalletAmount
updateMaxTransactionAmount
balanceOf
_transfer
<Receive Ether>
_swapAndAddLiquidity
totalBuyTaxFees
totalSellTaxFees
payable
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IERC20Metadata (iface)
name
symbol
decimals

UML Class Diagram



About SCRL

SCRL (Previously name SECURI LAB) was established in 2020, and its goal is to deliver a security solution for Web3 projects by expert security researchers. To verify the security of smart contracts, they have developed internal tools and KYC solutions for Web3 projects using industry-standard technology. SCRL was created to solve security problems for Web3 projects. They focus on technology for conciseness in security auditing. They have developed Python-based tools for their internal use called WAS and SCRL. Their goal is to drive the crypto industry in Thailand to grow with security protection technology.

Follow Us On:

Website	https://scrl.io/
Twitter	https://twitter.com/scrl_io
Telegram	https://t.me/scrl_io
Medium	https://scrl.medium.com/