AUDIT REPORT

SecureWise

COINFLIP (FLIP)

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

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 ?	No KYC

Page 10 for more details

CoinFlip (Flip) 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	1	0	0
Informational	3	0	0
Total	4	0	0

Contents

Overview

Token Name: CoinFlip (**Flip**)

Language: Solidity

Contract Address: 0x0A64CF5A0eFfCf86825a1e981a9a446226a405F5

Network: Ethereum

Supply: 10000000000

KYC: No KYC

Website: https://coinflip.vip

Twitter: https://twitter.com/CoinFlip_ETH

Telegram: https://t.me/Flip_Eth

Report Date: July 1, 2023

Testnet:

https://testnet.bscscan.com/address/0x4cB901764D34a3A94259C0C502767A3500ada510

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

⚠	Owner has the authority to change swap token amount to setting "0"
⚠	Owner has the authority to exclude account from fees

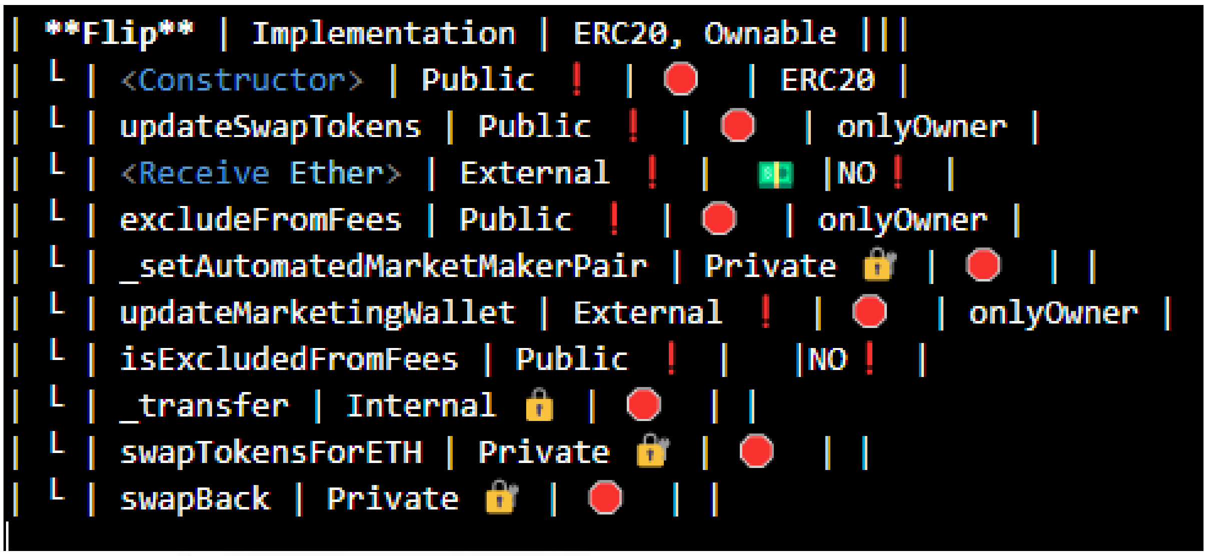
Logical Findings

⚠	Using an old version of solidity compiler
⚠	Lack of zero address check

Page 10 for more details

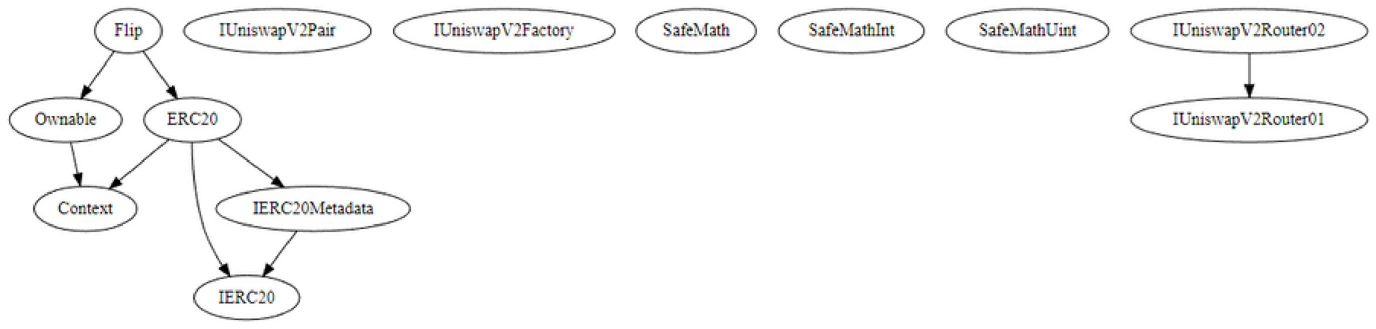
CoinFlip (Flip) as PASSED the smart contract audit.

Function Privileges

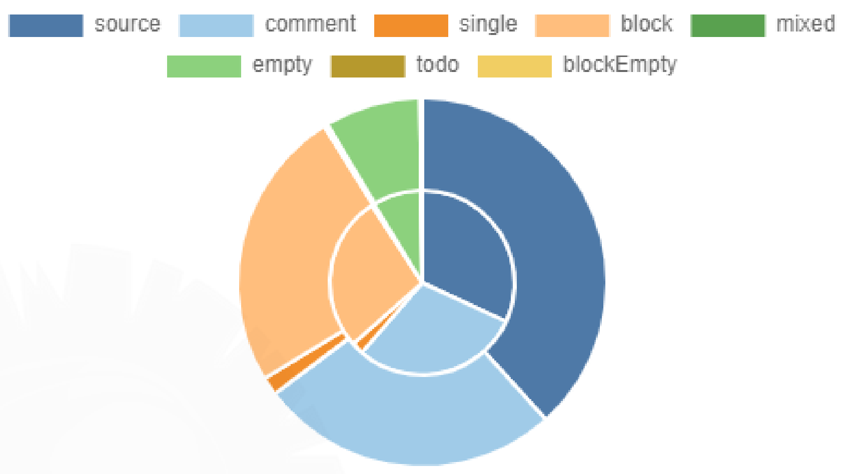


```
| **Flip** | Implementation | ERC20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  | ERC20 |
| └ | updateSwapTokens | Public ❗️ | 🛑  | onlyOwner |
| └ | <Receive Ether> | External ❗️ | 💵 |NO❗️ |
| └ | excludeFromFees | Public ❗️ | 🛑  | onlyOwner |
| └ | _setAutomatedMarketMakerPair | Private 🔐 | 🛑  | |
| └ | updateMarketingWallet | External ❗️ | 🛑  | onlyOwner |
| └ | isExcludedFromFees | Public ❗️ |   |NO❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | swapTokensForETH | Private 🔐 | 🛑  | |
| └ | swapBack | Private 🔐 | 🛑  | |
```

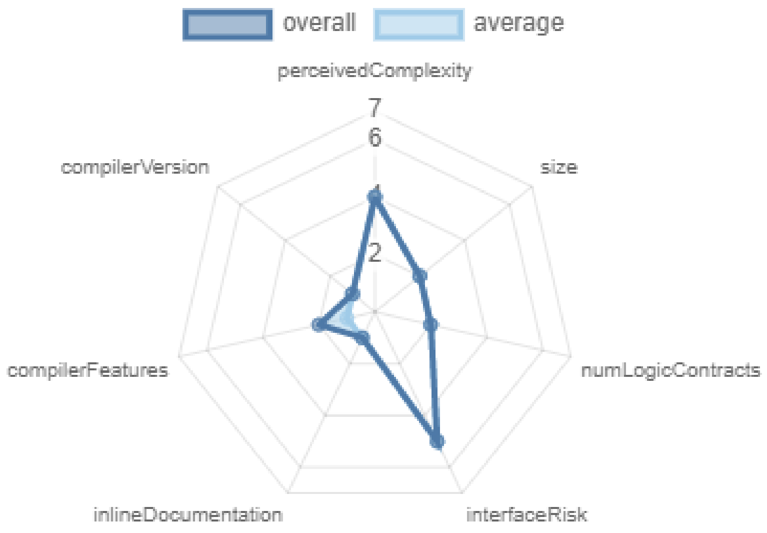
Inheritance Graph



Source Lines



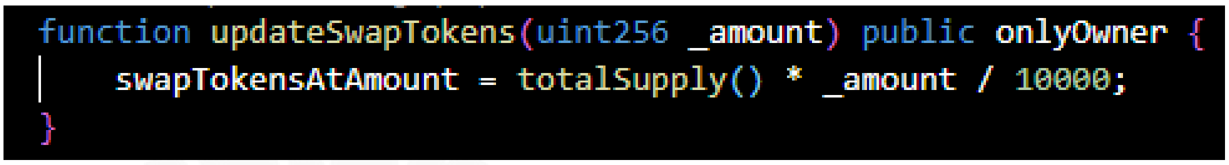
Risk



Manual Review

Low Risk

Owner has the authority to change swap token amount to setting "0"



```
function updateSwapTokens(uint256 _amount) public onlyOwner {
    swapTokensAtAmount = totalSupply() * _amount / 10000;
}
```

Description

swapTokensAtAmount *variable can be modified by the owner, potentially allowing them to disable token swapping or create imbalances in token distribution. This introduces the risk of unfair practices or market manipulation*

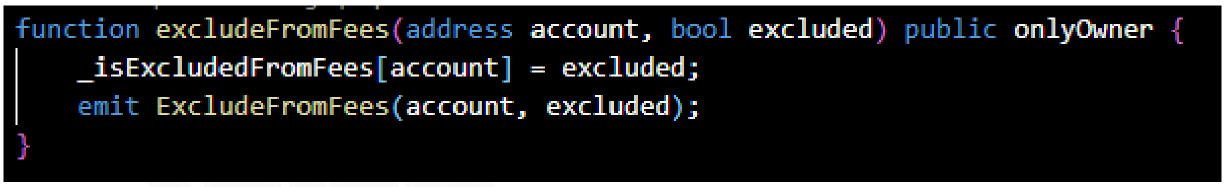
Recommendation

Consider implementing measures to limit the owner's ability to freely modify the swapTokensAtAmount variable, such as setting a reasonable upper lower bound or utilizing timelocks and multi-signature requirements for modifications.

Manual Review

Informational

Owner has the authority to exclude account from fees



```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    _isExcludedFromFees[account] = excluded;
    emit ExcludeFromFees(account, excluded);
}
```

Description

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

Recommendation

No specific recommendation is necessary for the ***excludeFromFees*** *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

Old Version of Solidity Compiler version

```
pragma solidity 0.8.11;
```

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.

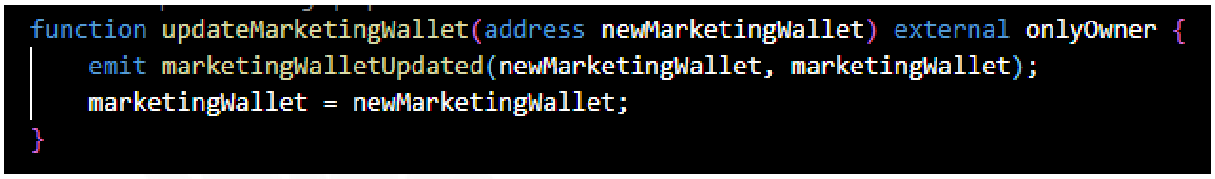
Recommendation

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Manual Review

Informational

Lack of zero address check



```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
    emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
    marketingWallet = newMarketingWallet;
}
```

Description

Detect missing zero address validation.

Recommendation

Check that the address is not zero.

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