Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	VatCap Coin
Website:	https://vatcapcoin.com
Platform:	Binance Smart Chain
Language:	Solidity
Date:	May 4th, 2022

Table of contents

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the VatCap Coin team to perform the Security audit of the VatCap Coin smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on May 4th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

The VatCap Coin contract is a BEP20 standard smart contract with dividend distribution. It has functions like approveMax, setMaxWalletPercent, setTxLimit, basicTransfer, etc.

Audit scope

Name	**Code Review and Security Analysis Report for VatCap Coin cSmart Contract**
Platform	**BSC / Solidity**
File	VatCapCoin.sol
File MD5 Hash	34FC240D92A9B0AACAC9AB16799538D5
Online Code Link	0x9a1B64E3536E8C210b6485CDDf11344130aecF9D
Audit Date	May 4th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: VatCap Coin • Symbol: VCC • Decimals: 18 • Total Supply: 5 Billion • Liquidity Fee: 2% • Reflection Fee: 3% • Marketing Fee: 2% • Ecosystem Fee: 1% • Burn Fee: 1% • Fee Denominator: 100 • Sell Multiplier: 100 • Target Liquidity: 85 • Target Liquidity Denominator: 100 • Distributor Gas: 0.5 Million • Cooldown Timer Interval: 1 minute	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 1 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 1 smart contract file. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in VatCap coin are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the VatCap Coin protocol.

The VatCap Coin team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not well** commented on smart contracts.

Documentation

We were given a VatCap Coin smart contract code in the form of a BSCScan Web Link.The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	authorized	modifier	Passed	No Issue
4	authorize	write	access only Owner	No Issue
5	unauthorize	write	Passed	No Issue
6	isOwner	read	Passed	No Issue
7	isAuthorized	read	Passed	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	receive	external	Passed	No Issue
10	totalSupply	external	Passed	No Issue
11	decimals	external	Passed	No Issue
12	symbol	external	Passed	No Issue
13	name	external	Passed	No Issue
14	getOwner	external	Passed	No Issue
15	balanceOf	read	Passed	No Issue
16	allowance	external	Passed	No Issue
17	approve	write	Passed	No Issue
18	approveMax	external	Passed	No Issue
19	transfer	external	Passed	No Issue
20	transferFrom	external	Passed	No Issue
21	setMaxWalletPercent_base1000	external	access only Owner	No Issue
22	setMaxTxPercent_base1000	external	access only Owner	No Issue
23	setTxLimit	external	access by authorized	No Issue
24	_transferFrom	internal	Passed	No Issue
25	_basicTransfer	internal	Passed	No Issue
26	checkTxLimit	internal	Passed	No Issue
27	shouldTakeFee	internal	Passed	No Issue
28	takeFee	internal	Passed	No Issue
29	shouldSwapBack	internal	Passed	No Issue
30	clearStuckBalance	external	access by authorized	No Issue
31	clearStuckBalance_sender	external	access by authorized	No Issue
32	set_sell_multiplier	external	access only Owner	No Issue
33	tradingStatus	write	access only Owner	No Issue
34	cooldownEnabled	write	access only Owner	No Issue
35	swapBack	internal	Passed	No Issue
36	setIsDividendExempt	external	Missing required error message	Refer audit findings
37	enable_blacklist	write	access only Owner	No Issue
38	manage_blacklist	write	Unbounded Loop	No Big Issue

39	setIsFeeExempt	external	access by authorized	No Issue
40	setIsTxLimitExempt	external	access by authorized	No Issue
41	setIsTimelockExempt	external	access by authorized	No Issue
42	setFees	external	access by authorized	No Issue
43	setFeeReceivers	external	access by authorized	No Issue
44	setSwapBackSettings	external	access by authorized	No Issue
45	setTargetLiquidity	external	access by authorized	No Issue
46	setDistributionCriteria	external	access by authorized	No Issue
47	setDistributorSettings	external	Missing required error message	Refer audit findings
48	getCirculatingSupply	read	Passed	No Issue
49	getLiquidityBacking	read	Passed	No Issue
50	isOverLiquified	read	Passed	No Issue
51	multiTransfer	external	access only Owner	No Issue
52	multiTransfer_fixed	external	access only Owner	No Issue
53	initialization	modifier	Missing required error message	Refer audit findings
54	onlyToken	modifier	Missing required error message	Refer audit findings
55	setShare	external	access only Token	No Issue
56	setDistributionCriteria	external	access only Token	No Issue
57	deposit	external	access only Token	No Issue
58	process	external	access only Token	No Issue
59	shouldDistribute	internal	Passed	No Issue
60	distributeDividend	internal	Passed	No Issue
61	claimDividend	external	Passed	No Issue
62	getUnpaidEarnings	read	Passed	No Issue
63	getCumulativeDividends	internal	Passed	No Issue
64	addShareholder	internal	Passed	No Issue
65	removeShareholder	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Missing events:

It is recommended to emit appropriate events in the functions where significant state is being changed. This helps the UI elements to interact with the blockchain. We suggest adding events in the following functions:

• authorize • unauthorize • setMaxWalletPercent_base1000 • setMaxTxPercent_base1000 • set_sell_multiplier • tradingStatus • cooldownEnabled • enable_blacklist • setTxLimit • clearStuckBalance • clearStuckBalance_sender	• setIsFeeExempt • setIsTxLimitExempt • setIsTimelockExempt • setFees • setFeeReceivers • setSwapBackSettings • setTargetLiquidity • setDistributionCriteria • setDistributorSettings • setIsDividendExempt

Status: Open

Very Low / Informational / Best practices:

(1) Missing required error message:

```
function setDistributorSettings(uint256 gas) external authorized {
    require(gas < 750000);
    distributorGas = gas;
}
```

```
function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
```

There are many places where there are no error messages in the required condition. It is best practice to put relevant error messages which will help identify the failure of the transactions.

Resolution: We suggest setting relevant error messages.

Status: Open

(2) Unbounded loop

```
function manage_blacklist(address[] calldata addresses, bool status) public onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        isBlacklisted[addresses[i]] = status;
    }
}
```

If the owner inputs many addresses, then it may hit the block's gas limit, and cause the transaction to fail. Solidity is a very resource constrained language and does not have as extensive resource allocation as other programming languages.

Resolution: We suggest setting a limit on the number of wallets used, or the owner can acknowledge as being careful and only use a limited number of wallets.

Status: Open

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setMaxWalletPercent_base1000: Owner can set max wallet percent base value.
- setMaxTxPercent_base1000: Owner can set max transaction percent base value.
- setTxLimit: Authorized person can set transaction limit.
- clearStuckBalance: Authorized person can clear stuck balance.
- clearStuckBalance_sender: Authorized person can clear stuck balance.
- set_sell_multiplier: Owner can set sell multiplier value.
- tradingStatus: Owner can set switch Trading status.
- cooldownEnabled: Owner can enable cooldown between trades.
- setIsDividendExempt: Authorized person can set distributor shares.
- enable_blacklist: Owner can enable blacklist status.
- manage_blacklist: Owner can manage blacklist.
- setIsFeeExempt: Authorized can set fee exempt address and status.
- setIsTxLimitExempt: Authorized can set transaction limit exempt.
- setIsTimelockExempt: Authorized can set time lock exempt.
- setFees: Authorized can set fees like: liquidity Fee, reflection Fee, marketing Fee, ecosystem fee, burn Fee, fee Denominator.
- setFeeReceivers: Authorized can set receivers fees like: autoLiquidity Receiver, marketingFee Receiver, ecosystem fee Receiver, burn Fee Receiver.
- setSwapBackSettings: Authorized can set swap back settings amount.
- setTargetLiquidity: Authorized can set target liquidity value.
- setDistributionCriteria: Authorized can set distribution criteria.
- setDistributorSettings: Authorized can set distribution settings.
- multiTransfer: Owner can set muti transfer token.
- multiTransfer_fixed: Owner can set muti transfer fixed token.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We have not observed any major issues. So, **the smart contract is good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

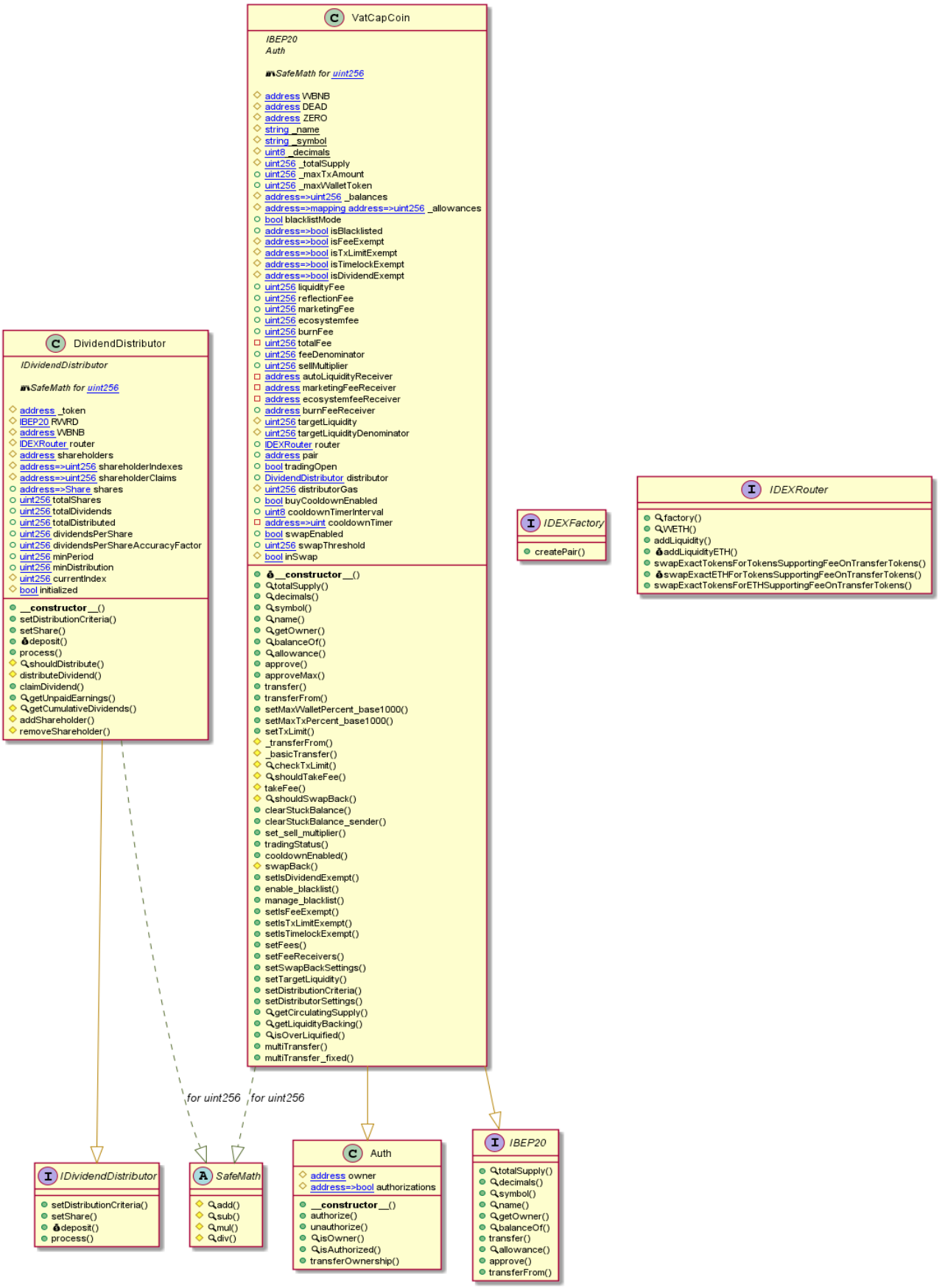
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - VatCap Coin



Slither Results Log

Slither log >> VatCapCoin.sol

```
INFO:Detectors:
DividendDistributor.setDistributionCriteria(uint256,uint256) (VatCapCoin.sol#209-212) should emit an event for:
        - minPeriod = _minPeriod (VatCapCoin.sol#210)
        - minDistribution = _minDistribution (VatCapCoin.sol#211)
VatCapCoin.setMaxTxPercent_base1000(uint256) (VatCapCoin.sol#447-449) should emit an event for:
        - _maxTxAmount = (_totalSupply * maxTXPercentage_base1000) / 1000 (VatCapCoin.sol#448)
VatCapCoin.setTxLimit(uint256) (VatCapCoin.sol#451-453) should emit an event for:
        - _maxTxAmount = amount (VatCapCoin.sol#452)
VatCapCoin.set_sell_multiplier(uint256) (VatCapCoin.sol#557-559) should emit an event for:
        - sellMultiplier = Multiplier (VatCapCoin.sol#558)
VatCapCoin.setFees(uint256,uint256,uint256,uint256,uint256,uint256) (VatCapCoin.sol#654-663) should emit an event for:
        - liquidityFee = _liquidityFee (VatCapCoin.sol#655)
        - reflectionFee = _reflectionFee (VatCapCoin.sol#656)
        - marketingFee = _marketingFee (VatCapCoin.sol#657)
        - ecosystemfee = _ecosystemfee (VatCapCoin.sol#658)
        - burnFee = _burnFee (VatCapCoin.sol#659)
        - totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee).add(_ecosystemfee).add(_burnFee) (VatCapCoin.sol#660)

        - feeDenominator = _feeDenominator (VatCapCoin.sol#661)
VatCapCoin.setSwapBackSettings(bool,uint256) (VatCapCoin.sol#672-675) should emit an event for:
        - swapThreshold = _amount (VatCapCoin.sol#674)
VatCapCoin.setTargetLiquidity(uint256,uint256) (VatCapCoin.sol#677-680) should emit an event for:
        - targetLiquidity = _target (VatCapCoin.sol#678)
        - targetLiquidityDenominator = _denominator (VatCapCoin.sol#679)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Auth.transferOwnership(address).adr (VatCapCoin.sol#91) lacks a zero-check on :
                - owner = adr (VatCapCoin.sol#92)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Auth.transferOwnership(address).adr (VatCapCoin.sol#91) lacks a zero-check on :
                - owner = adr (VatCapCoin.sol#92)
VatCapCoin.setFeeReceivers(address,address,address,address)._autoLiquidityReceiver (VatCapCoin.sol#665) lacks a zero-check on :
                - autoLiquidityReceiver = _autoLiquidityReceiver (VatCapCoin.sol#666)
VatCapCoin.setFeeReceivers(address,address,address,address)._marketingFeeReceiver (VatCapCoin.sol#665) lacks a zero-check on :
                - marketingFeeReceiver = _marketingFeeReceiver (VatCapCoin.sol#667)
VatCapCoin.setFeeReceivers(address,address,address,address)._ecosystemfeeReceiver (VatCapCoin.sol#665) lacks a zero-check on :
                - ecosystemfeeReceiver = _ecosystemfeeReceiver (VatCapCoin.sol#668)
VatCapCoin.setFeeReceivers(address,address,address,address)._burnFeeReceiver (VatCapCoin.sol#665) lacks a zero-check on :
                - burnFeeReceiver = _burnFeeReceiver (VatCapCoin.sol#669)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
VatCapCoin.multiTransfer(address,address[],uint256[]) (VatCapCoin.sol#706-730) has external calls inside a loop: distributor.se
tShare(addresses[i_scope_0],_balances[addresses[i_scope_0]]) (VatCapCoin.sol#722)
VatCapCoin.multiTransfer_fixed(address,address[],uint256) (VatCapCoin.sol#732-751) has external calls inside a loop: distributo
r.setShare(addresses[i],_balances[addresses[i]]) (VatCapCoin.sol#743)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in VatCapCoin.constructor() (VatCapCoin.sol#387-412):
        External calls:
        - pair = IDEXFactory(router.factory()).createPair(WBNB,address(this)) (VatCapCoin.sol#389)
        State variables written after the call(s):
        - _balances[msg.sender] = _totalSupply (VatCapCoin.sol#410)
        - autoLiquidityReceiver = msg.sender (VatCapCoin.sol#405)
        - burnFeeReceiver = DEAD (VatCapCoin.sol#408)
        - distributor = new DividendDistributor(address(router)) (VatCapCoin.sol#392)
        - ecosystemfeeReceiver = 0x1BAFaDB36a4180B59e0d63Ac9e39E4eFAE85D34a (VatCapCoin.sol#407)
        - isDividendExempt[pair] = true (VatCapCoin.sol#401)
```

```
                - shareholders.push(shareholder) (VatCapCoin.sol#315)
        - removeShareholder(shareholder) (VatCapCoin.sol#222)
                - shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length - 1] (VatCapCoin.sol#319)
                - shareholders.pop() (VatCapCoin.sol#321)
        - totalShares = totalShares.sub(shares[shareholder].amount).add(amount) (VatCapCoin.sol#225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in VatCapCoin._transferFrom(address,address,uint256) (VatCapCoin.sol#456-504):
        External calls:
        - swapBack() (VatCapCoin.sol#483)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp)
(VatCapCoin.sol#583-589)
                - distributor.deposit{value: amountBNBReflection}() (VatCapCoin.sol#600)
                - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (VatCapCoin.sol#601
)
                - (tmpSuccess,None) = address(ecosystemfeeReceiver).call{gas: 30000,value: amountBNBDev}() (VatCapCoin.sol#602)
                - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,blo
ck.timestamp) (VatCapCoin.sol#608-615)
        - distributor.setShare(sender,_balances[sender]) (VatCapCoin.sol#493)
        - distributor.setShare(recipient,_balances[recipient]) (VatCapCoin.sol#497)
        - distributor.process(distributorGas) (VatCapCoin.sol#500)
        External calls sending eth:
        - swapBack() (VatCapCoin.sol#483)
                - distributor.deposit{value: amountBNBReflection}() (VatCapCoin.sol#600)
                - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (VatCapCoin.sol#601
)
                - (tmpSuccess,None) = address(ecosystemfeeReceiver).call{gas: 30000,value: amountBNBDev}() (VatCapCoin.sol#602)
                - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,blo
ck.timestamp) (VatCapCoin.sol#608-615)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amountReceived) (VatCapCoin.sol#502)
Reentrancy in VatCapCoin._transferFrom(address,address,uint256) (VatCapCoin.sol#456-504):
        External calls:
        - swapBack() (VatCapCoin.sol#483)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp)
(VatCapCoin.sol#583-589)
                - distributor.deposit{value: amountBNBReflection}() (VatCapCoin.sol#600)
                - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (VatCapCoin.sol#601
```

```
INFO:Detectors:
DividendDistributor.shouldDistribute(address) (VatCapCoin.sol#276-279) uses timestamp for comparisons
        Dangerous comparisons:
        - shareholderClaims[shareholder] + minPeriod < block.timestamp && getUnpaidEarnings(shareholder) > minDistribution (Vat
CapCoin.sol#277-278)
VatCapCoin._transferFrom(address,address,uint256) (VatCapCoin.sol#456-504) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(cooldownTimer[recipient] < block.timestamp,Please wait for 1min between two buys) (VatCapCoin.so
l#476)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
DividendDistributor.process(uint256) (VatCapCoin.sol#250-274) has costly operations inside a loop:
        - currentIndex = 0 (VatCapCoin.sol#262)
DividendDistributor.process(uint256) (VatCapCoin.sol#250-274) has costly operations inside a loop:
        - currentIndex ++ (VatCapCoin.sol#271)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
VatCapCoin._maxTxAmount (VatCapCoin.sol#338) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.mul(1).div(100)
VatCapCoin._maxWalletToken (VatCapCoin.sol#339) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.mul(2).div(100)
VatCapCoin.totalFee (VatCapCoin.sol#357) is set pre-construction with a non-constant function or state variable:
        - marketingFee + reflectionFee + liquidityFee + ecosystemfee + burnFee
VatCapCoin.swapThreshold (VatCapCoin.sol#383) is set pre-construction with a non-constant function or state variable:
        - _totalSupply * 10 / 10000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.0 (VatCapCoin.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in VatCapCoin.swapBack() (VatCapCoin.sol#572-618):
        - (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (VatCapCoin.sol#601)
        - (tmpSuccess,None) = address(ecosystemfeeReceiver).call{gas: 30000,value: amountBNBDev}() (VatCapCoin.sol#602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Variable IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VatCapCoin.so
l#111) is too similar to IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesire
d (VatCapCoin.sol#112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
VatCapCoin.slitherConstructorVariables() (VatCapCoin.sol#325-755) uses literals with too many digits:
        - DEAD = 0x000000000000000000000000000000000000dEaD (VatCapCoin.sol#329)
VatCapCoin.slitherConstructorVariables() (VatCapCoin.sol#325-755) uses literals with too many digits:
        - ZERO = 0x0000000000000000000000000000000000000000 (VatCapCoin.sol#330)
VatCapCoin.slitherConstructorVariables() (VatCapCoin.sol#325-755) uses literals with too many digits:
        - distributorGas = 500000 (VatCapCoin.sol#376)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
DividendDistributor.WBNB (VatCapCoin.sol#171) should be constant
DividendDistributor.dividendsPerShareAccuracyFactor (VatCapCoin.sol#184) should be constant
VatCapCoin.DEAD (VatCapCoin.sol#329) should be constant
VatCapCoin.WBNB (VatCapCoin.sol#328) should be constant
VatCapCoin.ZERO (VatCapCoin.sol#330) should be constant
VatCapCoin._totalSupply (VatCapCoin.sol#336) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
authorize(address) should be declared external:
        - Auth.authorize(address) (VatCapCoin.sol#75-77)
unauthorize(address) should be declared external:
        - Auth.unauthorize(address) (VatCapCoin.sol#79-81)
transferOwnership(address) should be declared external:
        - Auth.transferOwnership(address) (VatCapCoin.sol#91-95)
approve(address,uint256) should be declared external:
        - VatCapCoin.approve(address,uint256) (VatCapCoin.sol#424-428)
tradingStatus(bool) should be declared external:
        - VatCapCoin.tradingStatus(bool) (VatCapCoin.sol#562-564)
cooldownEnabled(bool,uint8) should be declared external:
        - VatCapCoin.cooldownEnabled(bool,uint8) (VatCapCoin.sol#567-570)
enable_blacklist(bool) should be declared external:
        - VatCapCoin.enable_blacklist(bool) (VatCapCoin.sol#631-633)
```

```
enable_blacklist(bool) should be declared external:
        - VatCapCoin.enable_blacklist(bool) (VatCapCoin.sol#631-633)
manage_blacklist(address[],bool) should be declared external:
        - VatCapCoin.manage_blacklist(address[],bool) (VatCapCoin.sol#635-639)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VatCapCoin.sol analyzed (8 contracts with 75 detectors), 109 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

VatCapCoin.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 614:16:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 602:24:

Gas & Economy

Gas costs:

Gas requirement of function VatCapCoin.transferOwnership is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 91:4:

Gas costs:

Gas requirement of function VatCapCoin.multiTransfer is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 706:0:

Gas costs:

Gas requirement of function VatCapCoin.multiTransfer_fixed is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 732:0:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 636:8:

Email: audit@EtherAuthority.io

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 740:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 45:4:

Miscellaneous

Similar variable names:

VatCapCoin.swapBack() : Variables have very similar names "pair" and "path". Note: Modifiers are currently not considered by this static analysis.
Pos: 586:12:

No return:

VatCapCoin.approveMax(address): Defines a return type but never explicitly returns a value.
Pos: 430:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 708:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 709:4:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 662:27:

Solhint Linter

VatCapCoin.sol

```
VatCapCoin.sol:4:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
VatCapCoin.sol:62:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VatCapCoin.sol:106:5: Error: Function name must be in mixedCase
VatCapCoin.sol:159:1: Error: Contract has 17 states declarations but
allowed no more than 15
VatCapCoin.sol:162:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:170:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:170:12: Error: Variable name must be in mixedCase
VatCapCoin.sol:171:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:171:13: Error: Variable name must be in mixedCase
VatCapCoin.sol:172:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:174:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:175:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:176:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:189:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:191:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:202:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VatCapCoin.sol:241:13: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:277:61: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:288:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
VatCapCoin.sol:288:46: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:289:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
VatCapCoin.sol:290:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
VatCapCoin.sol:325:1: Error: Contract has 39 states declarations but
allowed no more than 15
VatCapCoin.sol:328:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:328:13: Error: Variable name must be in mixedCase
VatCapCoin.sol:329:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:329:13: Error: Variable name must be in mixedCase
VatCapCoin.sol:330:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:330:13: Error: Variable name must be in mixedCase
VatCapCoin.sol:332:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:332:21: Error: Constant name must be in capitalized
SNAKE_CASE
VatCapCoin.sol:333:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:333:21: Error: Constant name must be in capitalized
SNAKE_CASE
VatCapCoin.sol:334:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:334:20: Error: Constant name must be in capitalized
SNAKE_CASE
VatCapCoin.sol:336:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:341:5: Error: Explicitly mark visibility of state
```

```
VatCapCoin.sol:342:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:347:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:348:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:349:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:350:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:367:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:368:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:376:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:384:5: Error: Explicitly mark visibility of state
VatCapCoin.sol:387:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VatCapCoin.sol:414:32: Error: Code contains empty blocks
VatCapCoin.sol:430:66: Error: Code contains empty blocks
VatCapCoin.sol:444:5: Error: Function name must be in mixedCase
VatCapCoin.sol:444:43: Error: Variable name must be in mixedCase
VatCapCoin.sol:447:5: Error: Function name must be in mixedCase
VatCapCoin.sol:447:39: Error: Variable name must be in mixedCase
VatCapCoin.sol:476:48: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:477:40: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:493:65: Error: Code contains empty blocks
VatCapCoin.sol:493:74: Error: Code contains empty blocks
VatCapCoin.sol:497:71: Error: Code contains empty blocks
VatCapCoin.sol:497:80: Error: Code contains empty blocks
VatCapCoin.sol:500:49: Error: Code contains empty blocks
VatCapCoin.sol:500:58: Error: Code contains empty blocks
VatCapCoin.sol:552:5: Error: Function name must be in mixedCase
VatCapCoin.sol:557:5: Error: Function name must be in mixedCase
VatCapCoin.sol:557:34: Error: Variable name must be in mixedCase
VatCapCoin.sol:588:13: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:600:63: Error: Code contains empty blocks
VatCapCoin.sol:600:72: Error: Code contains empty blocks
VatCapCoin.sol:601:30: Error: Avoid using low level calls.
VatCapCoin.sol:602:25: Error: Avoid using low level calls.
VatCapCoin.sol:614:17: Error: Avoid to make time-based decisions in
your business logic
VatCapCoin.sol:631:5: Error: Function name must be in mixedCase
VatCapCoin.sol:635:5: Error: Function name must be in mixedCase
VatCapCoin.sol:711:5: Error: Variable name must be in mixedCase
VatCapCoin.sol:722:77: Error: Code contains empty blocks
VatCapCoin.sol:722:86: Error: Code contains empty blocks
VatCapCoin.sol:728:57: Error: Code contains empty blocks
VatCapCoin.sol:728:66: Error: Code contains empty blocks
VatCapCoin.sol:732:1: Error: Function name must be in mixedCase
VatCapCoin.sol:736:5: Error: Variable name must be in mixedCase
VatCapCoin.sol:743:77: Error: Code contains empty blocks
VatCapCoin.sol:743:86: Error: Code contains empty blocks
VatCapCoin.sol:749:57: Error: Code contains empty blocks
VatCapCoin.sol:749:66: Error: Code contains empty blocks
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io