expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

USR

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	0
Low	2
Informational	1

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner Can enable trading ?	Detected
Can Owner Disable Trades ?	Detected
Can Owner Mint ?	Not Detected
Can Owner Blacklist ?	Detected
Can Owner set Max Wallet amount ?	Not Detected
Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	Passed With High Risk
KYC Verification	-
Audit Date	**07 November 2023**

CONTRACT DETAILS

Token Address: 0xB24A7830cb83987A3b3034cf48210D0ac28fF50e

Name: USR

Symbol: USR

Decimals: 18

Network: Binance smart chain

Token Type: ERC20

Owner: 0xE9F5010e0709848e7aD1390A899968Fb8F9e62aA

Deployer: 0xe9f15dcfe5712bdc0fb1858bf8dbae4fb4a6bddc

Token Supply: 420000000000

Checksum: 0ff4376de461c7768aeb7a75b5a3f1ee

Testnet version:
The tests were performed using the contract deployed on the Binance smart chain Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xbfd18c0a03e41643f6d68ec943dcbd6a1dc1f960#code

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

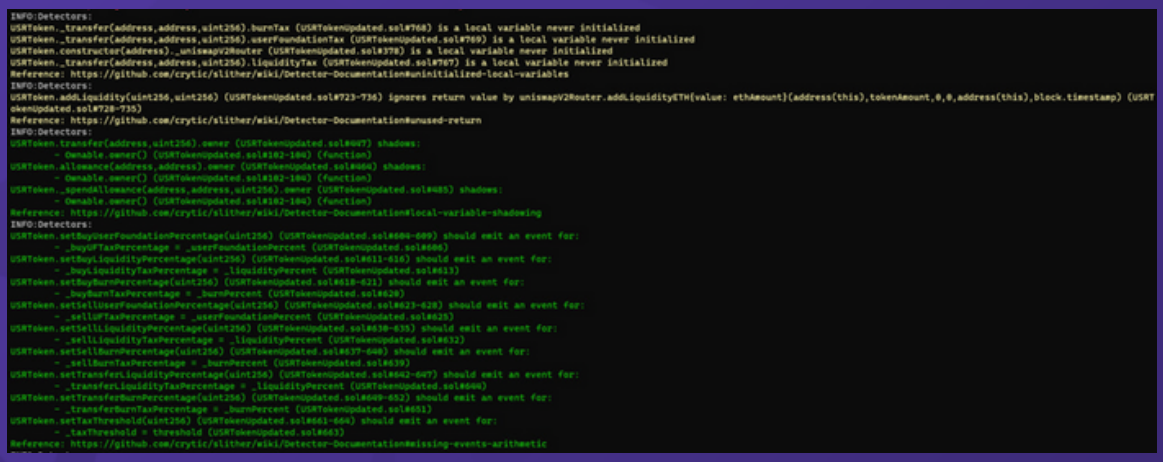
Issues on this level are minor details and warning that can remain unfixed.

Informational

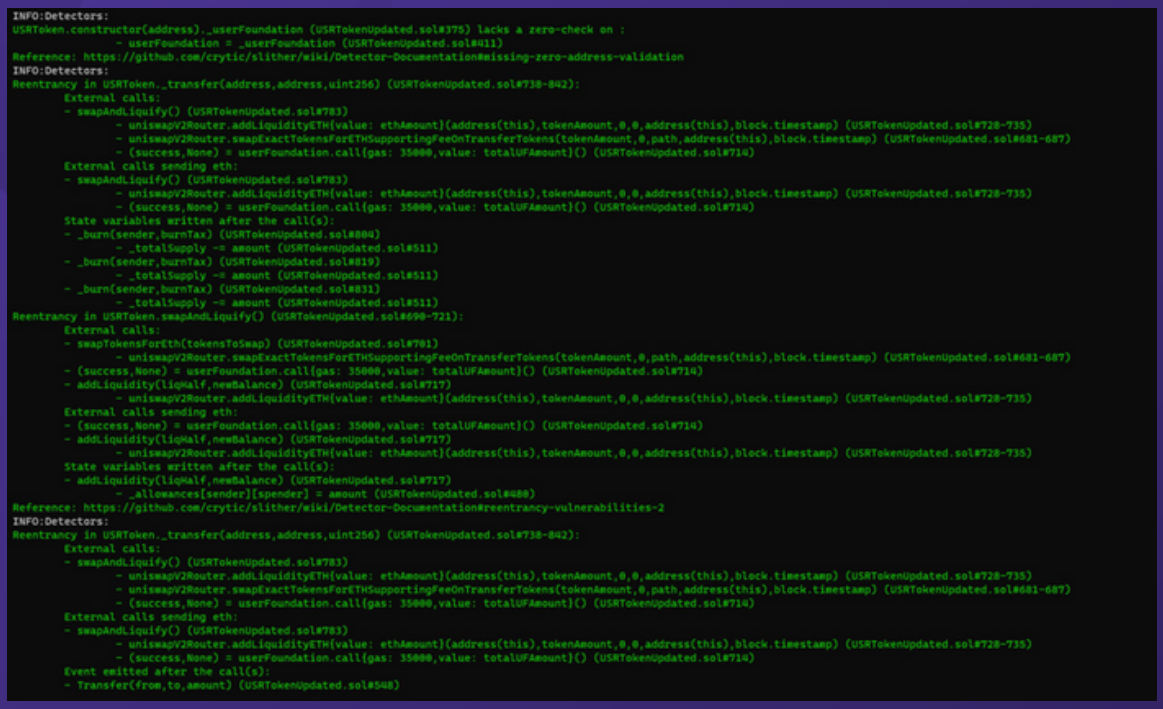
Issues on this level are minor details and warning that can remain unfixed.

expelee

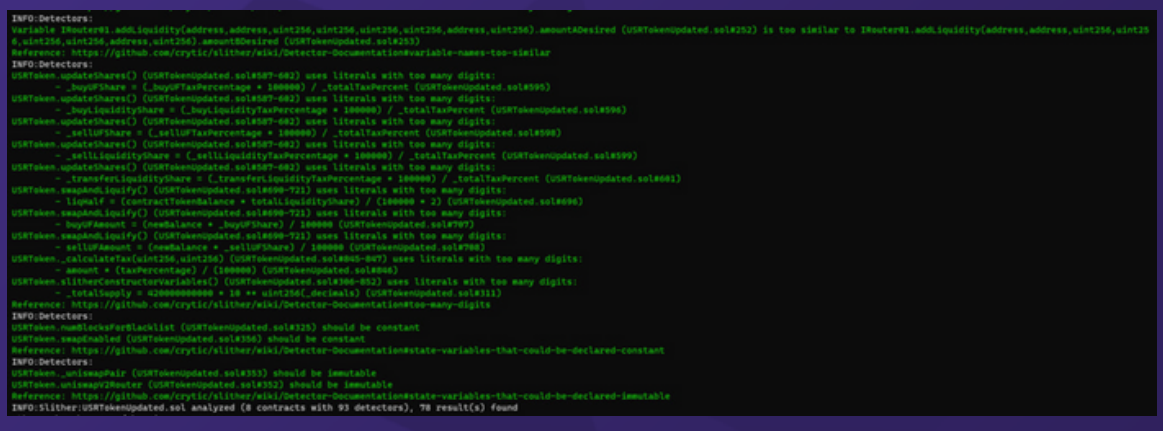
STATİC ANALYSİS



```
INFO:Detectors:
USRToken._transfer(address,address,uint256).burnTax (USRTokenUpdated.sol#768) is a local variable never initialized
USRToken._transfer(address,address,uint256).userFoundationTax (USRTokenUpdated.sol#769) is a local variable never initialized
USRToken.constructor(address)._uniswapV2Router (USRTokenUpdated.sol#378) is a local variable never initialized
USRToken._transfer(address,address,uint256).liquidityTax (USRTokenUpdated.sol#767) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
USRToken.addLiquidity(uint256,uint256) (USRTokenUpdated.sol#723-736) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRT
okenUpdated.sol#728-735)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
USRToken.transfer(address,uint256).owner (USRTokenUpdated.sol#447) shadows:
        - Ownable.owner() (USRTokenUpdated.sol#102-104) (function)
USRToken.allowance(address,address).owner (USRTokenUpdated.sol#464) shadows:
        - Ownable.owner() (USRTokenUpdated.sol#102-104) (function)
USRToken._spendAllowance(address,address,uint256).owner (USRTokenUpdated.sol#485) shadows:
        - Ownable.owner() (USRTokenUpdated.sol#102-104) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
USRToken.setBuyUserFoundationPercentage(uint256) (USRTokenUpdated.sol#606-609) should emit an event for:
        - _buyUFTaxPercentage = _userFoundationPercent (USRTokenUpdated.sol#606)
USRToken.setBuyLiquidityPercentage(uint256) (USRTokenUpdated.sol#611-616) should emit an event for:
        - _buyLiquidityTaxPercentage = _liquidityPercent (USRTokenUpdated.sol#613)
USRToken.setBuyBurnPercentage(uint256) (USRTokenUpdated.sol#618-621) should emit an event for:
        - _buyBurnTaxPercentage = _burnPercent (USRTokenUpdated.sol#620)
USRToken.setSellUserFoundationPercentage(uint256) (USRTokenUpdated.sol#623-628) should emit an event for:
        - _sellUFTaxPercentage = _userFoundationPercent (USRTokenUpdated.sol#625)
USRToken.setSellLiquidityPercentage(uint256) (USRTokenUpdated.sol#630-635) should emit an event for:
        - _sellLiquidityTaxPercentage = _liquidityPercent (USRTokenUpdated.sol#632)
USRToken.setSellBurnPercentage(uint256) (USRTokenUpdated.sol#637-640) should emit an event for:
        - _sellBurnTaxPercentage = _burnPercent (USRTokenUpdated.sol#639)
USRToken.setTransferLiquidityPercentage(uint256) (USRTokenUpdated.sol#642-647) should emit an event for:
        - _transferLiquidityTaxPercentage = _liquidityPercent (USRTokenUpdated.sol#644)
USRToken.setTransferBurnPercentage(uint256) (USRTokenUpdated.sol#649-652) should emit an event for:
        - _transferBurnTaxPercentage = _burnPercent (USRTokenUpdated.sol#651)
USRToken.setTaxThreshold(uint256) (USRTokenUpdated.sol#661-664) should emit an event for:
        - _taxThreshold = threshold (USRTokenUpdated.sol#663)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```



```
INFO:Detectors:
USRToken.constructor(address)._userFoundation (USRTokenUpdated.sol#375) lacks a zero-check on :
        - userFoundation = _userFoundation (USRTokenUpdated.sol#411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in USRToken._transfer(address,address,uint256) (USRTokenUpdated.sol#738-842):
        External calls:
        - swapAndLiquify() (USRTokenUpdated.sol#783)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRTokenUpdated.sol#728-735)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (USRTokenUpdated.sol#681-687)
                - (success,None) = userFoundation.call{gas: 35000,value: totalUFAmount}() (USRTokenUpdated.sol#714)
        External calls sending eth:
        - swapAndLiquify() (USRTokenUpdated.sol#783)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRTokenUpdated.sol#728-735)
                - (success,None) = userFoundation.call{gas: 35000,value: totalUFAmount}() (USRTokenUpdated.sol#714)
        State variables written after the call(s):
        - _burn(sender,burnTax) (USRTokenUpdated.sol#804)
                - _totalSupply -= amount (USRTokenUpdated.sol#511)
        - _burn(sender,burnTax) (USRTokenUpdated.sol#819)
                - _totalSupply -= amount (USRTokenUpdated.sol#511)
        - _burn(sender,burnTax) (USRTokenUpdated.sol#831)
                - _totalSupply -= amount (USRTokenUpdated.sol#511)
Reentrancy in USRToken.swapAndLiquify() (USRTokenUpdated.sol#690-721):
        External calls:
        - swapTokensForEth(tokensToSwap) (USRTokenUpdated.sol#701)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (USRTokenUpdated.sol#681-687)
        - (success,None) = userFoundation.call{gas: 35000,value: totalUFAmount}() (USRTokenUpdated.sol#714)
        - addLiquidity(liqHalf,newBalance) (USRTokenUpdated.sol#717)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRTokenUpdated.sol#728-735)
        External calls sending eth:
        - (success,None) = userFoundation.call{gas: 35000,value: totalUFAmount}() (USRTokenUpdated.sol#714)
        - addLiquidity(liqHalf,newBalance) (USRTokenUpdated.sol#717)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRTokenUpdated.sol#728-735)
        State variables written after the call(s):
        - addLiquidity(liqHalf,newBalance) (USRTokenUpdated.sol#717)
                - _allowances[sender][spender] = amount (USRTokenUpdated.sol#488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in USRToken._transfer(address,address,uint256) (USRTokenUpdated.sol#738-842):
        External calls:
        - swapAndLiquify() (USRTokenUpdated.sol#783)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRTokenUpdated.sol#728-735)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (USRTokenUpdated.sol#681-687)
                - (success,None) = userFoundation.call{gas: 35000,value: totalUFAmount}() (USRTokenUpdated.sol#714)
        External calls sending eth:
        - swapAndLiquify() (USRTokenUpdated.sol#783)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp) (USRTokenUpdated.sol#728-735)
                - (success,None) = userFoundation.call{gas: 35000,value: totalUFAmount}() (USRTokenUpdated.sol#714)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (USRTokenUpdated.sol#548)
```



```
INFO:Detectors:
Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (USRTokenUpdated.sol#252) is too similar to IRouter01.addLiquidity(address,address,uint256,uint25
6,uint256,uint256,address,uint256).amountBDesired (USRTokenUpdated.sol#253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
USRToken.updateShares() (USRTokenUpdated.sol#587-682) uses literals with too many digits:
        - _buyUFShare = (_buyUFTaxPercentage * 100000) / _totalTaxPercent (USRTokenUpdated.sol#595)
USRToken.updateShares() (USRTokenUpdated.sol#587-682) uses literals with too many digits:
        - _buyLiquidityShare = (_buyLiquidityTaxPercentage * 100000) / _totalTaxPercent (USRTokenUpdated.sol#596)
USRToken.updateShares() (USRTokenUpdated.sol#587-682) uses literals with too many digits:
        - _sellUFShare = (_sellUFTaxPercentage * 100000) / _totalTaxPercent (USRTokenUpdated.sol#598)
USRToken.updateShares() (USRTokenUpdated.sol#587-682) uses literals with too many digits:
        - _sellLiquidityShare = (_sellLiquidityTaxPercentage * 100000) / _totalTaxPercent (USRTokenUpdated.sol#599)
USRToken.updateShares() (USRTokenUpdated.sol#587-682) uses literals with too many digits:
        - _transferLiquidityShare = (_transferLiquidityTaxPercentage * 100000) / _totalTaxPercent (USRTokenUpdated.sol#601)
USRToken.swapAndLiquify() (USRTokenUpdated.sol#690-721) uses literals with too many digits:
        - liqHalf = (contractTokenBalance * totalLiquidityShare) / (100000 * 2) (USRTokenUpdated.sol#696)
USRToken.swapAndLiquify() (USRTokenUpdated.sol#690-721) uses literals with too many digits:
        - buyUFAmount = (newBalance * _buyUFShare) / 100000 (USRTokenUpdated.sol#707)
USRToken.swapAndLiquify() (USRTokenUpdated.sol#690-721) uses literals with too many digits:
        - sellUFAmount = (newBalance * _sellUFShare) / 100000 (USRTokenUpdated.sol#708)
USRToken._calculateTax(uint256,uint256) (USRTokenUpdated.sol#845-847) uses literals with too many digits:
        - amount * (taxPercentage) / (100000) (USRTokenUpdated.sol#846)
USRToken.slitherConstructorVariables() (USRTokenUpdated.sol#306-852) uses literals with too many digits:
        - _totalSupply = 420000000000 * 10 ** uint256(_decimals) (USRTokenUpdated.sol#311)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
USRToken.numBlocksForBlacklist (USRTokenUpdated.sol#325) should be constant
USRToken.swapEnabled (USRTokenUpdated.sol#356) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
USRToken._uniswapPair (USRTokenUpdated.sol#353) should be immutable
USRToken.uniswapV2Router (USRTokenUpdated.sol#352) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:USRTokenUpdated.sol analyzed (8 contracts with 93 detectors), 78 result(s) found
```

TESTNET VERSION

1- **Enable Tarding (passed):**
https://testnet.bscscan.com/tx/0xd6b337ff0e860b46813cf4f0e2274b86d335d541a9b7fedae3d98b25e4369802

2- Approve (passed):
https://testnet.bscscan.com/tx/0xd6b337ff0e860b46813cf4f0e2274b86d335d541a9b7fedae3d98b25e4369802

3- Set Buy Burn Percentage (passed):
https://testnet.bscscan.com/tx/0x1bb410d1fc6f49b0983aceecf5724b5bde23e98b0db44273feb6bff3de77af79

4- Set Buy Liquidity Percentage (passed):
https://testnet.bscscan.com/tx/0xc00da1ef6f8409a4a22ca79b49881dedeeeec1ec35e649ece3bdb336101bc33d

5- Set Buy User Foundation Percentage (passed):
https://testnet.bscscan.com/tx/0xc5c27dcd4670f4f53695750cf5a8b3e8f4fe7d0f5c7ceae401ef3c321005a32b

6- Set Excluded from Fee (passed):
https://testnet.bscscan.com/tx/0x0809f1cae2c49acbc26949c1074a41ed107721650de989bcbb42b35786c172d3

7- Transfer Ownership(passed):
https://testnet.bscscan.com/tx/0xfd8261dd2d37f15358886e0c33a60a61c507293776a09d46f85a043f29a62e03

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

HIGH RISK FINDING

Centralization – Owner can blacklist wallets.

Severity: High

function: addToBlacklis

Status: Open

Overview:

The owner can blacklist wallets from transferring of tokens for an indefinite period of time
which is not recommended. Which can lock user's token.

```
function addToBlacklist(address account) public onlyOwner{
    require(!blacklisted[account], "Account is already
blacklisted");
    require(_msgSender() != account, "Cannot blacklist self");
    blacklisted[account] = true.
  }
```

Suggestion:

There should be a locking period so that the wallet cannot be locked for an indefinite
Period of time.

LOW RISK FINDING

Severity: Low
subject: Missing Events
Status: Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it
transparent and easily accessible.

```
 function setExcludedFromFee(address account, bool excluded) external
onlyOwner {
   _isExcludedFromFee[account] = excluded;
 }

 function enableTrade() public onlyOwner
  trade_open = true;
 }

 function pauseTrade() public onlyOwner {
  trade_open = false;
 }

 function disableBurn(bool _status) public onlyOwner {
  burn_disable = _status;
 }
```

Suggestion:
Add an event to these important functions where address updation is happening. This can also be marked as indexed event for better off-chain tracking.

LOW RISK FINDING

Severity: Low
subject: Local variable shadowing
Status: Open

Overview:
Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice to avoid them.

```
function transfer(
    address to,
    uint256 amount
  ) public virtual returns (bool) {
    address owner = msg.sender;
    _transfer(owner, to, amount);
    return true;
  }
```

Suggestion:
To reduce high gas fees. It is suggested to remove. unused code from the contract.

INFORMATIONAL FINDING

Severity: Information
Subject: remove virtual if the function is not going to be overridden
Status: Open

Overview:

```
function _checkOwner() internal view virtual {
  require(owner() == _msgSender(), "Ownable: caller is not the owner");
}

/**
 * @dev Leaves the contract without owner. It will not be possible to call
 * `onlyOwner` functions anymore. Can only be called by the current owner.
 *
 * NOTE: Renouncing ownership will leave the contract without an owner,
 * thereby removing any functionality that is only available to the owner.
 */
function renounceOwnership() public virtual onlyOwner {
  _transferOwnership(address(0));
}

/**
 * @dev Transfers ownership of the contract to a new account (`newOwner`).
 * Can only be called by the current owner.
 */
function transferOwnership(address newOwner) public virtual onlyOwner { ///@audit remove virtual if the function is not going to be overridden -- info
  require(
    newOwner != address(0),
    "Ownable: new owner is the zero address"
  );
  _transferOwnership(newOwner);
}
```

Suggestion:
It is recommended to remove virtual if the function is not going to be overridden.

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.