



AUDIT REPORT

SecureWise

OG PEPE (OGPEPE)



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 fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification 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	0	0	0
Informational	4	0	0
Total	4	0	0

Page 6,10 for more details

Ownership Renounced

Contents

02	Findings
04	Overview
05	Auditing Approach and Methodologies
06	Findings Summary
07	Function Privileges
08	Inheritance Graph
10	Manual Review
14	Disclaimer

Overview

Token Name: OG Pepe (**OGPEPE**)

Language: Solidity

Contract Address: 0x39cd47fD517EE68DecbDdc69AF7cAED7927145f5

Network: Ethereum

Supply: 10000000000

KYC: No KYC

Website: <https://OGpepe.org>

Twitter: <https://twitter.com/ogpepecoin>

Telegram: <https://t.me/OGPEPEcoin>

Report Date: July 6, 2023

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 fixed as soon as possible.





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

Informational: A vulnerability that have informational character but is not affecting 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

	Ownership has been renounced and the bot feature has been implemented incorrectly and unnecessarily.
	Ownership has been renounced and the openTrading feature has been implemented incorrectly and unnecessarily logic.
	Ownership has been renounced and the removeLimits feature has been implemented incorrectly logic.
	Trading cooldown function and Anti_whale max tx and max wallet detected it safe range and Ownership Renounced

Page 10 for more details

Ownership Renounced

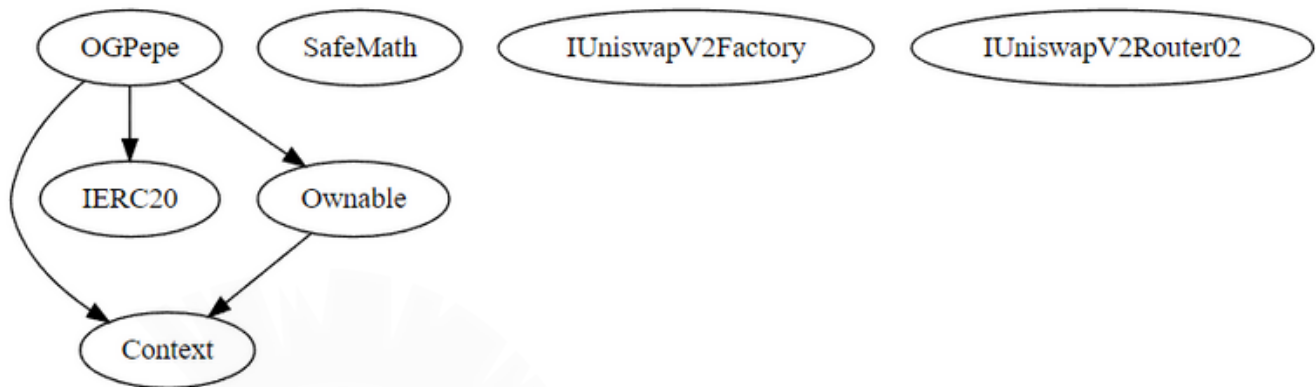
Function Privileges

```

||||| |
| **OGPepe** | Implementation | Context, IERC20, Ownable |||
| L | <Constructor> | Public ! | ● | NO ! |
| L | name | Public ! | | NO ! |
| L | symbol | Public ! | | NO ! |
| L | decimals | Public ! | | NO ! |
| L | totalSupply | Public ! | | NO ! |
| L | balanceOf | Public ! | | NO ! |
| L | transfer | Public ! | ● | NO ! |
| L | allowance | Public ! | | NO ! |
| L | approve | Public ! | ● | NO ! |
| L | transferFrom | Public ! | ● | NO ! |
| L | _approve | Private 🔒 | ● | |
| L | _transfer | Private 🔒 | ● | |
| L | min | Private 🔒 | | |
| L | swapTokensForEth | Private 🔒 | ● | lockTheSwap |
| L | removeLimits | External ! | ● | onlyOwner |
| L | sendETHToFee | Private 🔒 | ● | |
| L | isBot | Public ! | | NO ! |
| L | openTrading | External ! | ● | onlyOwner |
| L | <Receive Ether> | External ! | 🟢 | NO ! |
| L | isContract | Private 🔒 | | |
| L | manualSwap | External ! | ● | NO ! |
|||||

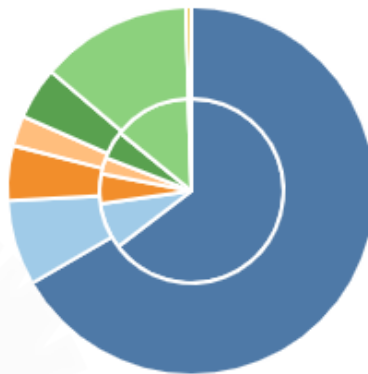
```

Inheritance Graph



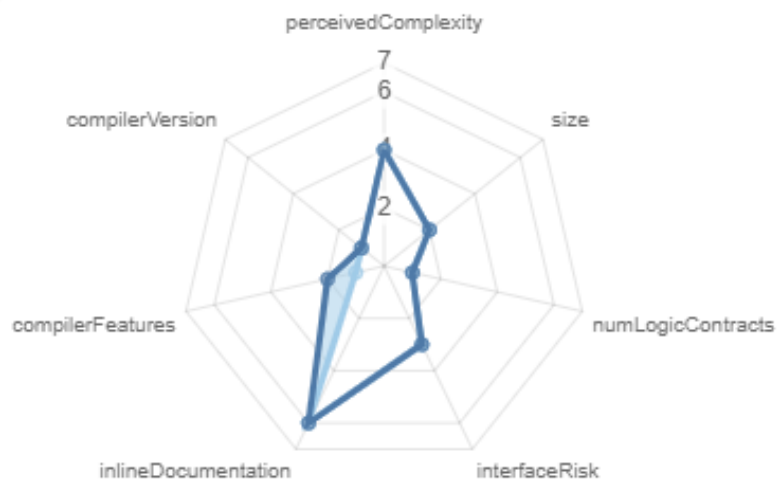
Source Lines

source comment single block mixed
empty todo blockEmpty



Risk

overall average



Manual Review

Informational

Trading cooldown function and Anti_whale max tx and max wallet detected it safe range and Ownership Renounced

Recommendation

No specific recommendation is necessary at this time.



Manual Review

Informational

Ownership has been renounced and the bot feature has been implemented incorrectly and unnecessarily.

```
function isBot(address a) public view returns (bool){  
    return bots[a];  
}
```

```
mapping (address => bool) private bots;  
2 references
```

```
if (from != owner() && to != owner()) {  
    require(!bots[from] && !bots[to]);  
}
```

Description

The function has been implemented for the bot, but it has been used unnecessarily. A structure that can be added to the bot has not been established.

Recommendation

Removing unnecessary and useless features to avoid bloating the code and making it meaningless

Manual Review

Informational

Ownership has been renounced and the openTrading feature has been implemented incorrectly and unnecessarily logic.

```
function openTrading() external onlyOwner() {
    require(!tradingOpen, "trading is already open");
    uniswapV2Router = IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
    _approve(address(this), address(uniswapV2Router), _tTotal);
    uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Router.addLiquidityETH{value: address(this).balance}(address(this), balanceOf(address(this)), 0, 0, owner(), block.timestamp);
    IERC20(uniswapV2Pair).approve(address(uniswapV2Router), type(uint).max);
    swapEnabled = true;
    tradingOpen = true;
}
```

```
function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {
    if(tokenAmount==0){return;}
    if(!tradingOpen){return;}
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
        address(this),
        block.timestamp
    );
}
```

Description

OpenTrading feature has been implemented incorrectly and unnecessarily logic.

Recommendation

If you want this feature you should use in **_transfer** function and adding require not in swapTokensForEth.

Manual Review

Informational

Ownership has been renounced and the `removeLimits` feature has been implemented incorrectly logic.

```
function removeLimits() external onlyOwner{
    _maxTxAmount = _tTotal;
    _maxWalletSize=_tTotal;
    transferDelayEnabled=false;
    emit MaxTxAmountUpdated(_tTotal);
}
```

```
bool public transferDelayEnabled = false;
```

```
if (transferDelayEnabled) { //@audit
    if (to != address(uniswapV2Router) && to != address(uniswapV2Pair)) {
        require(_holderLastTransferTimestamp[tx.origin] < block.number,"Only one transfer per block allowed.");
        _holderLastTransferTimestamp[tx.origin] = block.number;
    }
}
```

Description

The usage of `_maxTxAmount` and `_maxWalletSize` may be valid, but the `transferDelayEnabled` feature is unnecessary as it is initially set to false, and there is no implementation that sets it to true. It seems to be used unnecessarily in the code.

Recommendation

Since the `transferDelayEnabled` feature is not utilized in the code and serves no purpose, it is recommended to remove this feature altogether. Unnecessary features and variables increase code complexity and can potentially introduce bugs or security vulnerabilities. By removing `transferDelayEnabled`, you can simplify the codebase and improve its readability and maintainability.

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

