SCRL

Full Audit Report

BOTiFi.Ai Security Assessment

BOTiFi.Ai Security Assessment

FULL AUDIT REPORT

Security Assessment by SCRL on **Friday, October 6, 2023**

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

Executive Summary

For this security assessment, SCRL received a request on Sunday, October 1, 2023

Client	Language	Audit Method	Confidential	Network Chain	Contract
BOTiFi.Ai	**Solidity**	**Whitebox**	**Public**	**BNB-Chain**	**0xb1bf223D00a0c0086EE0AE6c8B1fcb1c2E4a479C**

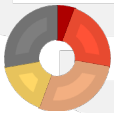
Report Version	Twitter	Telegram	Website
1.1	https://twitter.com/BoTiFi_Ai	https://t.me/BOTiFi_Ai	https://www.botifi.ai/

Scoring:

Scoring

0 1 2 3 4 5 6 7 8 9 10

Vulnerability Summary



Total Findings	Unresolved	Resolved	Mitigate	Acknowledge	Decline
18	18	0	0	18	0

1	Critical	1 Unresolved	Critical severity is assigned to security vulnerabilities that pose a severe threat to the smart contract and the entire blockchain ecosystem.
4	High	4 Unresolved	High-severity issues should be addressed quickly to reduce the risk of exploitation and protect users' funds and data.
5	Medium	5 Unresolved	It's essential to fix medium-severity issues in a reasonable timeframe to enhance the overall security of the smart contract.
3	Low	3 Unresolved	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.
5	Informational	5 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
3	Gas-optimization	3 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
src/BOTiFiAi.sol	cf3c152cab9785d6046ba3c418dc0f5f1c8ab858

Audit Version History:

Version	Date	Description
1.0	Monday, October 2, 2023	Preliminary Report
1.1	Friday, October 6, 2023	Full Audit Report

Audit information:

Request Date	Audit Date	Re-assessment Date
Sunday, October 1, 2023	Monday, October 2, 2023	-

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: 2023.10.06 19:34:52 +07'00'

Disclaimer

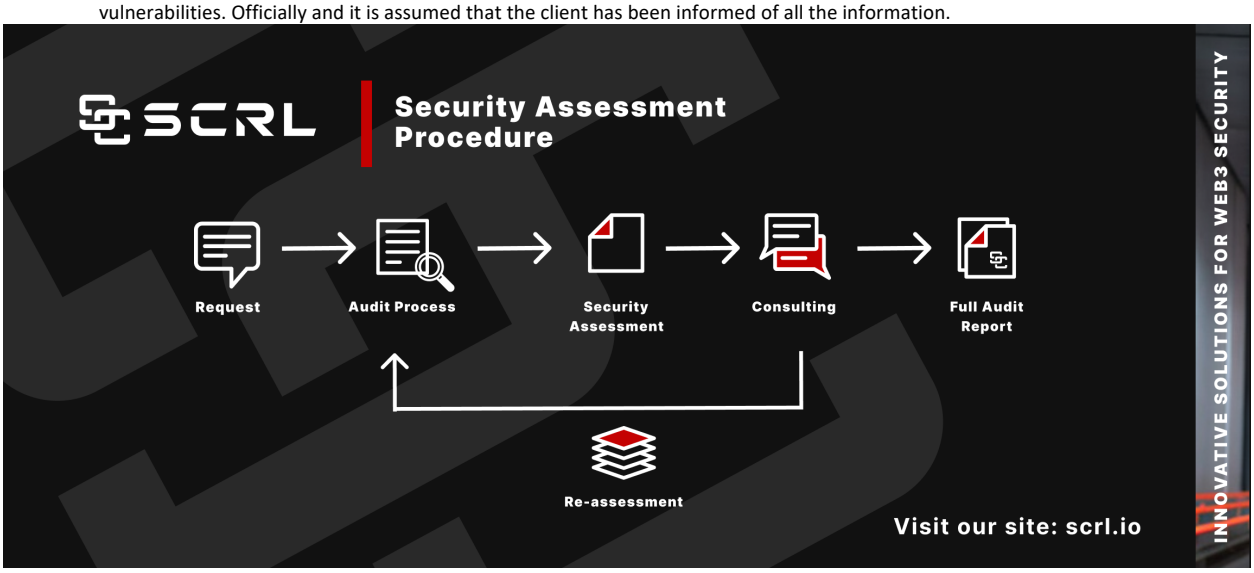
Regarding this security assessment, there are no guarantees about the security of the program instruction received from the client is hereinafter referred to as "**Source code**".
And **SCRL** hereinafter referred to as "**Service Provider**", the **Service Provider** will not be held liable for any 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

Gas-optimization | Informational | Very Low | Low | Medium | High | Critical

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 Code Detail

Source Units Analyzed: **1**

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

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍🎨	src/BOTiFiAi.sol	5	6	1094	799	364	386	424	💰💠🔆Σ
📝🔍🎨	**Totals**	**5**	**6**	**1094**	**799**	**364**	**386**	**424**	💰💠🔆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	6	3

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
93	6

External	Internal	Private	Pure	View
69	70	1	10	36

StateVariables

Total	🌐Public
35	2

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.8.9`		`yes`		

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

TryCatch	Σ Unchecked
	`yes`

Vulnerability Findings

ID	Vulnerability Detail	Severity	Category	Status
CEN-01	**Owner can set buy & sell tax up to 100%**	Critical	Centralization	**Acknowledge**
SEC-01	If **Owner** Set 'setSellTax' and **add address to exclude tax list** with function 'exclude' only **exclude tax address list** can **trading with out tax**	High	Centralization	**Acknowledge**
SEC-02	If 'setSellTax' & 'setBuyTax' **> 49%**, the tokens **will not be able to be trading**	High	Centralization	**Acknowledge**
SEC-03	Centralization Risk	High	Centralization	**Acknowledge**
SEC-04	No-slippage-check	High	Security Risk	**Acknowledge**
SEC-05	Unchecked low-level calls (unchecked-lowlevel)	Medium	Security Risk	**Acknowledge**
SEC-06	Unused return values (unused-return)	Medium	Best Practices	**Acknowledge**
SEC-07	Imprecise arithmetic operations order (divide-before-multiply)	Medium	Logical Issue	**Acknowledge**
SEC-08	Dangerous strict equalities (incorrect-equality)	Medium	Best Practices	**Acknowledge**
SEC-09	Uninitialized local variables (uninitialized-local)	Medium	Best Practices	**Acknowledge**
SEC-10	Reentrancy vulnerabilities leading to out-of-order Events (reentrancy-events)	Low	Best Practices	**Acknowledge**
SEC-11	Benign reentrancy vulnerabilities (reentrancy-benign)	Low	Best Practices	**Acknowledge**
SEC-12	Contract function does not emit event after the value is set (pess-event-setter)	Low	Best Practices	**Acknowledge**
SEC-13	Unused state variables (unused-state)	Informational	Best Practices	**Acknowledge**
SEC-14	Conformance to numeric notation best practices (too-many-digits)	Informational	Best Practices	**Acknowledge**
SEC-15	Low level calls (low-level-calls)	Informational	Best Practices	**Acknowledge**
SEC-16	Unlocked pragma	Informational	Best Practices	**Acknowledge**
SEC-17	Use-nested-if	Informational	Optimization	**Acknowledge**

GAS-01	Use Custom Errors	Gas-optimization	Gas Optimization	**Acknowledge**
GAS-02	Long revert strings	Gas-optimization	Gas Optimization	**Acknowledge**
GAS-03	Use != 0 instead of > 0 for unsigned integer comparison	Gas-optimization	Gas Optimization	**Acknowledge**

CEN-01: Owner can set buy & sell tax up to 100%

Vulnerability Detail	Severity	Location	Category	Status
Owner can set buy & sell tax up to 100%	Critical	Check on finding	Centralization	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

1043:       function setBuyTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {

1053:       function setSellTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {
```

Scenario: In cases where the **Buy/Sell Tax function does not have a maximum value function set**, the **Owner can set the value up to 100%,** resulting in the **trader not receiving any trade. And the benefit will go to the account that is designated to receive the tax fee**. This is a **serious centralized risk**.

We recommend checking Tax before every trade. And being able to set taxes as high as 100% is a very high risk. In the event that you purchase such coins It may prevent you from selling. or when making a sale The beneficiary belongs to the account list that receives the Tax Fees.

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.

and should add the maximum value setting function of 'setSellTax' & 'setBuyTax'

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-01: If Owner Set 'setSellTax' and add address to exclude tax list with function 'exclude' only exclude tax address list can trading with out tax

Vulnerability Detail	Severity	Location	Category	Status
If **Owner** Set 'setSellTax' and **add address to exclude tax list** with function 'exclude' only **exclude tax address list** can **trading with out tax**	High	Check on finding	Centralization	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

825:   contract BOTiFiAi is ERC20, Ownable, Pausable {

1053:       function setSellTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {

1027:       function exclude(address account) public onlyOwner {
```

Scenario: In case the function **'setSellTax'** is called with the **value already set**. **All transactions will be subject to the Tax calculation mechanism**. **Tax calculation will be exempted** for **accounts that are set** by the **'exclude'** function, such as the **Owner making Set the Tax very high to prevent trading by those who want to sell. But at the same time, tax-exempt addresses can sell a lot with out any tax**. This may create a risk regarding **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.

and should add the maximum value setting function of 'setSellTax' & 'setBuyTax'

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-02: If 'setSellTax' & 'setBuyTax' > 49%, the tokens will not be able to be trading

Vulnerability Detail	Severity	Location	Category	Status
If 'setSellTax' & 'setBuyTax' **> 49%**, the tokens **will not be able to be trading**	High	Check on finding	Centralization	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

1043:       function setBuyTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {

1053:       function setSellTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {
```

Scenario: When setting values in the 'setSellTax' or 'setBuyTax' functions, it may result in trading not being possible due to slipage greater than 49%. However, we have found that there have a function to prevent slipage from being checked. Reference with "SEC-04 No-slippage-check"

Recommendation:
The values for 'setSellTax' and 'setBuyTax' should not be set to maximum 49% and a function should be added to set values for Maximum 'setSellTax' & 'setBuyTax'. This will prevent potential problems.

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-03: Centralization Risk

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

Finding:

```
File: BOTiFiAi.sol

825:   contract BOTiFiAi is ERC20, Ownable, Pausable {

1014:       function triggerTax() public onlyOwner {

1027:       function exclude(address account) public onlyOwner {

1035:       function removeExclude(address account) public onlyOwner {

1043:       function setBuyTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {

1053:       function setSellTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {

1064:       function setTaxWallets(address dev, address marketing, address charity)
public onlyOwner {

1073:       function enableTax() public onlyOwner {

1081:       function disableTax() public onlyOwner {

```
```
```

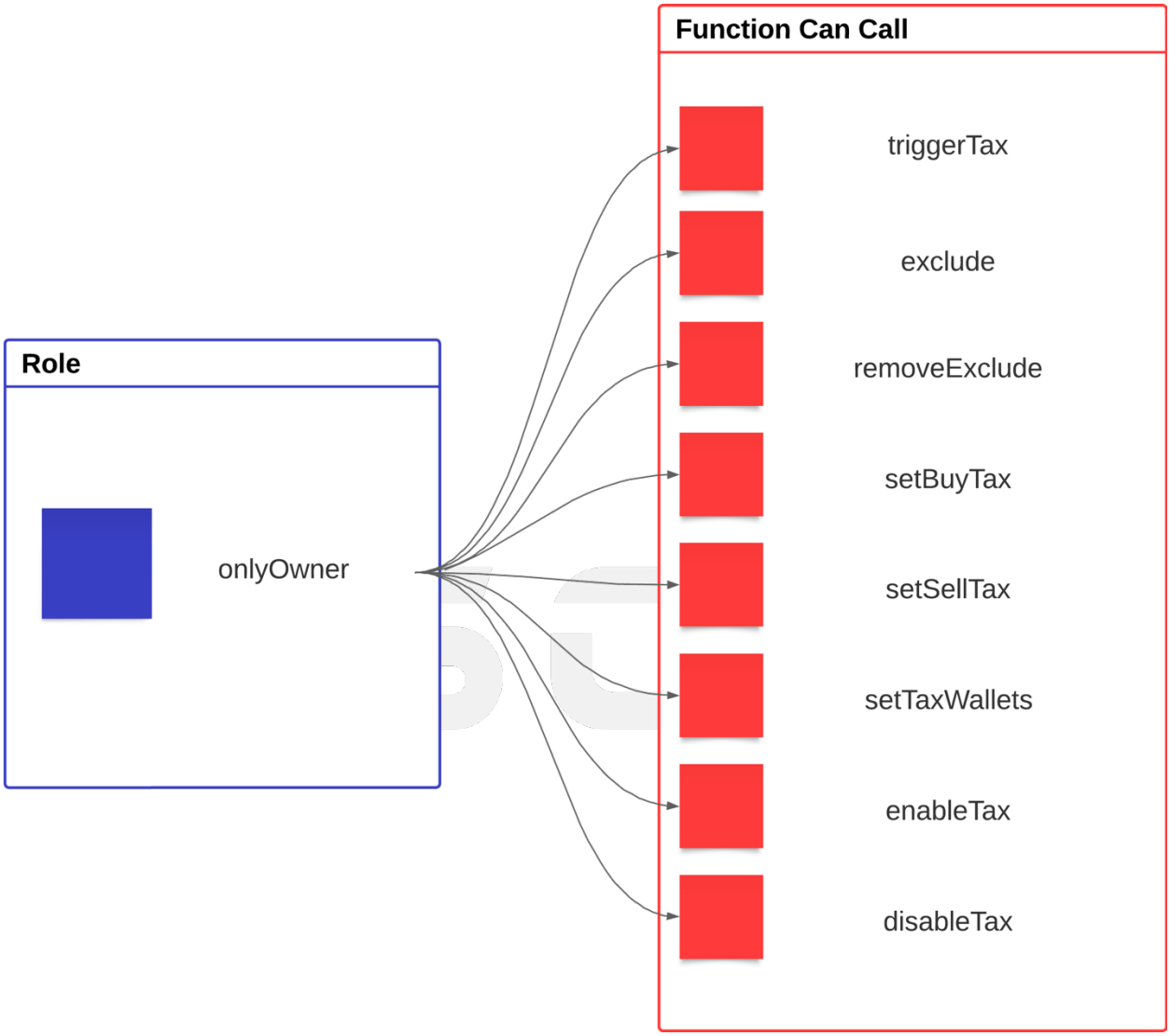
Explain Function Capability:

The contract provides several functions:

1. **triggerTax()** (Line 1014):
    - This function can be called by the contract owner (onlyOwner) to trigger the tax handling functionality.
    - It typically gets called when tokens are added to the liquidity pool or when the contract needs to redistribute taxes.
    - It ensures that the appropriate taxes are collected and processed according to the contract's tax logic.
2. **exclude(address account)** (Line 1027):
    - This function allows the contract owner to exclude a specific account from taxation.
    - When an account is excluded, it can transfer tokens without incurring taxes.
    - The function takes an **account** address as its parameter and adds it to the list of excluded accounts.
3. **removeExclude(address account)** (Line 1035):
    - This function is used by the contract owner to re-enable taxation for an account that was previously excluded using the **exclude** function.
    - It takes an **account** address as its parameter and removes it from the list of excluded accounts.
4. **setBuyTax(uint256 dev, uint256 marketing, uint256 liquidity, uint256 charity)** (Line 1043):
    - This function allows the contract owner to set tax percentages for buying tokens.
    - It takes four parameters: **dev**, **marketing**, **liquidity**, and **charity**, representing the tax percentages for different categories.
    - The taxes are applied when users buy tokens from the contract.
5. **setSellTax(uint256 dev, uint256 marketing, uint256 liquidity, uint256 charity)** (Line 1053):
    - Similar to **setBuyTax**, this function allows the contract owner to set tax percentages for selling tokens.
    - It also takes four parameters: **dev**, **marketing**, **liquidity**, and **charity**, representing the tax percentages for different categories.
    - The taxes are applied when users sell tokens to the contract.
6. **setTaxWallets(address dev, address marketing, address charity)** (Line 1064):
    - This function enables the contract owner to set the destination wallets for different tax categories: **dev**, **marketing**, and **charity**.
    - It allows the owner to specify where the collected taxes for each category should be sent.
7. **enableTax()** (Line 1073):
    - This function is used by the contract owner to globally enable the tax mechanism.
    - When taxes are enabled, they will be applied to all transfers according to the configured tax percentages.
    - It ensures that the **taxStatus** variable becomes **true**, enabling taxation.
8. **disableTax()** (Line 1081):
    - Conversely, this function allows the contract owner to globally disable the tax mechanism.
    - When taxes are disabled, no taxes will be applied to transfers.
    - It ensures that the **taxStatus** variable becomes **false**, disabling taxation.

These functions provide the contract owner with control over various aspects of the token's taxation and behavior, allowing for customization and adjustment of tax-related parameters and features.

Centralization Risk Contract BOTiFiAi



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:
BOTiFiAi Team has acknowledge this issue.

SEC-04: No-slippage-check

Vulnerability Detail	Severity	Location	Category	Status
No-slippage-check	High	Check on finding	Security Risk	Acknowledge

Finding:

```
FIle: BOTiFiAi.sol

L941:   uniswapV2Router02.swapExactTokensForETH(
```

No slippage check in a Uniswap v2/v3 trade
CWE-682: Incorrect Calculation

Recommendation:
We recommend that you add Slippage Protection functionality to prevent serious security issues such as sandwich attacks.

https://uniswapv3book.com/docs/milestone_3/slippage-protection/

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-05: Unchecked low-level calls (unchecked-lowlevel)

Vulnerability Detail	Severity	Location	Category	Status
Unchecked low-level calls (unchecked-lowlevel)	Medium	Check on finding	Security Risk	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

❌ BOTiFiAi.handleTax(address,address,uint256) (src/BOTiFiAi.sol:891-995) ignores
return value by taxWallets[charity].call{value: charityETH}() (src/BOTiFiAi.sol#977)
```

Recommendation:
Ensure that the return value of a low-level call is checked or logged.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-low-level-calls

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-06: Unused return values (unused-return)

Vulnerability Detail	Severity	Location	Category	Status
Unused return values (unused-return)	Medium	Check on finding	Best Practices	Acknowledge

Finding:

```
FIle: BOTiFiAi.sol

❌ BOTiFiAi.handleTax(address,address,uint256) (src/BOTiFiAi.sol:891-995) ignores
return value by
uniswapV2Router02.swapExactTokensForETH(toSell,0,sellPath,address(this),block.timestam
p) (src/BOTiFiAi.sol#941-947)
```

Recommendation:
Ensure that all the return values of the function calls are used.

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

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-07: Imprecise arithmetic operations order (divide-before-multiply)

Vulnerability Detail	Severity	Location	Category	Status
Imprecise arithmetic operations order (divide-before-multiply)	Medium	Check on finding	Logical Issue	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

❌ BOTiFiAi.handleTax(address,address,uint256) (src/BOTiFiAi.sol:891-995) performs a
multiplication on the result of a division:
    • baseUnit = amount / denominator (src/BOTiFiAi.sol#898)
    • charityTokens += baseUnit * buyTaxes[charity] (src/BOTiFiAi.sol#912)
```

Recommendation:
Consider ordering multiplication before division.

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

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-08: Dangerous strict equalities (incorrect-equality)

Vulnerability Detail	Severity	Location	Category	Status
Dangerous strict equalities (incorrect-equality)	Medium	Check on finding	Best Practices	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

✗ BOTiFiAi.handleTax(address,address,uint256) (src/BOTiFiAi.sol:891-995) uses a
dangerous strict equality:
    • taxSum == 0 (src/BOTiFiAi.sol#930)
```

Recommendation:
Don't use strict equality to determine if an account has enough Ether or tokens.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-09: Uninitialized local variables (uninitialized-local)

Vulnerability Detail	Severity	Location	Category	Status
Uninitialized local variables (uninitialized-local)	Medium	Check on finding	Best Practices	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

✕ BOTiFiAi.handleTax(address,address,uint256).tax (src/BOTiFiAi.sol:897) is a local
variable never initialized
```

Recommendation:
Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-10: Reentrancy vulnerabilities leading to out-of-order Events (reentrancy-events)

Vulnerability Detail	Severity	Location	Category	Status
Reentrancy vulnerabilities leading to out-of-order Events (reentrancy-events)	Low	Check on finding	Best Practices	Acknowledge

Finding:

```
FIle: BOTiFiAi.sol

❌ Reentrancy in BOTiFiAi._transfer(address,address,uint256) (src/BOTiFiAi.sol:997-
1009):
    • amount = handleTax(sender,recipient,amount) (src/BOTiFiAi.sol#1005)
    •
uniswapV2Router02.swapExactTokensForETH(toSell,0,sellPath,address(this),block.timestam
p) (src/BOTiFiAi.sol#941-947)
    • (amountToken,amountETH,liquidity) = uniswapV2Router02.addLiquidityETH{value:
liquidityETH}(address(this),liquidityToken,0,0,taxWallets[liquidity],block.timestamp)
(src/BOTiFiAi.sol#960-967)
    • taxWallets[marketing].call{value: marketingETH}() (src/BOTiFiAi.sol#975)
    • taxWallets[dev].call{value: devETH}() (src/BOTiFiAi.sol#976)
    • taxWallets[charity].call{value: charityETH}() (src/BOTiFiAi.sol#977)
    • taxWallets[marketing].call{value: ethGained - (marketingETH + devETH +
liquidityETH + charityETH)}() (src/BOTiFiAi.sol#980)
    • amount = handleTax(sender,recipient,amount) (src/BOTiFiAi.sol#1005)
    • (amountToken,amountETH,liquidity) = uniswapV2Router02.addLiquidityETH{value:
liquidityETH}(address(this),liquidityToken,0,0,taxWallets[liquidity],block.timestamp)
(src/BOTiFiAi.sol#960-967)
    • taxWallets[marketing].call{value: marketingETH}() (src/BOTiFiAi.sol#975)
    • taxWallets[dev].call{value: devETH}() (src/BOTiFiAi.sol#976)
    • taxWallets[charity].call{value: charityETH}() (src/BOTiFiAi.sol#977)
    • taxWallets[marketing].call{value: ethGained - (marketingETH + devETH +
liquidityETH + charityETH)}() (src/BOTiFiAi.sol#980)
    • Transfer(sender,recipient,amount) (src/BOTiFiAi.sol#353)
    • super._transfer(sender,recipient,amount) (src/BOTiFiAi.sol#1008)
```

Recommendation:
Apply the [`check-effects-interactions` pattern](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-11: Benign reentrancy vulnerabilities (reentrancy-benign)

Vulnerability Detail	Severity	Location	Category	Status
Benign reentrancy vulnerabilities (reentrancy-benign)	Low	Check on finding	Best Practices	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

❌ Reentrancy in BOTiFiAi.handleTax(address,address,uint256) (src/BOTiFiAi.sol:891-
995):
    • _transfer(from,address(this),tax) (src/BOTiFiAi.sol#920)
    •
uniswapV2Router02.swapExactTokensForETH(toSell,0,sellPath,address(this),block.timestam
p) (src/BOTiFiAi.sol#941-947)
    • (amountToken,amountETH,liquidity) = uniswapV2Router02.addLiquidityETH{value:
liquidityETH}(address(this),liquidityToken,0,0,taxWallets[liquidity],block.timestamp)
(src/BOTiFiAi.sol#960-967)
    • taxWallets[marketing].call{value: marketingETH}() (src/BOTiFiAi.sol#975)
    • taxWallets[dev].call{value: devETH}() (src/BOTiFiAi.sol#976)
    • taxWallets[charity].call{value: charityETH}() (src/BOTiFiAi.sol#977)
    • taxWallets[marketing].call{value: ethGained - (marketingETH + devETH +
liquidityETH + charityETH)}() (src/BOTiFiAi.sol#980)
    • _transfer(from,address(this),tax) (src/BOTiFiAi.sol#920)
    • (amountToken,amountETH,liquidity) = uniswapV2Router02.addLiquidityETH{value:
liquidityETH}(address(this),liquidityToken,0,0,taxWallets[liquidity],block.timestamp)
(src/BOTiFiAi.sol#960-967)
    • taxWallets[marketing].call{value: marketingETH}() (src/BOTiFiAi.sol#975)
    • taxWallets[dev].call{value: devETH}() (src/BOTiFiAi.sol#976)
    • taxWallets[charity].call{value: charityETH}() (src/BOTiFiAi.sol#977)
    • taxWallets[marketing].call{value: ethGained - (marketingETH + devETH +
liquidityETH + charityETH)}() (src/BOTiFiAi.sol#980)
    • _approve(address(this),address(uniswapV2Router02),toSell) (src/BOTiFiAi.sol#939)
    • _allowances[owner][spender] = amount (src/BOTiFiAi.sol#428)
```

Recommendation:
Apply the [`check-effects-interactions` pattern](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-12: Contract function does not emit event after the value is set (pess-event-setter)

Vulnerability Detail	Severity	Location	Category	Status
Contract function does not emit event after the value is set (pess-event-setter)	Low	Check on finding	Best Practices	Acknowledge

Finding:

```
FIle: BOTiFiAi.sol

❌ Setter function BOTiFiAi.setBuyTax(uint256,uint256,uint256,uint256)
(src/BOTiFiAi.sol:1043-1048) does not emit an event
❌ Setter function BOTiFiAi.setSellTax(uint256,uint256,uint256,uint256)
(src/BOTiFiAi.sol:1053-1059) does not emit an event
❌ Setter function BOTiFiAi.setTaxWallets(address,address,address)
(src/BOTiFiAi.sol:1064-1068) does not emit an event
```

Recommendation:
Emit events in setter functions

Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/event_setter.md

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-13: Unused state variables (unused-state)

Vulnerability Detail	Severity	Location	Category	Status
Unused state variables (unused-state)	Informational	Check on finding	Best Practices	Acknowledge

Finding:

```
FIle: BOTiFiAi.sol

❌ BOTiFiAi.charityTaxBuy (src/BOTiFiAi.sol:838) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.charityTaxSell (src/BOTiFiAi.sol:843) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.charityTaxWallet (src/BOTiFiAi.sol:848) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.devTaxBuy (src/BOTiFiAi.sol:835) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.devTaxSell (src/BOTiFiAi.sol:840) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.devTaxWallet (src/BOTiFiAi.sol:845) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.liquidityTaxBuy (src/BOTiFiAi.sol:837) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
uint256 private liquidityTaxSell
❌ BOTiFiAi.liquidityTaxWallet (src/BOTiFiAi.sol:847) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.marketingTaxBuy (src/BOTiFiAi.sol:836) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.marketingTaxSell (src/BOTiFiAi.sol:841) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
❌ BOTiFiAi.marketingTaxWallet (src/BOTiFiAi.sol:846) is never used in BOTiFiAi
(src/BOTiFiAi.sol#825-1095)
```

Recommendation:
Remove unused state variables.

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

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-14: Conformance to numeric notation best practices (too-many-digits)

Vulnerability Detail	Severity	Location	Category	Status
Conformance to numeric notation best practices (too-many-digits)	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

❌ BOTiFiAi.slitherConstructorVariables() (src/BOTiFiAi.sol:825-1095) uses literals
with too many digits:
    • swapThreshold = 500000000000 (src/BOTiFiAi.sol#833)
```

Recommendation:

Use:
- [Ether suffix](https://solidity.readthedocs.io/en/latest/units-and-global-variables.html#ether-units),
- [Time suffix](https://solidity.readthedocs.io/en/latest/units-and-global-variables.html#time-units), or
- [The scientific notation](https://solidity.readthedocs.io/en/latest/types.html#rational-and-integer-literals)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Alleviation:

BOTiFiAi Team has acknowledge this issue.

SEC-15: Low level calls (low-level-calls)

Vulnerability Detail	Severity	Location	Category	Status
Low level calls (low-level-calls)	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

✗ Low level call in BOTiFiAi.handleTax(address,address,uint256)
(src/BOTiFiAi.sol:891-995):
    • taxWallets[marketing].call{value: marketingETH}() (src/BOTiFiAi.sol#975)
    • taxWallets[dev].call{value: devETH}() (src/BOTiFiAi.sol#976)
    • taxWallets[charity].call{value: charityETH}() (src/BOTiFiAi.sol#977)
    • taxWallets[marketing].call{value: ethGained - (marketingETH + devETH +
liquidityETH + charityETH)}() (src/BOTiFiAi.sol#980)
```

Recommendation:
Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-16: Unlocked pragma

Vulnerability Detail	Severity	Location	Category	Status
Unlocked pragma	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

4:   pragma solidity ^0.8.9;
```

Recommendation:
Consider locking the compiler version to prevent unexpected behavior.

Reference: https://solidity.readthedocs.io/en/latest/layout-of-source-files.html#pragma-solidity

Alleviation:
BOTiFiAi Team has acknowledge this issue.

SEC-17: Use-nested-if

Vulnerability Detail	Severity	Location	Category	Status
Use-nested-if	Informational	Check on finding	Optimization	**Acknowledge**

Finding:

```
FIle: BOTiFiAi.sol

896:   if(!isExcluded(from) && !isExcluded(to)) {
```

Recommendation:
Using nested is cheaper than using && multiple check combinations. There are more advantages, such as easier to read code and better coverage reports.

Reference: https://code4rena.com/reports/2023-01-biconomy#g-18-use-nested-if-and-avoid-multiple-check-combinations

Alleviation:
BOTiFiAi Team has acknowledge this issue.

GAS-01: Use Custom Errors

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

Finding:

```
File: BOTiFiAi.sol

273:            require(currentAllowance >= amount, "ERC20: transfer amount exceeds
allowance");

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

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

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

347:            require(senderBalance >= amount, "ERC20: transfer amount exceeds
balance");

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

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

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

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

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

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

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

582:            require(!paused(), "Pausable: paused");

594:            require(paused(), "Pausable: not paused");
```

```
1028:           require(!isExcluded(account), "CoinToken: Account is already
excluded");

1036:           require(isExcluded(account), "CoinToken: Account is not excluded");

1074:           require(!taxStatus, "CoinToken: Tax is already enabled");

1082:           require(taxStatus, "CoinToken: Tax is already disabled");

```
```
```

Recommendation:

Instead of using error strings, to reduce deployment and runtime cost, you should use Custom Errors. This would save both deployment and runtime cost.

[Source](https://blog.soliditylang.org/2021/04/21/custom-errors/)

Alleviation:

BOTiFiAi Team has acknowledge this issue.

GAS-02: Long revert strings

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

Finding:

```
File: BOTiFiAi.sol

273:            require(currentAllowance >= amount, "ERC20: transfer amount exceeds
allowance");

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

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

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

347:            require(senderBalance >= amount, "ERC20: transfer amount exceeds
balance");

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

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

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

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

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

1028:            require(!isExcluded(account), "CoinToken: Account is already
excluded");

1036:            require(isExcluded(account), "CoinToken: Account is not excluded");

1074:            require(!taxStatus, "CoinToken: Tax is already enabled");
```

```
1082:            require(taxStatus, "CoinToken: Tax is already disabled");
```

Alleviation:

BOTiFiAi Team has acknowledge this issue.

GAS-03: 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: BOTiFiAi.sol

905:                    if(tax > 0) {

919:                    if(tax > 0) {

971:                        if(remainingTokens > 0) {

979:                        if(ethGained - (marketingETH + devETH + liquidityETH +
charityETH) > 0) {

```
```
```

Alleviation:

BOTiFiAi Team has acknowledge this issue.

SWC Findings

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

Contract	Type	Bases		
└	_setOwner	Internal 🔒	🛑	
Pausable	Implementation	Context		
└		Public ❗	🛑	NO ❗
└	paused	Public ❗		NO ❗
└	_pause	Internal 🔒	🛑	whenNot Paused
└	_unpause	Internal 🔒	🛑	whenPa used
IUniswapV 2Pair	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 ❗

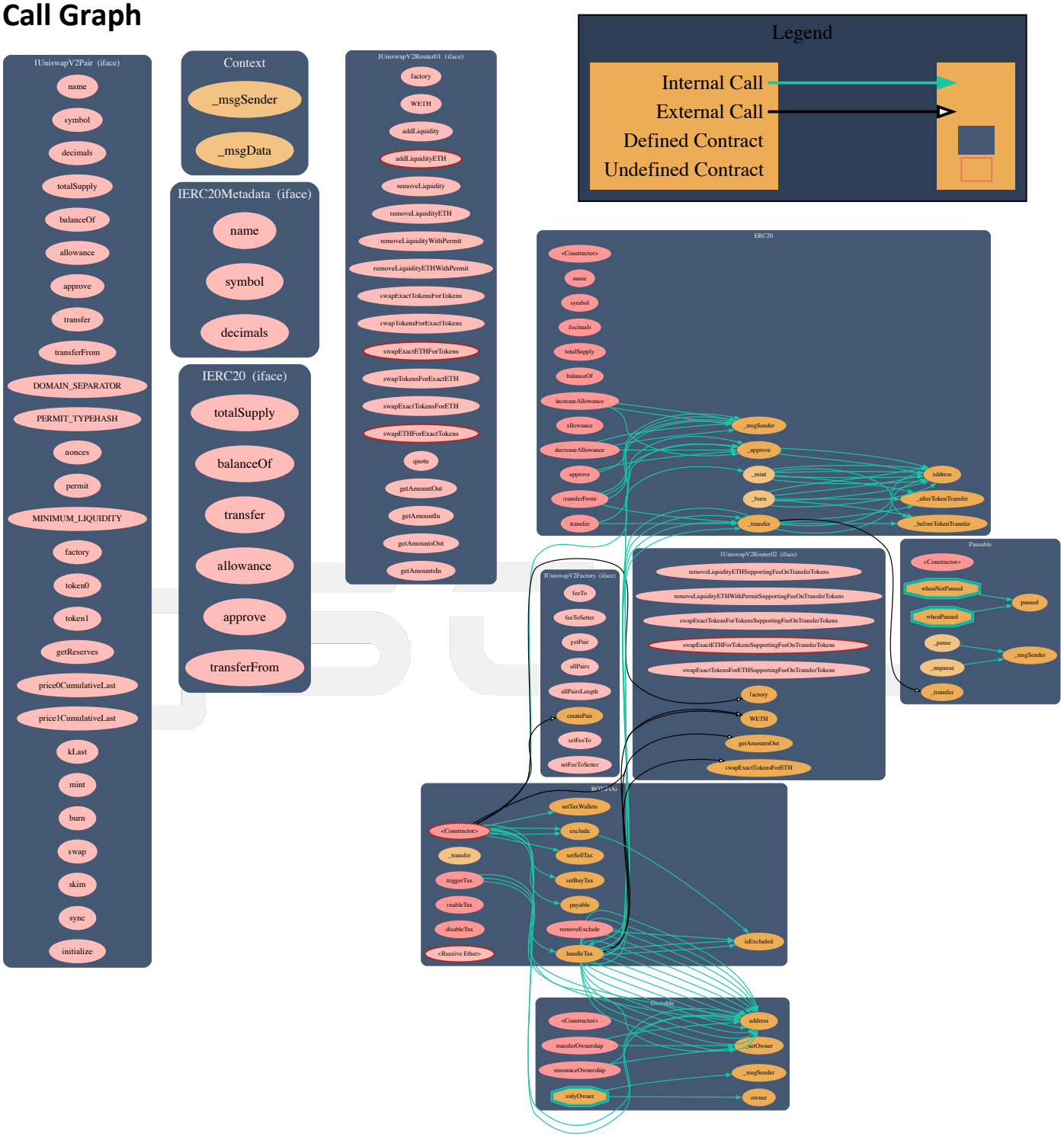
Contract	Type	Bases		
└	factory	External !		NO !
└	token0	External !		NO !
└	token1	External !		NO !
└	getReserves	External !		NO !
└	price0CumulativeLast	External !		NO !
└	price1CumulativeLast	External !		NO !
└	kLast	External !		NO !
└	mint	External !	🛑	NO !
└	burn	External !	🛑	NO !
└	swap	External !	🛑	NO !
└	skim	External !	🛑	NO !
└	sync	External !	🛑	NO !
└	initialize	External !	🛑	NO !
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 !

Contract	Type	Bases		
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		

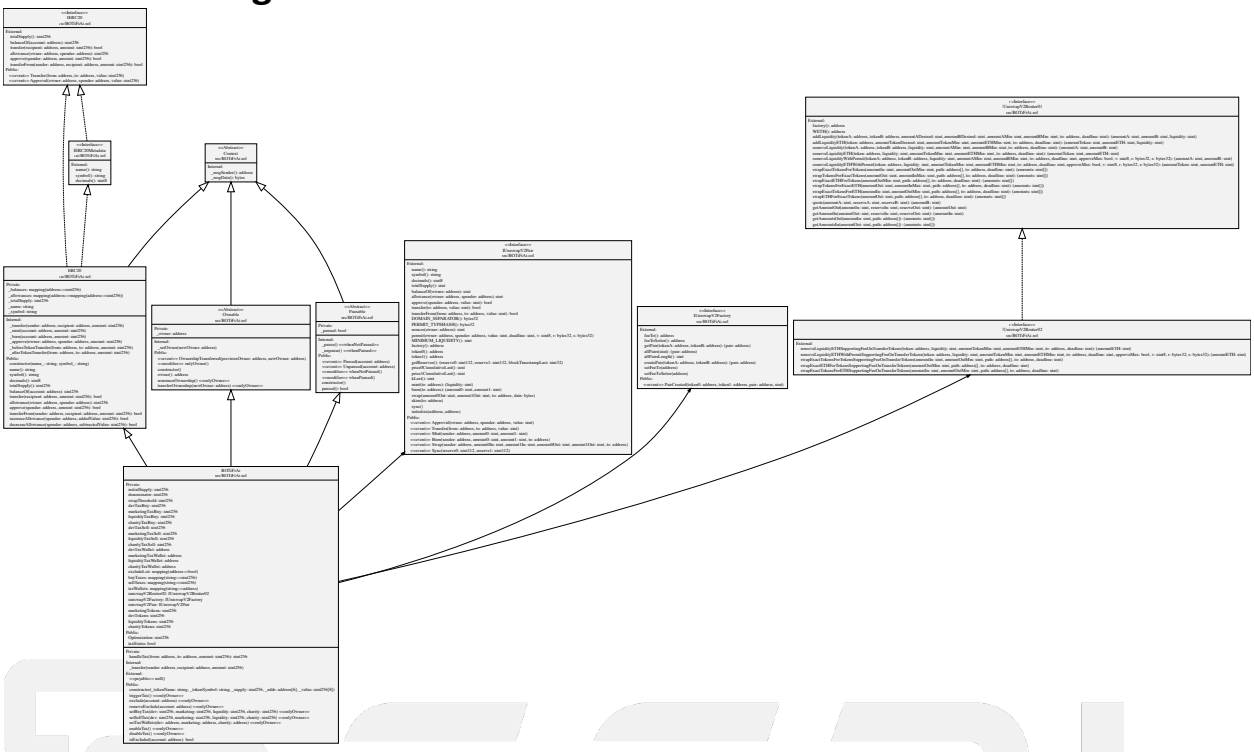
Contract	Type	Bases		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
BOTiFiAi	Implementation	ERC20, Ownable, Pausable		
└		Public ❗	💵	ERC20
└	handleTax	Private 🔐	🛑	
└	_transfer	Internal 🔒	🛑	
└	triggerTax	Public ❗	🛑	onlyOwner
└	exclude	Public ❗	🛑	onlyOwner
└	removeExclude	Public ❗	🛑	onlyOwner
└	setBuyTax	Public ❗	🛑	onlyOwner
└	setSellTax	Public ❗	🛑	onlyOwner
└	setTaxWallets	Public ❗	🛑	onlyOwner
└	enableTax	Public ❗	🛑	onlyOwner

Contract	Type	Bases		
└	disableTax	Public !	🛑	onlyOwner
└	isExcluded	Public !		NO !
└		External !	💵	NO !

Call Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Pair (iface)
name
symbol
decimals
totalSupply
balanceOf
allowance
approve
transfer
transferFrom
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
nonces
permit
MINIMUM_LIQUIDITY
factory
token0
token1
getReserves
price0CumulativeLast
price1CumulativeLast
kLast
mint
burn
swap
skim
sync
initialize
Context
_msgSender
_msgData
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
increaseAllowance
allowance
decreaseAllowance
approve
transferFrom
transfer
_msgSender
_approve
_mint
_burn
_transfer
address
_afterTokenTransfer
_beforeTokenTransfer
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
factory
WETH
getAmountsOut
swapExactTokensForETH
Pausable
<Constructor>
whenNotPaused
paused
whenPaused
_pause
_unpause
_msgSender
_transfer
BOTiFiAi
<Constructor>
_transfer
triggerTax
enableTax
disableTax
<Receive Ether>
setTaxWallets
exclude
setSellTax
setBuyTax
payable
removeExclude
handleTax
isExcluded
Ownable
<Constructor>
transferOwnership
renounceOwnership
onlyOwner
address
_setOwner
_msgSender
owner

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/