AUDIT REPORT

SecureWise

TUTENSTEIN INU (TUTEN)

Quick Result

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

Page 10 for more details

Tutenstein Inu (TUTEN) as PASSED the smart contract audit

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	0	0
Medium	0	0	0
Low	2	0	0
Informational	3	0	0
Total	5	0	0

Contents

Overview

Token Name: Tutenstein Inu **(TUTEN)**

Language: Solidity

Contract Address: 0x119360c866407EB3d54b59e1710Bc08e514408B3

Network: Binance Smart Chain

Supply: 990,000,000,000,000,000

KYC: Not done

Website: https://tutenstein.xyz

Twitter: https://twitter.com/tutensteininu

Telegram: https://t.me/tutensteininu

Report Date: June 29, 2023

Testnet:

https://testnet.bscscan.com/address/0xb7a457805b04d4a16Cd94bbc1781052Ad2997977

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

Centralization Findings

⚠	Accounts can be excluded by the owner from receiving rewards.
⚠	The current fee percentage is fixed 7% and cannot be changed.
⚠	Owner can exclude account from fees

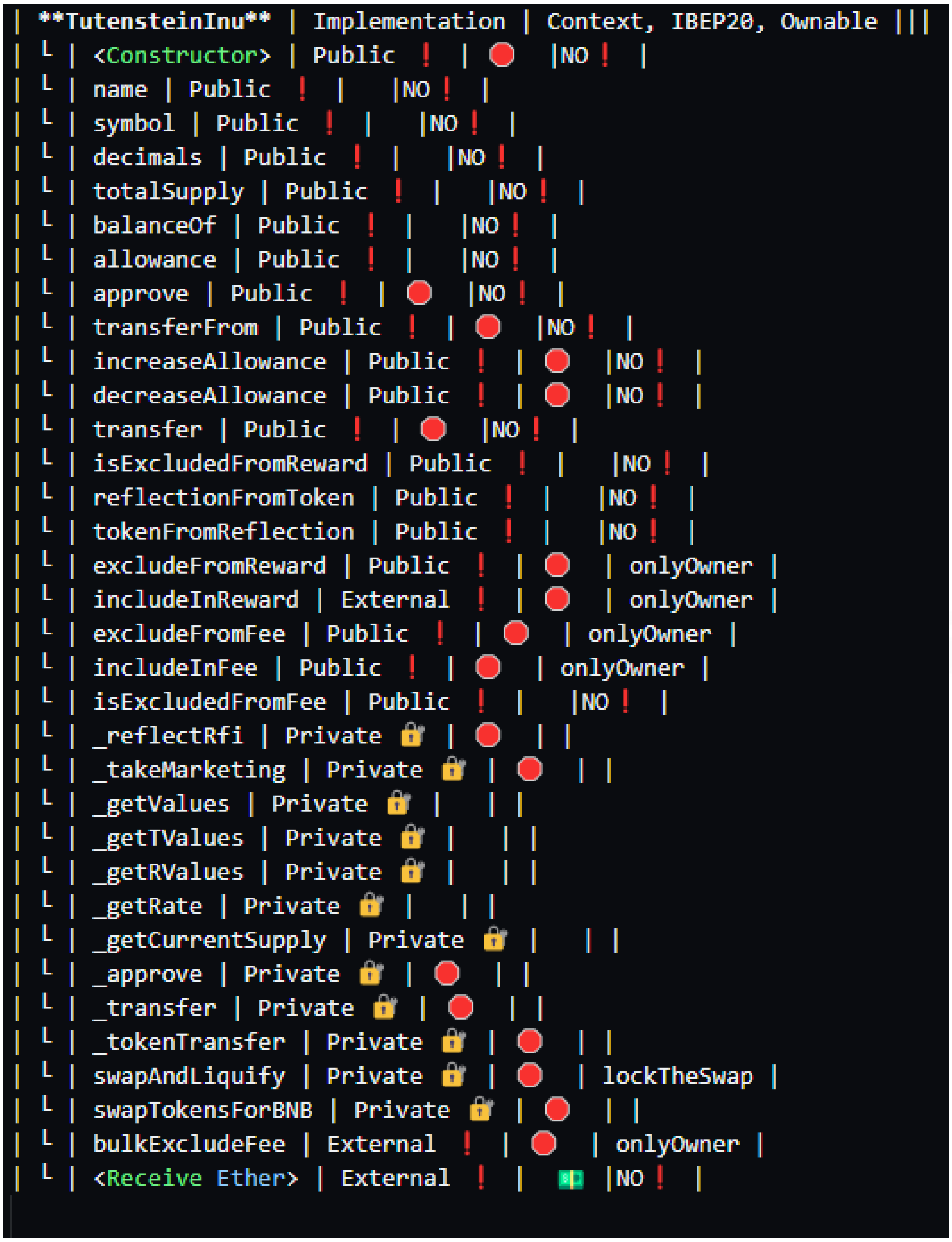
Logical Findings

⚠	Floating Pragma and Outdated Compiler Version
⚠	Missing zero address validation

Page 10 for more details

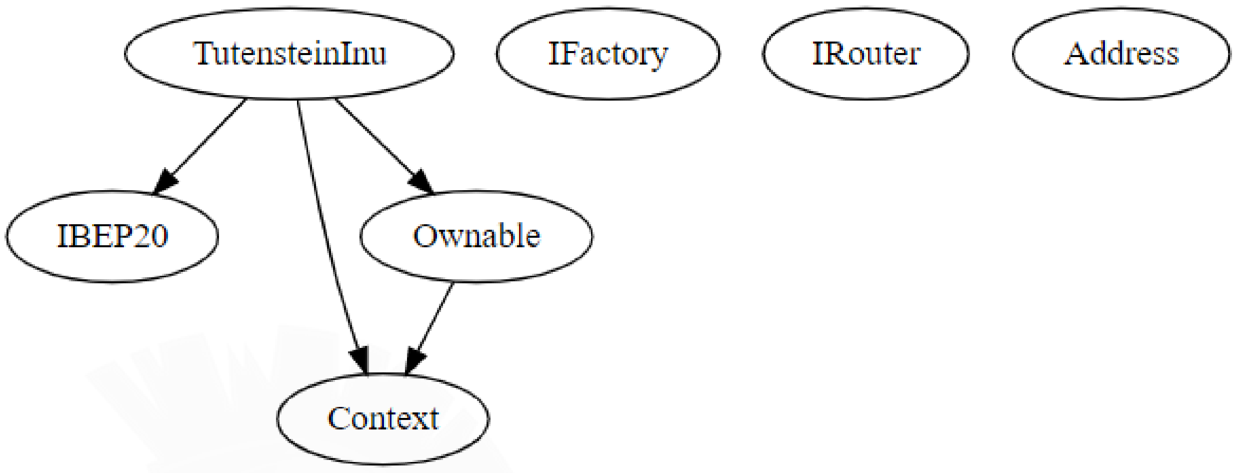
Tutenstein Inu (TUTEN) as PASSED the smart contract audit

Function Privileges

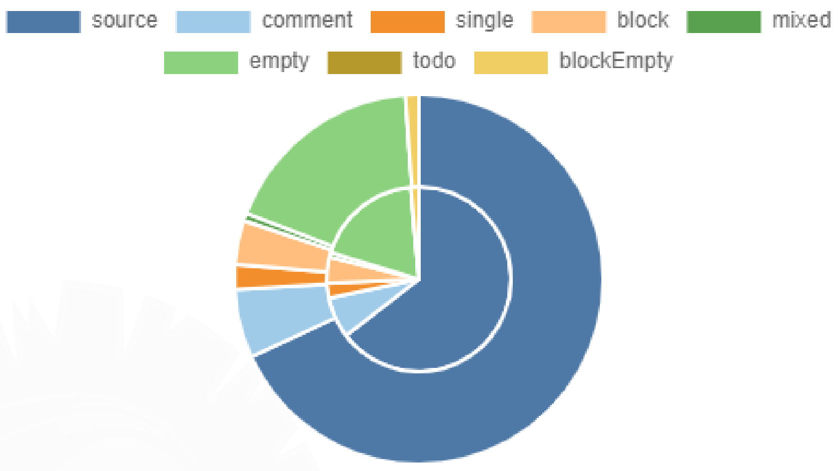


```
| **TutensteinInu** | Implementation | Context, IBEP20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO ❗️ |
| └ | name | Public ❗️ |   |NO ❗️ |
| └ | symbol | Public ❗️ |   |NO ❗️ |
| └ | decimals | Public ❗️ |   |NO ❗️ |
| └ | totalSupply | Public ❗️ |   |NO ❗️ |
| └ | balanceOf | Public ❗️ |   |NO ❗️ |
| └ | allowance | Public ❗️ |   |NO ❗️ |
| └ | approve | Public ❗️ | 🛑  |NO ❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO ❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO ❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO ❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO ❗️ |
| └ | isExcludedFromReward | Public ❗️ |   |NO ❗️ |
| └ | reflectionFromToken | Public ❗️ |   |NO ❗️ |
| └ | tokenFromReflection | Public ❗️ |   |NO ❗️ |
| └ | excludeFromReward | Public ❗️ | 🛑  | onlyOwner |
| └ | includeInReward | External ❗️ | 🛑  | onlyOwner |
| └ | excludeFromFee | Public ❗️ | 🛑  | onlyOwner |
| └ | includeInFee | Public ❗️ | 🛑  | onlyOwner |
| └ | isExcludedFromFee | Public ❗️ |   |NO ❗️ |
| └ | _reflectRfi | Private 🔐 | 🛑  | |
| └ | _takeMarketing | Private 🔐 | 🛑  | |
| └ | _getValues | Private 🔐 |   | |
| └ | _getTValues | Private 🔐 |   | |
| └ | _getRValues | Private 🔐 |   | |
| └ | _getRate | Private 🔐 |   | |
| └ | _getCurrentSupply | Private 🔐 |   | |
| └ | _approve | Private 🔐 | 🛑  | |
| └ | _transfer | Private 🔐 | 🛑  | |
| └ | _tokenTransfer | Private 🔐 | 🛑  | |
| └ | swapAndLiquify | Private 🔐 | 🛑  | lockTheSwap |
| └ | swapTokensForBNB | Private 🔐 | 🛑  | |
| └ | bulkExcludeFee | External ❗️ | 🛑  | onlyOwner |
| └ | <Receive Ether> | External ❗️ | 💵 |NO ❗️ |
```

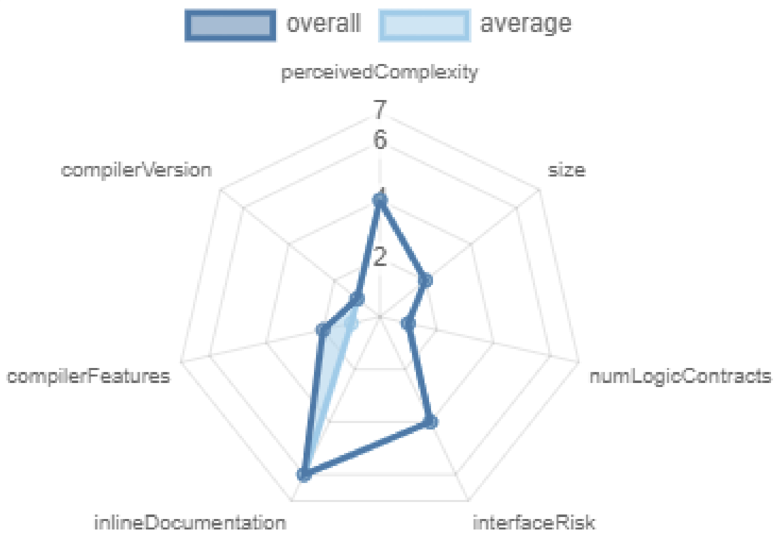
Inheritance Graph



Source Lines



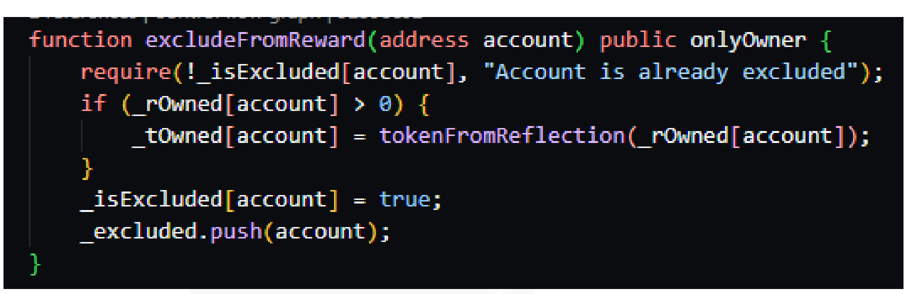
Risk



Manual Review

Low Risk

Accounts can be excluded by the owner from receiving 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);
}
```

Description

***excludeFromReward** function, designed to exclude a specified account from receiving dividends.*

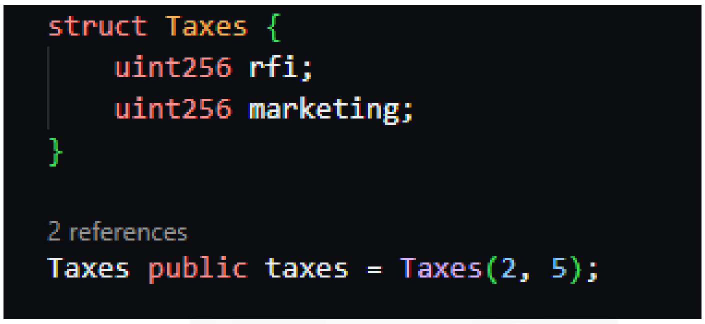
Recommendation

Implementing error-handling mechanisms to gracefully manage potential exceptions is also recommended. Ensure that appropriate access control mechanisms are in place to restrict the excludeFromDividends function to only be called by the contract owner

Manual Review

Low Risk

The current fee percentage is fixed 7% and cannot be changed.



```
struct Taxes {
    uint256 rfi;
    uint256 marketing;
}

2 references
Taxes public taxes = Taxes(2, 5);
```

Description

Reflection rfi is 2 and Marketing fee is 5 overall buy/sell fee is 7% and it can not be change by the owner

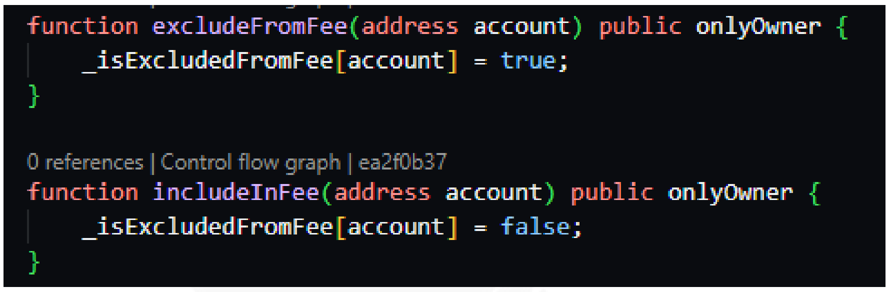
Recommendation

No specific recommendation is necessary for the taxes management at this time. Taxes are reasonable limit

Manual Review

Informational

Owner has the ability to exclude accounts from being charged fees.



```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

0 references | Control flow graph | ea2f0b37
function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

Description

excludeFromFee *allows the contract owner to modify the exclusion status of an account from fees by updating the _isExcludedFromFee mapping.*

Recommendation

No specific recommendation is necessary for the ***excludeFromFee*** *function at this time. 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.*

Manual Review

Informational

Missing zero address validation

routerAddress on constructor arg parameter

Description

Detect missing zero address validation.

Recommendation

Check that the address is not zero.

Manual Review

Informational

Old Version of Solidity Compiler and Floating pragma

```
pragma solidity ^0.8.17;
```

Description

Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statement. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing. Lock the pragma version and also consider known bugs or the compiler version that is chosen.

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