EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Hunter Token
Website:	archerswap.finance/hunt
Platform:	Core Chain
Language:	Solidity
Date:	March 5th, 2023

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 Hunter Token team to perform the Security audit of the Hunter 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 March 5th, 2023.

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

- ArcherSwap Hunter Token is a crypto world for users to trade, earn. It will be run on **Core Chain** with features including AMM.

Audit scope

Name	**Code Review and Security Analysis Report for Hunter Token Smart Contract**
Platform	**Core Chain / Solidity**
File	Hunter.sol
File MD5 Hash	F50FF9F445078AD33815FDE203D4A892
Audit Date	March 5th, 2023

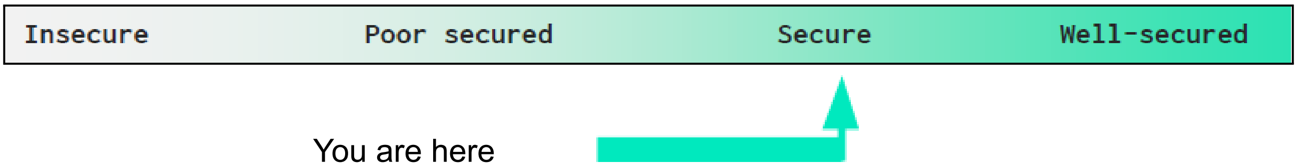
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File: **Hunter Contract** **Tokenomics:** ● Name: Hunter ● Symbol: HUNT ● Decimals: 18 **Total BPS Fee: 4%** ● Treasury Fee BPS: 1% ● Liquidity BPS Fee: 1% ● Dividend BPS Fee: 2% ● Maximum Transactions BPS: 49 ● Maximum Wallet BPS: 200 **Owner Specifications:** ● Open trading can approve status true by the owner. ● Divided amount sent manually by the owner. ● marketing Wallet address and liquidity Wallet address set by the owner. ● A market maker pair address can be set with a value by the owner. ● Treasury fees, liquidity fees, dividend fees can be set by the owner. ● An owner can enable swapping. ● Owner can set the tax status enabled. ● Owner can set the compounding status enabled. ● Owner can update divided settings. ● It is possible for the owner to update the maximum transaction BPS value. ● A maximum transaction limit can be set for an account address.	**YES, This is valid.**

● The maximum wallet BPS can be set by the owner.	
File: DividendTracker Contract **Tokenomics:** ● Name: Hunter_DividendTracker ● Symbol: Hunter_DividendTracker ● Decimals: 18 ● minimum Token Balance For Dividends: 10 **Owner Specifications:** ● New balance will be updated by the owner. ● Dividends account wallet addresses can be excluded from dividends account wallet addresses. ● Divided amount sent to the holder address manually by the owner. ● Owner can set the process account address status true. ● Owner can set the compound account address status true.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity based 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 0 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	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	Moderated
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. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Hunter 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 Hunter Token.

The Hunter 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 in the smart contracts. Ethereum's NatSpec commenting style is used, which is a good thing.

Documentation

We were given a Hunter Token smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. But the logic is straightforward. So it is 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://archerswap.finance/hunt 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 not used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	receive	external	Passed	No Issue
8	name	read	Passed	No Issue
9	symbol	read	Passed	No Issue
10	decimals	write	Passed	No Issue
11	totalSupply	read	Passed	No Issue
12	balanceOf	read	Passed	No Issue
13	allowance	read	Passed	No Issue
14	approve	write	Passed	No Issue
15	increaseAllowance	write	Passed	No Issue
16	decreaseAllowance	write	Passed	No Issue
17	transfer	write	Passed	No Issue
18	transferFrom	write	Passed	No Issue
19	openTrading	external	access only Owner	No Issue
20	_transfer	internal	Passed	No Issue
21	_executeTransfer	write	Passed	No Issue
22	_approve	write	Passed	No Issue
23	_mint	write	Passed	No Issue
24	_burn	write	Passed	No Issue
25	swapTokensForNative	write	Passed	No Issue
26	addLiquidity	write	Passed	No Issue
27	includeToWhiteList	write	Passed	No Issue
28	_executeSwap	write	Passed	No Issue
29	excludeFromFees	write	access only Owner	No Issue
30	isExcludedFromFees	read	Passed	No Issue
31	manualSendDividend	external	access only Owner	No Issue
32	excludeFromDividends	write	access only Owner	No Issue
33	isExcludedFromDividends	read	Passed	No Issue
34	setWallet	external	access only Owner	No Issue
35	setAutomatedMarketMakerPair	write	access only Owner	No Issue
36	setFee	external	access only Owner	No Issue
37	_setAutomatedMarketMakerPair	write	Passed	No Issue
38	updateUniswapV2Router	write	access only Owner	No Issue
39	claim	write	Transferred 0 amount	Refer Audit Findings

40	compound	write	Passed	No Issue
41	withdrawableDividendOf	read	Passed	No Issue
42	withdrawnDividendOf	read	Passed	No Issue
43	accumulativeDividendOf	read	Passed	No Issue
44	getAccountInfo	read	Passed	No Issue
45	getLastClaimTime	read	Passed	No Issue
46	setSwapEnabled	external	access only Owner	No Issue
47	setTaxEnabled	external	access only Owner	No Issue
48	setCompoundingEnabled	external	access only Owner	No Issue
49	updateDividendSettings	external	access only Owner	No Issue
50	setMaxTxBPS	external	Default value and Range validation mismatch	Refer Audit Findings
51	excludeFromMaxTx	write	access only Owner	No Issue
52	isExcludedFromMaxTx	read	Passed	No Issue
53	setMaxWalletBPS	external	access only Owner	No Issue
54	excludeFromMaxWallet	write	access only Owner	No Issue
55	isExcludedFromMaxWallet	read	Passed	No Issue
56	rescueToken	external	access only Owner	No Issue
57	rescueETH	external	access only Owner	No Issue
58	receive	external	Passed	No Issue
59	distributeDividends	write	Passed	No Issue
60	setBalance	external	access only Owner	No Issue
61	excludeFromDividends	external	access only Owner	No Issue
62	isExcludedFromDividends	read	Passed	No Issue
63	manualSendDividend	external	access only Owner	No Issue
64	_setBalance	external	Passed	No Issue
65	_mint	write	Passed	No Issue
66	_burn	write	Passed	No Issue
67	processAccount	write	access only Owner	No Issue
68	_withdrawDividendOfUser	write	Passed	No Issue
69	compoundAccount	write	access only Owner	No Issue
70	_compoundDividendOfUser	write	Passed	No Issue
71	withdrawableDividendOf	read	Passed	No Issue
72	withdrawnDividendOf	read	Passed	No Issue
73	accumulativeDividendOf	read	Passed	No Issue
74	getAccountInfo	read	Passed	No Issue
75	getLastClaimTime	read	Passed	No Issue
76	name	read	Passed	No Issue
77	symbol	read	Passed	No Issue
78	decimals	write	Passed	No Issue
79	totalSupply	read	Passed	No Issue
80	balanceOf	read	Passed	No Issue
81	transfer	write	Passed	No Issue
82	allowance	write	Passed	No Issue
83	approve	write	Passed	No Issue
84	transferFrom	write	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

No High severity vulnerabilities were found.

Medium

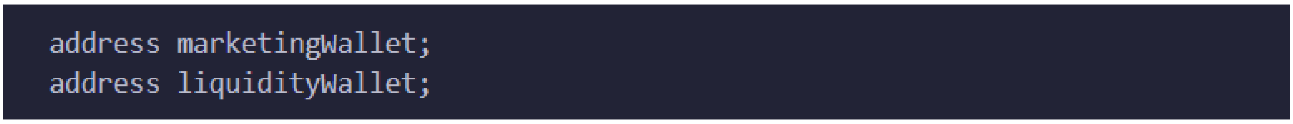
No Medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Variable visibility:



```
address marketingWallet;
address liquidityWallet;
```

Variables are defined by default visibility which is private. So no way to view those addresses.

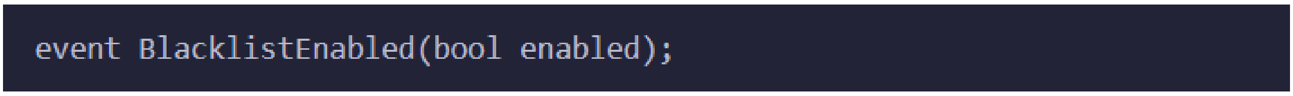
Resolution: We suggest defining the variable as public. Ignore if it is a part of the plan.

(2) Transferred 0 amount:

claim() function allows to transfer 0 tokens.

Resolution: We suggest avoiding 0 amounts to get transferred.

(3) Unused event is defined:

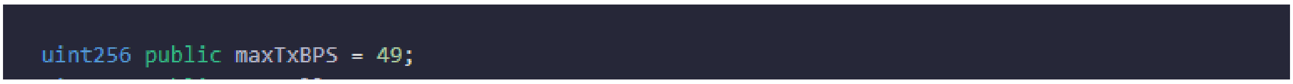


```
event BlacklistEnabled(bool enabled);
```

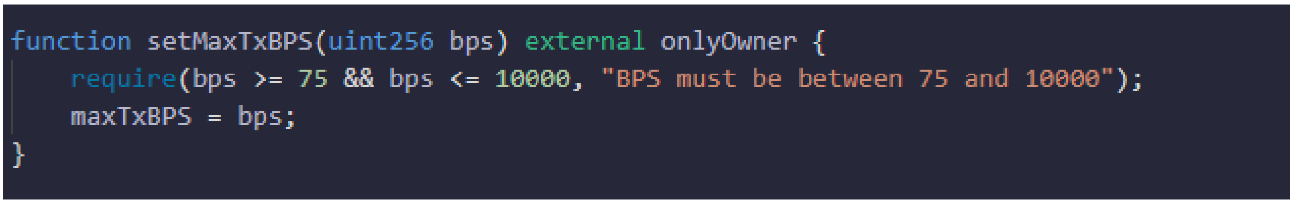
There is a "BlacklistEnabled" event defined in the smart contract but not used anywhere.

Resolution: Remove unused events from the smart contract.

(4) Default value and Range validation mismatch:



```
uint256 public maxTxBPS = 49;
```



```
function setMaxTxBPS(uint256 bps) external onlyOwner {
    require(bps >= 75 && bps <= 10000, "BPS must be between 75 and 10000");
    maxTxBPS = bps;
}
```

Default value of maxTxBPS is 49, but the range is 75 to 10000. As a result, the range does not match the default.

Resolution: We suggest correcting the values.

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:

Hunter.sol

- openTrading: Open trading can approve status true by the owner.
- excludeFromFees: Owner can set an account address which excludes the fee account address.
- manualSendDividend: Divided amount sent manually by the owner.
- excludeFromDividends: Owner can exclude the amount from the dividends account wallet address.

- setWallet: marketing Wallet address and liquidity Wallet address set by the owner.
- setAutomatedMarketMakerPair: Owner can set an automated market maker pair address with value.
- setFee: Treasury fees, liquidity fees, dividend fees can be set by the owner.
- updateUniswapV2Router: Owner can update a new uniswapV2Router address.
- setSwapEnabled: Owner can set the swap status enabled.
- setTaxEnabled: Owner can set the tax status enabled.
- setCompoundingEnabled: Owner can set the compounding status enabled.
- updateDividendSettings: Owner can update divided settings.
- setMaxTxBPS: Owner can update the maximum transaction BPS value.
- excludeFromMaxTx: Owner can set an account address which excludes maximum transactions.
- setMaxWalletBPS: Owner can set the maximum wallet BPS.
- excludeFromMaxWallet: Owner can set an account address which excludes maximum wallet addresses.
- rescueToken: Owner can rescue the token.
- rescueETH: Owner can rescue ether amount.

DividendTracker.sol

- setBalance: New balance will be updated by the owner.
- excludeFromDividends: Owner can exclude the amount from the dividends account wallet address.
- manualSendDividend: Divided amount sent to the holder address manually by the owner.
- processAccount: Owner can set process account address status true.
- compoundAccount: Owner can set compound account address status true.

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 in the form of a file and we have used all possible tests based on given objects as files. We have observed some informational severity issues in the token smart contract. But those issues are not critical. So, **it's good to go for the mainnet deployment**.

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.

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

The security state of the reviewed smart 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).

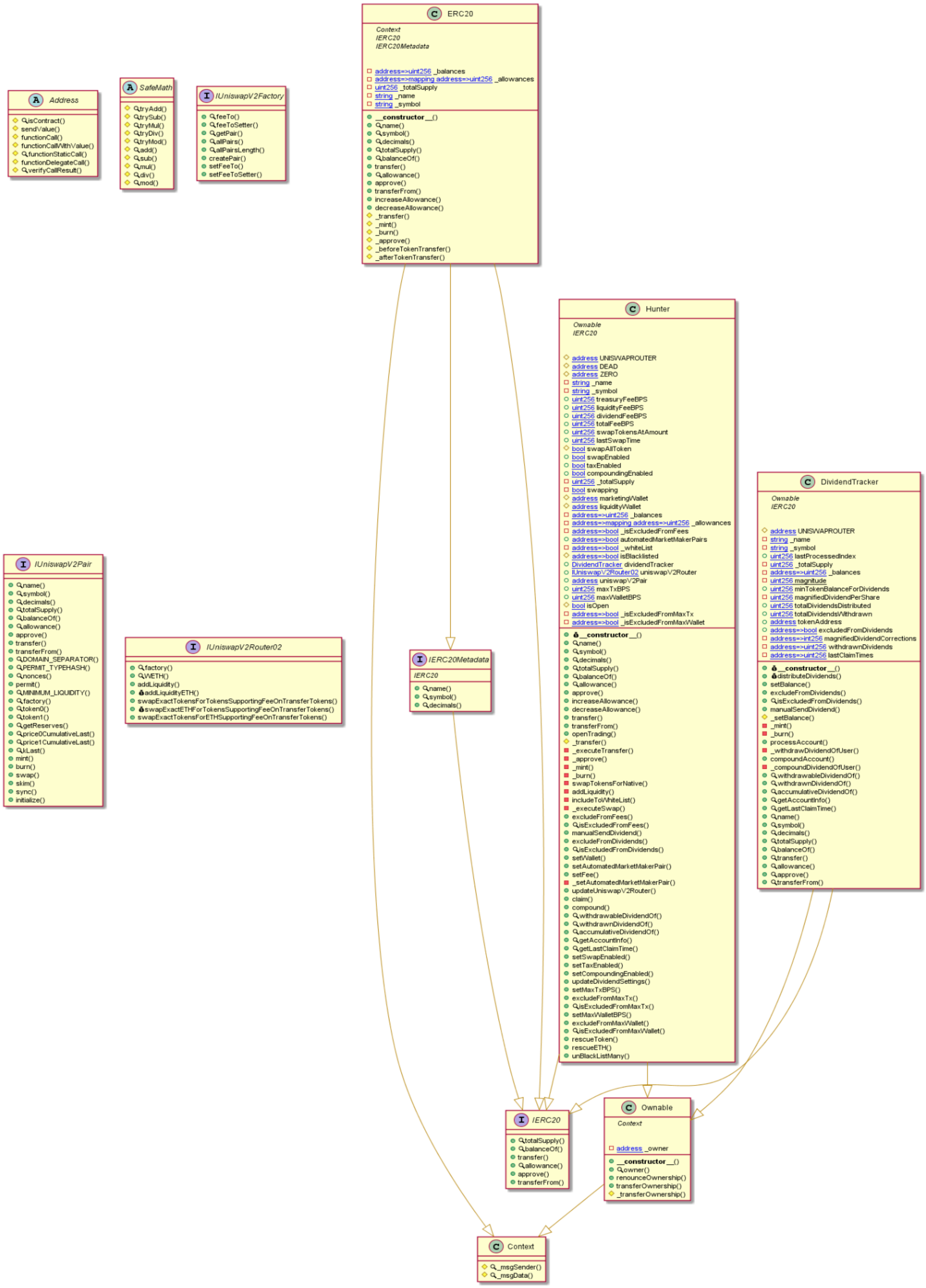
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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

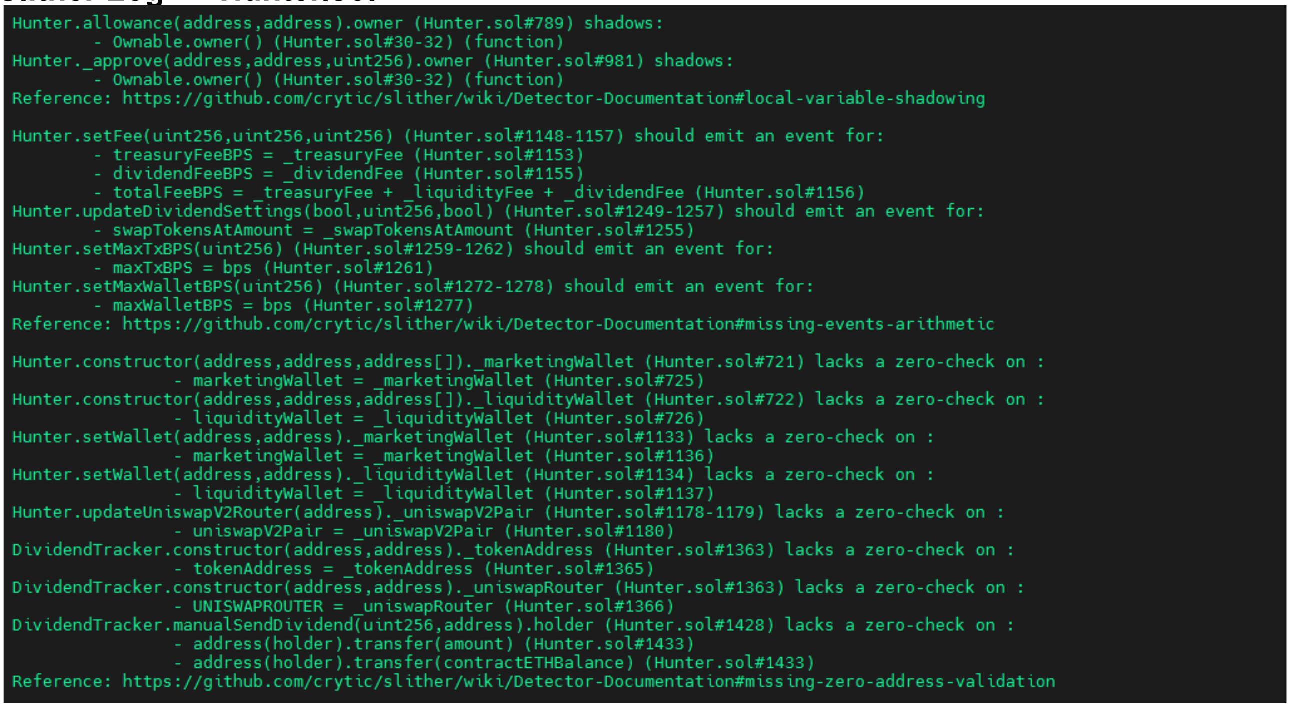
Appendix

Code Flow Diagram - Hunter Token

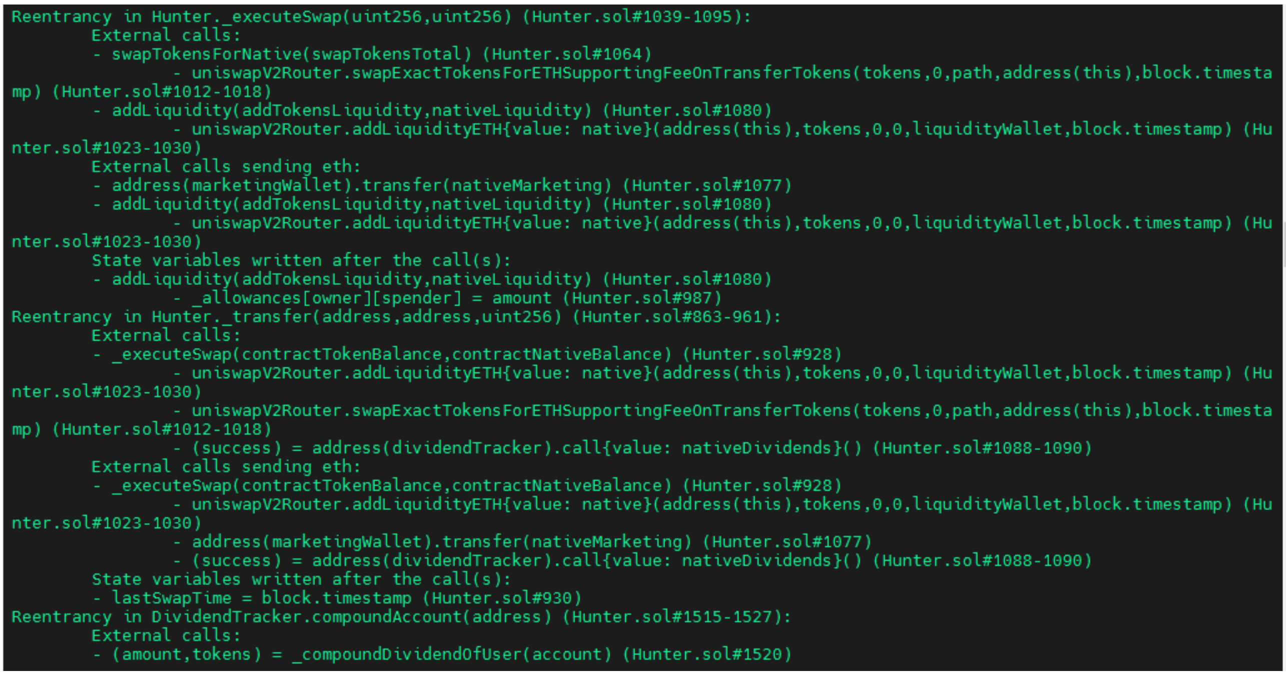


Slither Results Log

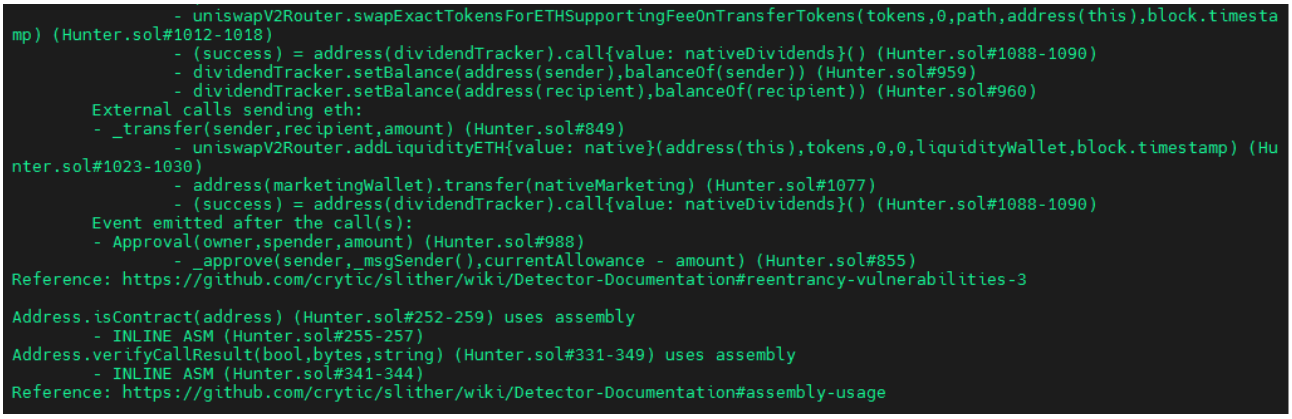
Slither Log >> Hunter.sol



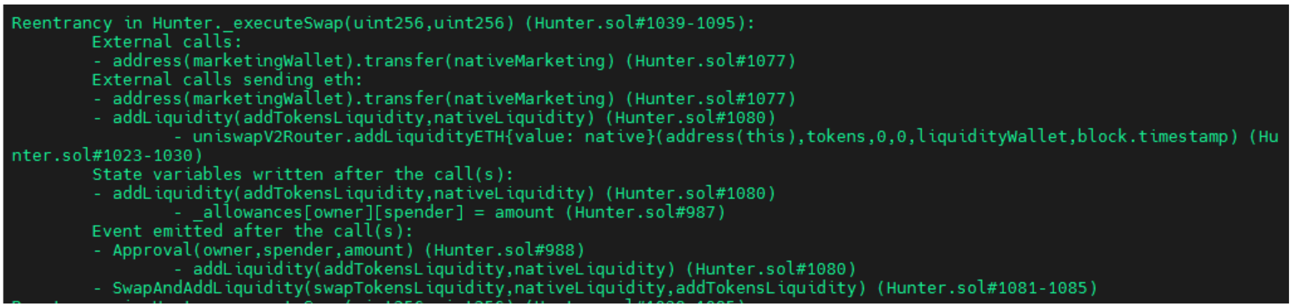
```
Hunter.allowance(address,address).owner (Hunter.sol#789) shadows:
        - Ownable.owner() (Hunter.sol#30-32) (function)
Hunter._approve(address,address,uint256).owner (Hunter.sol#981) shadows:
        - Ownable.owner() (Hunter.sol#30-32) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Hunter.setFee(uint256,uint256,uint256) (Hunter.sol#1148-1157) should emit an event for:
        - treasuryFeeBPS = _treasuryFee (Hunter.sol#1153)
        - dividendFeeBPS = _dividendFee (Hunter.sol#1155)
        - totalFeeBPS = _treasuryFee + _liquidityFee + _dividendFee (Hunter.sol#1156)
Hunter.updateDividendSettings(bool,uint256,bool) (Hunter.sol#1249-1257) should emit an event for:
        - swapTokensAtAmount = _swapTokensAtAmount (Hunter.sol#1255)
Hunter.setMaxTxBPS(uint256) (Hunter.sol#1259-1262) should emit an event for:
        - maxTxBPS = bps (Hunter.sol#1261)
Hunter.setMaxWalletBPS(uint256) (Hunter.sol#1272-1278) should emit an event for:
        - maxWalletBPS = bps (Hunter.sol#1277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Hunter.constructor(address,address,address[])._marketingWallet (Hunter.sol#721) lacks a zero-check on :
                - marketingWallet = _marketingWallet (Hunter.sol#725)
Hunter.constructor(address,address,address[])._liquidityWallet (Hunter.sol#722) lacks a zero-check on :
                - liquidityWallet = _liquidityWallet (Hunter.sol#726)
Hunter.setWallet(address,address)._marketingWallet (Hunter.sol#1133) lacks a zero-check on :
                - marketingWallet = _marketingWallet (Hunter.sol#1136)
Hunter.setWallet(address,address)._liquidityWallet (Hunter.sol#1134) lacks a zero-check on :
                - liquidityWallet = _liquidityWallet (Hunter.sol#1137)
Hunter.updateUniswapV2Router(address)._uniswapV2Pair (Hunter.sol#1178-1179) lacks a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (Hunter.sol#1180)
DividendTracker.constructor(address,address)._tokenAddress (Hunter.sol#1363) lacks a zero-check on :
                - tokenAddress = _tokenAddress (Hunter.sol#1365)
DividendTracker.constructor(address,address)._uniswapRouter (Hunter.sol#1363) lacks a zero-check on :
                - UNISWAPROUTER = _uniswapRouter (Hunter.sol#1366)
DividendTracker.manualSendDividend(uint256,address).holder (Hunter.sol#1428) lacks a zero-check on :
                - address(holder).transfer(amount) (Hunter.sol#1433)
                - address(holder).transfer(contractETHBalance) (Hunter.sol#1433)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



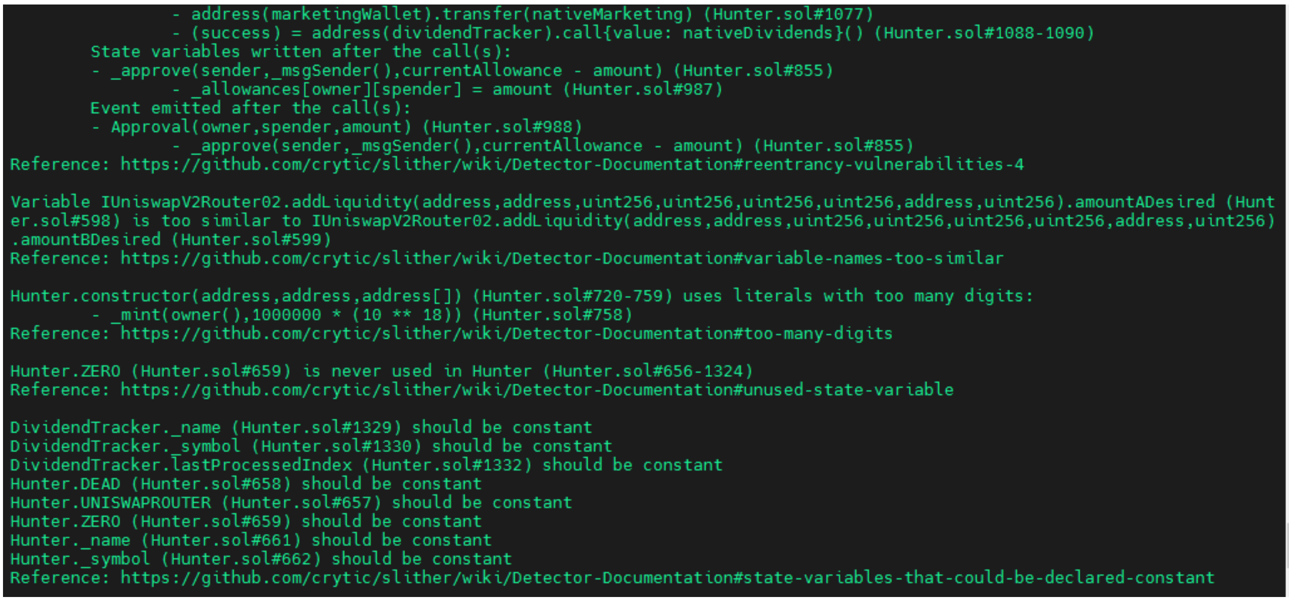
```
Reentrancy in Hunter._executeSwap(uint256,uint256) (Hunter.sol#1039-1095):
        External calls:
        - swapTokensForNative(swapTokensTotal) (Hunter.sol#1064)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokens,0,path,address(this),block.timesta
mp) (Hunter.sol#1012-1018)
        - addLiquidity(addTokensLiquidity,nativeLiquidity) (Hunter.sol#1080)
                - uniswapV2Router.addLiquidityETH{value: native}(address(this),tokens,0,0,liquidityWallet,block.timestamp) (Hu
nter.sol#1023-1030)
        External calls sending eth:
        - address(marketingWallet).transfer(nativeMarketing) (Hunter.sol#1077)
        - addLiquidity(addTokensLiquidity,nativeLiquidity) (Hunter.sol#1080)
                - uniswapV2Router.addLiquidityETH{value: native}(address(this),tokens,0,0,liquidityWallet,block.timestamp) (Hu
nter.sol#1023-1030)
        State variables written after the call(s):
        - addLiquidity(addTokensLiquidity,nativeLiquidity) (Hunter.sol#1080)
                - _allowances[owner][spender] = amount (Hunter.sol#987)
Reentrancy in Hunter._transfer(address,address,uint256) (Hunter.sol#863-961):
        External calls:
        - _executeSwap(contractTokenBalance,contractNativeBalance) (Hunter.sol#928)
                - uniswapV2Router.addLiquidityETH{value: native}(address(this),tokens,0,0,liquidityWallet,block.timestamp) (Hu
nter.sol#1023-1030)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokens,0,path,address(this),block.timesta
mp) (Hunter.sol#1012-1018)
                - (success) = address(dividendTracker).call{value: nativeDividends}() (Hunter.sol#1088-1090)
        External calls sending eth:
        - _executeSwap(contractTokenBalance,contractNativeBalance) (Hunter.sol#928)
                - uniswapV2Router.addLiquidityETH{value: native}(address(this),tokens,0,0,liquidityWallet,block.timestamp) (Hu
nter.sol#1023-1030)
                - address(marketingWallet).transfer(nativeMarketing) (Hunter.sol#1077)
                - (success) = address(dividendTracker).call{value: nativeDividends}() (Hunter.sol#1088-1090)
        State variables written after the call(s):
        - lastSwapTime = block.timestamp (Hunter.sol#930)
Reentrancy in DividendTracker.compoundAccount(address) (Hunter.sol#1515-1527):
        External calls:
        - (amount,tokens) = _compoundDividendOfUser(account) (Hunter.sol#1520)
```



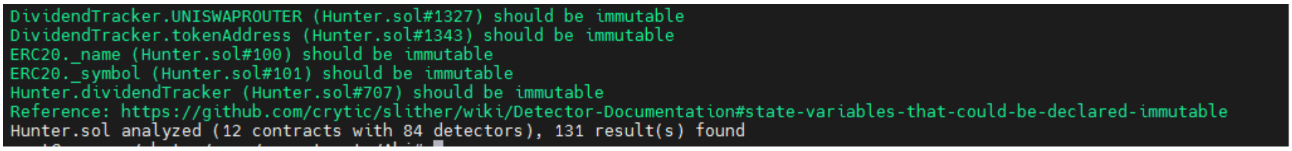
```
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokens,0,path,address(this),block.timesta
mp) (Hunter.sol#1012-1018)
                - (success) = address(dividendTracker).call{value: nativeDividends}() (Hunter.sol#1088-1090)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (Hunter.sol#959)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient)) (Hunter.sol#960)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Hunter.sol#849)
                - uniswapV2Router.addLiquidityETH{value: native}(address(this),tokens,0,0,liquidityWallet,block.timestamp) (Hu
nter.sol#1023-1030)
                - address(marketingWallet).transfer(nativeMarketing) (Hunter.sol#1077)
                - (success) = address(dividendTracker).call{value: nativeDividends}() (Hunter.sol#1088-1090)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Hunter.sol#988)
                - _approve(sender,_msgSender(),currentAllowance - amount) (Hunter.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (Hunter.sol#252-259) uses assembly
        - INLINE ASM (Hunter.sol#255-257)
Address.verifyCallResult(bool,bytes,string) (Hunter.sol#331-349) uses assembly
        - INLINE ASM (Hunter.sol#341-344)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



```
Reentrancy in Hunter._executeSwap(uint256,uint256) (Hunter.sol#1039-1095):
        External calls:
        - address(marketingWallet).transfer(nativeMarketing) (Hunter.sol#1077)
        External calls sending eth:
        - address(marketingWallet).transfer(nativeMarketing) (Hunter.sol#1077)
        - addLiquidity(addTokensLiquidity,nativeLiquidity) (Hunter.sol#1080)
                - uniswapV2Router.addLiquidityETH{value: native}(address(this),tokens,0,0,liquidityWallet,block.timestamp) (Hu
nter.sol#1023-1030)
        State variables written after the call(s):
        - addLiquidity(addTokensLiquidity,nativeLiquidity) (Hunter.sol#1080)
                - _allowances[owner][spender] = amount (Hunter.sol#987)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Hunter.sol#988)
                - addLiquidity(addTokensLiquidity,nativeLiquidity) (Hunter.sol#1080)
        - SwapAndAddLiquidity(swapTokensLiquidity,nativeLiquidity,addTokensLiquidity) (Hunter.sol#1081-1085)
```



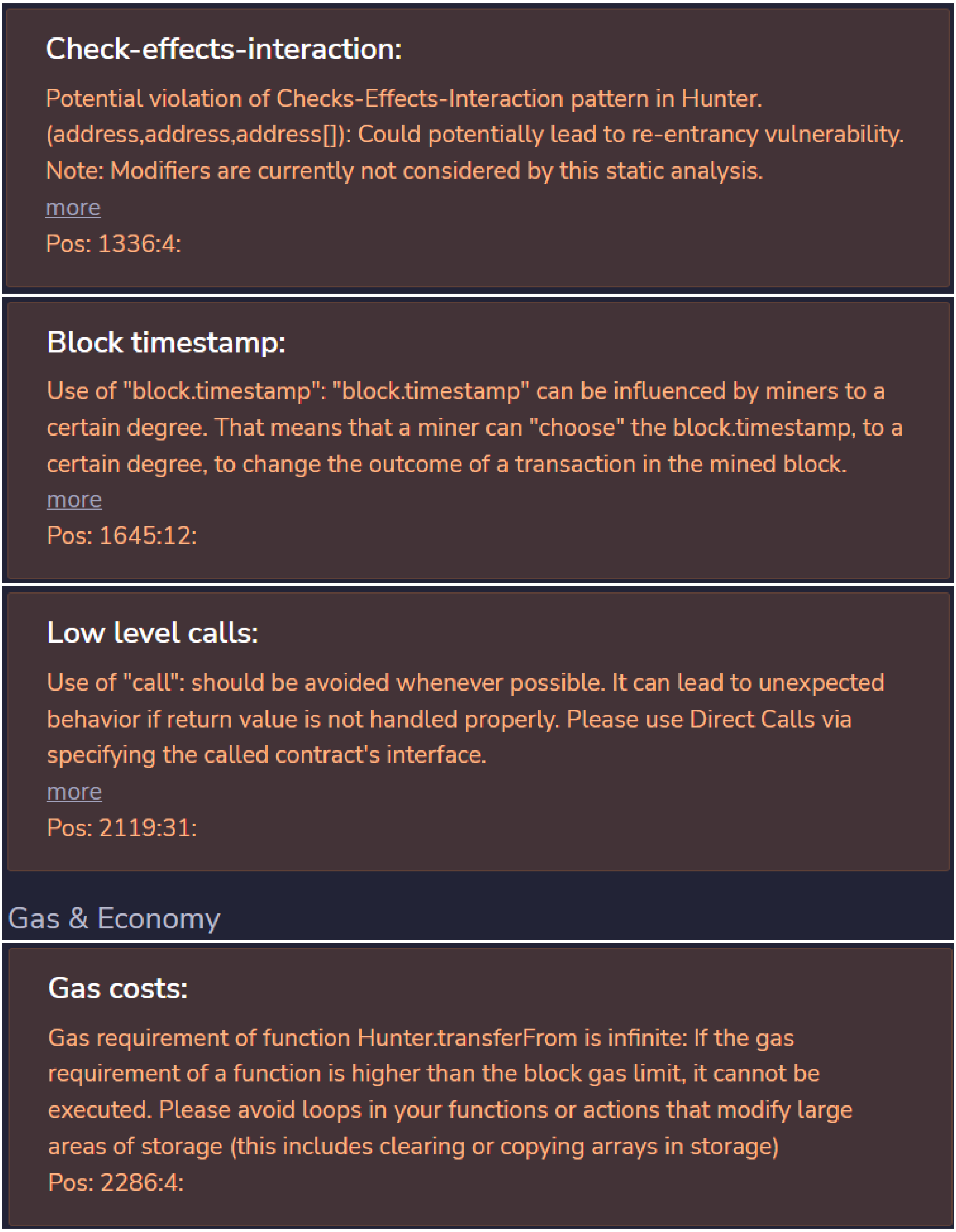
```
                - address(marketingWallet).transfer(nativeMarketing) (Hunter.sol#1077)
                - (success) = address(dividendTracker).call{value: nativeDividends}() (Hunter.sol#1088-1090)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (Hunter.sol#855)
                - _allowances[owner][spender] = amount (Hunter.sol#987)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Hunter.sol#988)
                - _approve(sender,_msgSender(),currentAllowance - amount) (Hunter.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Hunt
er.sol#598) is too similar to IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
.amountBDesired (Hunter.sol#599)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Hunter.constructor(address,address,address[]) (Hunter.sol#720-759) uses literals with too many digits:
        - _mint(owner(),1000000 * (10 ** 18)) (Hunter.sol#758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Hunter.ZERO (Hunter.sol#659) is never used in Hunter (Hunter.sol#656-1324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

DividendTracker._name (Hunter.sol#1329) should be constant
DividendTracker._symbol (Hunter.sol#1330) should be constant
DividendTracker.lastProcessedIndex (Hunter.sol#1332) should be constant
Hunter.DEAD (Hunter.sol#658) should be constant
Hunter.UNISWAPROUTER (Hunter.sol#657) should be constant
Hunter.ZERO (Hunter.sol#659) should be constant
Hunter._name (Hunter.sol#661) should be constant
Hunter._symbol (Hunter.sol#662) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```



```
DividendTracker.UNISWAPROUTER (Hunter.sol#1327) should be immutable
DividendTracker.tokenAddress (Hunter.sol#1343) should be immutable
ERC20._name (Hunter.sol#100) should be immutable
ERC20._symbol (Hunter.sol#101) should be immutable
Hunter.dividendTracker (Hunter.sol#707) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Hunter.sol analyzed (12 contracts with 84 detectors), 131 result(s) found
```

Solidity Static Analysis

Hunter.sol



Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hunter.(address,address,address[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1336:4:

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: 1645:12:

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

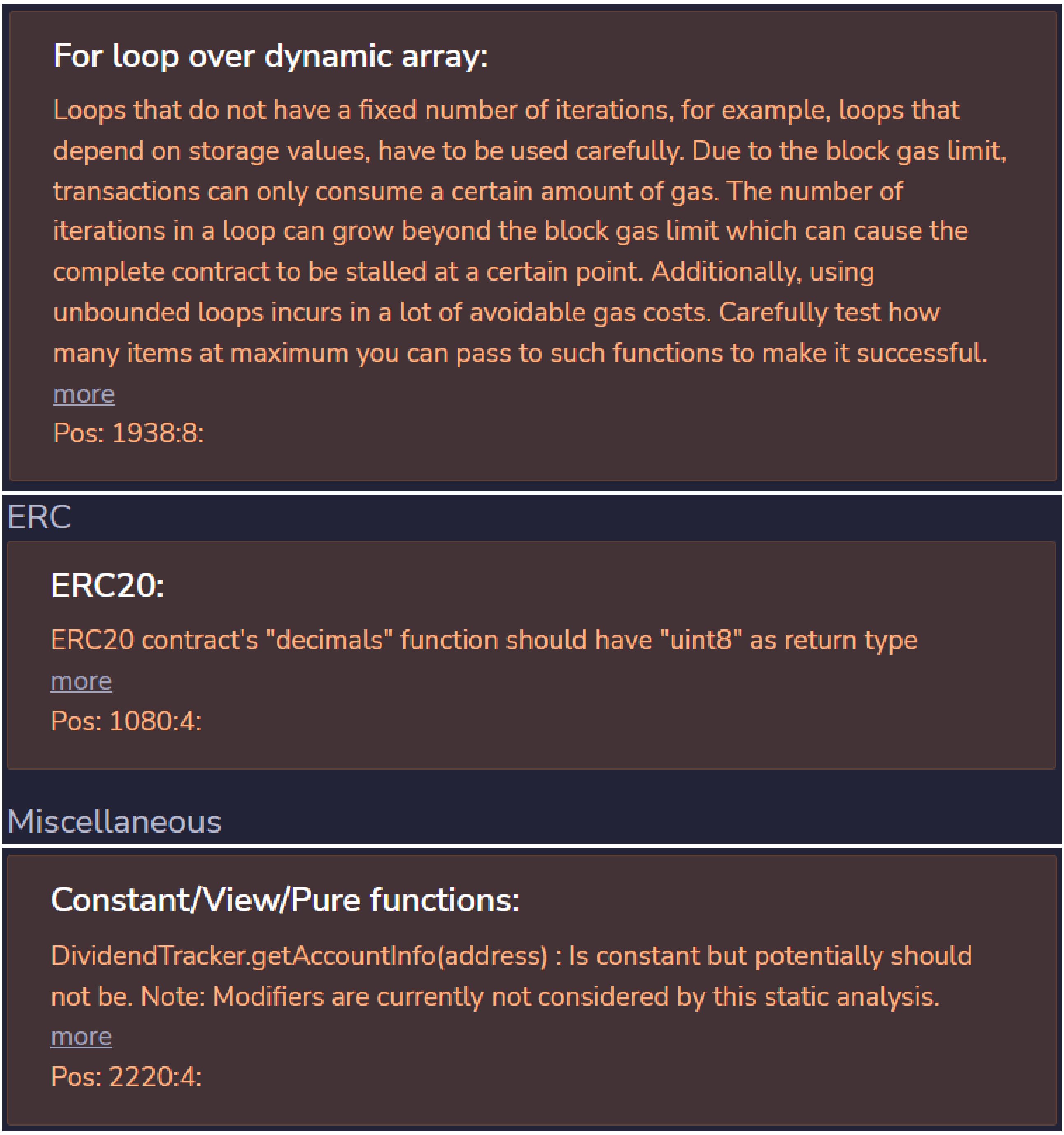
Pos: 2119:31:

Gas & Economy

Gas costs:

Gas requirement of function Hunter.transferFrom 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: 2286:4:



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: 1938:8:

ERC

ERC20:

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

more

Pos: 1080:4:

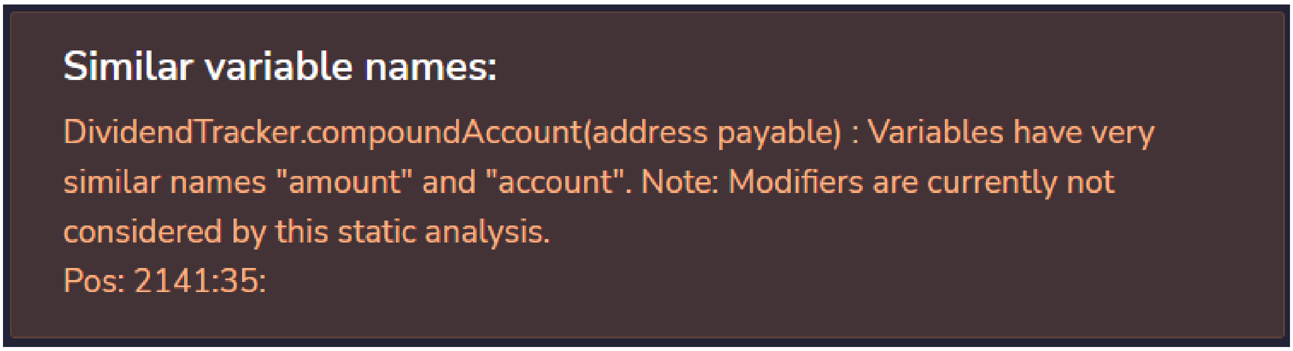
Miscellaneous

Constant/View/Pure functions:

DividendTracker.getAccountInfo(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

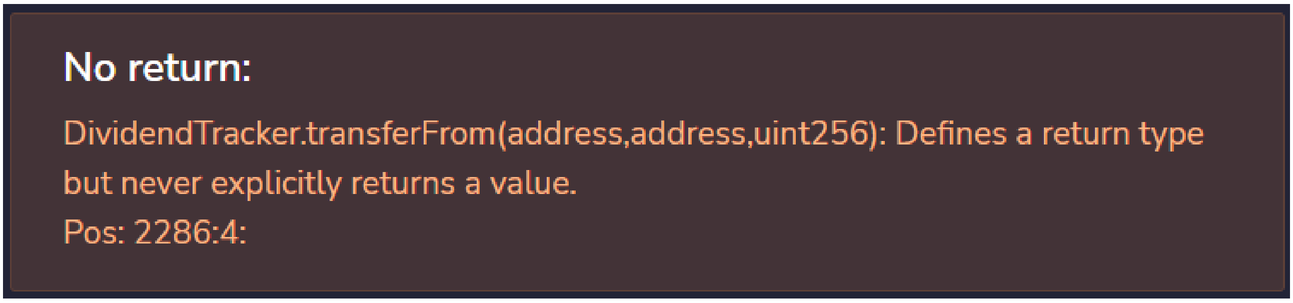
more

Pos: 2220:4:



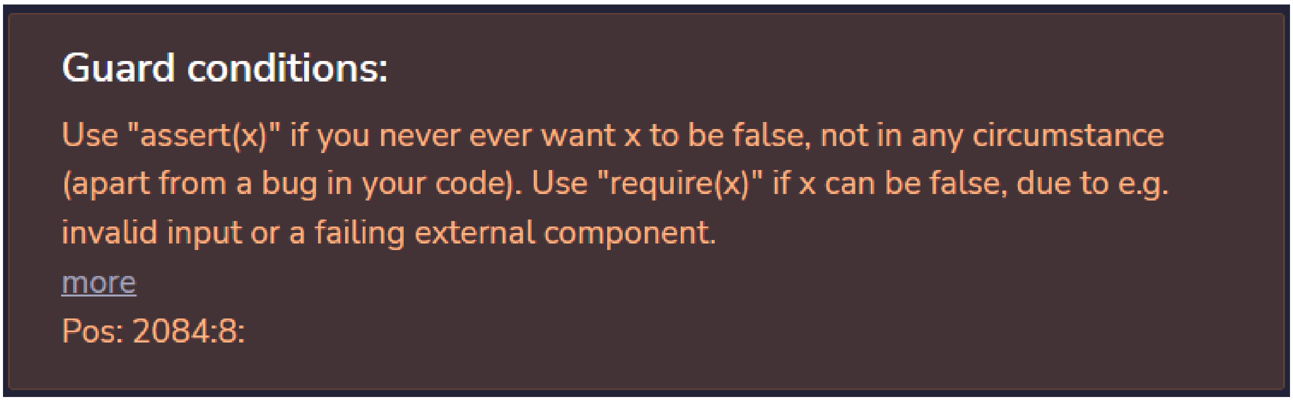
Similar variable names:

DividendTracker.compoundAccount(address payable) : Variables have very similar names "amount" and "account". Note: Modifiers are currently not considered by this static analysis.
Pos: 2141:35:



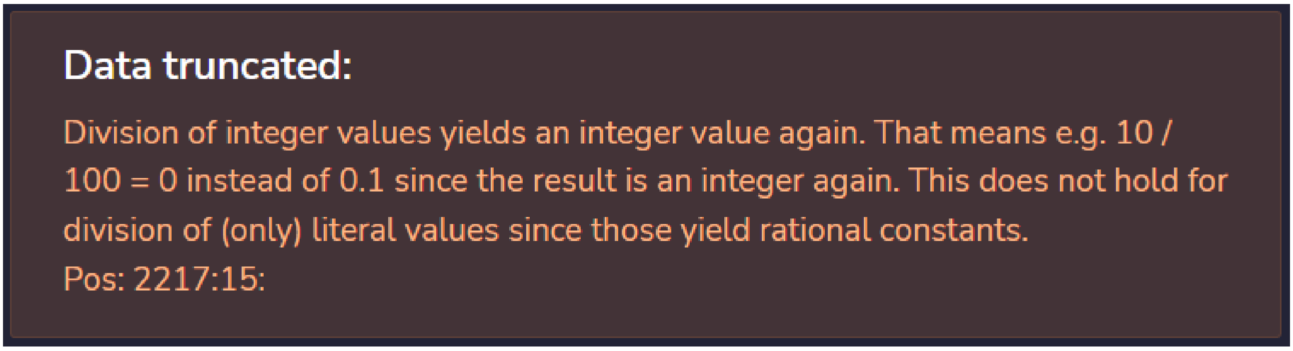
No return:

DividendTracker.transferFrom(address,address,uint256): Defines a return type but never explicitly returns a value.
Pos: 2286:4:



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: 2084:8:

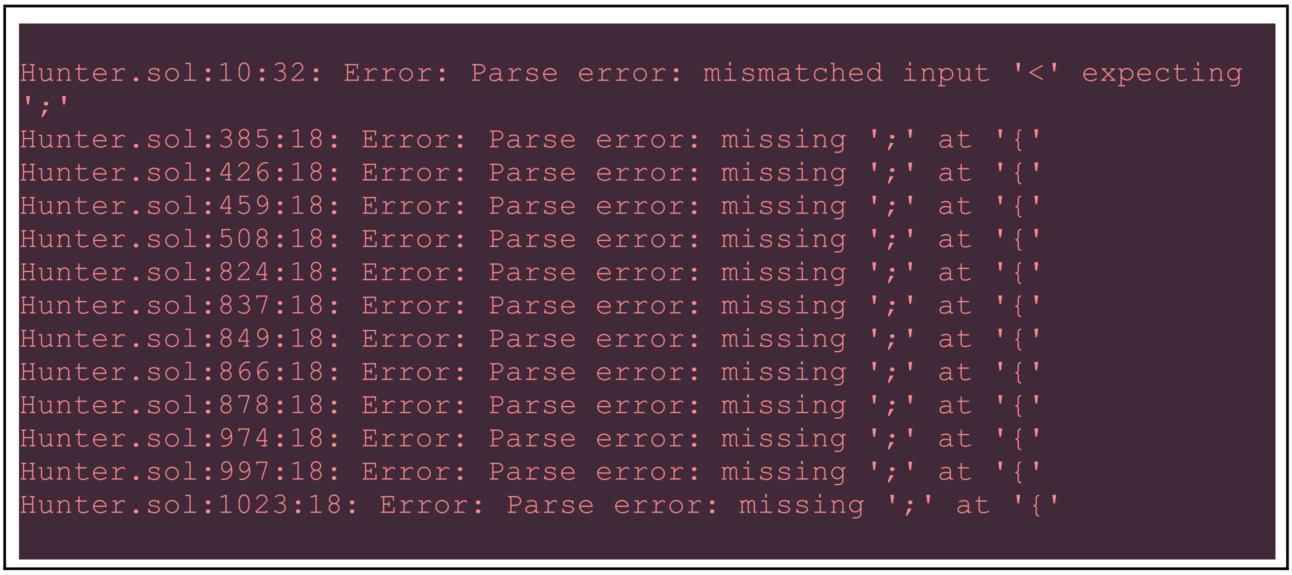


Data truncated:

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: 2217:15:

Solhint Linter

Hunter.sol



```
Hunter.sol:10:32: Error: Parse error: mismatched input '<' expecting
';'
Hunter.sol:385:18: Error: Parse error: missing ';' at '{'
Hunter.sol:426:18: Error: Parse error: missing ';' at '{'
Hunter.sol:459:18: Error: Parse error: missing ';' at '{'
Hunter.sol:508:18: Error: Parse error: missing ';' at '{'
Hunter.sol:824:18: Error: Parse error: missing ';' at '{'
Hunter.sol:837:18: Error: Parse error: missing ';' at '{'
Hunter.sol:849:18: Error: Parse error: missing ';' at '{'
Hunter.sol:866:18: Error: Parse error: missing ';' at '{'
Hunter.sol:878:18: Error: Parse error: missing ';' at '{'
Hunter.sol:974:18: Error: Parse error: missing ';' at '{'
Hunter.sol:997:18: Error: Parse error: missing ';' at '{'
Hunter.sol:1023:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

EtherAuthority