Nikola43
https://github.com/nikola43
12 June 2023

STONKERS

Audit Report

This is an audit report for the provided Solidity smart contract.
https://etherscan.io/address/0x706abfF8Bb86bA0b38E66A9043e06a424B9e6BB5

Summary

The contract implement the ERC20 token standard. It defines the required functions and events specified by the standard. The contract has been written using the Solidity version 0.8.0.

Findings

1. The contract defines an interface for the ERC20 standard (**IERC20**) and an interface for the optional metadata functions (**IERC20Metadata**).

2. The **Context** contract is used to provide information about the current execution context.

3. The **ERC20** contract inherits from the Context contract and implements the **IERC20** and **IERC20Metadata** interfaces.

4. The contract defines several storage variables:

_balances: A mapping of addresses to token balances.

_allowances: A mapping of addresses to a mapping of spender addresses and allowed token amounts.

_totalSupply: The total supply of the token.

_name: The name of the token.

_symbol: The symbol of the token.

5. The contract implements the required functions specified by the **IERC20** interface:

totalSupply(): Returns the total supply of the token.

balanceOf(**address**): Returns the token balance of the specified address.

transfer(**address, uint256**): Moves tokens from the caller's account to the specified address.

allowance(**address, address**): Returns the amount of tokens that the spender is allowed to spend on behalf of the owner.

approve(**address, uint256**): Sets the allowance for the spender to spend the specified amount of tokens on behalf of the owner.

transferFrom(**address, address, uint256**): Moves tokens from one address to another using the allowance mechanism.

6. The contract implements the optional metadata functions specified by the **IERC20Metadata** interface:

name(): Returns the name of the token.

symbol(): Returns the symbol of the token.

decimals(): Returns the number of decimal places used by the token.

7. The contract defines internal functions **_transfer**, **_mint**, and **_burn** to handle the token transfer, minting, and burning operations respectively but they are internal functions which cannot be called from the outside once the contract is deployed.

8. The contract emits the **Transfer** and **Approval** events as required by the ERC20 standard.

9. The contract does not implement any access control mechanisms, such as role-based access control or whitelisting, which means contract have not ownership and makes it impossible to change the configuration of the contract once it has been deployed.

10. The contract does not include any functionality for pausing or freezing token transfers, which may be necessary in certain situations to prevent unauthorized transfers.

11. The contract does not include any functionality for upgrading the token contract or managing token migrations.

12. The contract does not include any additional features or logic beyond the standard ERC20 functions.

Recommendations

Based on the audit conducted, the following recommendations are provided:

1. Consider implementing access control mechanisms, such as role-based access control or whitelisting, to restrict certain operations to authorized addresses only (only if the project needs it).

2. Consider adding functionality to pause or freeze token transfers in case of emergency situations or security vulnerabilities.

3. Consider including upgradeability features or a mechanism for managing token migrations in case of future contract upgrades or improvements(only if the contract is to be upgraded in the future).

4. Add some kind of anti-bot / anti-MEV / anti-whale protection to secure the price of the token at launch time.

https://www.pinksale.finance/antibot

https://docs.pinksale.finance/pink-anti-bot/pink-anti-bot-guide

Note that this audit report is based on the provided source code and does not guarantee the absence of bugs or vulnerabilities. It is important to conduct additional testing and security reviews to ensure the contract's reliability and security.