Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	Vetter Token
Website:	https://vetter.ai
Platform:	Binance Smart Chain
Language:	Solidity
Date:	October 17th, 2021

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.

Introduction

EtherAuthority was contracted by the Vetter team to perform the Security audit of the Vetter Token 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 October 17th, 2021.

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

Vetter Token (VETTER) reinvents the crypto-research industry allowing investors to sort for 2x to 100x gems and get paid to scout, vet and vote on projects in the crowdsourcing meets A.I. interface. With a proven use case, the dApp and intuitive oracle highlights projects worth researching by ranking the poster's performance. Add in compensation, points, tipping and a bad-ass interface, and you have the best crypto research tool at your fingertips.

Audit scope

Name	**Code Review and Security Analysis Report for Vetter Token Smart Contract**
Platform	**BSC / Solidity**
File	Vetter.sol
File MD5 Hash	449D613C8DC0C8618F0B87C4D8AB64AF
Audit Date	October 17th, 2021
Revised Code MD5	FD4B5EFE407187E51212552CDBBFC8E3
Revision Date	October 19th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: Vetter Token ● Symbol: VETTER ● Decimals: 9 ● Total Tokens: 4 Billion Tokens	**YES, This is valid.**
● Initial Founder: 200 Million Tokens ● Initial Stable: 200 Million Tokens ● Initial Private Sale: 2.3 Billion Tokens ● Initial Pre Sale: 645 Million Tokens ● Initial Liquidity: 645 Million Tokens ● Number Of Tiers: 4 ● Initial Tier 1: 10,000 ● Initial Tier 2: 100,000 ● Initial Tier 3: 250,000 ● Initial Tier 4: 500,000 ● Stable Release: 90 days ● Stable Percent: 10	**YES, This is valid.**
● Marketing Buy Tax: 5% ● Marketing Sell Tax: 5% ● Market Tax Cap: 5% Max ● Participatory Buy Tax: 2% ● Participatory Sell Tax: 5% ● Participatory Cap: 5% max ● Collect Vetter Pool: 100 ● Vetter Buy Tax: 2% ● Vetter Sell Tax: 5% ● Vetter Cap: 5% max ● Collect Liquidity: 100 ● Liquidity Buy Tax: 2%	**YES, This is valid.** **Owner authorized wallet (onlyArchitect) can set some percentages value and we suggest to handle the private key of that wallet securely.**

● Liquidity Sell Tax: 2% ● Liquidity Cap: 2% max ● Collect Reflections: 100 ● Reflection Buy Tax: 1% ● Reflection Sell Tax: 1% ● Reflection Cap: 2% max ● Reflection Min: 1% min	

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.

Insecure
Poor secured
Secure
Well-secured
You are here

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, 1 high, 0 medium and 1 low and some very low level issues. These issues are fixed in the revised smart contract.

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	Moderated
	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	Passed
	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	Passed
	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

Code Quality

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

The libraries in Vetter Token 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 Vetter Token.

The Vetter Token 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 Vetter Token smart contracts code in the form of a file.The hashes of that code are 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.

Another source of information was its official website https://vetter.ai which provided rich information about the project architecture and tokenomics.

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	lockTheSwap	modifier	Passed	No Issue
3	totalSupply	external	Passed	No Issue
4	balanceOf	read	Passed	No Issue
5	_preTaxAmount	read	Passed	No Issue
6	transfer	external	Passed	No Issue
7	allowance	read	Passed	No Issue
8	approve	write	Passed	No Issue
9	transferFrom	external	Passed	No Issue
10	name	external	Passed	No Issue
11	symbol	external	Passed	No Issue
12	decimals	external	Passed	No Issue
13	increaseAllowance	external	Passed	No Issue
14	decreaseAllowance	external	Passed	No Issue
15	addLiquidity	write	Passed	No Issue
16	_approve	write	Passed	No Issue
17	_transfer	write	Passed	No Issue
18	swapMarketingTokens	write	Passed	No Issue
19	swapParticipatoryPoolTokens	external	access only Architect	No Issue
20	swapParticipatoryTokens	write	Passed	No Issue
21	swapVetterPoolTokens	external	Passed	No Issue
22	swapVetterTokens	write	Passed	No Issue
23	swapLiquidityPoolTokens	external	Passed	No Issue
24	swapLiquidityTokens	write	Passed	No Issue
25	swapReflectionPoolTokens	external	access only Architect	No Issue
26	swapReflectionTokens	write	Passed	No Issue
27	swapTokensForBNB	write	Passed	No Issue
28	getBuyTaxes	read	Passed	No Issue
29	getSellTaxes	read	Passed	No Issue
30	takeTaxes	external	Passed	No Issue
31	setAllTaxes	write	Passed	No Issue
32	setInitialLaunchTaxes	write	Passed	No Issue
33	getTotalBuyTax	read	Passed	No Issue
34	getTotalSellTax	read	Passed	No Issue
35	setAllCollectionRates	write	access only Architect	No Issue
36	setAllReserves	write	access only Architect	No Issue
37	rewardParticipatory	external	Passed	No Issue
38	rewardVetter	external	Passed	No Issue

39	getPoolSize	read	Passed	No Issue
40	distributePresaleTokens	external	access only Architect	No Issue
41	isExcludedFromFee	external	Passed	No Issue
42	excludeFromFee	external	Passed	No Issue
43	includeInFee	external	Passed	No Issue
44	getSellBnBAmount	read	Passed	No Issue
45	setParticipatoryAddress	external	Passed	No Issue
46	setVetterAddress	external	Passed	No Issue
47	setLiquidityAddress	external	Passed	No Issue
48	setReflectionAddress	external	Passed	No Issue
49	transferBNBToAddress	write	Passed	No Issue
50	getPairAddress	external	access only Owner	No Issue
51	changeRouterVersion	external	Passed	No Issue
52	receive	external	Passed	No Issue
53	transferForeignToken	external	Passed	No Issue
54	_setRankContract	external	Passed	No Issue
55	isArchitect	read	Passed	No Issue
56	addArchitect	external	access only Architect	No Issue
57	removeArchitect	external	Passed	No Issue
58	adjustArchitect	external	Passed	No Issue
59	distributeFounderTokens	external	Passed	No Issue
60	onlyArchitect	modifier	Passed	No Issue
61	isAdmin	read	Passed	No Issue
62	isFullAdmin	read	Passed	No Issue
63	addAdmin	external	Passed	No Issue
64	removeAdmin	external	Passed	No Issue
65	onlyAdmin	modifier	Passed	No Issue
66	isVetter	read	Passed	No Issue
67	isTrialVetter	read	Passed	No Issue
68	isFullVetter	read	Passed	No Issue
69	addOrAdjustVetter	write	access only Admin	No Issue
70	removeVetter	write	Passed	No Issue
71	addToStableList	external	access only Admin	No Issue
72	getStableListEntry	external	Passed	No Issue
73	adjustStableListEnrty	external	access only Admin	No Issue
74	getStableShares	read	Passed	No Issue
75	getPrivateShares	read	Passed	No Issue
76	distributeStableTokens	external	Passed	No Issue
77	airdropPrivateTokens	external	Passed	No Issue
78	_getMaxTier	read	Passed	No Issue
79	_getTierLevel	read	Passed	No Issue
80	_getAllTiers	read	Passed	No Issue

81	_setTierLevel	external	access only Architect	No Issue
82	_addNewTier	external	access only Architect	No Issue
83	_getWalletTier	read	Passed	No Issue
84	_getUserInfo	external	Passed	No Issue
85	transferTipAmount	external	access only RankContract	No Issue
86	addToReflections	external	access only RankContract	No Issue
87	distributeReflectionTokens	external	Passed	No Issue
88	distributeVetterPoolTokens	external	Passed	No Issue
89	owner	read	Passed	No Issue
90	isOwner	read	Passed	No Issue
91	onlyOwner	modifier	Passed	No Issue
92	transferOwnership	write	access only Owner	No Issue
93	getTime	read	Passed	No Issue
94	pullLiquidityPoolTokens	external	Passed	No Issue
95	addLiquidityPoolTokensToLP	external	Passed	No Issue
96	setPresaleAddress	external	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.

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) Infinite loops, Out of Gas issue:

As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that data directly, instead of looping through all the elements to find an element.

Resolution: Adjust logic to replace loops with mapping or other code structure.

- swapMarketingTokens() - _founderList.length
- removeArchitect() - _founderList.length
- adjustArchitect() - _founderList.length
- distributeFounderTokens() - _founderList.length
- getStableShares() - _stableList.length
- getPrivateShares() - _stableList.length
- distributeStableTokens() - _stableList.length
- airdropPrivateTokens() - _stableList.length
- _getAllTiers() - maxTier = _getMaxTier()
- distributeReflectionTokens() - holders = _reflectionsTo.length
- distributeVetterPoolTokens() - numVetters = _vetterList.length

Update: This issue is resolved in the revised contract code

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:

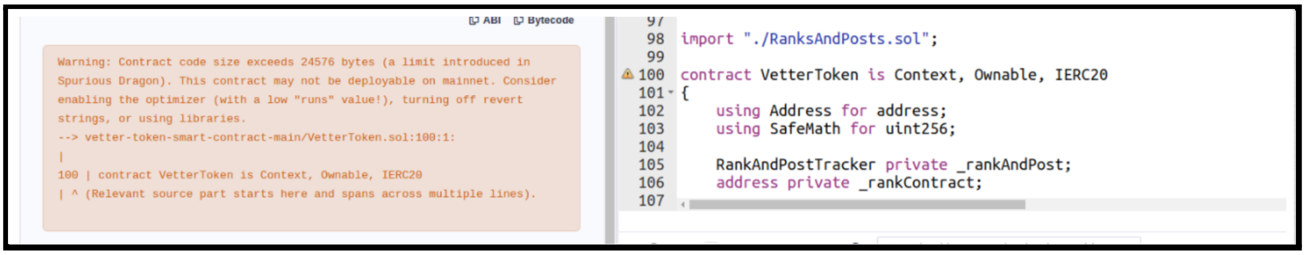
There are several places in the smart contracts, where a critical function call event log was not added. We suggest to add appropriate event log in the following functions:

- swapMarketingTokens(),
- swapVetterPoolTokens(),
- pullLiquidityPoolTokens(),
- swapLiquidityPoolTokens(),
- addLiquidityPoolTokensToLP(),
- takeTaxes(),
- setAllTaxes(),
- setInitialLaunchTaxes(),
- rewardParticipatory(),
- rewardVetter(),
- excludeFromFee(),
- includeInFee(),
- setParticipatoryAddress(),
- setVetterAddress(),
- setLiquidityAddress(),
- setReflectionAddress(),
- setPresaleAddress(),
- changeRouterVersion(),
- transferForeignToken(),
- _setRankContract(),
- removeArchitect(),
- adjustArchitect(),
- addAdmin(),
- removeAdmin(),
- removeVetter(),
- distributeStableTokens()

Update: This issue is resolved in the revised contract code

Very Low / Informational / Best practices:

(1) VetterToken contract code size limit:



```
Warning: Contract code size exceeds 24576 bytes (a limit introduced in
Spurious Dragon). This contract may not be deployable on mainnet. Consider
enabling the optimizer (with a low "runs" value!), turning off revert
strings, or using libraries.
--> vetter-token-smart-contract-main/VetterToken.sol:100:1:
|
100 | contract VetterToken is Context, Ownable, IERC20
| ^ (Relevant source part starts here and spans across multiple lines).
```

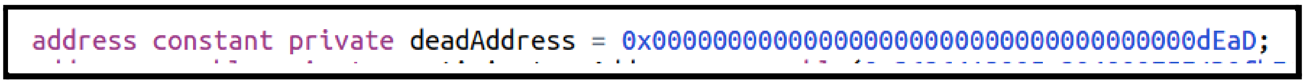
```
import "./RanksAndPosts.sol";

contract VetterToken is Context, Ownable, IERC20
{
    using Address for address;
    using SafeMath for uint256;

    RankAndPostTracker private _rankAndPost;
    address private _rankContract;
```

Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off revert strings, or using libraries.

Resolution: We suggest removing unnecessary code blocks and optimizing the code.

Update: This issue is resolved in the revised contract code

(2) Unused variables:



```
address constant private deadAddress = 0x000000000000000000000000000000000000dEaD;
```

There is a deadAddress variable defined but not used anywhere.

Resolution: Remove unwanted variables from the contract code.

Update: This issue is resolved in the revised contract code

(3) Hard coded Values:

Too many hard coded values.

Resolution: Must be double checked just before deploying to production.

Update: This issue is resolved in the revised contract code

Centralization

These smart contracts have 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:

- swapParticipatoryPoolTokens: The Architect owner can swap participatory pool tokens.
- swapVetterPoolTokens: The Architect owner can swap vetter pool tokens.
- swapLiquidityPoolTokens: The Architect owner can swap liquidity pool tokens.
- swapReflectionPoolTokens: The Architect owner can swap reflection pool tokens.
- takeTaxes: The Architect owner can take taxes.
- setAllTaxes: The Architect owner can set all taxes.
- setInitialLaunchTaxes: The Architect owner can set initial launch taxes.
- setAllCollectionRates: The Architect owner can set all collection rates.
- setAllReserves: The Architect owner can set all reserves.
- rewardParticipatory: The RankContract owner can reward participants.
- rewardVetter: The RankContract owner can reward vetter.
- distributePresaleTokens: The Architect owner can distribute presale tokens.
- excludeFromFee: The Architect owner can exclude from fee.
- includeInFee: The Architect owner can include from fee.
- setParticipatoryAddress: The Architect owner can set a participatory address.
- setVetterAddress: The Architect owner can set a vetter address.
- setLiquidityAddress: The Architect owner can set a liquidity address.
- setReflectionAddress: The Architect owner can set a reflection address.
- changeRouterVersion: The Owner can change the router version.
- transferForeignToken: The Architect owner can transfer foreign tokens.
- _setRankContract: The Owner can set rank contract address.
- addArchitect: The Architect owner can add the architect address.
- removeArchitect: The Architect owner can remove the architect address.
- adjustArchitect: The Architect owner can add a new wallet address to the list.
- distributeFounderTokens: The Architect owner can distribute founder tokens.
- addAdmin: The Architect owner can add a new wallet address to the list.
- removeAdmin: The Architect owner can remove wallet address to list.

- addOrAdjustVetter: The Admin owner can add a new wallet address to the list.
- removeVetter: The Admin owner can remove a wallet address from list.
- addToStableList: The Admin owner can add a stable address to the list.
- adjustStableListEnrty:The Admin owner can adjust a stable list entry.
- distributeStableTokens: The Architect owner can distribute stable tokens.
- airdropPrivateTokens: The Architect owner can airdrop private tokens.
- _setTierLevel: The Architect owner can set tier level.
- _addNewTier: The Architect owner can add a new tier level.
- transferTipAmount: The RankContract can transfer tip amount.
- addToReflections: The RankContract can add to reflection.
- distributeReflectionTokens: The Architect owner can distribute reflection tokens.
- distributeVetterPoolTokens: The Architect owner can distribute vetter pool tokens.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those issues are fixed/acknowledged in the revised contract code. So, **it's 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).

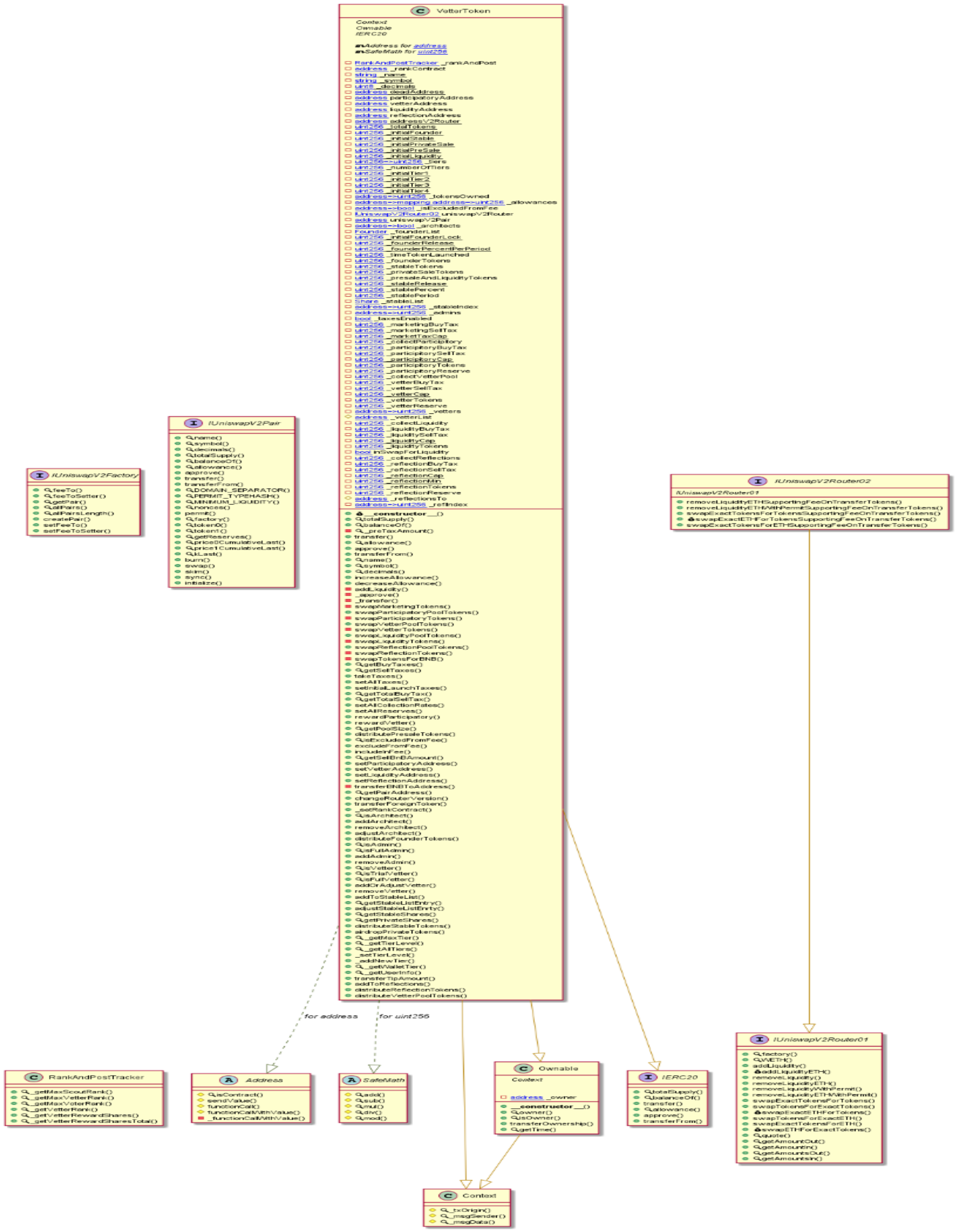
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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.

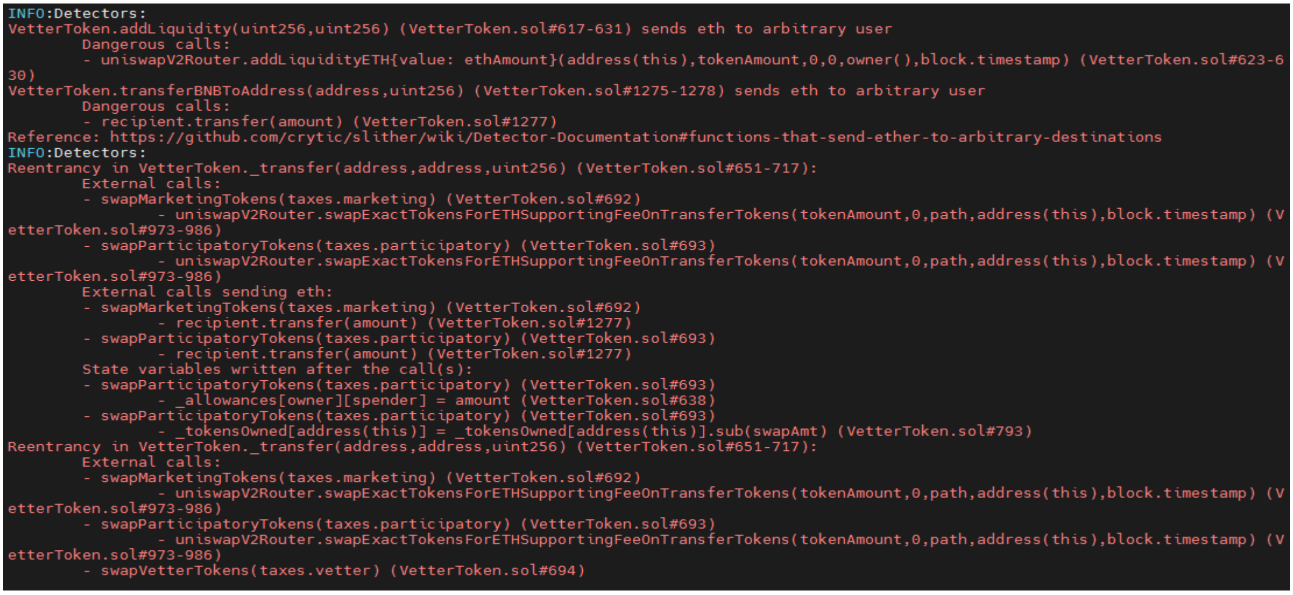
Appendix

Code Flow Diagram - Vetter Token

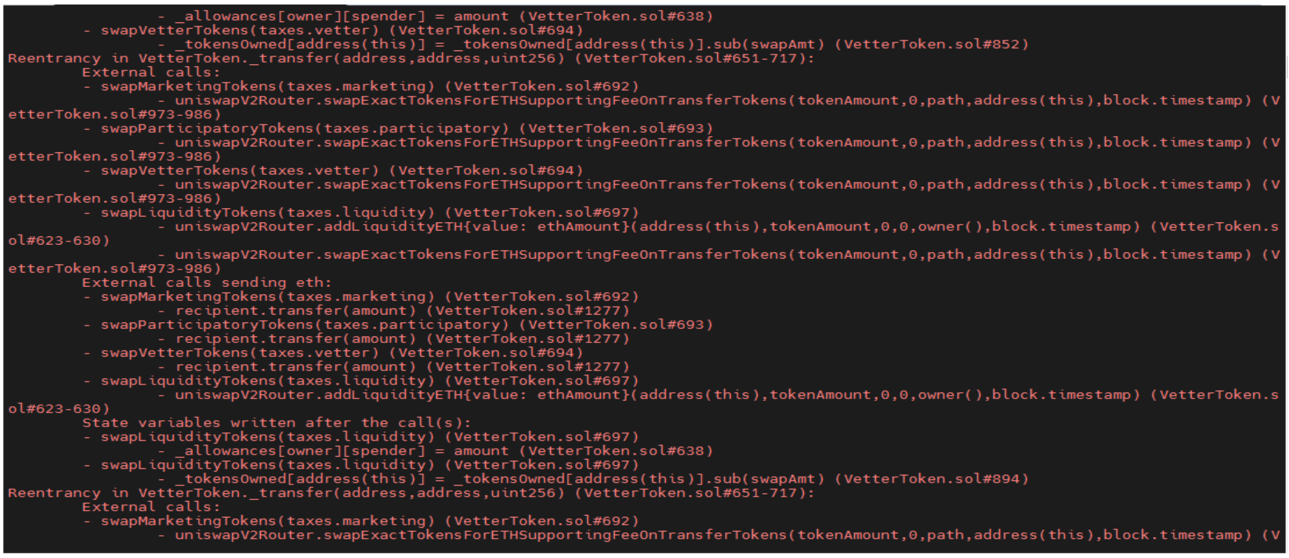


Slither Results Log

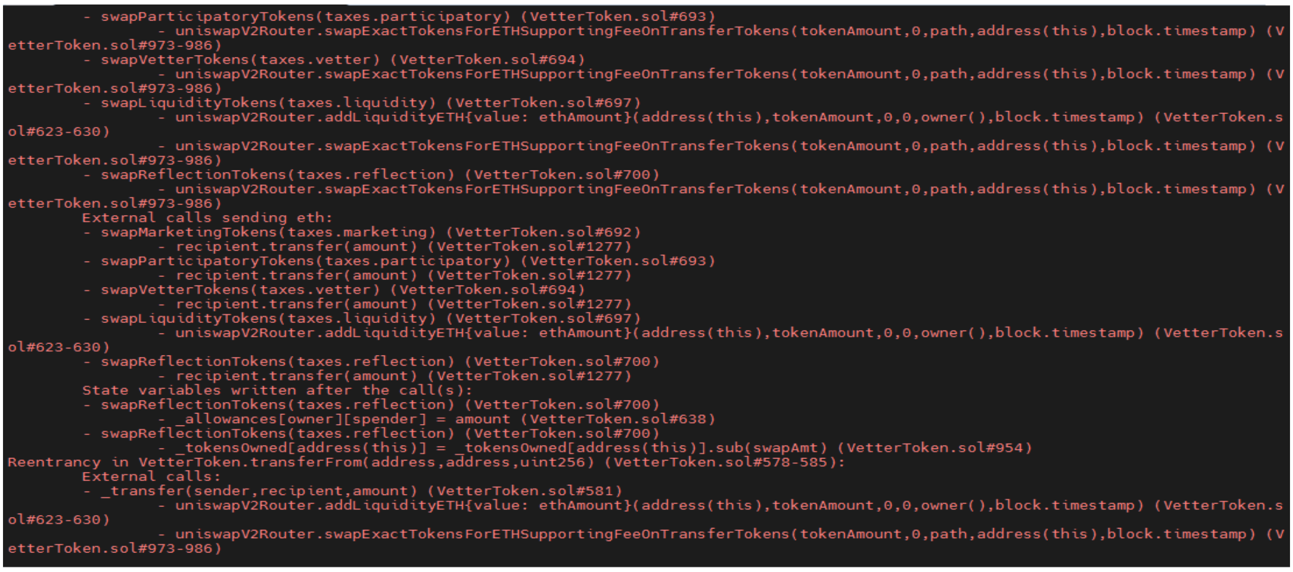
Slither log >> Vetter.sol



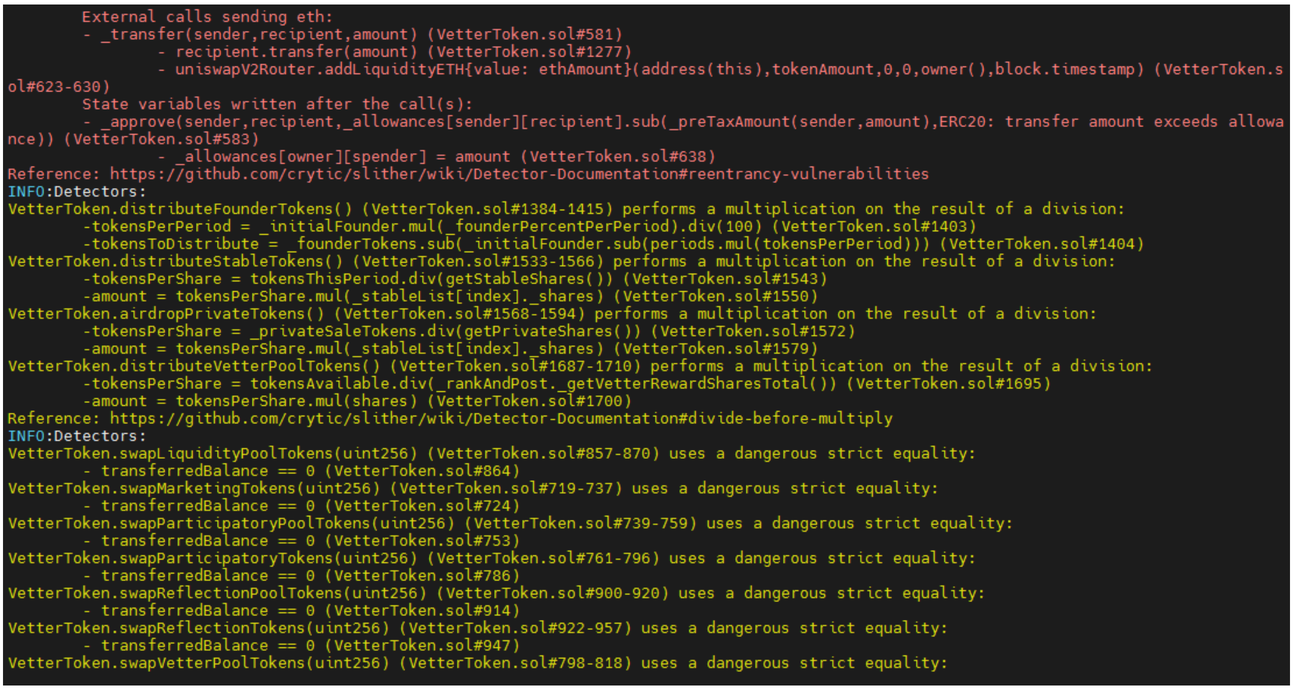
```
INFO:Detectors:
VetterToken.addLiquidity(uint256,uint256) (VetterToken.sol#617-631) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.sol#623-6
30)
VetterToken.transferBNBToAddress(address,uint256) (VetterToken.sol#1275-1278) sends eth to arbitrary user
        Dangerous calls:
        - recipient.transfer(amount) (VetterToken.sol#1277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        External calls sending eth:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        State variables written after the call(s):
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#793)
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
```



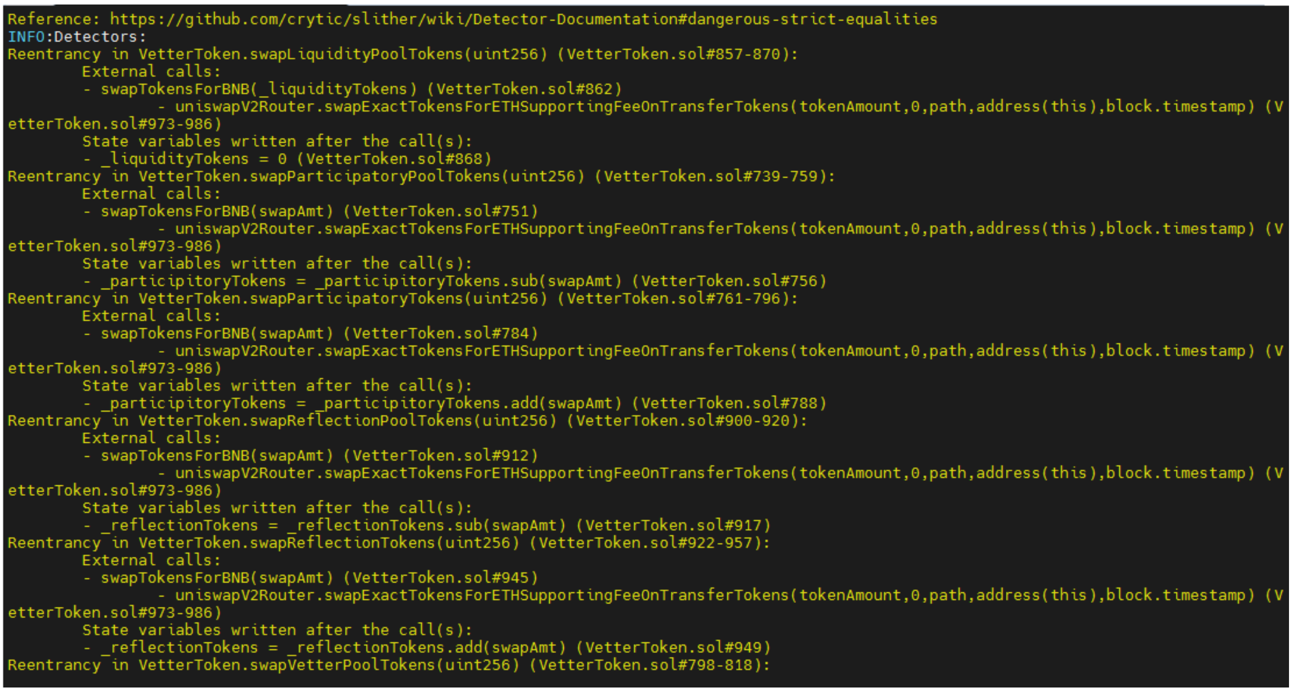
```
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#852)
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        External calls sending eth:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
        State variables written after the call(s):
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#894)
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
```



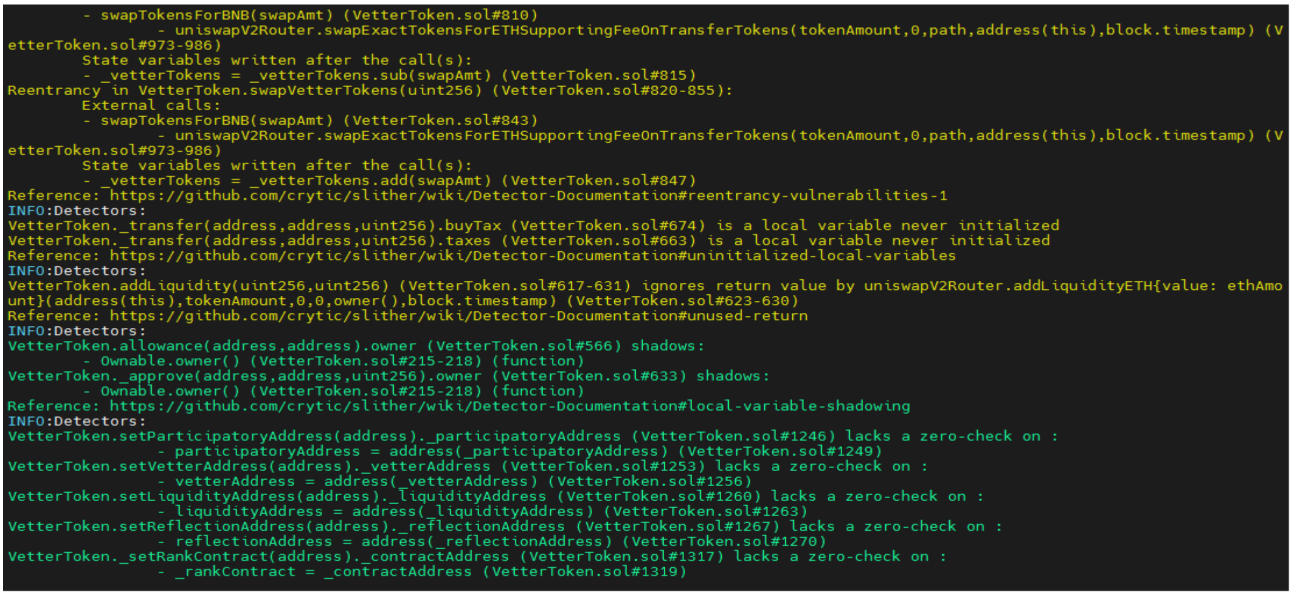
```
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapReflectionTokens(taxes.reflection) (VetterToken.sol#700)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        External calls sending eth:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
        - swapReflectionTokens(taxes.reflection) (VetterToken.sol#700)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        State variables written after the call(s):
        - swapReflectionTokens(taxes.reflection) (VetterToken.sol#700)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        - swapReflectionTokens(taxes.reflection) (VetterToken.sol#700)
                - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#954)
Reentrancy in VetterToken.transferFrom(address,address,uint256) (VetterToken.sol#578-585):
        External calls:
        - _transfer(sender,recipient,amount) (VetterToken.sol#581)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
```



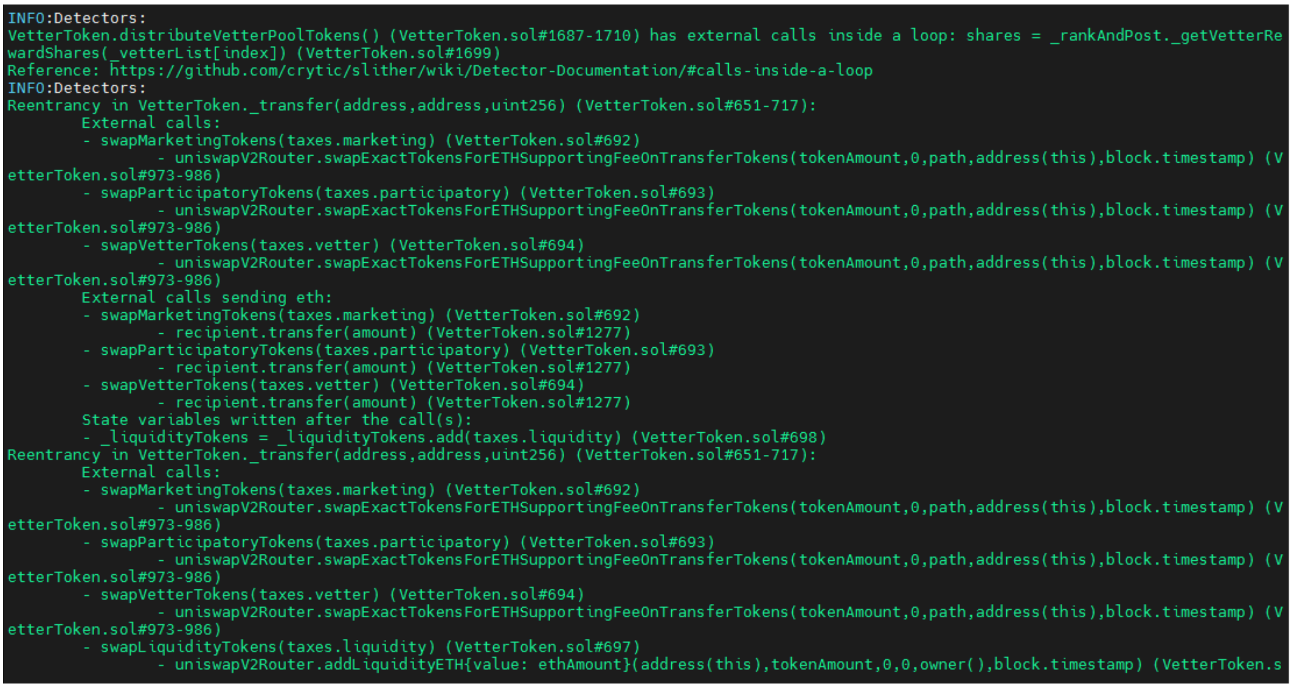
```
        External calls sending eth:
        - _transfer(sender,recipient,amount) (VetterToken.sol#581)
                - recipient.transfer(amount) (VetterToken.sol#1277)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
        State variables written after the call(s):
        - _approve(sender,recipient,_allowances[sender][recipient].sub(_preTaxAmount(sender,amount),ERC20: transfer amount exceeds allowa
nce)) (VetterToken.sol#583)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
VetterToken.distributeFounderTokens() (VetterToken.sol#1384-1415) performs a multiplication on the result of a division:
        -tokensPerPeriod = _initialFounder.mul(_founderPercentPerPeriod).div(100) (VetterToken.sol#1403)
        -tokensToDistribute = _founderTokens.sub(_initialFounder.sub(periods.mul(tokensPerPeriod))) (VetterToken.sol#1404)
VetterToken.distributeStableTokens() (VetterToken.sol#1533-1566) performs a multiplication on the result of a division:
        -tokensPerShare = tokensThisPeriod.div(getStableShares()) (VetterToken.sol#1543)
        -amount = tokensPerShare.mul(_stableList[index]._shares) (VetterToken.sol#1550)
VetterToken.airdropPrivateTokens() (VetterToken.sol#1568-1594) performs a multiplication on the result of a division:
        -tokensPerShare = _privateSaleTokens.div(getPrivateShares()) (VetterToken.sol#1572)
        -amount = tokensPerShare.mul(_stableList[index]._shares) (VetterToken.sol#1579)
VetterToken.distributeVetterPoolTokens() (VetterToken.sol#1687-1710) performs a multiplication on the result of a division:
        -tokensPerShare = tokensAvailable.div(_rankAndPost._getVetterRewardSharesTotal()) (VetterToken.sol#1695)
        -amount = tokensPerShare.mul(shares) (VetterToken.sol#1700)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
VetterToken.swapLiquidityPoolTokens(uint256) (VetterToken.sol#857-870) uses a dangerous strict equality:
        - transferredBalance == 0 (VetterToken.sol#864)
VetterToken.swapMarketingTokens(uint256) (VetterToken.sol#719-737) uses a dangerous strict equality:
        - transferredBalance == 0 (VetterToken.sol#724)
VetterToken.swapParticipatoryPoolTokens(uint256) (VetterToken.sol#739-759) uses a dangerous strict equality:
        - transferredBalance == 0 (VetterToken.sol#753)
VetterToken.swapParticipatoryTokens(uint256) (VetterToken.sol#761-796) uses a dangerous strict equality:
        - transferredBalance == 0 (VetterToken.sol#786)
VetterToken.swapReflectionPoolTokens(uint256) (VetterToken.sol#900-920) uses a dangerous strict equality:
        - transferredBalance == 0 (VetterToken.sol#914)
VetterToken.swapReflectionTokens(uint256) (VetterToken.sol#922-957) uses a dangerous strict equality:
        - transferredBalance == 0 (VetterToken.sol#947)
VetterToken.swapVetterPoolTokens(uint256) (VetterToken.sol#798-818) uses a dangerous strict equality:
```



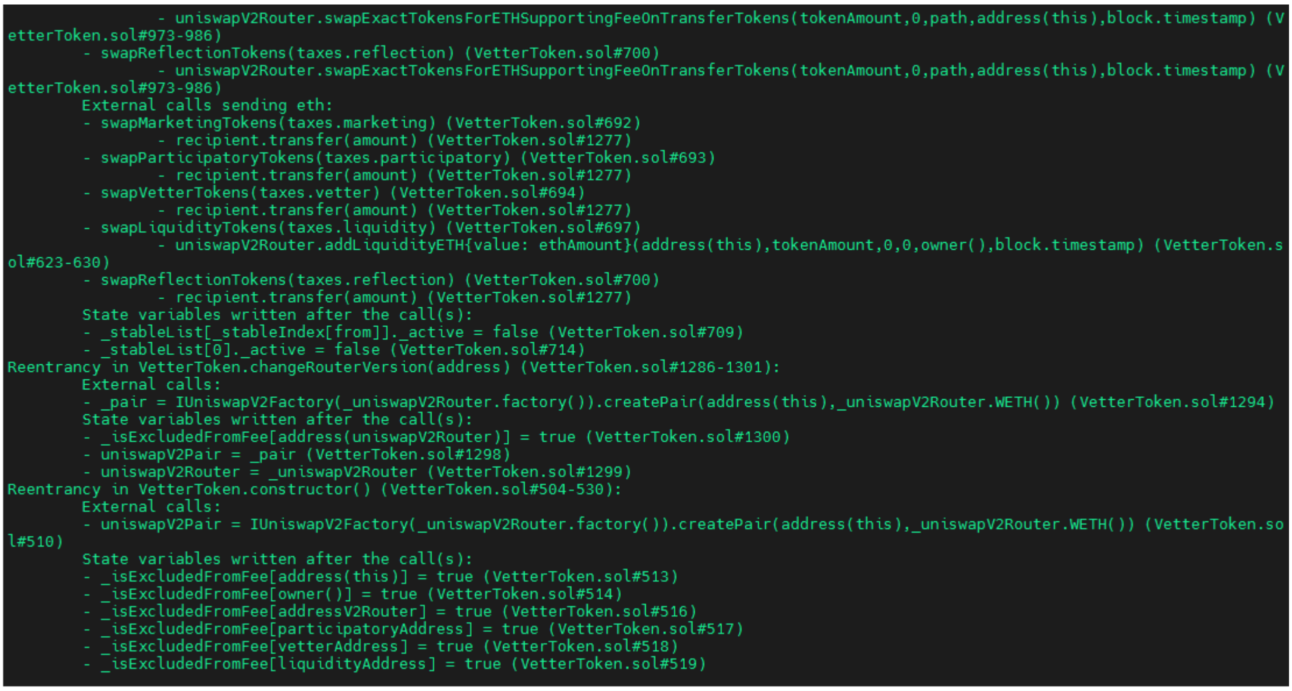
```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in VetterToken.swapLiquidityPoolTokens(uint256) (VetterToken.sol#857-870):
        External calls:
        - swapTokensForBNB(_liquidityTokens) (VetterToken.sol#862)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _liquidityTokens = 0 (VetterToken.sol#868)
Reentrancy in VetterToken.swapParticipatoryPoolTokens(uint256) (VetterToken.sol#739-759):
        External calls:
        - swapTokensForBNB(swapAmt) (VetterToken.sol#751)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _participitoryTokens = _participitoryTokens.sub(swapAmt) (VetterToken.sol#756)
Reentrancy in VetterToken.swapParticipatoryTokens(uint256) (VetterToken.sol#761-796):
        External calls:
        - swapTokensForBNB(swapAmt) (VetterToken.sol#784)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _participitoryTokens = _participitoryTokens.add(swapAmt) (VetterToken.sol#788)
Reentrancy in VetterToken.swapReflectionPoolTokens(uint256) (VetterToken.sol#900-920):
        External calls:
        - swapTokensForBNB(swapAmt) (VetterToken.sol#912)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _reflectionTokens = _reflectionTokens.sub(swapAmt) (VetterToken.sol#917)
Reentrancy in VetterToken.swapReflectionTokens(uint256) (VetterToken.sol#922-957):
        External calls:
        - swapTokensForBNB(swapAmt) (VetterToken.sol#945)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _reflectionTokens = _reflectionTokens.add(swapAmt) (VetterToken.sol#949)
Reentrancy in VetterToken.swapVetterPoolTokens(uint256) (VetterToken.sol#798-818):
```



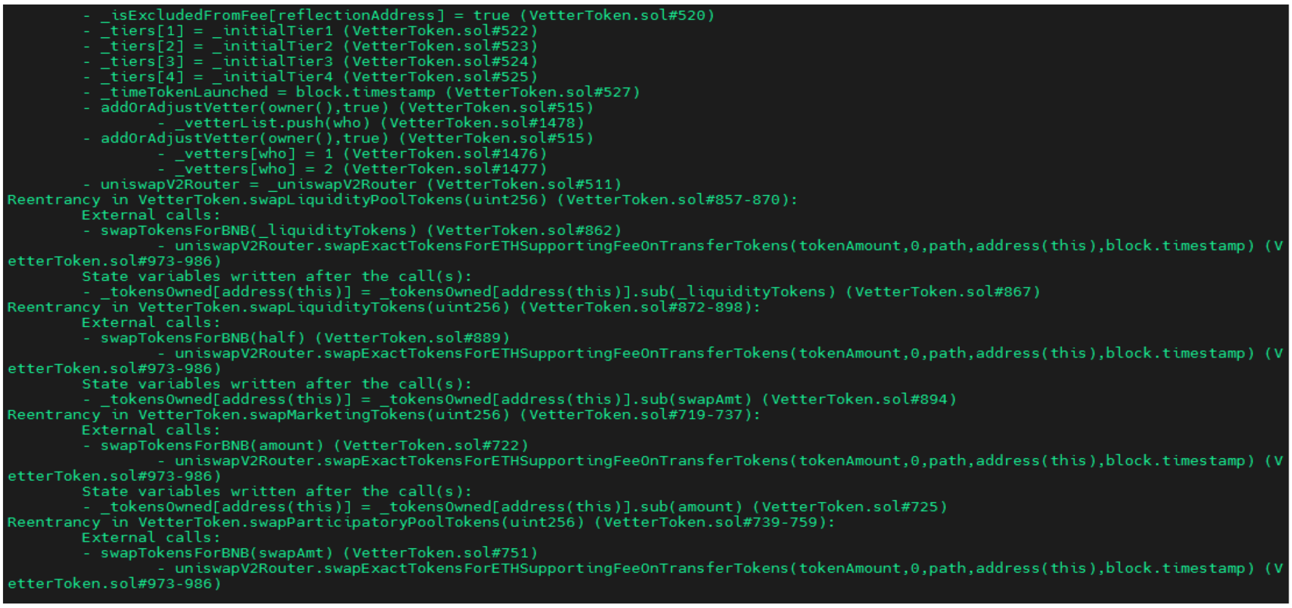
```
        - swapTokensForBNB(swapAmt) (VetterToken.sol#810)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _vetterTokens = _vetterTokens.sub(swapAmt) (VetterToken.sol#815)
Reentrancy in VetterToken.swapVetterTokens(uint256) (VetterToken.sol#820-855):
        External calls:
        - swapTokensForBNB(swapAmt) (VetterToken.sol#843)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _vetterTokens = _vetterTokens.add(swapAmt) (VetterToken.sol#847)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
VetterToken._transfer(address,address,uint256).buyTax (VetterToken.sol#674) is a local variable never initialized
VetterToken._transfer(address,address,uint256).taxes (VetterToken.sol#663) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
VetterToken.addLiquidity(uint256,uint256) (VetterToken.sol#617-631) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmo
unt}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.sol#623-630)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
VetterToken.allowance(address,address).owner (VetterToken.sol#566) shadows:
        - Ownable.owner() (VetterToken.sol#215-218) (function)
VetterToken._approve(address,address,uint256).owner (VetterToken.sol#633) shadows:
        - Ownable.owner() (VetterToken.sol#215-218) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
VetterToken.setParticipatoryAddress(address)._participatoryAddress (VetterToken.sol#1246) lacks a zero-check on :
                - participatoryAddress = address(_participatoryAddress) (VetterToken.sol#1249)
VetterToken.setVetterAddress(address)._vetterAddress (VetterToken.sol#1253) lacks a zero-check on :
                - vetterAddress = address(_vetterAddress) (VetterToken.sol#1256)
VetterToken.setLiquidityAddress(address)._liquidityAddress (VetterToken.sol#1260) lacks a zero-check on :
                - liquidityAddress = address(_liquidityAddress) (VetterToken.sol#1263)
VetterToken.setReflectionAddress(address)._reflectionAddress (VetterToken.sol#1267) lacks a zero-check on :
                - reflectionAddress = address(_reflectionAddress) (VetterToken.sol#1270)
VetterToken._setRankContract(address)._contractAddress (VetterToken.sol#1317) lacks a zero-check on :
                - _rankContract = _contractAddress (VetterToken.sol#1319)
```



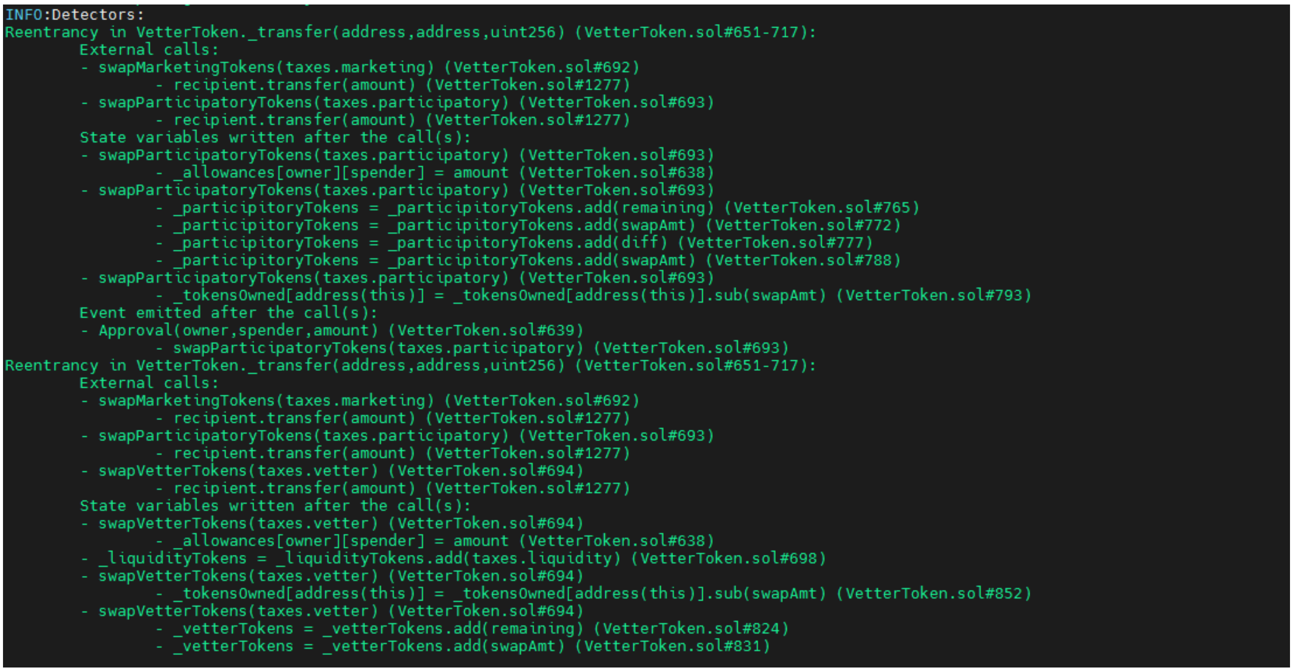
```
INFO:Detectors:
VetterToken.distributeVetterPoolTokens() (VetterToken.sol#1687-1710) has external calls inside a loop: shares = _rankAndPost._getVetterRe
wardShares(_vetterList[index]) (VetterToken.sol#1699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        External calls sending eth:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        State variables written after the call(s):
        - _liquidityTokens = _liquidityTokens.add(taxes.liquidity) (VetterToken.sol#698)
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
```



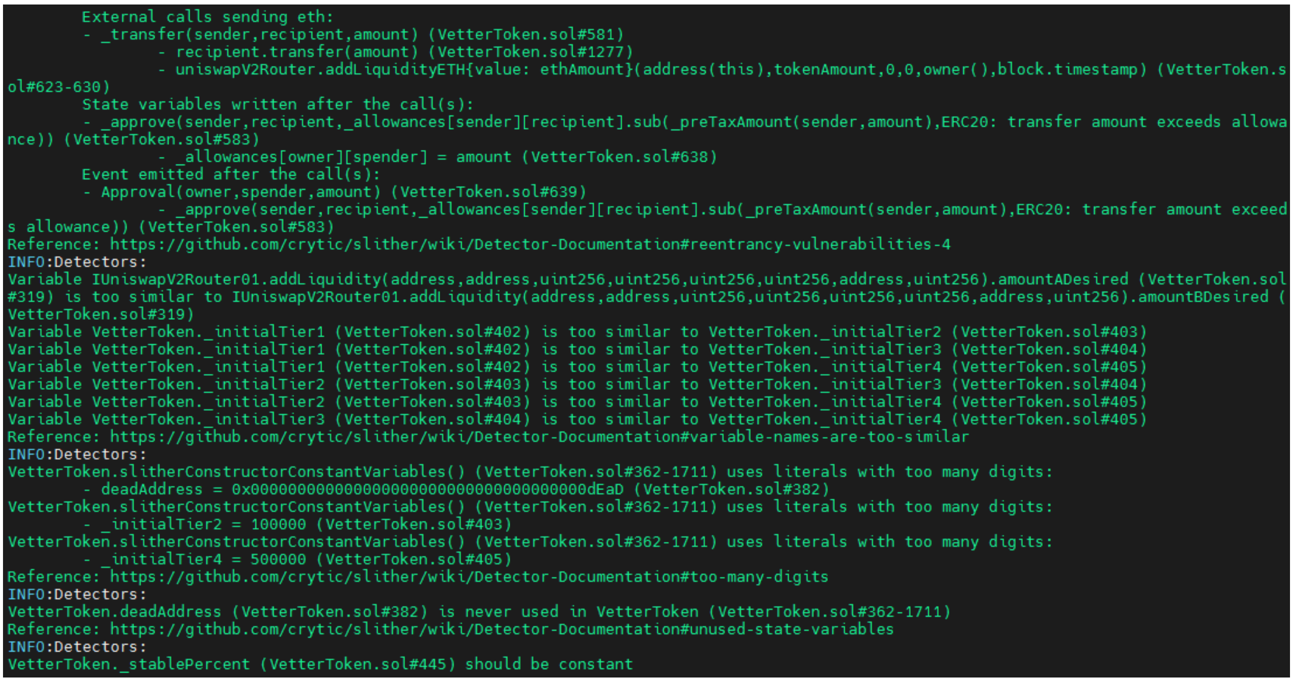
```
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        - swapReflectionTokens(taxes.reflection) (VetterToken.sol#700)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        External calls sending eth:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapLiquidityTokens(taxes.liquidity) (VetterToken.sol#697)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
        - swapReflectionTokens(taxes.reflection) (VetterToken.sol#700)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        State variables written after the call(s):
        - _stableList[_stableIndex[from]]._active = false (VetterToken.sol#709)
        - _stableList[0]._active = false (VetterToken.sol#714)
Reentrancy in VetterToken.changeRouterVersion(address) (VetterToken.sol#1286-1301):
        External calls:
        - _pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (VetterToken.sol#1294)
        State variables written after the call(s):
        - _isExcludedFromFee[address(uniswapV2Router)] = true (VetterToken.sol#1300)
        - uniswapV2Pair = _pair (VetterToken.sol#1298)
        - uniswapV2Router = _uniswapV2Router (VetterToken.sol#1299)
Reentrancy in VetterToken.constructor() (VetterToken.sol#504-530):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (VetterToken.so
l#510)
        State variables written after the call(s):
        - _isExcludedFromFee[address(this)] = true (VetterToken.sol#513)
        - _isExcludedFromFee[owner()] = true (VetterToken.sol#514)
        - _isExcludedFromFee[addressV2Router] = true (VetterToken.sol#516)
        - _isExcludedFromFee[participatoryAddress] = true (VetterToken.sol#517)
        - _isExcludedFromFee[vetterAddress] = true (VetterToken.sol#518)
        - _isExcludedFromFee[liquidityAddress] = true (VetterToken.sol#519)
```



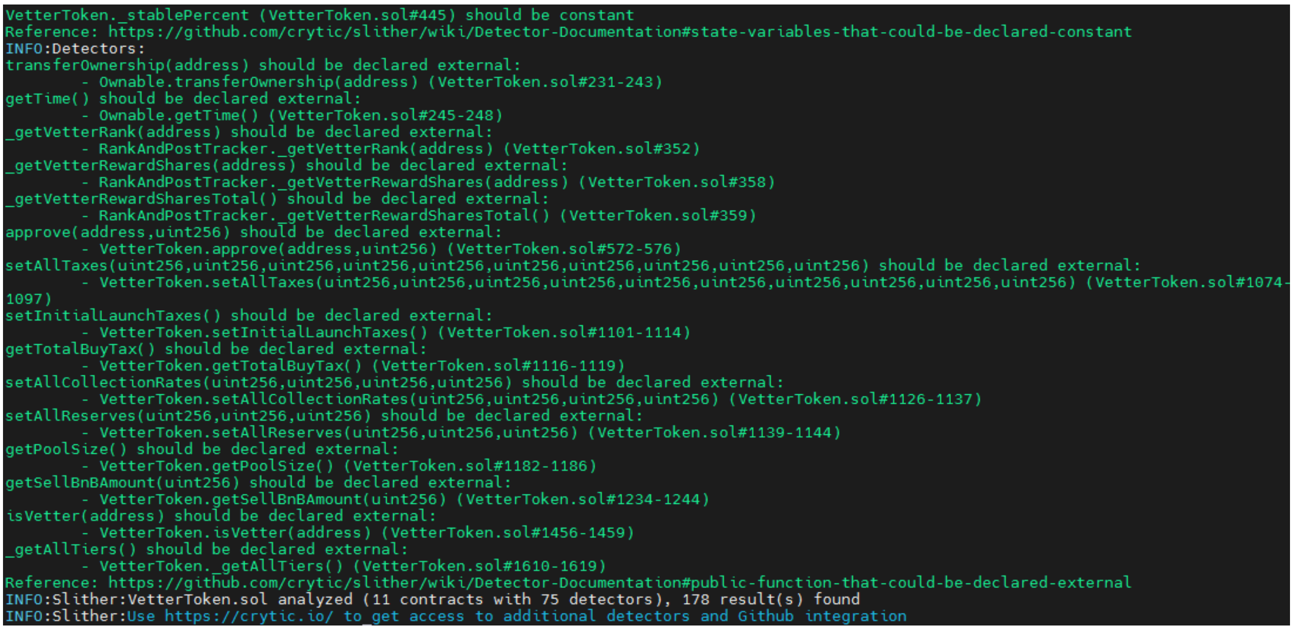
```
        - _isExcludedFromFee[reflectionAddress] = true (VetterToken.sol#520)
        - _tiers[1] = _initialTier1 (VetterToken.sol#522)
        - _tiers[2] = _initialTier2 (VetterToken.sol#523)
        - _tiers[3] = _initialTier3 (VetterToken.sol#524)
        - _tiers[4] = _initialTier4 (VetterToken.sol#525)
        - _timeTokenLaunched = block.timestamp (VetterToken.sol#527)
        - addOrAdjustVetter(owner(),true) (VetterToken.sol#515)
                - _vetterList.push(who) (VetterToken.sol#1478)
        - addOrAdjustVetter(owner(),true) (VetterToken.sol#515)
                - _vetters[who] = 1 (VetterToken.sol#1476)
                - _vetters[who] = 2 (VetterToken.sol#1477)
        - uniswapV2Router = _uniswapV2Router (VetterToken.sol#511)
Reentrancy in VetterToken.swapLiquidityPoolTokens(uint256) (VetterToken.sol#857-870):
        External calls:
        - swapTokensForBNB(_liquidityTokens) (VetterToken.sol#862)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(_liquidityTokens) (VetterToken.sol#867)
Reentrancy in VetterToken.swapLiquidityTokens(uint256) (VetterToken.sol#872-898):
        External calls:
        - swapTokensForBNB(half) (VetterToken.sol#889)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#894)
Reentrancy in VetterToken.swapMarketingTokens(uint256) (VetterToken.sol#719-737):
        External calls:
        - swapTokensForBNB(amount) (VetterToken.sol#722)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
        State variables written after the call(s):
        - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(amount) (VetterToken.sol#725)
Reentrancy in VetterToken.swapParticipatoryPoolTokens(uint256) (VetterToken.sol#739-759):
        External calls:
        - swapTokensForBNB(swapAmt) (VetterToken.sol#751)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (V
etterToken.sol#973-986)
```



```
INFO:Detectors:
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        State variables written after the call(s):
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - _participitoryTokens = _participitoryTokens.add(remaining) (VetterToken.sol#765)
                - _participitoryTokens = _participitoryTokens.add(swapAmt) (VetterToken.sol#772)
                - _participitoryTokens = _participitoryTokens.add(diff) (VetterToken.sol#777)
                - _participitoryTokens = _participitoryTokens.add(swapAmt) (VetterToken.sol#788)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#793)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (VetterToken.sol#639)
                - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
Reentrancy in VetterToken._transfer(address,address,uint256) (VetterToken.sol#651-717):
        External calls:
        - swapMarketingTokens(taxes.marketing) (VetterToken.sol#692)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapParticipatoryTokens(taxes.participatory) (VetterToken.sol#693)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - recipient.transfer(amount) (VetterToken.sol#1277)
        State variables written after the call(s):
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        - _liquidityTokens = _liquidityTokens.add(taxes.liquidity) (VetterToken.sol#698)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - _tokensOwned[address(this)] = _tokensOwned[address(this)].sub(swapAmt) (VetterToken.sol#852)
        - swapVetterTokens(taxes.vetter) (VetterToken.sol#694)
                - _vetterTokens = _vetterTokens.add(remaining) (VetterToken.sol#824)
                - _vetterTokens = _vetterTokens.add(swapAmt) (VetterToken.sol#831)
```



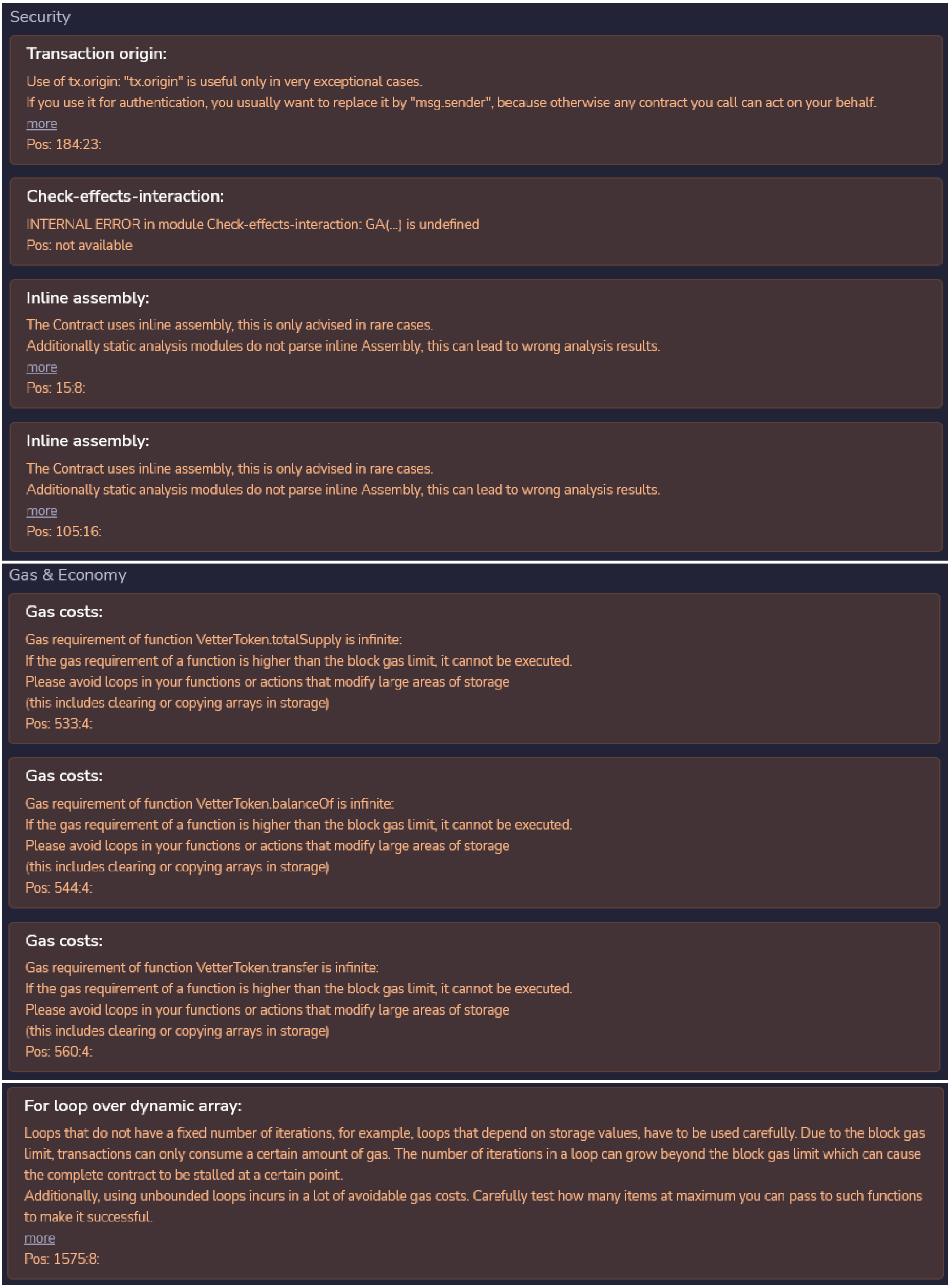
```
        External calls sending eth:
        - _transfer(sender,recipient,amount) (VetterToken.sol#581)
                - recipient.transfer(amount) (VetterToken.sol#1277)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (VetterToken.s
ol#623-630)
        State variables written after the call(s):
        - _approve(sender,recipient,_allowances[sender][recipient].sub(_preTaxAmount(sender,amount),ERC20: transfer amount exceeds allowa
nce)) (VetterToken.sol#583)
                - _allowances[owner][spender] = amount (VetterToken.sol#638)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (VetterToken.sol#639)
                - _approve(sender,recipient,_allowances[sender][recipient].sub(_preTaxAmount(sender,amount),ERC20: transfer amount exceed
s allowance)) (VetterToken.sol#583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VetterToken.sol
#319) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (
VetterToken.sol#319)
Variable VetterToken._initialTier1 (VetterToken.sol#402) is too similar to VetterToken._initialTier2 (VetterToken.sol#403)
Variable VetterToken._initialTier1 (VetterToken.sol#402) is too similar to VetterToken._initialTier3 (VetterToken.sol#404)
Variable VetterToken._initialTier1 (VetterToken.sol#402) is too similar to VetterToken._initialTier4 (VetterToken.sol#405)
Variable VetterToken._initialTier2 (VetterToken.sol#403) is too similar to VetterToken._initialTier3 (VetterToken.sol#404)
Variable VetterToken._initialTier2 (VetterToken.sol#403) is too similar to VetterToken._initialTier4 (VetterToken.sol#405)
Variable VetterToken._initialTier3 (VetterToken.sol#404) is too similar to VetterToken._initialTier4 (VetterToken.sol#405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
VetterToken.slitherConstructorConstantVariables() (VetterToken.sol#362-1711) uses literals with too many digits:
        - deadAddress = 0x000000000000000000000000000000000000dEaD (VetterToken.sol#382)
VetterToken.slitherConstructorConstantVariables() (VetterToken.sol#362-1711) uses literals with too many digits:
        - _initialTier2 = 100000 (VetterToken.sol#403)
VetterToken.slitherConstructorConstantVariables() (VetterToken.sol#362-1711) uses literals with too many digits:
        - _initialTier4 = 500000 (VetterToken.sol#405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
VetterToken.deadAddress (VetterToken.sol#382) is never used in VetterToken (VetterToken.sol#362-1711)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
VetterToken._stablePercent (VetterToken.sol#445) should be constant
```



```
VetterToken._stablePercent (VetterToken.sol#445) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (VetterToken.sol#231-243)
getTime() should be declared external:
        - Ownable.getTime() (VetterToken.sol#245-248)
_getVetterRank(address) should be declared external:
        - RankAndPostTracker._getVetterRank(address) (VetterToken.sol#352)
_getVetterRewardShares(address) should be declared external:
        - RankAndPostTracker._getVetterRewardShares(address) (VetterToken.sol#358)
_getVetterRewardSharesTotal() should be declared external:
        - RankAndPostTracker._getVetterRewardSharesTotal() (VetterToken.sol#359)
approve(address,uint256) should be declared external:
        - VetterToken.approve(address,uint256) (VetterToken.sol#572-576)
setAllTaxes(uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256) should be declared external:
        - VetterToken.setAllTaxes(uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256) (VetterToken.sol#1074-
1097)
setInitialLaunchTaxes() should be declared external:
        - VetterToken.setInitialLaunchTaxes() (VetterToken.sol#1101-1114)
getTotalBuyTax() should be declared external:
        - VetterToken.getTotalBuyTax() (VetterToken.sol#1116-1119)
setAllCollectionRates(uint256,uint256,uint256,uint256) should be declared external:
        - VetterToken.setAllCollectionRates(uint256,uint256,uint256,uint256) (VetterToken.sol#1126-1137)
setAllReserves(uint256,uint256,uint256) should be declared external:
        - VetterToken.setAllReserves(uint256,uint256,uint256) (VetterToken.sol#1139-1144)
getPoolSize() should be declared external:
        - VetterToken.getPoolSize() (VetterToken.sol#1182-1186)
getSellBnBAmount(uint256) should be declared external:
        - VetterToken.getSellBnBAmount(uint256) (VetterToken.sol#1234-1244)
isVetter(address) should be declared external:
        - VetterToken.isVetter(address) (VetterToken.sol#1456-1459)
_getAllTiers() should be declared external:
        - VetterToken._getAllTiers() (VetterToken.sol#1610-1619)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VetterToken.sol analyzed (11 contracts with 75 detectors), 178 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

Vetter.sol



Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 184:23:

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: GA(...) is undefined
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 15:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 105:16:

Gas & Economy

Gas costs:

Gas requirement of function VetterToken.totalSupply 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: 533:4:

Gas costs:

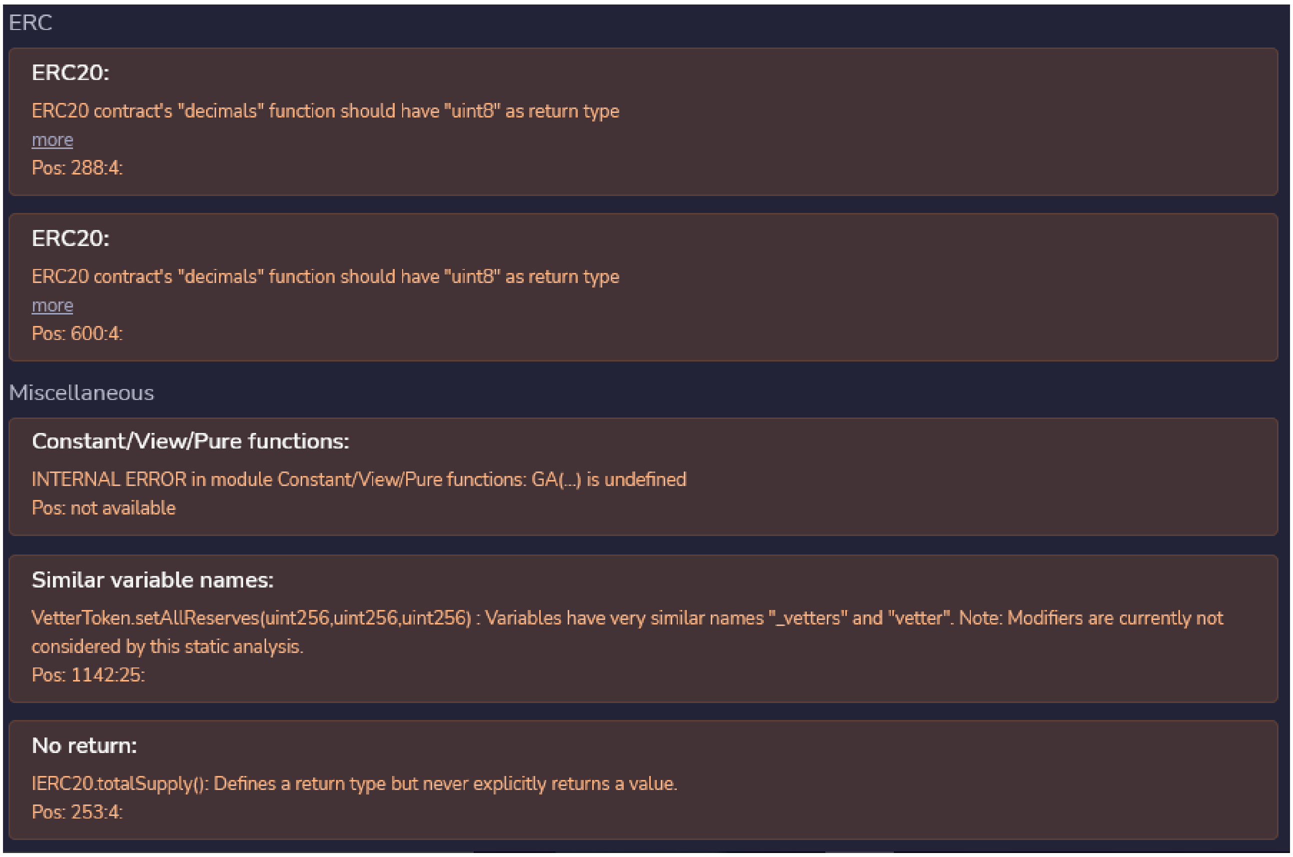
Gas requirement of function VetterToken.balanceOf is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 544:4:

Gas costs:

Gas requirement of function VetterToken.transfer 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: 560:4:

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: 1575:8:



ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 288:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 600:4:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: GA(...) is undefined

Pos: not available

Similar variable names:

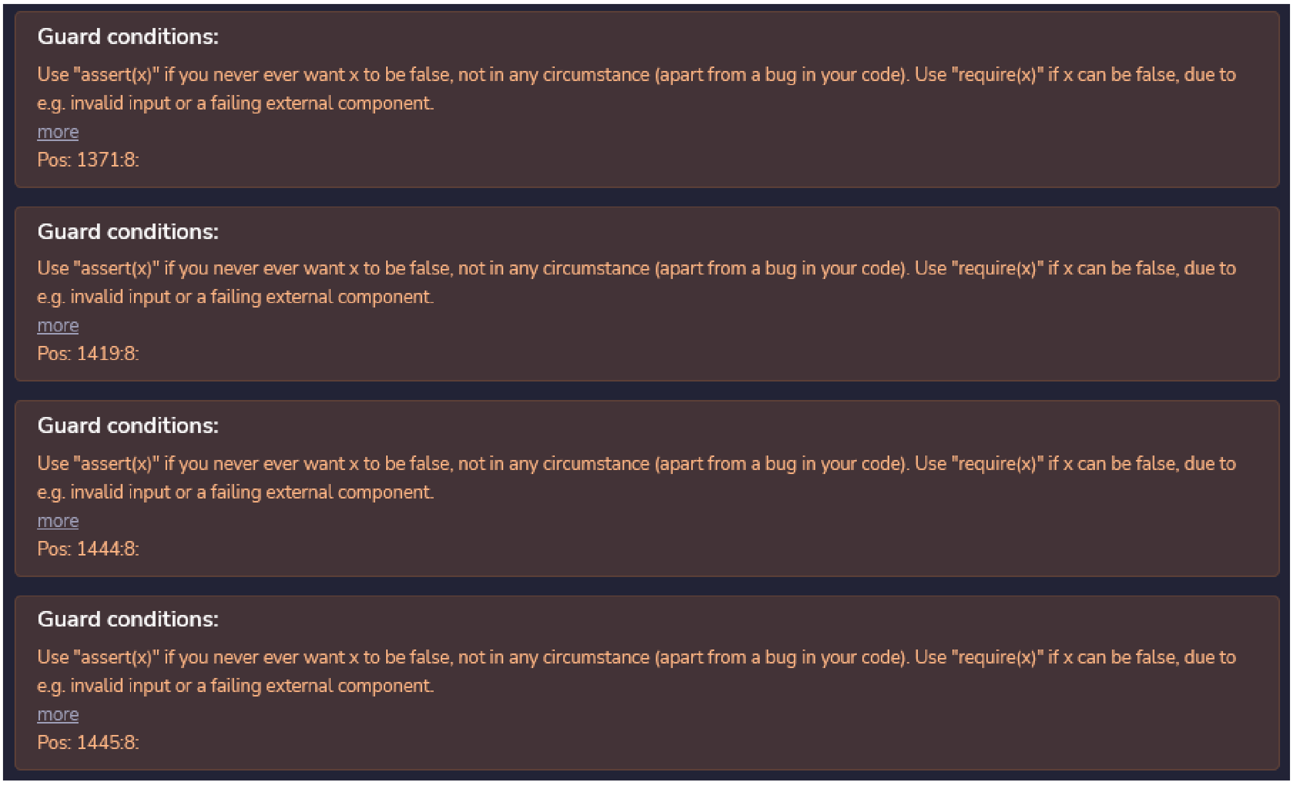
VetterToken.setAllReserves(uint256,uint256,uint256) : Variables have very similar names "_vetters" and "vetter". Note: Modifiers are currently not considered by this static analysis.

Pos: 1142:25:

No return:

IERC20.totalSupply(): Defines a return type but never explicitly returns a value.

Pos: 253:4:



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: 1371:8:

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: 1419:8:

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

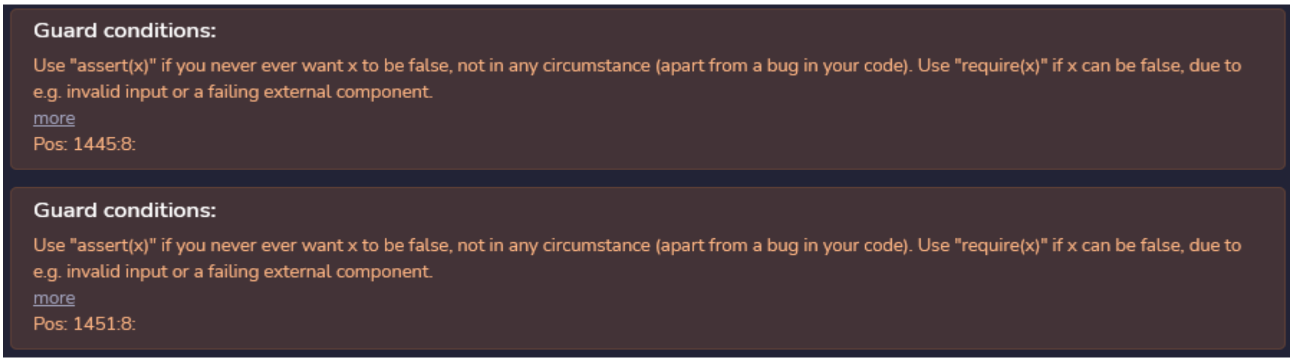
Pos: 1444:8:

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: 1445:8:



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: 1445:8:

Guard conditions:

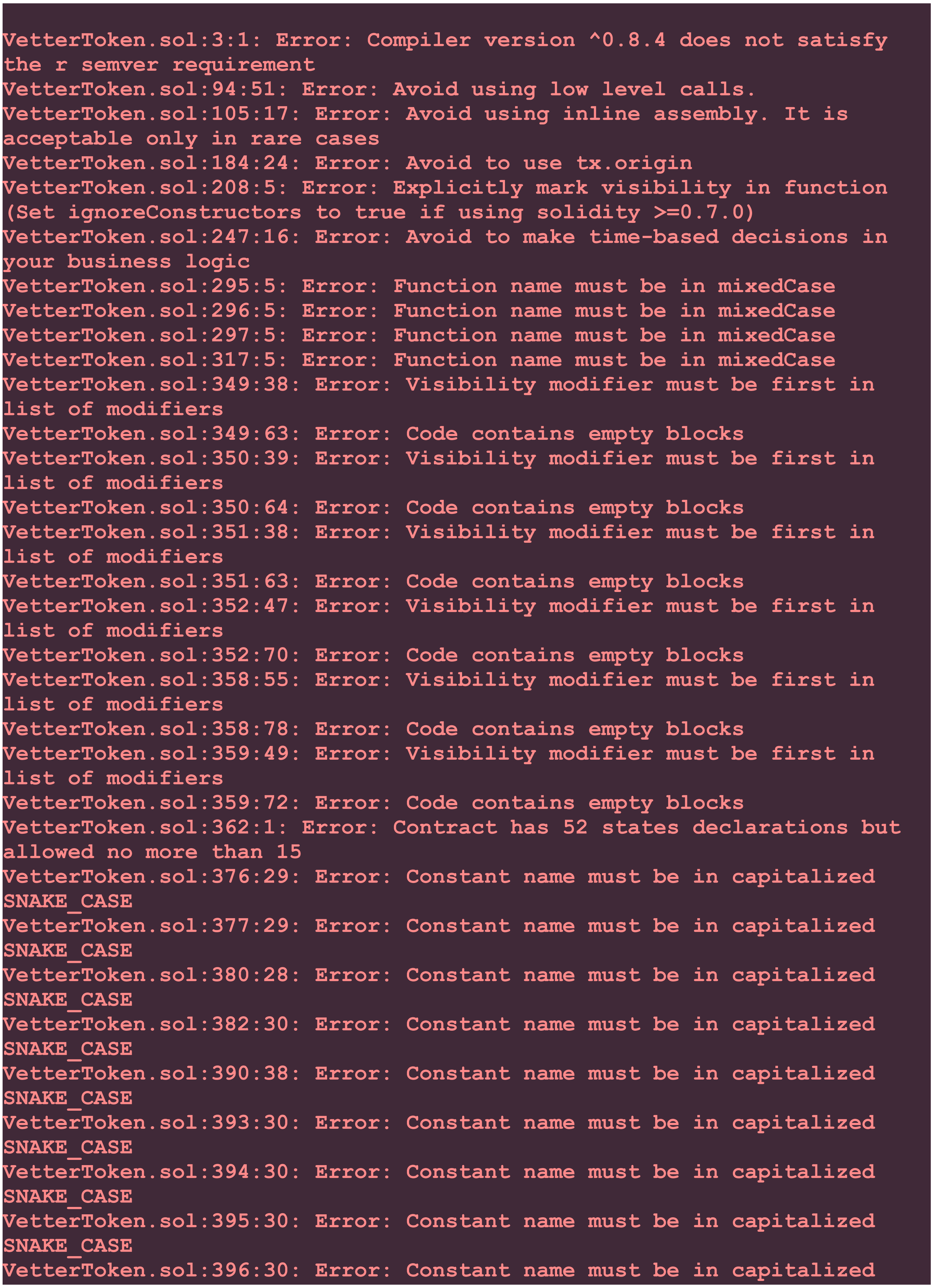
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

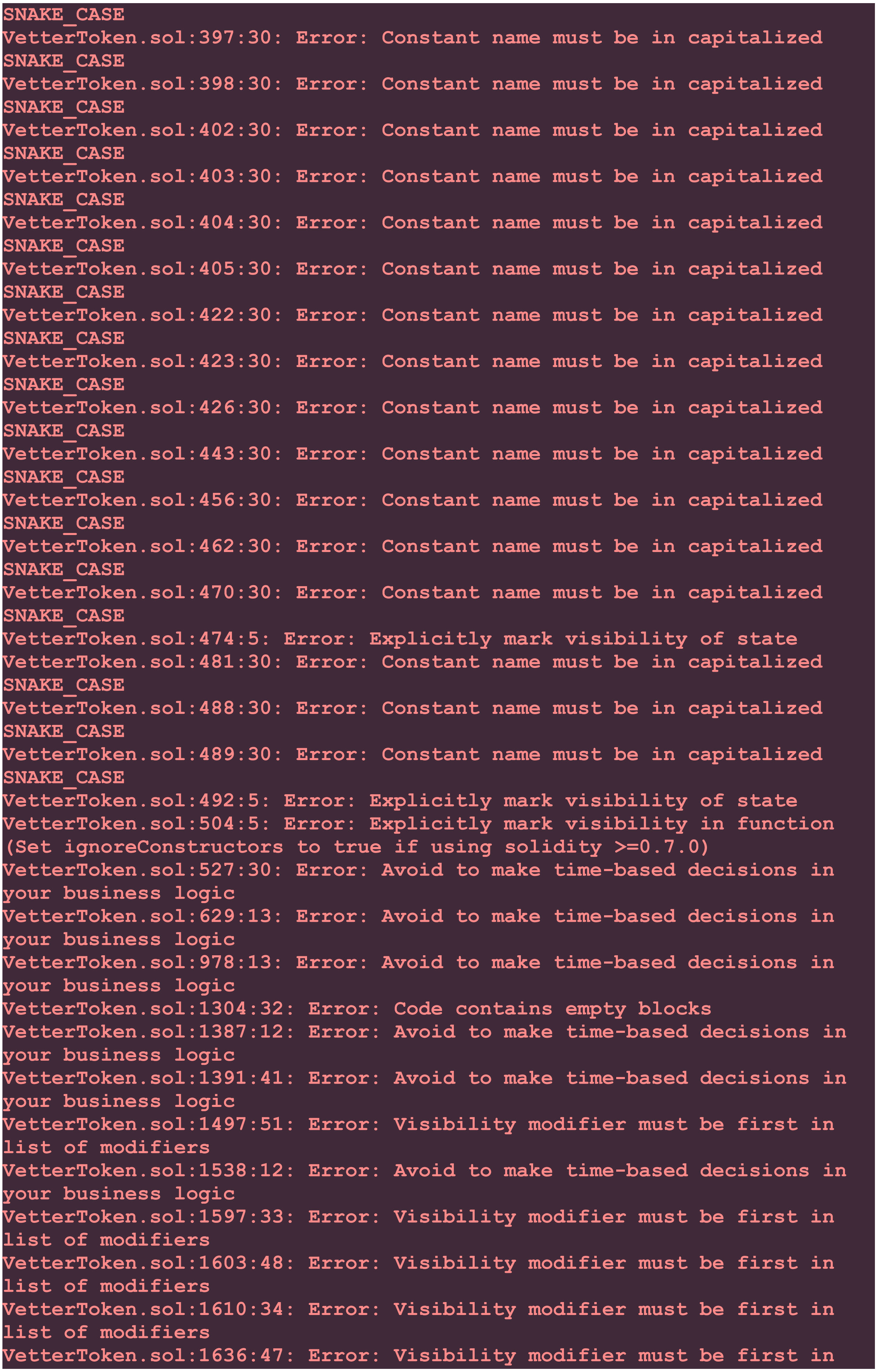
Pos: 1451:8:

Solhint Linter

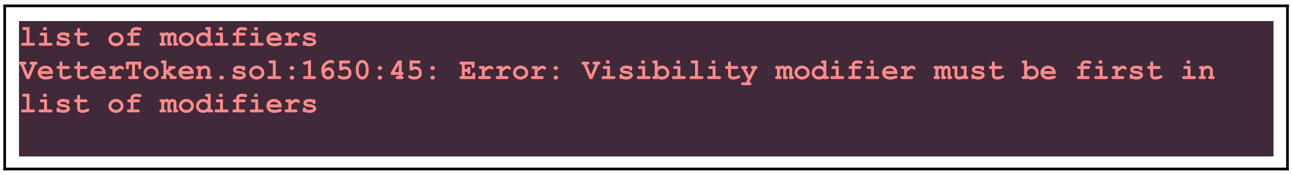
Vetter.sol



```
VetterToken.sol:3:1: Error: Compiler version ^0.8.4 does not satisfy
the r semver requirement
VetterToken.sol:94:51: Error: Avoid using low level calls.
VetterToken.sol:105:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
VetterToken.sol:184:24: Error: Avoid to use tx.origin
VetterToken.sol:208:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VetterToken.sol:247:16: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:295:5: Error: Function name must be in mixedCase
VetterToken.sol:296:5: Error: Function name must be in mixedCase
VetterToken.sol:297:5: Error: Function name must be in mixedCase
VetterToken.sol:317:5: Error: Function name must be in mixedCase
VetterToken.sol:349:38: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:349:63: Error: Code contains empty blocks
VetterToken.sol:350:39: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:350:64: Error: Code contains empty blocks
VetterToken.sol:351:38: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:351:63: Error: Code contains empty blocks
VetterToken.sol:352:47: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:352:70: Error: Code contains empty blocks
VetterToken.sol:358:55: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:358:78: Error: Code contains empty blocks
VetterToken.sol:359:49: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:359:72: Error: Code contains empty blocks
VetterToken.sol:362:1: Error: Contract has 52 states declarations but
allowed no more than 15
VetterToken.sol:376:29: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:377:29: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:380:28: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:382:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:390:38: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:393:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:394:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:395:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:396:30: Error: Constant name must be in capitalized
```



```
SNAKE_CASE
VetterToken.sol:397:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:398:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:402:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:403:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:404:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:405:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:422:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:423:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:426:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:443:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:456:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:462:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:470:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:474:5: Error: Explicitly mark visibility of state
VetterToken.sol:481:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:488:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:489:30: Error: Constant name must be in capitalized
SNAKE_CASE
VetterToken.sol:492:5: Error: Explicitly mark visibility of state
VetterToken.sol:504:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VetterToken.sol:527:30: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:629:13: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:978:13: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:1304:32: Error: Code contains empty blocks
VetterToken.sol:1387:12: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:1391:41: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:1497:51: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:1538:12: Error: Avoid to make time-based decisions in
your business logic
VetterToken.sol:1597:33: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:1603:48: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:1610:34: Error: Visibility modifier must be first in
list of modifiers
VetterToken.sol:1636:47: Error: Visibility modifier must be first in
```



```
list of modifiers
VetterToken.sol:1650:45: Error: Visibility modifier must be first in
list of modifiers
```

Software analysis result:

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

EtherAuthority

Email: audit@EtherAuthority.io