Cyberscope

Audit Report

YachtingVerse

December 2023

Network	ARBITRUM
Address	0xde70aed3d14d39b4955147efcf272334bdb75ab5
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	UOD	Unnecessary Override Declaration	Unresolved
	RC	Redundant Calculation	Unresolved
	RCS	Redundant Code Segments	Unresolved
	RSD	Redundant Swap Duplication	Unresolved
	PVC	Price Volatility Concern	Unresolved
	DDP	Decimal Division Precision	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	MEM	Misleading Error Messages	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L15	Local Scope Variable Shadowing	Unresolved
	L16	Validate Variable Setters	Unresolved
	L18	Multiple Pragma Directives	Unresolved
	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	YachtingVerse
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://arbiscan.io/address/0xde70aed3d14d39b4955147efcf272334bdb75ab5
Address	0xde70aed3d14d39b4955147efcf272334bdb75ab5
Network	ARBITRUM
Symbol	YACHT
Decimals	18
Total Supply	7,500,000

Audit Updates

Initial Audit	12 Feb 2022 https://github.com/cyberscope-io/audits/blob/main/yacht/v1/audit.pdf
Corrected Phase 2	26 Feb 2022 https://github.com/cyberscope-io/audits/blob/main/yacht/v2/audit.pdf
Corrected Phase 3	19 Apr 2023 https://github.com/cyberscope-io/audits/blob/main/yacht/v3/audit.pdf
Corrected Phase 4	25 Apr 2023

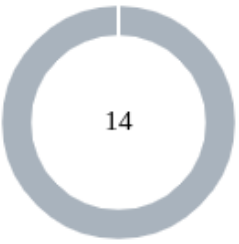
	https://github.com/cyberscope-io/audits/blob/main/yacht/v4/audit.pdf
Corrected Phase 5	07 Dec 2023

Source Files

Filename	SHA256
Token.sol	638458be55641a2ee6a5e22f2f555000f4679a6fc437e8f6e33d9f27e5983055
Ownable.sol	33422e7771fefe5fbfe8934837515097119d82a50eda0e49b38e4d6a64a1c25d
Initializable.sol	b05c26d897c4178cbdb35ad113527e463e1bdeae5764869318a54f93c8b98a94
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81498c1d98120e2877a1439f0c31b5a
IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
IERC20Metadata.sol	b10e2f8bcc3ed53a5d9a82a29b1ad3209225331bb4de4a0459862a762cf83a1a
IERC20.sol	7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587
ERC20Burnable.sol	480b22ce348050fdb85a693e38ed6b4767a94e4776fc6806d6808a0ec171177e
ERC20.sol	f70c6ae5f2dda91a37e17cfcbec390cc59515ed0d34e316f036f5431b5c0a3f2

Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
CoinDividendTracker.sol	5043f58deb43d616907a389eac505e16312ed894d29224b833b57973832f2ec7

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	14

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	14	0	0	0

UOD - Unnecessary Override Declaration

Criticality	Minor / Informative
Location	Token.sol#L152
Status	Unresolved

Description

The contract is currently implementing an override of the decimals function, which simply returns the value 18. This override is redundant since the extending token contract already specifies 18 decimals as its standard. In the context of ERC-20 tokens, 18 decimals is a common default, and overriding this function to return the same value adds unnecessary complexity to the contract. This redundancy does not contribute to the functionality of the contract and could potentially lead to confusion about the necessity of this override.

```
    function decimals() public pure override returns (uint8) {
        return 18;
    }
```

Recommendation

It is recommended to remove the `override` keyword from the decimals function, assuming the extending token contract already defines 18 decimals. This action will simplify the contract by eliminating redundant code, thereby enhancing its clarity and efficiency. The removal of this unnecessary override will not impact the contract's functionality but will contribute to a cleaner and more maintainable codebase.

RC - Redundant Calculation

Criticality	Minor / Informative
Location	Token.sol#L266
Status	Unresolved

Description

The contract sets the variable `token2Swap` is to be equal to `_marketingPending`. This assignment makes the subsequent calculation of marketingPortion using `coinsReceived * _marketingPending / token2Swap redundant`, as `token2Swap` is always equal to `_marketingPending`. This redundancy not only adds unnecessary complexity to the contract but also potentially wastes gas, as it involves a division operation where the divisor and dividend are always the same.

```
if (false || _marketingPending > 0) {
   uint256 token2Swap = 0 + _marketingPending;
   bool success = false;

   _swapTokensForCoin(token2Swap);
    uint256 coinsReceived = address(this).balance;

   uint256 marketingPortion = coinsReceived * _marketingPending /
token2Swap;
```

Recommendation

It is recommended to simplify the calculation of marketingPortion by removing the division operation, as `token2Swap` is always equal to `_marketingPending`. The marketingPortion can be directly set to coinsReceived, assuming the intention is to distribute all received coins as marketing rewards. This change will streamline the contract's logic, reduce gas costs, and enhance overall code efficiency and readability.

RCS - Redundant Code Segments

Criticality	Minor / Informative
Location	Token.sol#L126,259,260
Status	Unresolved

Description

The contract is implementing an `if` statement that contains the `false` keyword in the if condition. This inclusion of false does not impact the logical outcome of the condition, as the false value in a logical OR (||) operation is effectively ignored. The result of this condition solely depends on the evaluation of `_marketingPending > 0`. As a result, the presence of the `false` keyword in this context is redundant. Additionally, the contract includes unnecessary addition of zero to variables. These additions of zero are redundant since adding zero to any variable does not change its value and only serves to complicate the code without providing any functional benefit.

```
if (false || _marketingPending > 0)
return 0 + _marketingPending + _liquidityPending + _rewardsPending;
0 + _marketingPending;
```

Recommendation

It is recommended to remove the `false` keyword from the `if` statement and the redundant zero additions from the expressions to enhance code clarity and maintainability. Simplifying the `if` condition and the calculations will make the code more readable and straightforward. These changes will not alter the functionality of the contract but will improve the overall quality and understandability of the code, as well as potentially reduce gas costs by eliminating unnecessary operations.

RSD - Redundant Swap Duplication

Criticality	Minor / Informative
Location	Token.sol#L154,201,263
Status	Unresolved

Description

The contract contains multiple swap methods that individually perform token swaps and transfer promotional amounts to specific addresses and features. This redundant duplication of code introduces unnecessary complexity and increases dramatically the gas consumption. By consolidating these operations into a single swap method, the contract can achieve better code readability, reduce gas costs, and improve overall efficiency.

```
   function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        ...
         _swapTokensForCoin(token2Swap);
                ...
            }

            if (_liquidityPending > 0) {
                _swapAndLiquify(_liquidityPending);
                _liquidityPending = 0;
            }

            if (_rewardsPending > 0 && getNumberOfDividendTokenHolders()
> 0) {
                _sendDividends(_rewardsPending);
                _rewardsPending = 0;
            }
            ...
        }

    function _swapAndLiquify(uint256 tokenAmount) private returns
(uint256 leftover) {
        // Sub-optimal method for supplying liquidity
        uint256 halfAmount = tokenAmount / 2;
        uint256 otherHalf = tokenAmount - halfAmount;

        _swapTokensForCoin(halfAmount);
        ...
    }

    function _sendDividends(uint256 tokenAmount) private {
        _swapTokensForCoin(tokenAmount);
     ...
    }
```

Recommendation

A more optimized approach could be adopted to perform the token swap operation once for the total amount of tokens and distribute the proportional amounts to the corresponding addresses, eliminating the need for separate swaps.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	Token.sol#L263
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
        bool canSwap = getAllPending() >= swapThreshold;

         if (!_swapping && !AMMPairs[from] && canSwap) {
             _swapping = true;

             if (false || _marketingPending > 0) {
                 uint256 token2Swap = 0 + _marketingPending;
                 bool success = false;

                 _swapTokensForCoin(token2Swap);
             ...
             }
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	Token.sol#L307
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
_marketingPending += fees * marketingFees[txType] / totalFees[txType];

_liquidityPending += fees * liquidityFees[txType] / totalFees[txType];

_rewardsPending += fees * rewardsFees[txType] / totalFees[txType];
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Token.sol#L242
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
    function excludeFromFees(address account, bool isExcluded)
public onlyOwner {
        isExcludedFromFees[account] = isExcluded;

        emit ExcludeFromFees(account, isExcluded);
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	CoinDividendTracker.sol#L72
Status	Unresolved

Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(totalSupply() > 0)
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	CoinDividendTracker.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L49,53,54,55,58,59,62,63,99,119,129,138,183,189,219 CoinDividendTracker.sol#L56,250,383
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
    mapping (address => bool) public AMMPairs;
    event marketingAddressUpdated(address marketingAddress);
    event marketingFeesUpdated(uint16 buyFee, uint16 sellFee,
uint16 transferFee);
    event marketingFeeSent(address recipient, uint256 amount);
    event liquidityFeesUpdated(uint16 buyFee, uint16 sellFee,
uint16 transferFee);
    event rewardsFeesUpdated(uint16 buyFee, uint16 sellFee,
uint16 transferFee);
    event rewardsFeeSent(uint256 amount);
    function initialize(address _router) initializer external {
    ...
    uint256 constant internal magnitude = 2**128;
    ...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	CoinDividendTracker.sol#L65
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
string memory _name
string memory _symbol
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	Token.sol#L184
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
TokensReceiver = _newAddress;
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	Token.sol#L13 Ownable.sol#L4 IUniswapV2Router02.sol#L1 IUniswapV2Router01.sol#L1 IUniswapV2Pair.sol#L1 IUniswapV2Factory.sol#L1 Initializable.sol#L3 IERC20Metadata.sol#L4 IERC20.sol#L4 ERC20B urnable.sol#L4 ERC20.sol#L4 Context.sol#L4 CoinDividendTracker.sol#L6
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity 0.8.19;
pragma solidity >=0.5.0;
pragma solidity >=0.6.2;
pragma solidity ^0.8.0;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	CoinDividendTracker.sol#L6
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
YachtingVerse	Implementation	ERC20, ERC20Burnable, Ownable, DividendTrackerFunctions, Initializable		
		Public	✓	ERC20
	initialize	External	✓	initializer
		External	Payable	-
	decimals	Public		-
	_swapTokensForCoin	Private	✓	
	updateSwapThreshold	Public	✓	onlyOwner
	getAllPending	Public		-
	marketingAddressSetup	Public	✓	onlyOwner
	marketingFeesSetup	Public	✓	onlyOwner
	_swapAndLiquify	Private	✓	
	_addLiquidity	Private	✓	
	addLiquidityFromLeftoverTokens	External	✓	onlyOwner
	lpTokensReceiverSetup	Public	✓	onlyOwner
	liquidityFeesSetup	Public	✓	onlyOwner
	_sendDividends	Private	✓	

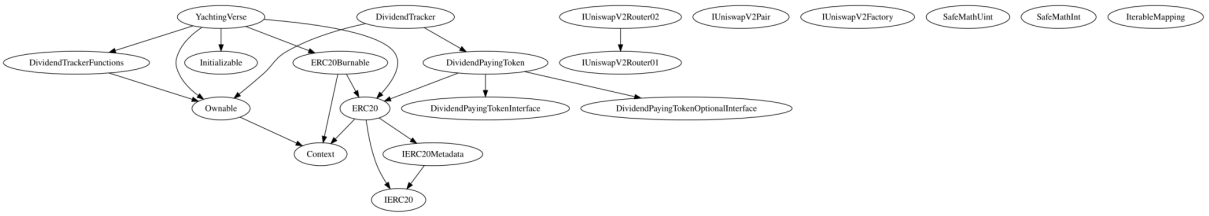
	excludeFromDividends	External	✓	onlyOwner
	_excludeFromDividends	Internal	✓	
	rewardsFeesSetup	Public	✓	onlyOwner
	_burn	Internal	✓	
	_mint	Internal	✓	
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	_updateRouterV2	Private	✓	
	setAMMPair	External	✓	onlyOwner
	_setAMMPair	Private	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
SafeMathUint	Library			
	toInt256Safe	Internal		
SafeMathInt	Library			
	toUint256Safe	Internal		
DividendPayingTokenInterface	Interface			
	dividendOf	External		-
DividendPayingTokenOptionalInterface	Interface			

	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
		Public	✓	ERC20
		External	Payable	-
	distributeDividends	Public	Payable	-
	_withdrawDividend	Internal	✓	
	dividendOf	Public		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	accumulativeDividendOf	Public		-
	_mint	Internal	✓	
	_burn	Internal	✓	
	_setBalance	Internal	✓	
IterableMapping	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-

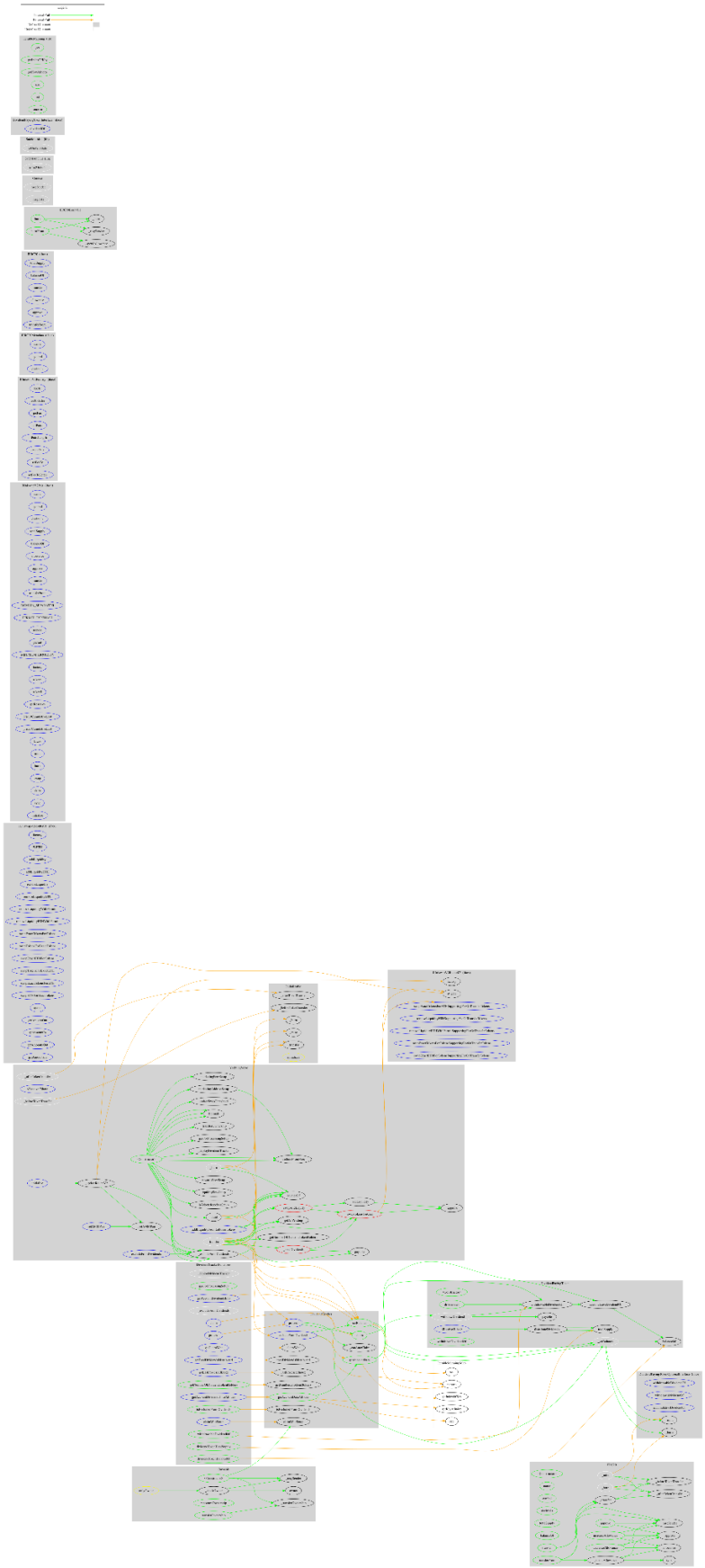
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-
DividendTracker	Implementation	Ownable, DividendPayingToken		
		Public	✓	DividendPayingToken
	excludeFromDividends	External	✓	onlyOwner
	claimWaitSetup	Public	✓	onlyOwner
	getNumberOfTokenHolders	External		-
	getAccountData	Public		-
	getAccountDataAtIndex	Public		-
	claim	Public	✓	onlyOwner
	_canAutoClaim	Private		
	setBalance	Public	✓	onlyOwner
	process	External	✓	onlyOwner
DividendTrackerFunctions	Implementation	Ownable		
	_deployDividendTracker	Internal	✓	
	gasForProcessingSetup	Public	✓	onlyOwner
	claimWaitSetup	External	✓	onlyOwner
	_excludeFromDividends	Internal	✓	
	isExcludedFromDividends	Public		-

	claim	External	✓	-
	getClaimWait	External		-
	getTotalDividendsDistributed	External		-
	withdrawableDividendOf	Public		-
	dividendTokenBalanceOf	Public		-
	dividendTokenTotalSupply	Public		-
	getAccountDividendsInfo	External		-
	getAccountDividendsInfoAtIndex	External		-
	getLastProcessedIndex	External		-
	getNumberOfDividendTokenHolders	Public		-
	process	External	✓	-

Inheritance Graph



Flow Graph



Summary

YachtingVerse contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. YachtingVerse is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 25% fees.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io