Revoluzion Audit

Audit Report

Name	: World Of Rewards
Symbol	: WOR
Decimals	: 18
Address	: 0x9d622799F6eC7D8FCcb08A87c3c0f1c71b9315a3
Owner	: 0xc733a1E6B794B85591Ae3Bd359A32920C30B7E57
Network	: Binance Smart Chain
Type	: ERC20
Audited on	: 13 February 2024
Audited Score	: 76%

Revoluzion Audit

Table of Contents

Project Overview

Name	World Of Rewards
Symbol	WOR
Decimals	18
Total Supply	21,000,000
Tax	11% For Buy & Sell
Compiler Version	v0.8.18+commit.87f61d96
Optimization	Yes with 200 runs
License Type	MIT
Explorer Link	https://bscscan.com/address/0x9d622799f6ec7d8fccb08a87c3c0f1c71b9315a3
Create Tx	https://bscscan.com/tx/0x1c9dd3c95f6a86e37548861330613b26ca84b3ed07006a7b76ad2ab4c7427803
Creator	0xc733a1E6B794B85591Ae3Bd359A32920C30B7E57
Featured Wallet	MarketingWallet address : 0xCE5605079BB56d65eb5a03AF71D8951afCA3a393 LPWallet address : 0xc733a1E6B794B85591Ae3Bd359A32920C30B7E57 NFTFunds address : 0x0000000000000000000000000000000000000000 DevelopmentWallet1 address : 0x50E2f8A6643bB179Cb0A09E119472866FF242C89

	DevelopmentWallet2 address : 0x6A8bB8B42fCC1F8bd452D70A7cb71BcbFaBc400C DevelopmentWallet3 address : 0x607AbA106cFD33cD15aF8c67F3193DF07eadF3D6 DevelopmentWallet4 address : 0xDa6523Da05F966708B37b72635ad16BBc1B35A96 DevelopmentWallet5 address : 0x924a70eb81F1309d2691e0960068B758CBfA4179

Project Description

According to their website

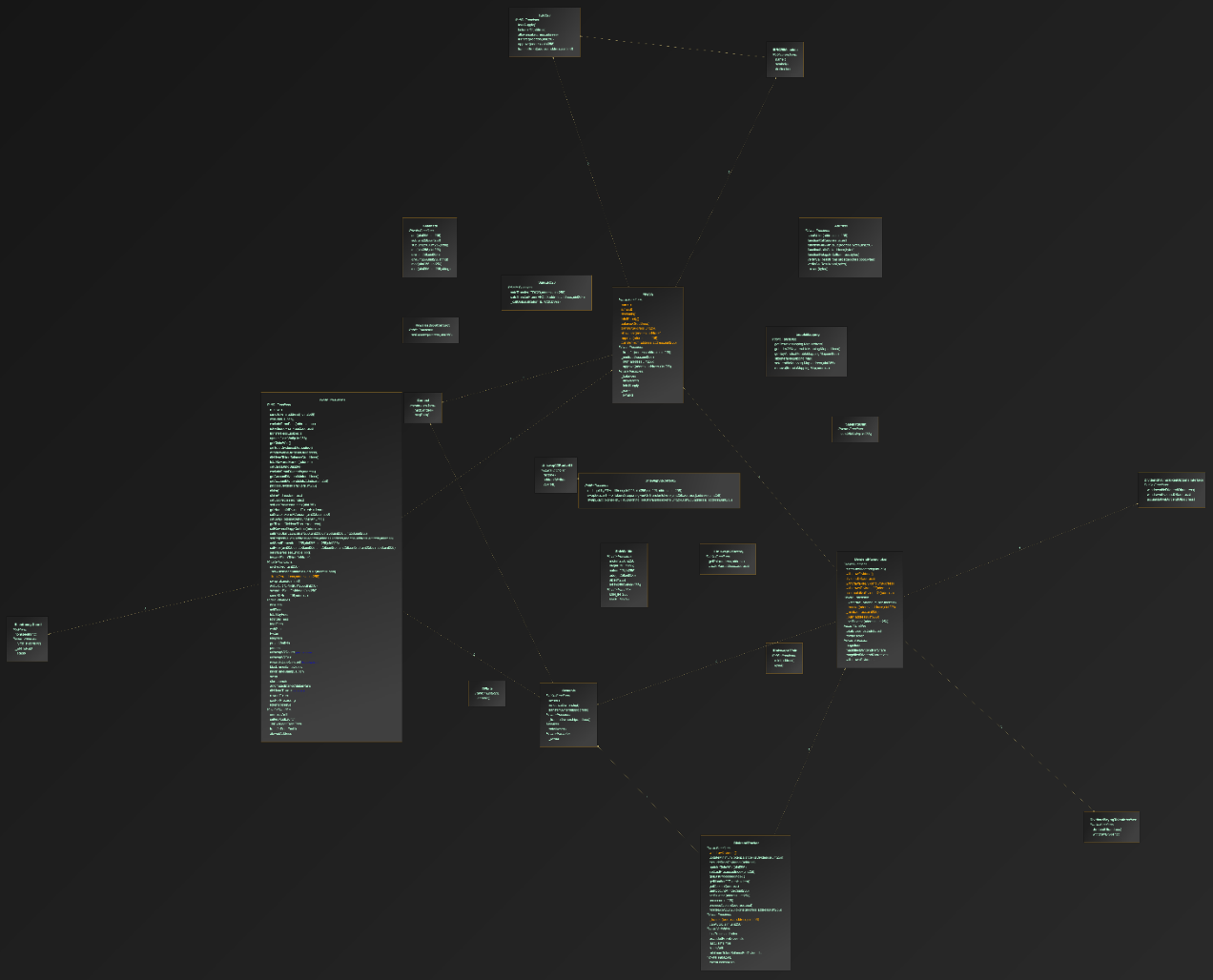
World of Rewards (WOR) is a rewards platform based on block chains that aims to create an ecosystem decentralized, transparent, and fair reward system for users. The project is based on the BSC and ETH block chains and uses smart contracts to automate the distribution of rewards to its holders on WOR DAPP

Release Date : TBA

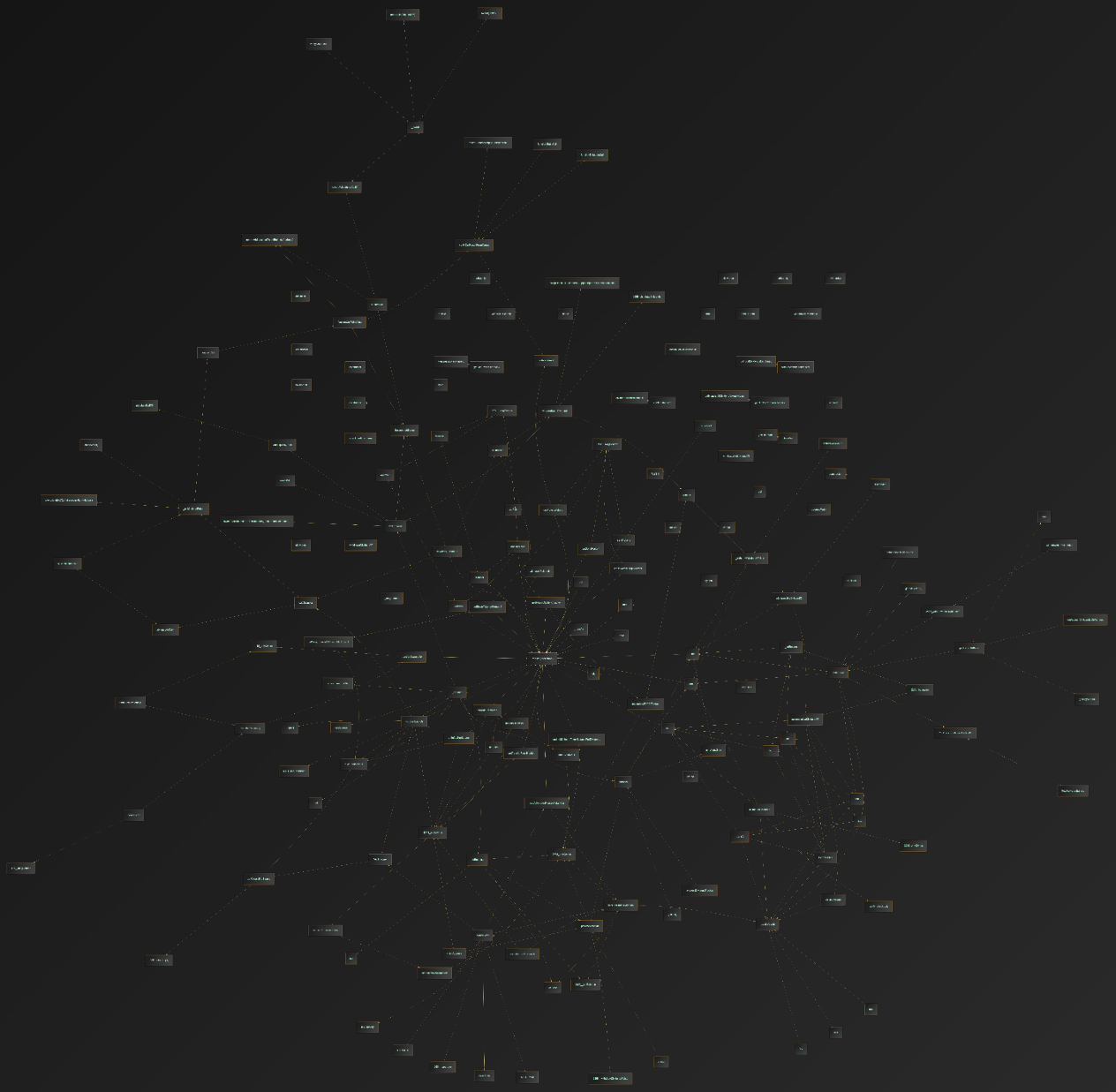
Category : DeFi

Contract Functions Interaction

Inheritance Graph



Call Graph (All)



Audit Overview

Threat Level

When conducting audit on smart contract(s), we first look for known vulnerabilities and issues within the code because any exploitation on such vulnerabilities and issues by malicious actors could potentially result in serious financial damage to the projects. All the issues and vulnerabilities will be categorized into the categories as provided below.

Critical

This category provides issues and vulnerabilities that are critical to the performance/functionality of the smart contract and should be fixed by project creator before moving to a live environment.

Medium

This category provides issues and vulnerabilities that are not that critical to the performance/functionality of the smart contract but is recommended to be fixed by project creator before moving to a live environment.

Minor

This category provides issues and vulnerabilities that are minor to the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment.

Informational

This category provides issues and vulnerability that have insignificant effect on the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment. However, fixing them can further improve the efficacy or security for features with a risk-free factor.

Notable Information

- Contract is a token contract with dividend, dividend payout are in Binance-Peg BSC-USD (BSC-USD)
- Dividends automated based on 300,000 gas to process during swapback and threshold payout is every 1 hour at once and token amount min > 10.
- Project owner and users should be aware that upon the initiation of worStartLaunch function, the lp fee will be change to 0% for both buy and sell fee.
- Project owner and users should be aware that upon the initiation of worStartLaunch function, the rewards fee will be change to 0% for both buy and sell fee
- Project owner and users should be aware that upon the initiation of the worStartLaunch function, the deploy timestamp will be modified to the time at which the function was initiated.
- Project owner and users should be aware that upon deployment, the smart contract owner and developmentWallet5 address were included into the list of alowedAddress.
- Project owner and users should be aware that upon initiating setProjectWallets with new addresses, the function will not automatically include the previous address for fee and this need to be done manually if the previous address is not supposed to be excluded from fees anymore.
- Project owner need to remember not to initiate the setSendPercent, setInfosStartLaunch, setSwapTokensAtAmount and setFees function with all the current value since the function to change the values does not have a restriction to prevent such action which is just a waste of gas for the owner.
- Project owner need to remember not to initiate the setProjectWallets function with all the current address since the function to change the addresses does not have a restriction to prevent such action which is just a waste of gas for the owner.

- Project owner and users should be aware that sendTokens, setSwapTokensAtAmount, setRewardsDappContract, setProjectWallets and setSendPercent will only be accessible by an address that was already added into the list of alowedAddress.

- Project owner and users should be aware that the restriction implementation within the setSwapTokensAtAmount function relies heavily on the native token balance of the main pair address and the new restriction will only takes place for the value update and the previous value can still remain active even when the balance in the pair address increase or decrease.

- Project owner and users should be aware that the amount of circulating supply returned by getCirculatingSupply function does not include the token supply that is available in pair, owner and token address and the amount of tokensReserve.

- Project owner can further improve the gas involved in deployment of the smart contract by removing SendBNB, UpdateUniswapV2Router, UpdateDividendTracker, GasForProcessingUpdated and SettedNftFunds events since they are not being used in any part of the smart contract.

- Project owner and users should be aware that updateMinimumTokenBalanceForDividends function cannot be use by anyone else except for the token smart contract and since there is no function in the token smart contract that is utilising this function, the function is totally inaccessible.

- Project owner need to remember not to initiate the setSendPercent values to a total of 1000 so that developmentWallet5 can still get some percentage from the distribution provided that the smart contract does not have any ether stuck in it since any ether stuck in the contract will be included in the fund to be transferred to the developmentWallet5.

- Project owner need to remember not to initiate the setRewardsDappContract address with an address that is not the address for rewardsDappContract since it could potentially lead to some undesired situation or worse turning the contract into a honey pot.

- Project owner and users should be aware that swapAndSendDividends function could potentially send more reward token to dividendTracker contract provided that the token contract already have some reward token stuck in it

since the logic of this function will be using the balance in the contract instead of the amount received from the swap.

- Project owner need to remember not to remove the main pair address from automatedMarketMakerPairs since the _setAutomatedMarketMakerPair function does not have a restriction to prevent such action although this can already be omitted since the function is a private function and no users will have access to it.

- Project owner should be aware that no other token pair can be added as automatedMarketMakerPairs after the deployment of the smart contract since there is no function to do so.

- Project owner should be aware that developmentWallet5 might get more fund during the swap instead of the intended percentage among the addresses provided that there are already native fund inside the smart contract since the whole remaining balance of the smart contract will be transferred based on the code logic of the smart contract.

- Project owner and users should be aware that anyone can initiate the auto distribution of the dividend as long as they are willing to pay the gas for the transaction by initiating the processDividendTracker function.

- Project owner and users should be aware that no fee will involve if at least one address that's excluded from fee is involved in the transaction.

- Project owner should be aware that despite the implementation of the bypass for the trade enable/pause feature, some newly updated automatic contract analyzers might still be able to detect the smart contract having such feature.

- Project owner should be aware that since the _setAutomatedMarketMakerPair function is a private function, there will be no other pair address that can be added into the list, hence no tax could possibility be obtained from transaction involving those pairs.

- Project owner and users should be aware that the swapTokens could potentially cause some transaction to fail depending on the value it will use for the calculation or not swapping the exact amount it is supposed to swap due to the nature of Solidity.

- Project owner and users should be aware that distribution of the dividend can be prevented by the project owner if the value minimumTokenBalanceForDividends was updated to a value that's impossible for any one of the addresses to have or at least reached.

- Project owner and users should be aware that any external call on the transfer function from anywhere might fail if the amount involved is 0 since the logic require the amount to be more that 0 at all time.

- Project owner and users should be aware that any transfer transaction that involve two addresses that are not excluded from fees will have the transfer fee incurred on the amount of the exact percentage similar to the fees incur on sell transaction.

- Project owner and users should be aware that the amount of tokensReserve is already excluded from the circulating supply although this supply of token is will technically still be circulating in the market as the amount was not locked anywhere and even if it was locked, the value might not properly being accounted for by the logic of the smart contract.

Caution Information

- Project owner and users should be aware that any ether or token, except the WOR token, received by the smart contract can be withdrawn to developmentWallet5 anytime by anyone as the function can be triggered by everyone.

- Project owner and users should be aware that when project owner initiate worStartLaunch function, project owner will be able to set marketingFees and/or developmentFees up to a total of 100% or more provided that the setInfosStartLaunch function was initiated with such value before the liquidity pool was created.

- Project owner and users should be aware that even when the worStartLaunch function should only be able to be initiated once, if the token in the liquidity pool has been drained to 0, this function can be initiated again.

- Project owner and users should be aware that based on the value used by the project owner when initiating worStartLaunch function, the tier fee structure based on the user's earnings could potentially turn the contract into a honeypot although this would only occur, if and only if, marketingFees and/or developmentFees were set up to a total of 100% or more by initiating the setInfosStartLaunch function such value before the liquidity pool was created

- Project owner need to be careful when initiating setProjectWallets function so as not to use a smart contract address that could not receive any ether.

- Project owner need to be careful when initiating setInfosStartLaunch function before the liquidity pool was created so as not to set marketingFees and/or developmentFees up to a total of 100% or more.

- Project owner and users should be aware that once the securityTime has passed and there is no more dividend being distributed for at least five days, anyone can initiate getTokensDividendTracker function to distribute a specific type of token from dividendTracker to the developmentWallet5 address even when the contract has not been renounced.

- Project owner and users should be aware that once it has been more than 360 days has passed since the blockTimestampDeploy timestamp, anyone can

initiate getTokensDividendTracker function to distribute a specific type of token from dividendTracker to the developmentWallet5 address even when the contract has not been renounced.

- Project owner should be aware that the updateMinimumTokenBalanceForDividends function is not accessible at all since the owner of the contract is actually the token contract instead of the project owner.

- Project owner and users should be aware that any transaction involving at least one address that's excluded from fee will still be possible to be conducted before liquidity was added.

- Project owner and users should be aware that any address that are already in the list of alowedAddres will be able to trigger sendTokens, setSwapTokensAtAmount, setRewardsDappContract, setProjectWallets and setSendPercent functions even after the ownership of the smart contract has been renounced.

- Project owner and users should be aware that once an address is excluded from dividend it will remain excluded forever since there is no function to be used to include it back for dividend.

- Project owner and users should be aware that even when there is no trade enable/pause feature in this smart contract, project owner will still be able to prevent trade by completely removing the token from main pair of liquidity pool.

- Project owner and users should be aware that the tax on buy or sell transaction can be bypassed if it involved other pair addresses that could potentially exist such as WOR/BUSD or WOR/USDT as examples since such pair addresses are not being stored in the smart contract to be used by the logic unless if the transferFee is enabled which will then utilise the sell fees.

- Users should be aware that project owner can manually adjust the lastProcessedIndex, which could potentially result in some users to be skipped or need to wait longer for their dividend auto distribution if the function is being misused.

- Users should be aware that project owner can help to manually distribute the dividend for users by manually initiating the claimAddress function.

- Users should be aware that this smart contract implemented a custom trade enabled/pause logic that bypass currently available automatic contract analyzers from detecting such feature and showing the smart contract as not having such feature.

- Project owner and users should be aware that this audit does not include any vulnerabilities that could potentially occur in conjunction to the logic of the function from rewardsDappContract since the source code of the said contracts were not provided to be audited along with this smart contract.

Bugs and Optimizations Detection

This table is based on the result obtained from running the smart contract through Slither's Solidity static analysis.

What it detects	Impact	Confidence	Status
Storage abiencoderv2 array	High	High	Passed
transferFrom uses arbitrary from	High	High	Passed
Modifying storage array by value	High	High	Passed
The order of parameters in a shift instruction is incorrect.	High	High	Passed
Multiple constructor schemes	High	High	Passed
Contract's name reused	High	High	Passed
Detected unprotected variables	High	High	Passed
Public mappings with nested variables	High	High	Passed
Right-To-Left-Override control character is used	High	High	Passed
State variables shadowing	High	High	Passed
Functions allowing anyone to destruct the contract	High	High	Passed
Uninitialized state variables	High	High	Passed
Uninitialized storage variables	High	High	Passed

Unprotected upgradeable contract	High	High	Passed
transferFrom uses arbitrary from with permit	High	Medium	Passed
Functions that send Ether to arbitrary destinations	High	Medium	Moderated
Tainted array length assignment	High	Medium	Passed
Controlled delegatecall destination	High	Medium	Passed
Payable functions using delegatecall inside a loop	High	Medium	Passed
msg.value inside a loop	High	Medium	Passed
Reentrancy vulnerabilities (theft of ethers)	High	Medium	Moderated
Signed storage integer array compiler bug	High	Medium	Passed
Unchecked tokens transfer	High	Medium	Moderated
Weak PRNG	High	Medium	Passed
Detects ERC20 tokens that have a function whose signature collides with EIP-2612's DOMAIN_SEPARATOR()	Medium	High	Passed
Detect dangerous enum conversion	Medium	High	Passed
Incorrect ERC20 interfaces	Medium	High	Passed
Incorrect ERC721 interfaces	Medium	High	Passed
Dangerous strict equalities	Medium	High	Moderated

Contracts that lock ether	Medium	High	Passed
Deletion on mapping containing a structure	Medium	High	Passed
State variables shadowing from abstract contracts	Medium	High	Passed
Tautology or contradiction	Medium	High	Passed
Unused write	Medium	High	Passed
Misuse of Boolean constant	Medium	Medium	Passed
Constant functions using assembly code	Medium	Medium	Passed
Constant functions changing the state	Medium	Medium	Passed
Imprecise arithmetic operations order	Medium	Medium	Moderated
Reentrancy vulnerabilities (no theft of ethers)	Medium	Medium	Moderated
Reused base constructor	Medium	Medium	Passed
Dangerous usage of tx.origin	Medium	Medium	Passed
Unchecked low-level calls	Medium	Medium	Passed
Unchecked send	Medium	Medium	Passed
Uninitialized local variables	Medium	Medium	Moderated
Unused return values	Medium	Medium	Moderated
Modifiers that can return the default value	Low	High	Passed

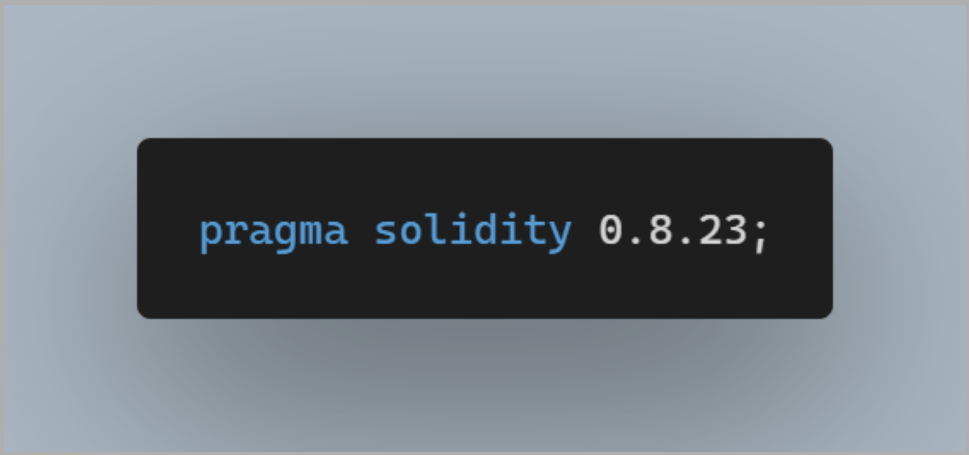
Built-in symbol shadowing	Low	High	Passed
Local variables shadowing	Low	High	Moderated
Uninitialized function pointer calls in constructors	Low	High	Passed
Local variables used prior their declaration	Low	High	Passed
Constructor called not implemented	Low	High	Passed
Multiple calls in a loop	Low	Medium	Moderated
Missing Events Access Control	Low	Medium	Passed
Missing Events Arithmetic	Low	Medium	Moderated
Dangerous unary expressions	Low	Medium	Passed
Missing Zero Address Validation	Low	Medium	Moderated
Benign reentrancy vulnerabilities	Low	Medium	Moderated
Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium	Moderated
Dangerous usage of block.timestamp	Low	Medium	Moderated
Assembly usage	Informational	High	Moderated
Assert state change	Informational	High	Passed
Comparison to boolean constant	Informational	High	Passed
Deprecated Solidity Standards	Informational	High	Passed
Un-indexed ERC20 event parameters	Informational	High	Passed

Function initializing state variables	Informational	High	Passed
Low level calls	Informational	High	Moderated
Missing inheritance	Informational	High	Moderated
Conformity to Solidity naming conventions	Informational	High	Moderated
If different pragma directives are used	Informational	High	Passed
Redundant statements	Informational	High	Passed
Incorrect Solidity version	Informational	High	Passed
Unimplemented functions	Informational	High	Passed
Unused state variables	Informational	High	Moderated
Costly operations in a loop	Informational	Medium	Passed
Functions that are not used	Informational	Medium	Moderated
Reentrancy vulnerabilities through send and transfer	Informational	Medium	Moderated
Variable names are too similar	Informational	Medium	Moderated
Conformance to numeric notation best practices	Informational	Medium	Passed
State variables that could be declared constant	Optimization	High	Moderated
Public function that could be declared external	Optimization	High	Passed

Contract Diagnostic

CODE	SEVERITY	DESCRIPTION
MI	Informational	Missing inheritance

SV — Incorrect Solidity Version

SEVERITY	Informational
LOCATION(S)	L818-L1038
	
DESCRIPTION	We identified that the DividendTracker contract (lines 818-1038 in contract.sol) does not inherit from the IRewardsDappContract interface (lines 338-340 in contract.sol). This deviation from best practices could potentially impact the functionality and interoperability of the contracts involved.
RECOMMENDATIONS	We recommend to ensure that the DividendTracker contract properly inherits from the IRewardsDappContract interface as required. This adjustment will enhance the clarity and maintainability of the codebase, aligning it with industry standards and reducing the likelihood of future compatibility issues.
STATUS	**Revoluzion acknowledgement:** Unresolved and should not have any major effect..

Constructor Calls

```
contract DividendTracker is Ownable, DividendPayingToken {
    using SafeMath for uint256;
    using SafeMathInt for int256;
    using IterableMapping for IterableMapping.Map;

    IterableMapping.Map private tokenHoldersMap;
    uint256 public lastProcessedIndex;

    mapping (address => bool) public excludedFromDividends;
    mapping (address => uint256) public lastClaimTimes;

    uint256 public claimWait;
    uint256 public minimumTokenBalanceForDividends;

    event ExcludeFromDividends(address indexed account);
    event ClaimWaitUpdated(uint256 indexed newValue, uint256 indexed oldValue);

    event Claim(address indexed account, uint256 amount, bool indexed automatic);

    constructor(uint256 minBalance, address _rewardToken) DividendPayingToken(
        "World Of Rewards Dividends", "WOR", _rewardToken
        ) {
        claimWait = 3600;
        minimumTokenBalanceForDividends = minBalance * 10 ** 18;
    }

    function _transfer(address, address, uint256) internal pure override {
        require(false, "No transfers allowed");
    }

    function withdrawDividend() public pure override {
        require(false, "withdrawDividend disabled. Use the 'claim' function on the main contract.");
    }

    function updateMinimumTokenBalanceForDividends(uint256 _newMinimumBalance) external onlyOwner {
        require(_newMinimumBalance != minimumTokenBalanceForDividends, "New minimum balance for dividend cannot be same as current minimum balance");
        minimumTokenBalanceForDividends = _newMinimumBalance;
    }

    function excludeFromDividends(address account) external onlyOwner {
        require(!excludedFromDividends[account]);
        excludedFromDividends[account] = true;

        _setBalance(account, 0);
        tokenHoldersMap.remove(account);

        emit ExcludeFromDividends(account);
    }

    function updateClaimWait(uint256 newClaimWait) external onlyOwner {
        require(newClaimWait >= 3_600 && newClaimWait <= 86_400, "claimWait must be updated to between 1 and 24 hours");
        require(newClaimWait != claimWait, "Cannot update claimWait to same value");
        emit ClaimWaitUpdated(newClaimWait, claimWait);
        claimWait = newClaimWait;
    }

    function setLastProcessedIndex(uint256 index) external onlyOwner {
        lastProcessedIndex = index;
    }

    function getLastProcessedIndex() external view returns(uint256) {
        return lastProcessedIndex;
    }

    function getNumberOfTokenHolders() external view returns(uint256) {
        return tokenHoldersMap.keys.length;
    }

    function getAccount(address _account)
        public view returns (
            address account,
            int256 index,
```

```
            int256 iterationsUntilProcessed,
            uint256 withdrawableDividends,
            uint256 totalDividends,
            uint256 lastClaimTime,
            uint256 nextClaimTime,
            uint256 secondsUntilAutoClaimAvailable) {
        account = _account;

        index = tokenHoldersMap.getIndexOfKey(account);

        iterationsUntilProcessed = -1;

        if(index >= 0) {
            if(uint256(index) > lastProcessedIndex) {
                iterationsUntilProcessed = index.sub(int256(lastProcessedIndex));
            }
            else {
                uint256 processesUntilEndOfArray = tokenHoldersMap.keys.length > lastProcessedIndex ?
                                                        tokenHoldersMap.keys.length.sub(lastProcessedIndex) :
                                                        0;


                iterationsUntilProcessed = index.add(int256(processesUntilEndOfArray));
            }
        }


        withdrawableDividends = withdrawableDividendOf(account);
        totalDividends = accumulativeDividendOf(account);

        lastClaimTime = lastClaimTimes[account];

        nextClaimTime = lastClaimTime > 0 ?
                                    lastClaimTime.add(claimWait) :
                                    0;

        secondsUntilAutoClaimAvailable = nextClaimTime > block.timestamp ?
                                                 nextClaimTime.sub(block.timestamp) :
                                                 0;
    }

    function getAccountAtIndex(uint256 index)
        public view returns (
            address,
            int256,
            int256,
            uint256,
            uint256,
            uint256,
            uint256,
            uint256,
            uint256) {
        if(index >= tokenHoldersMap.size()) {
            return (0x0000000000000000000000000000000000000000, -1, -1, 0, 0, 0, 0, 0);
        }

        address account = tokenHoldersMap.getKeyAtIndex(index);

        return getAccount(account);
    }

    function canAutoClaim(uint256 lastClaimTime) private view returns (bool) {
        if(lastClaimTime > block.timestamp)  {
            return false;
        }

        return block.timestamp.sub(lastClaimTime) >= claimWait;
    }

    function setBalance(address payable account, uint256 newBalance) external onlyOwner {
        if(excludedFromDividends[account]) {
            return;
        }

        if(newBalance >= minimumTokenBalanceForDividends) {
            _setBalance(account, newBalance);
```

```
            tokenHoldersMap.set(account, newBalance);
        }
        else {
            _setBalance(account, 0);
            tokenHoldersMap.remove(account);
        }

        processAccount(account, true);
    }

    function process(uint256 gas) public returns (uint256, uint256, uint256) {
        uint256 numberOfTokenHolders = tokenHoldersMap.keys.length;

        if(numberOfTokenHolders == 0) {
            return (0, 0, lastProcessedIndex);
        }

        uint256 _lastProcessedIndex = lastProcessedIndex;

        uint256 gasUsed = 0;

        uint256 gasLeft = gasleft();

        uint256 iterations = 0;
        uint256 claims = 0;

        while(gasUsed < gas && iterations < numberOfTokenHolders) {
            _lastProcessedIndex++;

            if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
                _lastProcessedIndex = 0;
            }

            address account = tokenHoldersMap.keys[_lastProcessedIndex];

            if(canAutoClaim(lastClaimTimes[account])) {
                if(processAccount(payable(account), true)) {
                    claims++;
                }
            }

            iterations++;

            uint256 newGasLeft = gasleft();

            if(gasLeft > newGasLeft) {
                gasUsed = gasUsed.add(gasLeft.sub(newGasLeft));
            }

            gasLeft = newGasLeft;
        }

        lastProcessedIndex = _lastProcessedIndex;

        return (iterations, claims, lastProcessedIndex);
    }

    function processAccount(address payable account, bool automatic) public onlyOwner returns (bool) {
        uint256 amount = _withdrawDividendOfUser(account);

        if(amount > 0) {
            lastClaimTimes[account] = block.timestamp;
            emit Claim(account, amount, automatic);
            return true;
        }

        return false;
    }

    function rescueAnyBEP20Tokens(address _tokenAddr,address _to, uint256 amount) external onlyOwner {
        IERC20(_tokenAddr).transfer(_to, amount);
    }

}
```

Disclaimer

This report only shows findings based on our limited project analysis according to the good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall online presence and team transparency details of which are set out in this report. To get a full view of our analysis, **it is important for you to read the full report**. Under no circumstances did Revoluzion Audit receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. **Our team provides no guarantees against the sale of team tokens or the removal of liquidity by the project** audited in this document.

While **we have done our best to conduct thorough analysis to produce this report**, it is crucial to note that you should not rely solely on this report and use the content provided in this document as financial advice or a reason to buy any investment. The Our team disclaims any liability against us for the resulting losses based on the you decision made by relying on the content of this report. **You must conduct your own independent investigations before making any decisions** to protect yourselves from being scammed. We go into more detail on this in the disclaimer clause in the next page — please make sure to read it in full.

Full Disclaimer Clause

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 all copies of this report downloaded and/or printed by you. This report is provided for information purposes only, on a non-reliance basis and does not constitute to any investment advice.

No one shall have any right to rely on the report or its contents, and Revoluzion Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (collectively known as Revoluzion) owe no duty of care towards you or any other person, nor do we 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 Revoluzion 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, Revoluzion hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Revoluzion, 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.

The analysis of the security is purely based on the smart contracts, website, social media, and team.