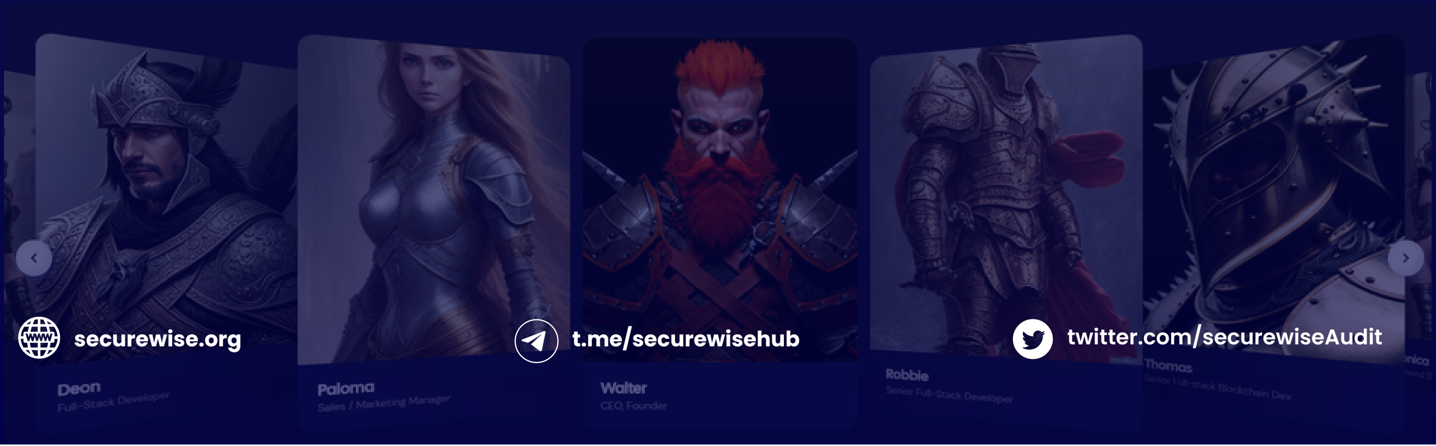
AUDIT REPORT

TYRION TOKEN (TYON)


Deon
Full-Stack Developer
Paloma
Sales / Marketing Manager
Walter
CEO, Founder
Robbie
Senior Full-Stack Developer
Thomas
Senior Full-stack Blockchain Dev.

securewise.org t.me/securewisehub twitter.com/securewiseAudit

Quick Result

Quick Result	Status
Owner can mint new token?	Not Detected
Owner can update tax over 25% ?	Not Detected
Owner can pause trade ?	Yes
Owner can enable trading ?	Not Detected
Owner can add Blacklist ?	Not Detected
Owner can set Max Tx ?	Yes
Owner can set Max Wallet Amount ?	Not Detected
KYC ?	Yes

Page 6,10 for more details

Tyrion Token (TYON) as PASSED the smart contract audit with Critical Risk

Findings

Risk Classification	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.
Low	Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.
Informational	A vulnerability that have informational character but is not effecting any of the code

Severity	Found	Pending	Resolved
High	0	3	0
Medium	0	0	0
Low	0	0	0
Informational	0	3	0
Total	0	6	0

Contents

Overview

Token Name: Tyrion Token (**TYON**)

Language: Solidity

Contract Address: 0x7Ee43f72b5431082993AE81356472AfbB42F9dAc

Network: Binance Smart Chain

Total Supply: 500000000

KYC: Yes

Website: https://www.tyrion.io/

Twitter: https://twitter.com/Tyrion_TYON

Telegram: https://t.me/tyriontyonchat

Report Date: August 10, 2023

Testnet link:

https://testnet.bscscan.com/address/0xe4f8e690e6db38895d1db8a96a47c16c73fa6f3e

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.

Low: Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.

Informational: A vulnerability that have informational character but is not effecting any of the code

Findings Summary

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

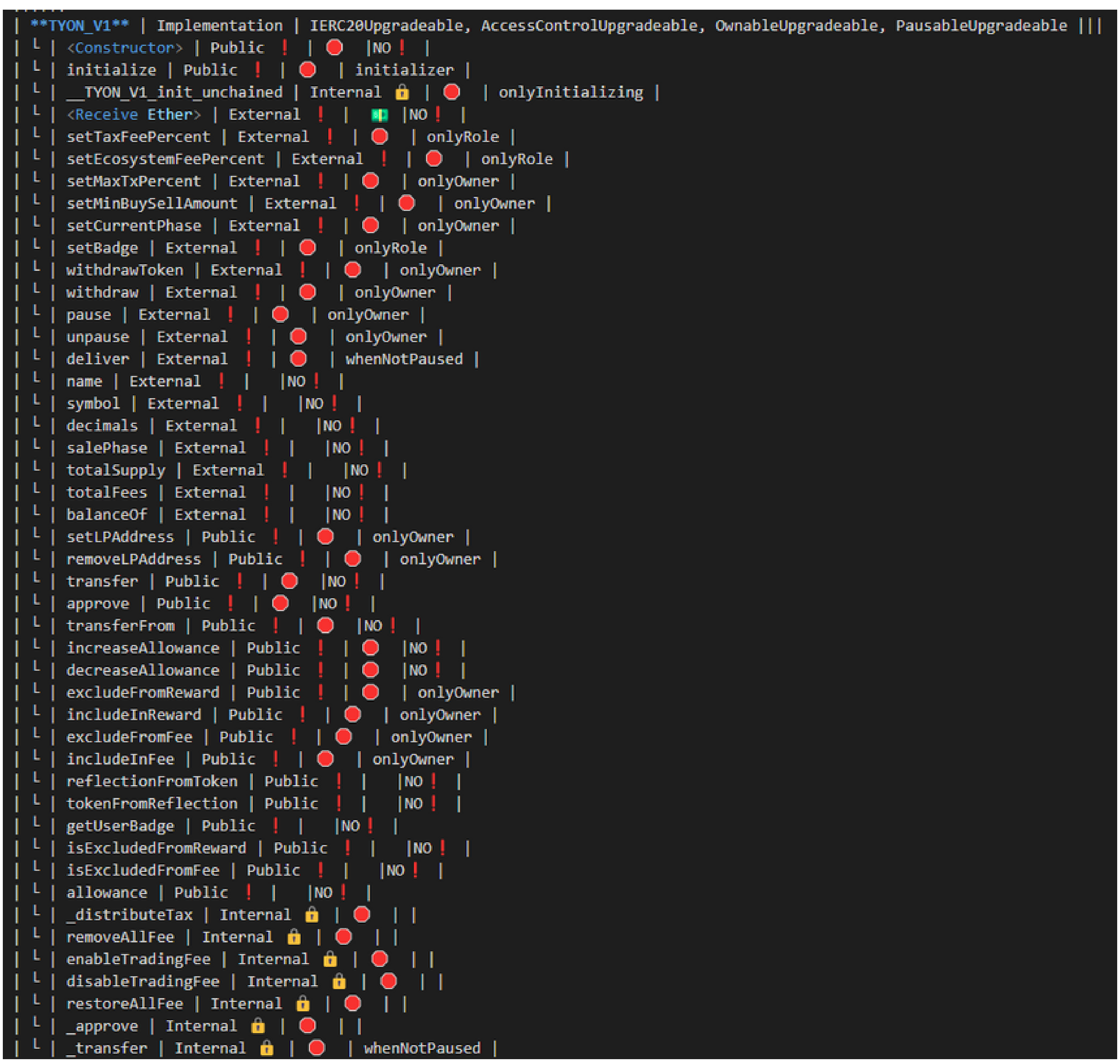
Findings

🔺	The owner has the authority to pause trading
🔺	The owner has the authority to set the maximum transaction amount to 0.
🔺	The owner has the authority to set a minimum buy and sell amount without limit.
⚠	The owner has the authority to change total fees by a maximum of 20%.
⚠	The owner has the authority to exclude addresses from rewards.
⚠	The owner has the authority to exclude addresses from fees.

Page 10 for more details

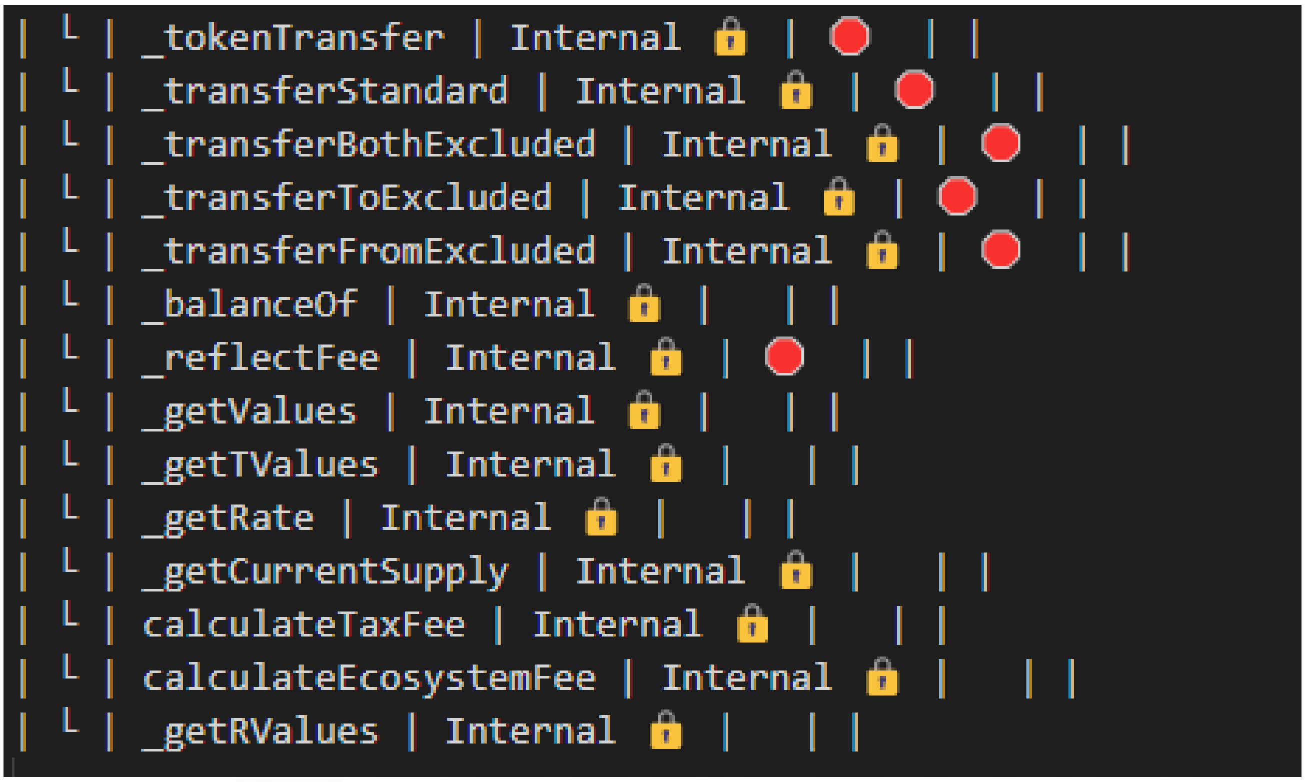
Tyrion Token (TYON) as PASSED the smart contract audit with Critical Risk

Function Privileges



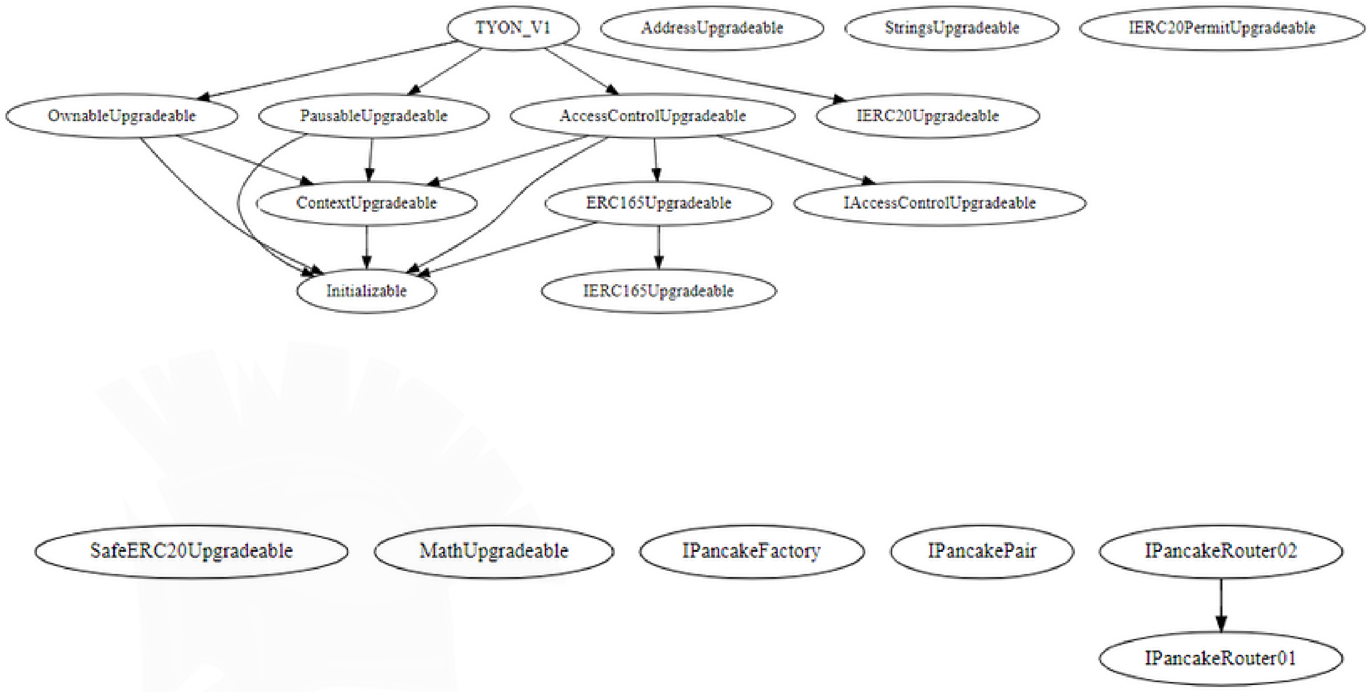
```
| **TYON_V1** | Implementation | IERC20Upgradeable, AccessControlUpgradeable, OwnableUpgradeable, PausableUpgradeable |||
| └ | <Constructor> | Public ❗ | 🔴 |NO ❗ |
| └ | initialize | Public ❗ | 🔴 | initializer |
| └ | __TYON_V1_init_unchained | Internal 🔒 | 🔴 | onlyInitializing |
| └ | <Receive Ether> | External ❗ | 💵 |NO ❗ |
| └ | setTaxFeePercent | External ❗ | 🔴 | onlyRole |
| └ | setEcosystemFeePercent | External ❗ | 🔴 | onlyRole |
| └ | setMaxTxPercent | External ❗ | 🔴 | onlyOwner |
| └ | setMinBuySellAmount | External ❗ | 🔴 | onlyOwner |
| └ | setCurrentPhase | External ❗ | 🔴 | onlyOwner |
| └ | setBadge | External ❗ | 🔴 | onlyRole |
| └ | withdrawToken | External ❗ | 🔴 | onlyOwner |
| └ | withdraw | External ❗ | 🔴 | onlyOwner |
| └ | pause | External ❗ | 🔴 | onlyOwner |
| └ | unpause | External ❗ | 🔴 | onlyOwner |
| └ | deliver | External ❗ | 🔴 | whenNotPaused |
| └ | name | External ❗ |   |NO ❗ |
| └ | symbol | External ❗ |   |NO ❗ |
| └ | decimals | External ❗ |   |NO ❗ |
| └ | salePhase | External ❗ |   |NO ❗ |
| └ | totalSupply | External ❗ |   |NO ❗ |
| └ | totalFees | External ❗ |   |NO ❗ |
| └ | balanceOf | External ❗ |   |NO ❗ |
| └ | setLPAddress | Public ❗ | 🔴 | onlyOwner |
| └ | removeLPAddress | Public ❗ | 🔴 | onlyOwner |
| └ | transfer | Public ❗ | 🔴 |NO ❗ |
| └ | approve | Public ❗ | 🔴 |NO ❗ |
| └ | transferFrom | Public ❗ | 🔴 |NO ❗ |
| └ | increaseAllowance | Public ❗ | 🔴 |NO ❗ |
| └ | decreaseAllowance | Public ❗ | 🔴 |NO ❗ |
| └ | excludeFromReward | Public ❗ | 🔴 | onlyOwner |
| └ | includeInReward | Public ❗ | 🔴 | onlyOwner |
| └ | excludeFromFee | Public ❗ | 🔴 | onlyOwner |
| └ | includeInFee | Public ❗ | 🔴 | onlyOwner |
| └ | reflectionFromToken | Public ❗ |   |NO ❗ |
| └ | tokenFromReflection | Public ❗ |   |NO ❗ |
| └ | getUserBadge | Public ❗ |   |NO ❗ |
| └ | isExcludedFromReward | Public ❗ |   |NO ❗ |
| └ | isExcludedFromFee | Public ❗ |   |NO ❗ |
| └ | allowance | Public ❗ |   |NO ❗ |
| └ | _distributeTax | Internal 🔒 | 🔴 | |
| └ | removeAllFee | Internal 🔒 | 🔴 | |
| └ | enableTradingFee | Internal 🔒 | 🔴 | |
| └ | disableTradingFee | Internal 🔒 | 🔴 | |
| └ | restoreAllFee | Internal 🔒 | 🔴 | |
| └ | _approve | Internal 🔒 | 🔴 | |
| └ | _transfer | Internal 🔒 | 🔴 | whenNotPaused |
```

Function Privileges

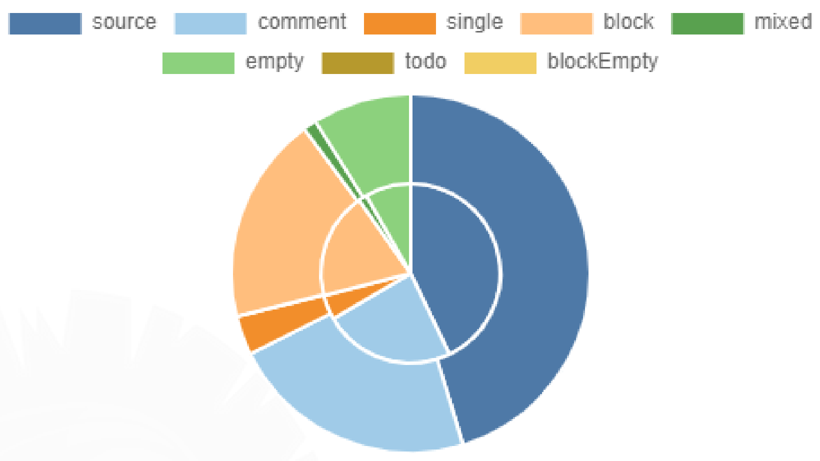


```
| └ | _tokenTransfer | Internal 🔒 | 🛑  | |
| └ | _transferStandard | Internal 🔒 | 🛑  | |
| └ | _transferBothExcluded | Internal 🔒 | 🛑  | |
| └ | _transferToExcluded | Internal 🔒 | 🛑  | |
| └ | _transferFromExcluded | Internal 🔒 | 🛑  | |
| └ | _balanceOf | Internal 🔒 |   | |
| └ | _reflectFee | Internal 🔒 | 🛑  | |
| └ | _getValues | Internal 🔒 |   | |
| └ | _getTValues | Internal 🔒 |   | |
| └ | _getRate | Internal 🔒 |   | |
| └ | _getCurrentSupply | Internal 🔒 |   | |
| └ | calculateTaxFee | Internal 🔒 |   | |
| └ | calculateEcosystemFee | Internal 🔒 |   | |
| └ | _getRValues | Internal 🔒 |   | |
```

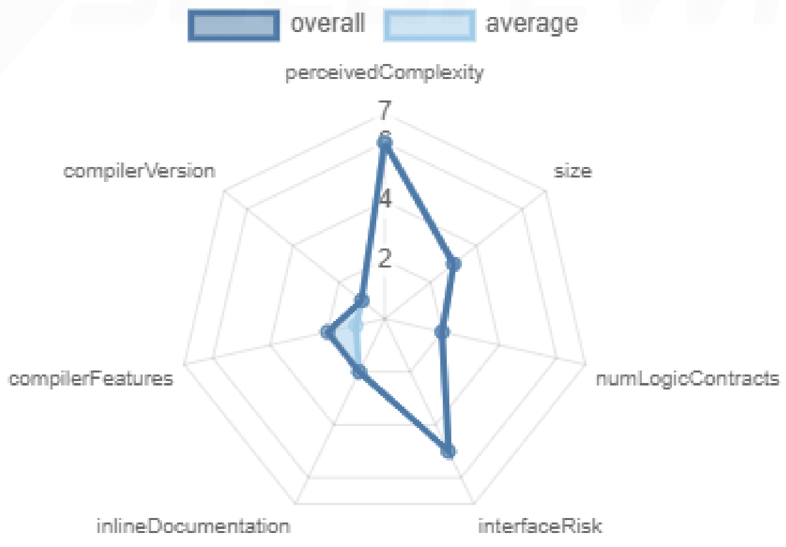
Inheritance Graph



Source Lines



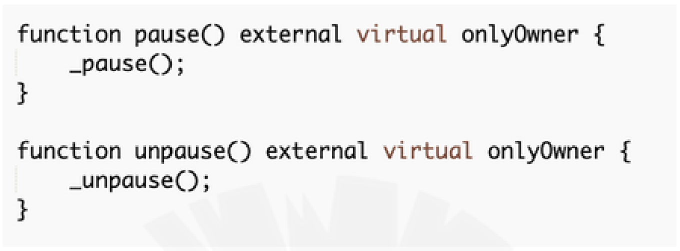
Risk



Manuel Review

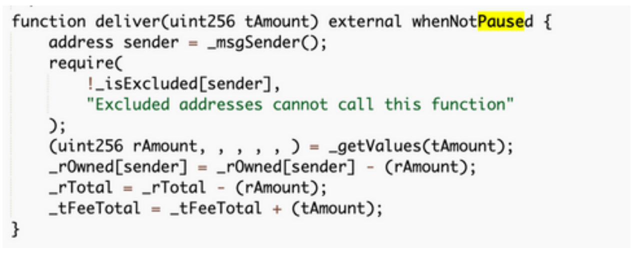
High Risk

The owner has the authority to pause trading

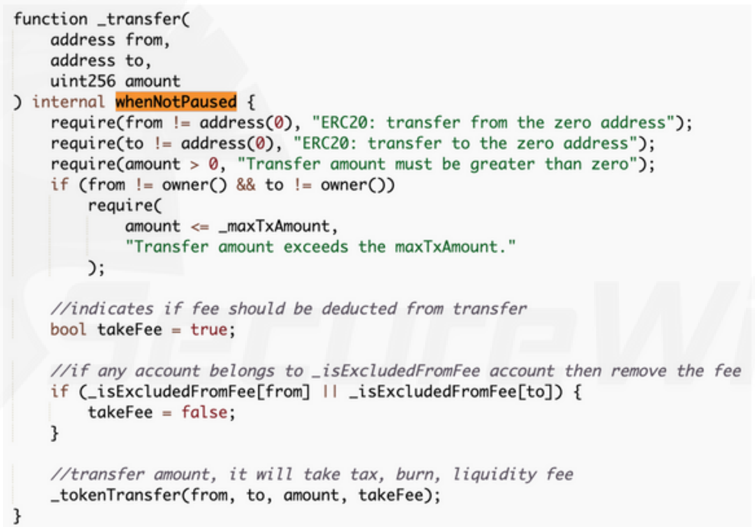


```
function pause() external virtual onlyOwner {
    _pause();
}

function unpause() external virtual onlyOwner {
    _unpause();
}
```



```
function deliver(uint256 tAmount) external whenNotPaused {
    address sender = _msgSender();
    require(
        !_isExcluded[sender],
        "Excluded addresses cannot call this function"
    );
    (uint256 rAmount, , , , , ) = _getValues(tAmount);
    _rOwned[sender] = _rOwned[sender] - (rAmount);
    _rTotal = _rTotal - (rAmount);
    _tFeeTotal = _tFeeTotal + (tAmount);
}
```



```
function _transfer(
    address from,
    address to,
    uint256 amount
) internal whenNotPaused {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    if (from != owner() && to != owner())
        require(
            amount <= _maxTxAmount,
            "Transfer amount exceeds the maxTxAmount."
        );

    //indicates if fee should be deducted from transfer
    bool takeFee = true;

    //if any account belongs to _isExcludedFromFee account then remove the fee
    if (_isExcludedFromFee[from] || _isExcludedFromFee[to]) {
        takeFee = false;
    }

    //transfer amount, it will take tax, burn, liquidity fee
    _tokenTransfer(from, to, amount, takeFee);
}
```

Description

Owner can pause or unpause the trading with these functions. If owner pause the contract so no one can trade(buy+sell) except owner.

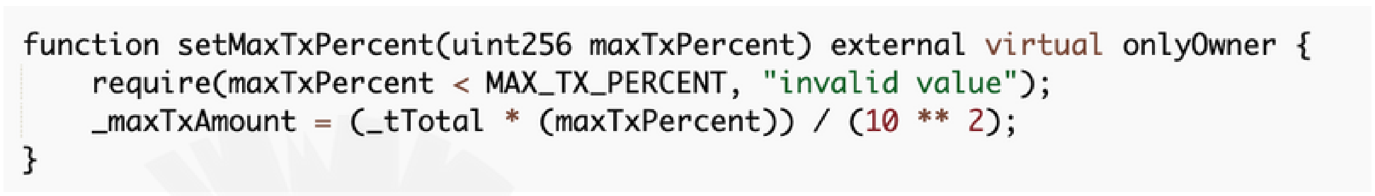
Recommendation

With this authority, the owner can pause or unpause the contract whenever he wants, and this is a critical risk for investors, we recommend that you remove this authority from the contract.

Manuel Review

High Risk

The owner has the authority to set the maximum transaction amount to 0.



```
function setMaxTxPercent(uint256 maxTxPercent) external virtual onlyOwner {
    require(maxTxPercent < MAX_TX_PERCENT, "invalid value");
    _maxTxAmount = (_tTotal * (maxTxPercent)) / (10 ** 2);
}
```

Description

If owner set maxTxPercent to "0". It impossible to trade for anyone except the owner.

Recommendation

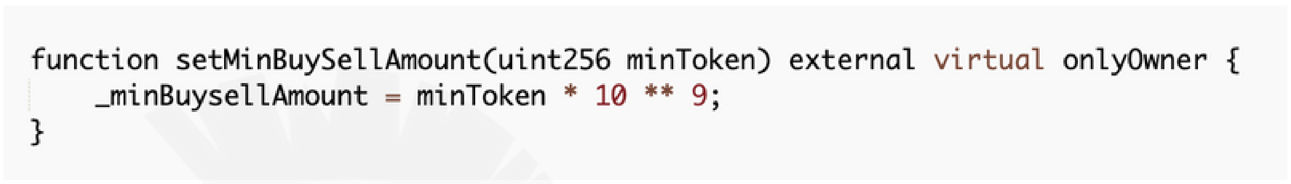
Add an additional condition that requires the maximum transaction percentage to be greater than 0.

```
function setMaxTxPercent(uint256 maxTxPercent) external virtual onlyOwner {
  require(maxTxPercent < MAX_TX_PERCENT && maxTxPercent > 0, "invalid value");
  _maxTxAmount = (_tTotal * maxTxPercent) / (10 ** 2);
}
```

Manuel Review

High Risk

The owner has the authority to set a minimum buy and sell amount without limit.



```
function setMinBuySellAmount(uint256 minToken) external virtual onlyOwner {
    _minBuysellAmount = minToken * 10 ** 9;
}
```

Description

If owner set minToken to "0". This makes it impossible to trade, including for the owner.

Recommendation

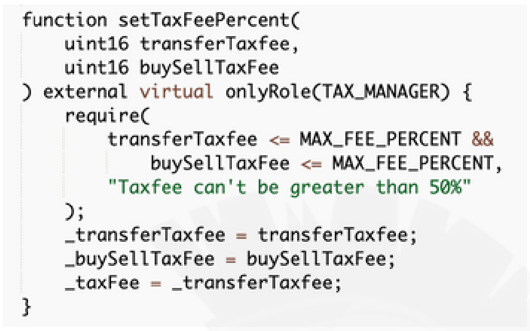
Add an upper limit to the `require` statement and ensure that it is a logical limit, such as 0.001%.

```
function setMinBuySellAmount(uint256 minToken) external virtual onlyOwner {
  require(minToken <= _tTotal / 100000, "TYON_V1 Error: minToken must be less than 0.001% of total supply");
  _minBuysellAmount = minToken * 10 ** 9;
}
```

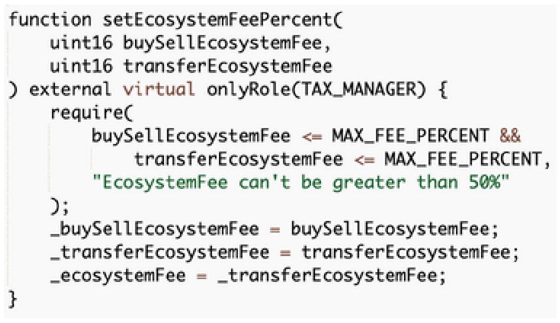
Manuel Review

Informational

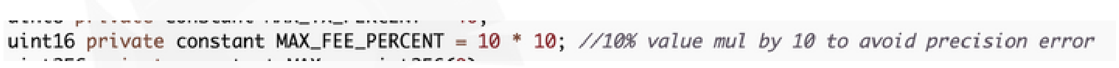
The owner has the authority to change total fees by a maximum of 20%.



```
function setTaxFeePercent(
    uint16 transferTaxfee,
    uint16 buySellTaxFee
) external virtual onlyRole(TAX_MANAGER) {
    require(
        transferTaxfee <= MAX_FEE_PERCENT &&
            buySellTaxFee <= MAX_FEE_PERCENT,
        "Taxfee can't be greater than 50%"
    );
    _transferTaxfee = transferTaxfee;
    _buySellTaxFee = buySellTaxFee;
    _taxFee = _transferTaxfee;
}
```



```
function setEcosystemFeePercent(
    uint16 buySellEcosystemFee,
    uint16 transferEcosystemFee
) external virtual onlyRole(TAX_MANAGER) {
    require(
        buySellEcosystemFee <= MAX_FEE_PERCENT &&
            transferEcosystemFee <= MAX_FEE_PERCENT,
        "EcosystemFee can't be greater than 50%"
    );
    _buySellEcosystemFee = buySellEcosystemFee;
    _transferEcosystemFee = transferEcosystemFee;
    _ecosystemFee = _transferEcosystemFee;
}
```



```
uint16 private constant MAX_FEE_PERCENT = 10 * 10; //10% value mul by 10 to avoid precision error
```

Description

Owner can change total fees by a maximum of 20%.

Recommendation

No specific recommendation is necessary. However, it is important to ensure that the function is being used appropriately and that the owner's ability to change the taxes is clearly documented and understood.

Manuel Review

Informational

The owner has the authority to exclude addresses from rewards.

```
function excludeFromReward(address account) public onlyOwner {
    require(!_isExcluded[account], "Account is already excluded");
    if (_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}
```

```
function includeInReward(address account) public onlyOwner {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            delete _tOwned[account];
            delete _isExcluded[account];
            _excluded.pop();
            break;
        }
    }
}
```

Description

Designed to exclude include a specified account from receiving dividends.

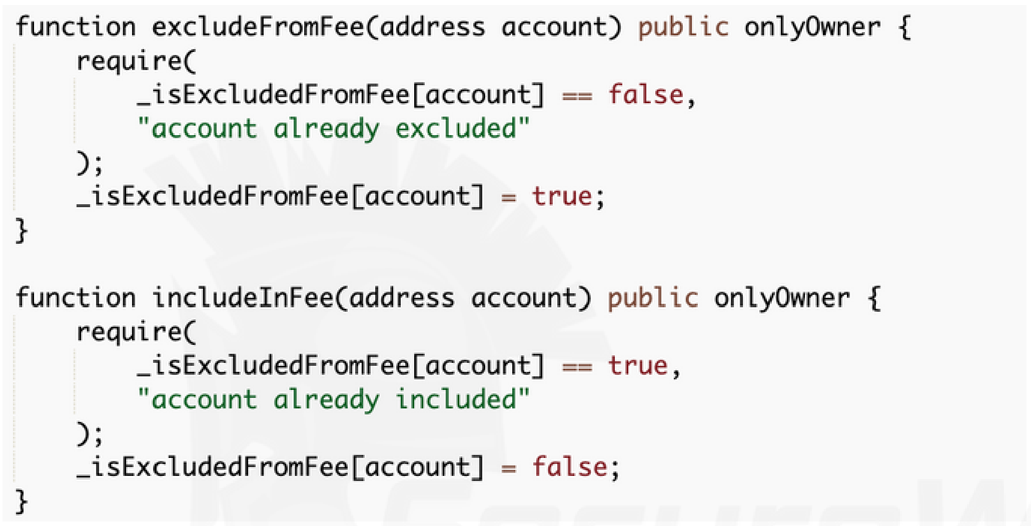
Recommendation

*While assuming input validation occurs before invoking this function, it is advisable to explicitly validate the account parameter to ensure its conformity as a valid address. Implementing error-handling mechanisms to gracefully manage potential exceptions is also recommended. Ensure that appropriate access control mechanisms are in place to restrict the **excludeFromReward and includeInReward** functions to only be called by the contract owner*

Manuel Review

Informational

The owner has the authority to exclude addresses from fees.



```
function excludeFromFee(address account) public onlyOwner {
    require(
        _isExcludedFromFee[account] == false,
        "account already excluded"
    );
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    require(
        _isExcludedFromFee[account] == true,
        "account already included"
    );
    _isExcludedFromFee[account] = false;
}
```

Description

*Its allows the contract owner to modify the exclude or include status of an account from fees by updating the **_isExcludedFromFee** mapping.*

Recommendation

No specific recommendation is necessary. However, it is important to ensure that the function is being used appropriately and that the owner's ability to exclude or include accounts from fees is clearly documented and understood.

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

AUDIT REPORT

SecureWise

securewise.org

t.me/securewisehub

twitter.com/securewiseAudit

SecureWise Scanner

t.me/pinksalefreechecks