EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	LiveCGI MarketPlace
Platform:	Binance Smart Chain
Language:	Solidity
Date:	August 24th, 2022

Table of contents

Introduction

EtherAuthority was contracted by LiveCGI MarketPlace to perform the Security audit of the LiveCGI MarketPlace smart contracts 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 August 24th, 2022.

The purpose of this audit was to address the following:

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

Project Background

- LiveCGI MarketPlace is a smart contract which has functions like initialize, cancel, transfer, simpleMatch, validate, transfer, subFee, subFeeInBp, etc.
- The LiveCGI MarketPlace contract inherits the Initializable, OwnableUpgradeable, AddressUpgradeable, ContextUpgradeable, IERC721Upgradeable, draft-EIP712Upgradeable, IERC20Upgradeable, IERC721Upgradeable, IERC1155Upgradeable, IERC2981. standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

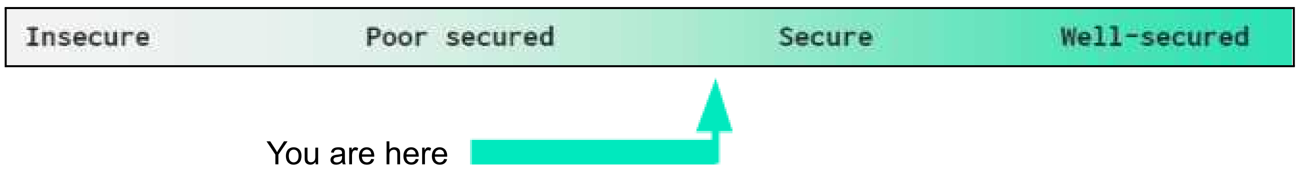
Name	Code Review and Security Analysis Report for LiveCGI MarketPlace Protocol Smart Contracts
Platform	**BSC / Solidity**
File 1	ExchangeV2.sol
File 1 MD5 Hash	A4BAFAEED2D87820A9FC53644C1D6A09
File 2	AssetMatcher.sol
File 2 MD5 Hash	1008E88646064CC83CBA373E536BCFCE
File 3	.ExchangeV2Core.sol
File 3 MD5 Hash	2CCE0C3B7B5818F67C647CC4111DD37F
File 4	OrderValidator.sol
File 4 MD5 Hash	249D2E04B2FCC85A7564C04506713E48
File 5	TransferExecutor.sol
File 5 MD5 Hash	F300524C66950449E341074525DF0059
File 6	RaribleTransferManager.sol
File 6 MD5 Hash	7787602FC51FF211E0DED57B7AB7BC01
File 7	ERC20Token.sol
File 7 MD5 Hash	EO3J524C66950449E341074525DF8745
File 8	PaymentSafe.sol
File 8 MD5 Hash	EAPN602FC51FF211E0DED57B7AB739547
Audit Date	August 24th,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 ExchangeV2.sol ● ExchangeV2 can initialize protocol fee, default fee receiver address, new contract owner address, owner fee.	**YES, This is valid.**
File 2 AssetMatcher.sol ● AssetMatcher can set asset matcher address,	**YES, This is valid.**
File 3 ExchangeV2Core.sol ● ExchangeV2Core owner can cancel order maker. ● ExchangeV2Core can generate sellOrder and buyOrder from parameters and call validateAndMatch() for purchase and bid transactions.	**YES, This is valid.**
File 4 OrderValidator.sol ● OrderValidator has functions like: validate.	**YES, This is valid.**
File 5 TransferExecutor.sol ● TransferExecutor has functions like: transfer.	**YES, This is valid.**
File 6 RaribleTransferManager.sol ● RaribleTransferManager owner can set owner address and fee, protocol fee, default Fee receiver address.	**YES, This is valid.**
File 7 ERC20Token.sol ● Token Name: USDC ● Token Symbol: USDC ● Total Supply: Unlimited minting by any users	**YES, This is valid.**
File 8 PaymentSafe.sol ● Owner can pay royalties	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Not Secured”**. Also, these contracts do contain owner control, which does not make them 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 1 critical, 1 high, 0 medium and 3 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	Moderated
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	Not Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Not Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: NOT PASSED

Code Quality

This audit scope has 6 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in LiveCGI MarketPlace 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 LiveCGI MarketPlace.

The LiveCGI MarketPlace team has provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

All code parts are well commented on smart contracts.

Documentation

We were given a LiveCGI MarketPlace smart contract code in the form of a Github weblink. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented. And 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.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts 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

ExchangeV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	external	Passed	No Issue
3	getProtocolFee	internal	Passed	No Issue
4	cancel	external	Passed	No Issue
5	directPurchase	external	Passed	No Issue
6	directAcceptBid	external	Passed	No Issue
7	matchOrders	external	Passed	No Issue
8	validateOrders	internal	Passed	No Issue
9	matchAndTransfer	internal	Passed	No Issue
10	getMaxFee	internal	Passed	No Issue
11	getDealData	internal	Passed	No Issue
12	getSumFees	internal	Passed	No Issue
13	setFillEmitMatch	internal	Passed	No Issue
14	getOrderFill	internal	Passed	No Issue
15	matchAssets	internal	Passed	No Issue
16	validateFull	internal	Passed	No Issue
17	getProtocolFee	internal	Passed	No Issue
18	getProtocolFeeConditional	internal	Passed	No Issue
19	__Ownable_init	internal	access only Initializing	No Issue
20	__Ownable_init_unchained	internal	access only Initializing	No Issue
21	onlyOwner	modifier	Passed	No Issue
22	owner	read	Passed	No Issue
23	_checkOwner	internal	Passed	No Issue
24	renounceOwnership	write	access only Owner	No Issue
25	transferOwnership	write	access only Owner	No Issue
26	_transferOwnership	internal	Passed	No Issue
27	__RaribleTransferManager_init_unchained	internal	access only Initializing	No Issue
28	setContractOwnerAndFee	external	access only Owner	No Issue
29	setProtocolFee	external	access only Owner	No Issue
30	setDefaultFeeReceiver	external	access only Owner	No Issue
31	setFeeReceiver	external	access only Owner	No Issue
32	getFeeReceiver	internal	Passed	No Issue
33	doTransfers	internal	Passed	No Issue
34	doTransfersWithFees	internal	Passed	No Issue
35	transferProtocolFee	internal	Passed	No Issue
36	transferRoyalties	internal	Passed	No Issue
37	getRoyaltiesByAssetType	internal	Passed	No Issue

38	getRoyalties	internal	Passed	No Issue
39	transferFees	internal	Passed	No Issue
40	transferPayouts	internal	Passed	No Issue
41	calculateTotalAmount	internal	Passed	No Issue
42	subFeeInBp	internal	Passed	No Issue
43	subFee	internal	Passed	No Issue

AssetMatcher.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	setAssetMatcher	external	access only Owner	No Issue
11	matchAssets	internal	Passed	No Issue
12	matchAssetOneSide	read	Passed	No Issue
13	simpleMatch	write	Passed	No Issue

ExchangeV2Core.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	cancel	external	Passed	No Issue
3	directPurchase	external	Passed	No Issue
4	directAcceptBid	external	Passed	No Issue
5	matchOrders	external	Passed	No Issue
6	validateOrders	internal	Passed	No Issue
7	matchAndTransfer	internal	Passed	No Issue
8	getMaxFee	internal	Passed	No Issue
9	getDealData	internal	Passed	No Issue
10	getSumFees	internal	Passed	No Issue
11	setFillEmitMatch	internal	Passed	No Issue
12	getOrderFill	internal	Passed	No Issue
13	matchAssets	internal	Passed	No Issue
14	validateFull	internal	Passed	No Issue
15	getProtocolFee	internal	Passed	No Issue

16	getProtocolFeeConditional	internal	Passed	No Issue
17	__Ownable_init	internal	access only Initializing	No Issue
18	__Ownable_init_unchained	internal	access only Initializing	No Issue
19	onlyOwner	modifier	Passed	No Issue
20	owner	read	Passed	No Issue
21	_checkOwner	internal	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue
24	_transferOwnership	internal	Passed	No Issue
25	setAssetMatcher	external	access only Owner	No Issue
26	matchAssets	internal	Passed	No Issue
27	matchAssetOneSide	read	Passed	No Issue
28	simpleMatch	write	Passed	No Issue
29	transfer	internal	Passed	No Issue
30	__Context_init	internal	access only Initializing	No Issue
31	__Context_init_unchained	internal	access only Initializing	No Issue
32	_msgSender	internal	Passed	No Issue
33	_msgData	internal	Passed	No Issue
34	__OrderValidator_init_unchained	internal	access only Initializing	No Issue
35	validate	internal	Passed	No Issue

OrderValidator.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Context_init	internal	access only Initializing	No Issue
3	__Context_init_unchained	internal	access only Initializing	No Issue
4	_msgSender	internal	Passed	No Issue
5	_msgData	internal	Passed	No Issue
6	__OrderValidator_init_unchained	internal	access only Initializing	No Issue
7	validate	internal	Passed	No Issue

TransferExecutor.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue

2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	transfer	internal	Passed	No Issue

RaribleTransferManager.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	__RaribleTransferManager_init_unchained	internal	access only Initializing	No Issue
11	setContractOwnerAndFee	external	access only Owner	No Issue
12	setProtocolFee	external	access only Owner	No Issue
13	setDefaultFeeReceiver	external	access only Owner	No Issue
14	setFeeReceiver	external	access only Owner	No Issue
15	getFeeReceiver	internal	Passed	No Issue
16	doTransfers	internal	Passed	No Issue
17	doTransfersWithFees	internal	Passed	No Issue
18	transferProtocolFee	internal	Passed	No Issue
19	transferRoyalties	internal	Passed	No Issue
20	getRoyaltiesByAssetType	internal	Passed	No Issue
21	getRoyalties	internal	Passed	No Issue
22	transferFees	internal	Passed	No Issue
23	transferPayouts	internal	Passed	No Issue
24	calculateTotalAmount	internal	Passed	No Issue
25	subFeeInBp	internal	Passed	No Issue
26	subFee	internal	Passed	No Issue

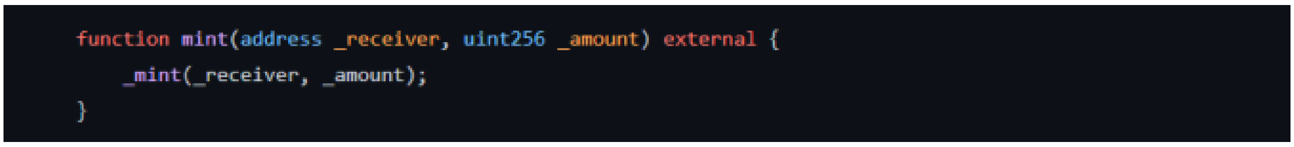
Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens loss
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

(1) Unlimited and uncontrolled token minting in ERC20Token.sol



```
function mint(address _receiver, uint256 _amount) external {
    _mint(_receiver, _amount);
}
```

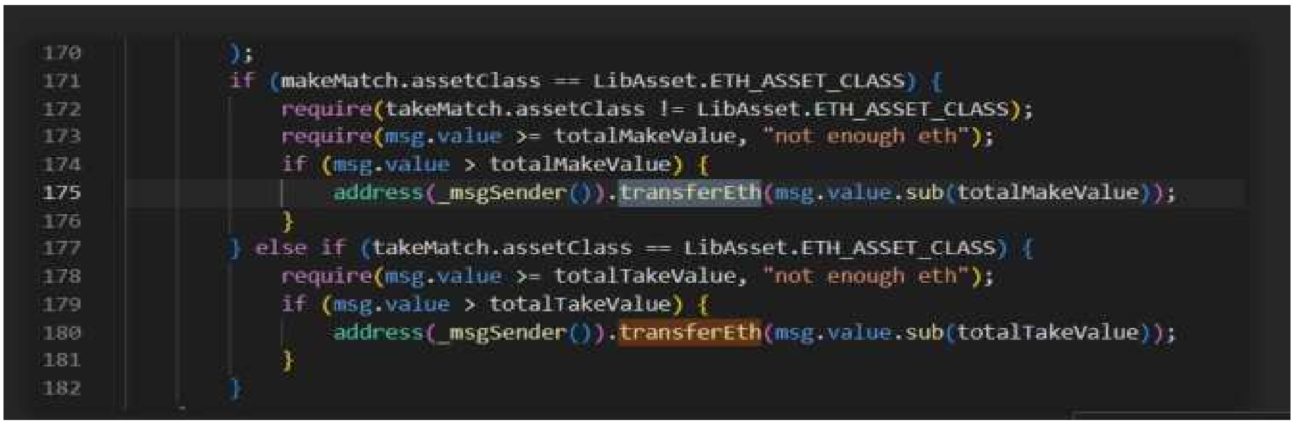
Any user can call this mint function and can mint unlimited tokens for himself. This is a major vulnerability as any user can simply inflate the value of a token by minting any amount of tokens.

Resolution: we advise making this function owner only. And setting any max minting limit. So, it can be used in harmony with your tokenomics.

High Severity

(1) Open To Reenterancy:

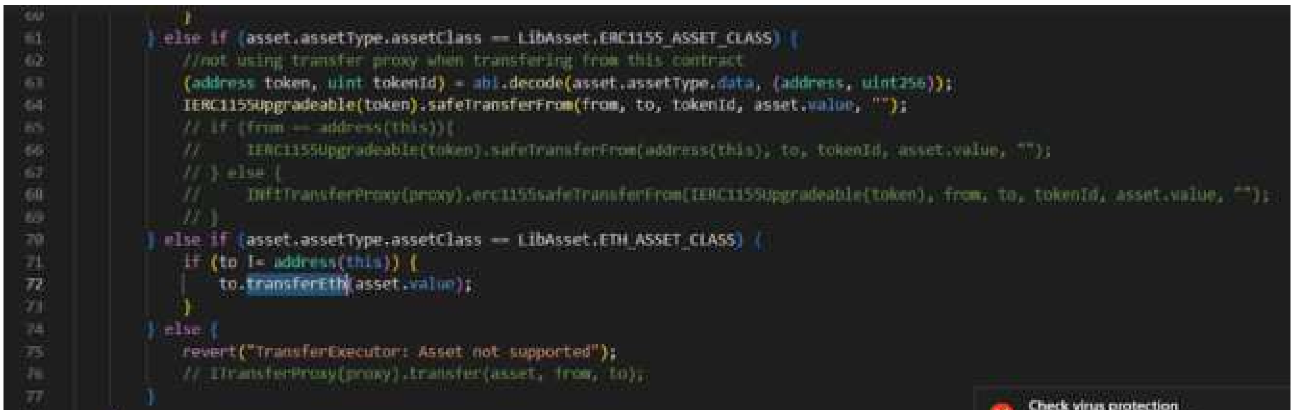
TransferExecuter.sol



```
170            );
171            if (makeMatch.assetClass == LibAsset.ETH_ASSET_CLASS) {
172                require(takeMatch.assetClass != LibAsset.ETH_ASSET_CLASS);
173                require(msg.value >= totalMakeValue, "not enough eth");
174                if (msg.value > totalMakeValue) {
175                    address(_msgSender()).transferEth(msg.value.sub(totalMakeValue));
176                }
177            } else if (takeMatch.assetClass == LibAsset.ETH_ASSET_CLASS) {
178                require(msg.value >= totalTakeValue, "not enough eth");
179                if (msg.value > totalTakeValue) {
180                    address(_msgSender()).transferEth(msg.value.sub(totalTakeValue));
181                }
182            }
```

A ".call" is used to transfer coins without gas limit which is being called via library in line no 175 & 180 in ExchangeV2Core.sol.

ExchangeV2.sol



```
60            }
61        } else if (asset.assetType.assetClass == LibAsset.ERC1155_ASSET_CLASS) {
62            //not using transfer proxy when transfering from this contract
63            (address token, uint tokenId) = abi.decode(asset.assetType.data, (address, uint256));
64            IERC1155Upgradeable(token).safeTransferFrom(from, to, tokenId, asset.value, "");
65            // if (from == address(this)){
66            //     IERC1155Upgradeable(token).safeTransferFrom(address(this), to, tokenId, asset.value, "");
67            // } else {
68            //     INftTransferProxy(proxy).erc1155safeTransferFrom(IERC1155Upgradeable(token), from, to, tokenId, asset.value, "");
69            // }
70        } else if (asset.assetType.assetClass == LibAsset.ETH_ASSET_CLASS) {
71            if (to != address(this)) {
72                to.transferEth(asset.value);
73            }
74        } else {
75            revert("TransferExecutor: Asset not supported");
76            // ITransferProxy(proxy).transfer(asset, from, to);
77        }
```

".call" is used to transfer coins without gas limit which is being called via library in line no 72 in transferExecutor.sol.

Resolution: Remove it and use ".transfer" instead or apply a gas amount.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Upgradability: **ExchangeV2.sol, AssetMatcher.sol, ExchangeV2Core.sol, OrderValidator.sol, TransferExecutor.sol, RaribleTransferManager.sol**

File Description: This feature applies to the entire projects derived by the nature of the feature; at many places empty memory slots have been left to apply upgrades if required in future.

Description: These libraries and contracts are set to upgrade ECDSAUpgradeable, AddressUpgradeable, ContextUpgradeable, OwnableUpgradeable, MathUpgradeable and which are inherited by its' breaks the concept of decentralization, however, in some cases it may help to solve emergency recovery or repair of the contracts.

Resolution: No need, use or misuse depends on the contract owner, the contract user must know it before using the smart contract.

(2) Centralization: **ExchangeV2.sol, AssetMatcher.sol, ExchangeV2Core.sol, OrderValidator.sol, TransferExecutor.sol, RaribleTransferManager.sol**

File Description: This feature applies to the entire projects derived by the nature of the feature; at many places empty memory slots have been left to apply upgrades, if required in future.

Description: Some of the functions can be called only by authorized users (like: owner, etc.) which is a central control which may alter contract behavior at desire.

Resolution: No need, use or misuse depends on the contract owner, the contract user must know it before using the smart contract.

(3) Infinite loop possibility in PaymentSafe.sol

The function distributeRoyalties() uses a loop without any limit. So, if the owner adds more records in one transaction, then it might hit the block's gas limit. The owner can acknowledge this by making sure they do not input so many records in one transaction.

Very Low / Informational / Best practices:

(1) unused code blocks: **ExchangeV2.sol, AssetMatcher.sol, ExchangeV2Core.sol, OrderValidator.sol, TransferExecutor.sol, RaribleTransferManager.sol**

File Description: Every library contains many functions callable, when those open source imported into the project, they contain many code blocks, functions which nowhere used/called in the entire project for example AddressUpgradeable library, stringUpgradable library and others.

Description: Too many code blocks specially inside library nowhere used

Resolution: Can be removed to make compiled size small.

(2) Hard Coded Values: **ExchangeV2.sol, AssetMatcher.sol, ExchangeV2Core.sol, OrderValidator.sol, TransferExecutor.sol, RaribleTransferManager.sol**

File Description: Any hardCoded address, numerical values, etc. should be double checked, it is used in many places even in some libraries.

Description: Some places contain hard coded values, it may lead to lasting bugs being injected.

Resolution: Before going to production all hard coded values must double be checked for accuracy.

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:

- setContractOwnerAndFee: RaribleTransferManager owner set contract owner address and fee.
- setProtocolFee: RaribleTransferManager owner can set protocol fee.
- setDefaultFeeReceiver: RaribleTransferManager owner can set default fee receiver address.
- setFeeReceiver: RaribleTransferManager owner can set receiver fee address.
- setAssetMatcher: AssetMatcher owner can set asset matcher address.
- cancel: ExchangeV2Core owner can cancel order.

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 Github weblink. And we have used all possible tests based on given objects as files. We have observed 1 high Severity issue, 2 low Severity issue and some informational issues in smart contracts. **So, the smart contracts are ready for the mainnet deployment after fixing those issues**.

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 **"Insecure".**

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.

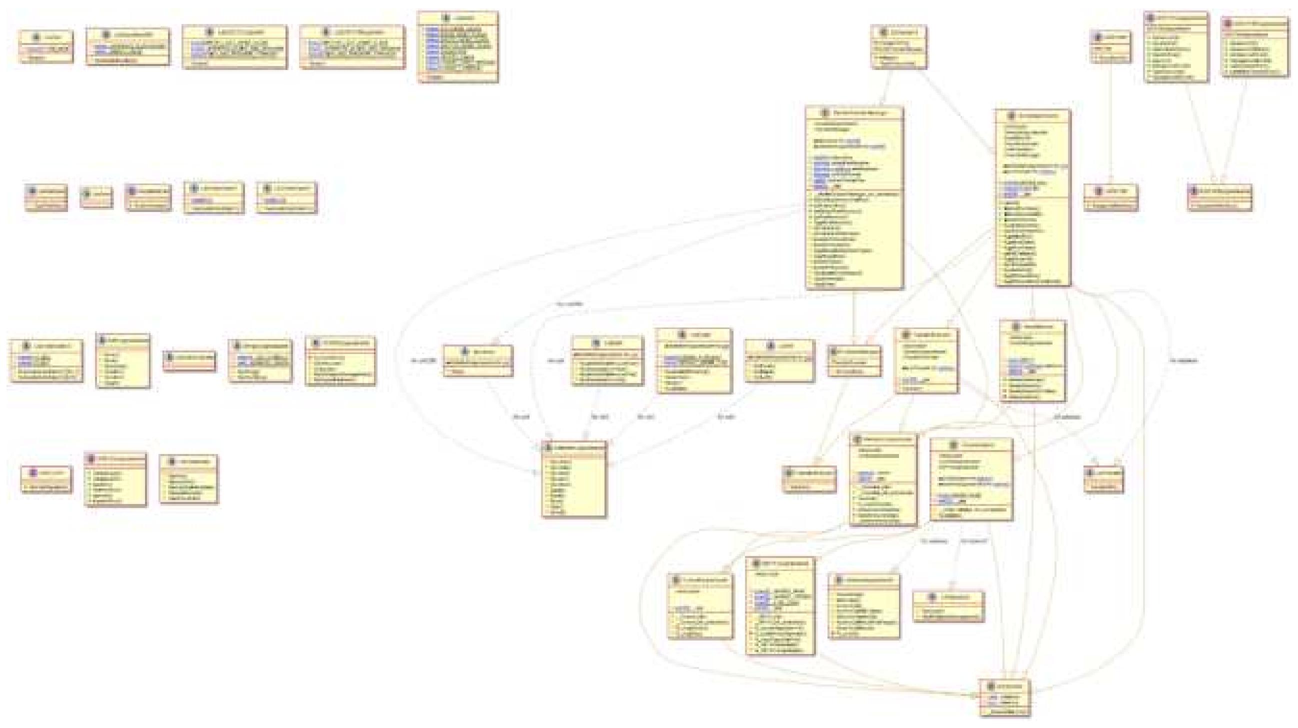
Technical Disclaimer

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.

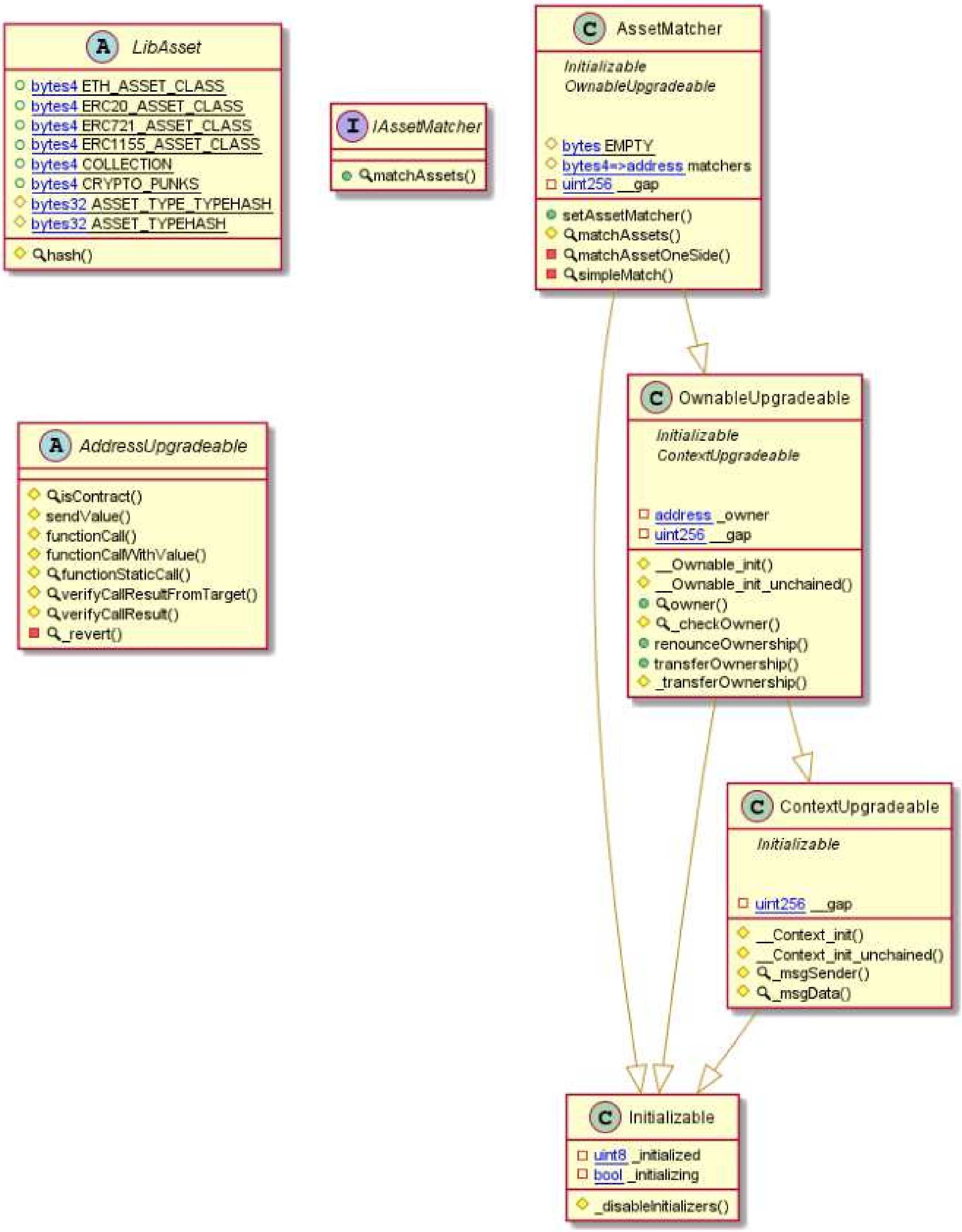
Appendix

Code Flow Diagram - LiveCGI MarketPlace Protocol

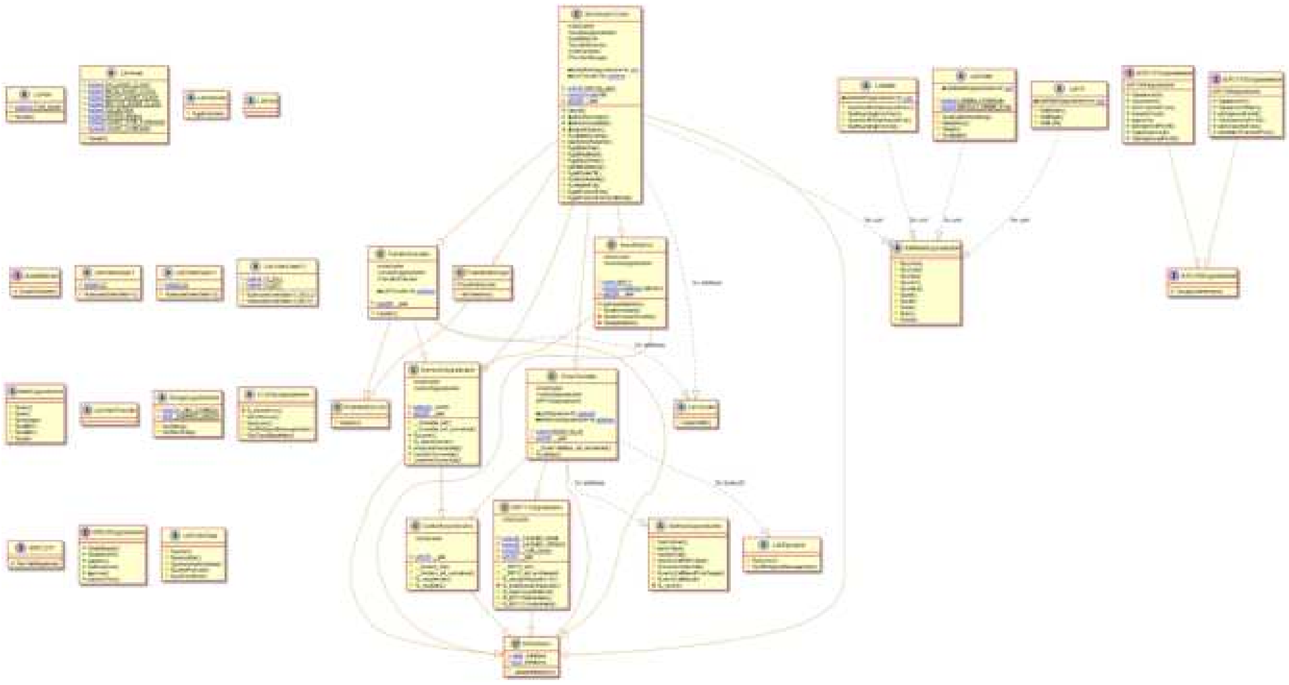
ExchangeV2 Diagram



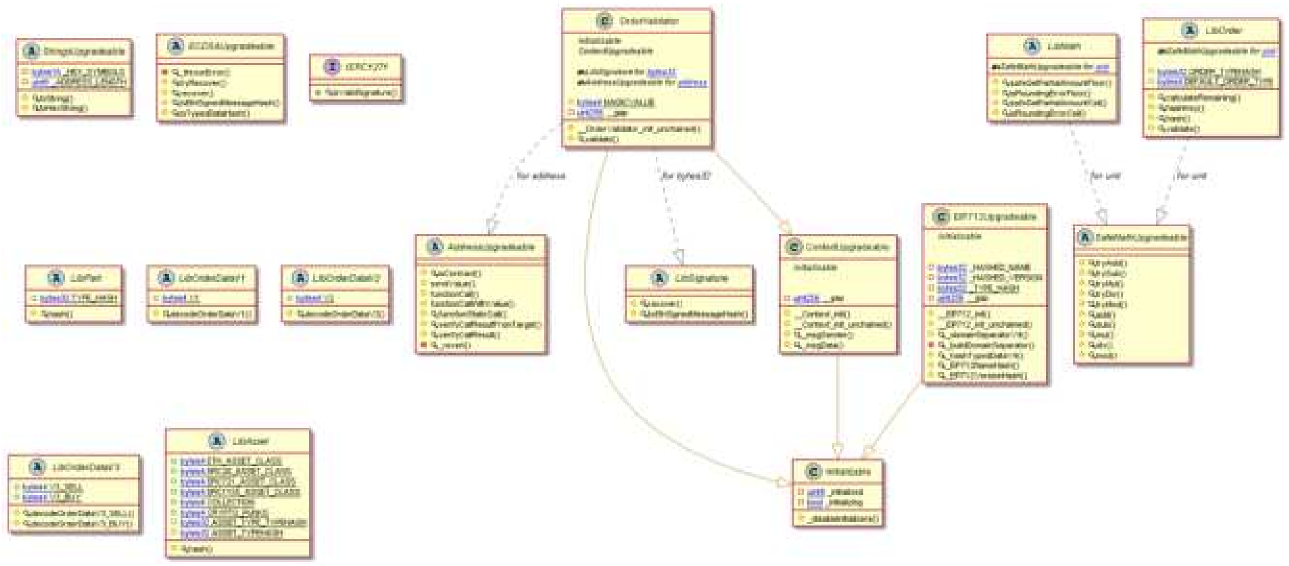
AssetMatcher Diagram



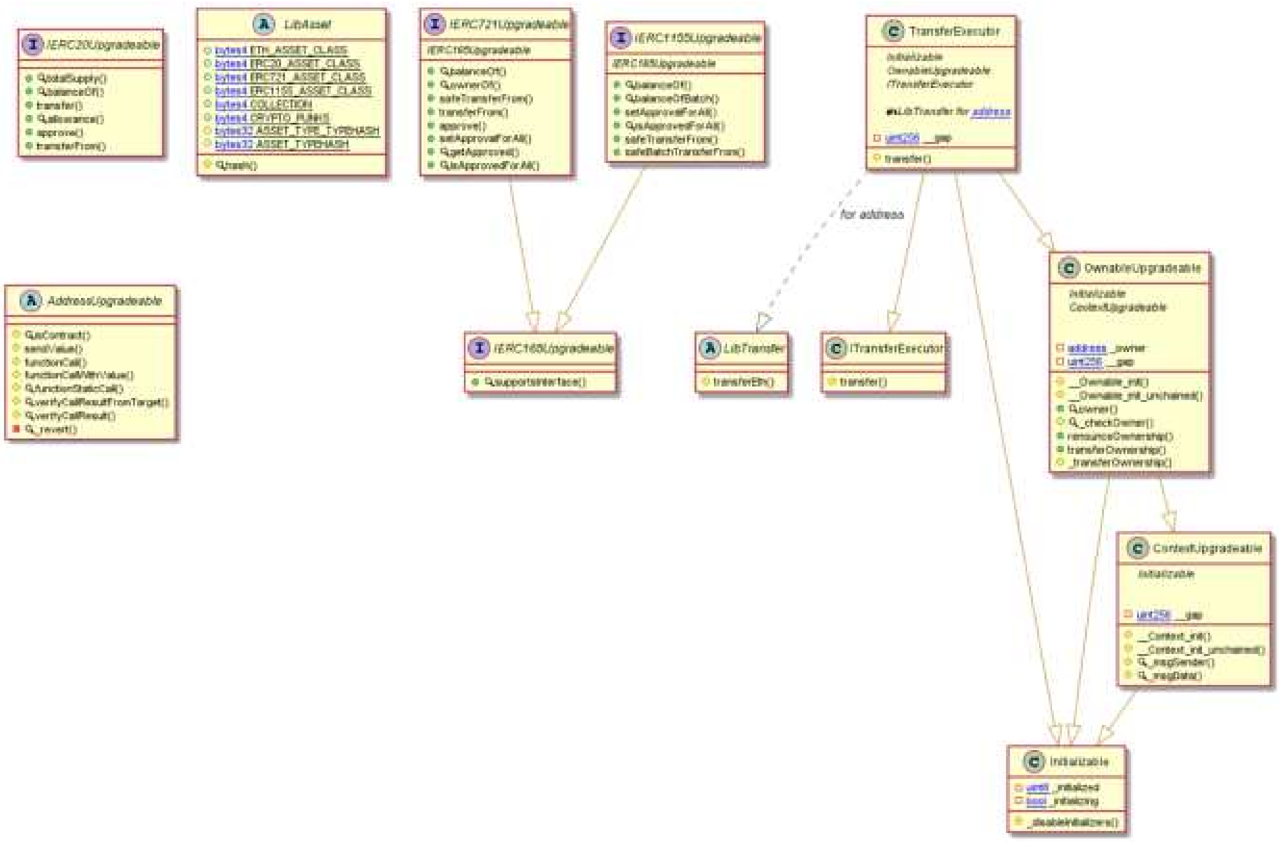
ExchangeV2Core Diagram



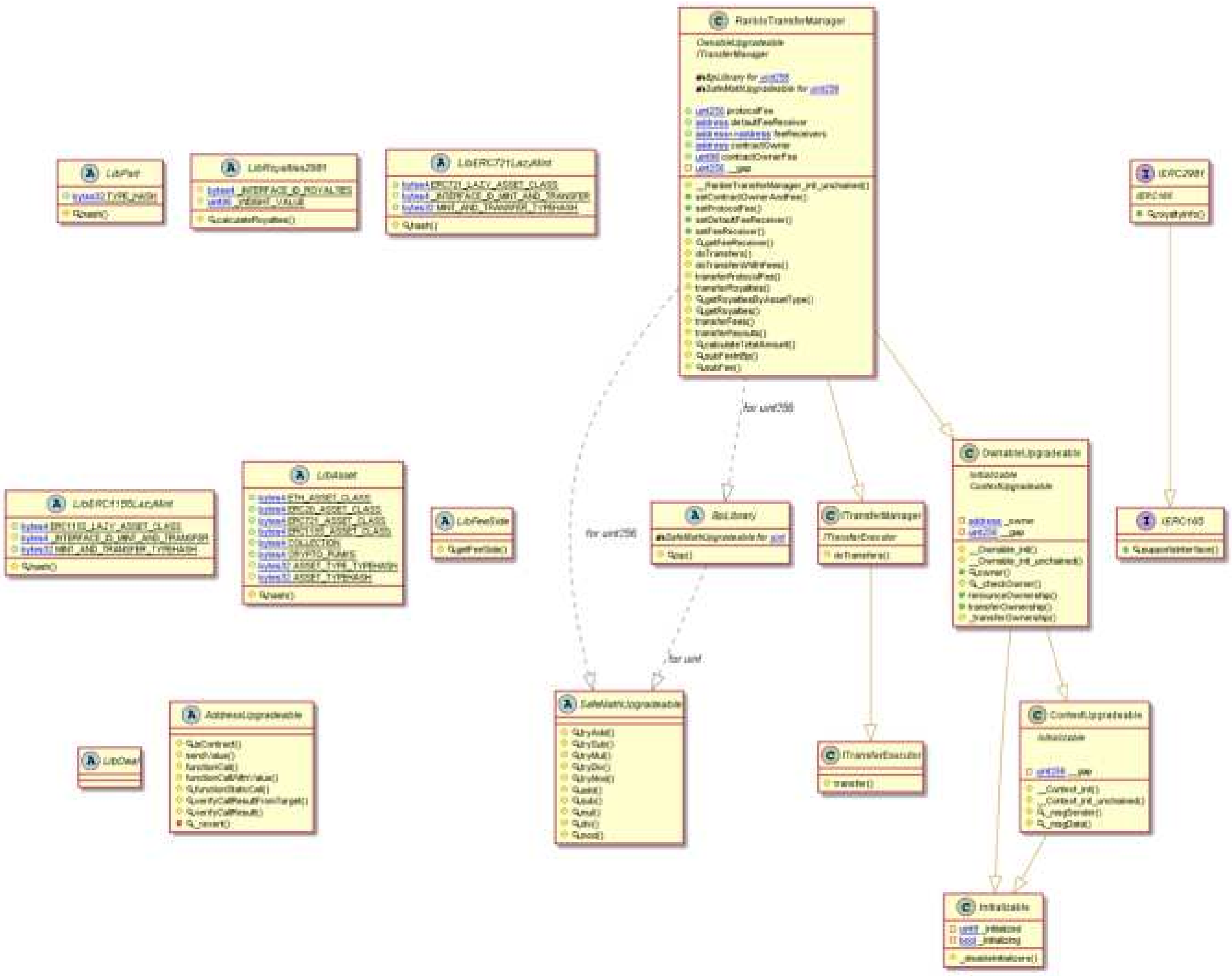
OrderValidator Diagram



TransferExecutor Diagram

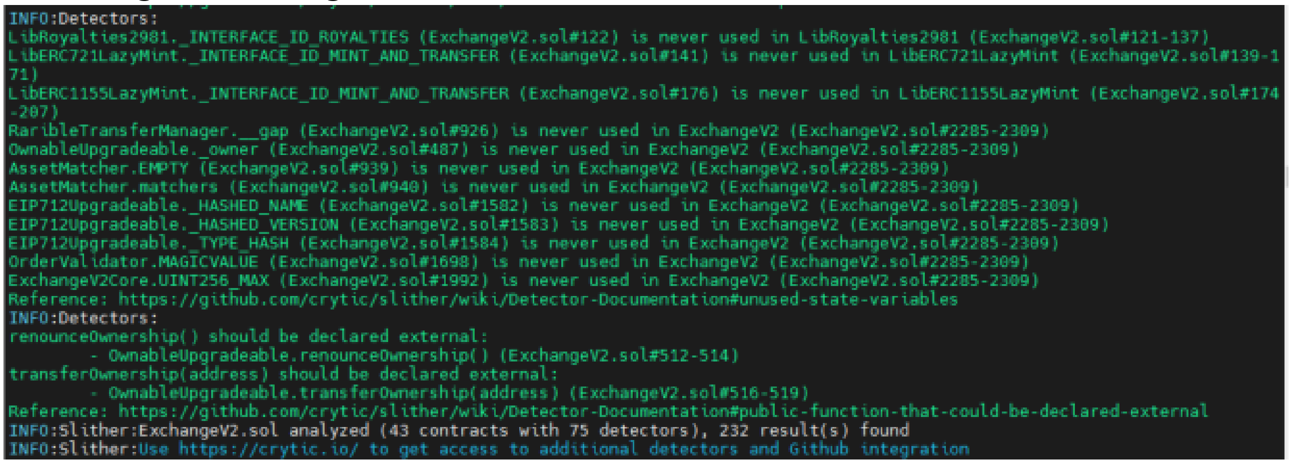


RaribleTransferManager Diagram



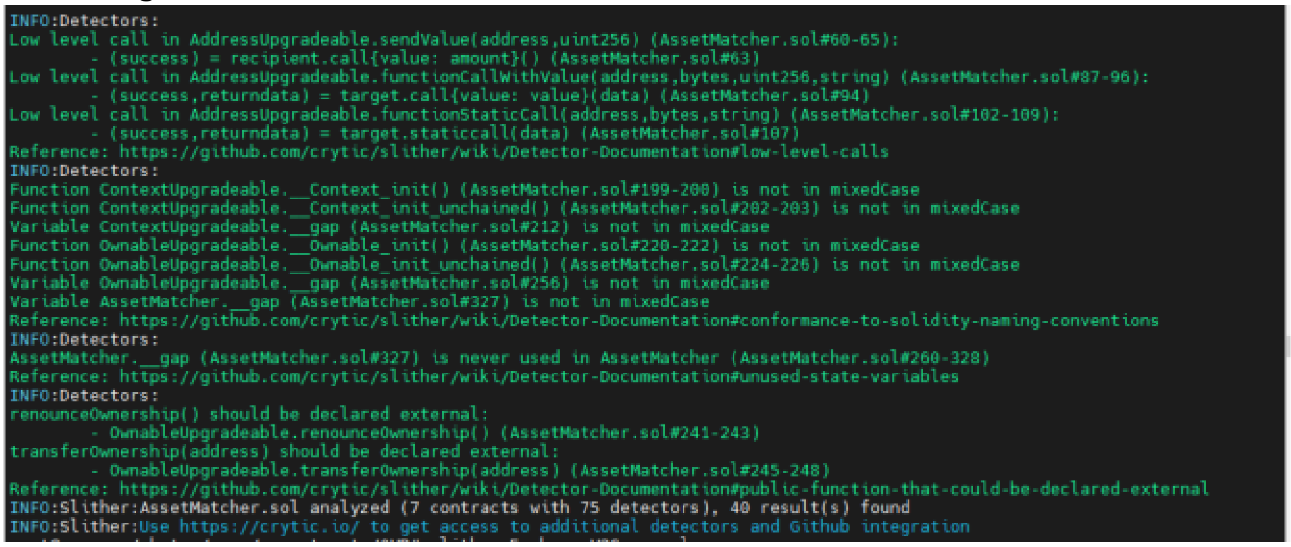
Slither Results Log

Slither log >> ExchangeV2.sol



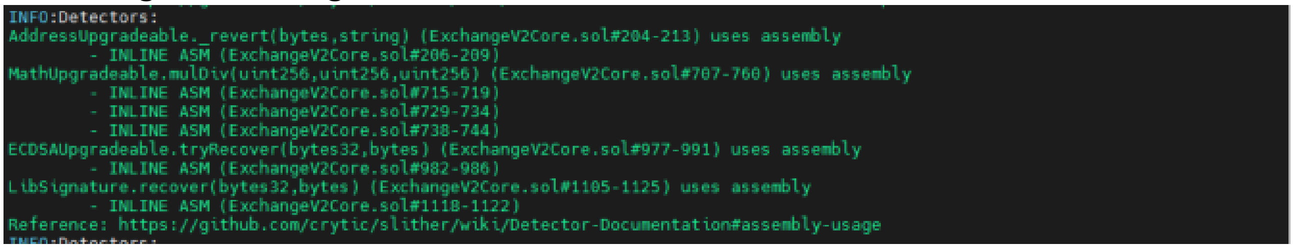
```
INFO:Detectors:
LibRoyalties2981._INTERFACE_ID_ROYALTIES (ExchangeV2.sol#122) is never used in LibRoyalties2981 (ExchangeV2.sol#121-137)
LibERC721LazyMint._INTERFACE_ID_MINT_AND_TRANSFER (ExchangeV2.sol#141) is never used in LibERC721LazyMint (ExchangeV2.sol#139-1
71)
LibERC1155LazyMint._INTERFACE_ID_MINT_AND_TRANSFER (ExchangeV2.sol#176) is never used in LibERC1155LazyMint (ExchangeV2.sol#174
-207)
RaribleTransferManager.__gap (ExchangeV2.sol#926) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
OwnableUpgradeable._owner (ExchangeV2.sol#487) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
AssetMatcher.EMPTY (ExchangeV2.sol#939) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
AssetMatcher.matchers (ExchangeV2.sol#940) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
EIP712Upgradeable._HASHED_NAME (ExchangeV2.sol#1582) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
EIP712Upgradeable._HASHED_VERSION (ExchangeV2.sol#1583) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
EIP712Upgradeable._TYPE_HASH (ExchangeV2.sol#1584) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
OrderValidator.MAGICVALUE (ExchangeV2.sol#1698) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
ExchangeV2Core.UINT256_MAX (ExchangeV2.sol#1992) is never used in ExchangeV2 (ExchangeV2.sol#2285-2309)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (ExchangeV2.sol#512-514)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (ExchangeV2.sol#516-519)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ExchangeV2.sol analyzed (43 contracts with 75 detectors), 232 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> AssetMatcher.sol

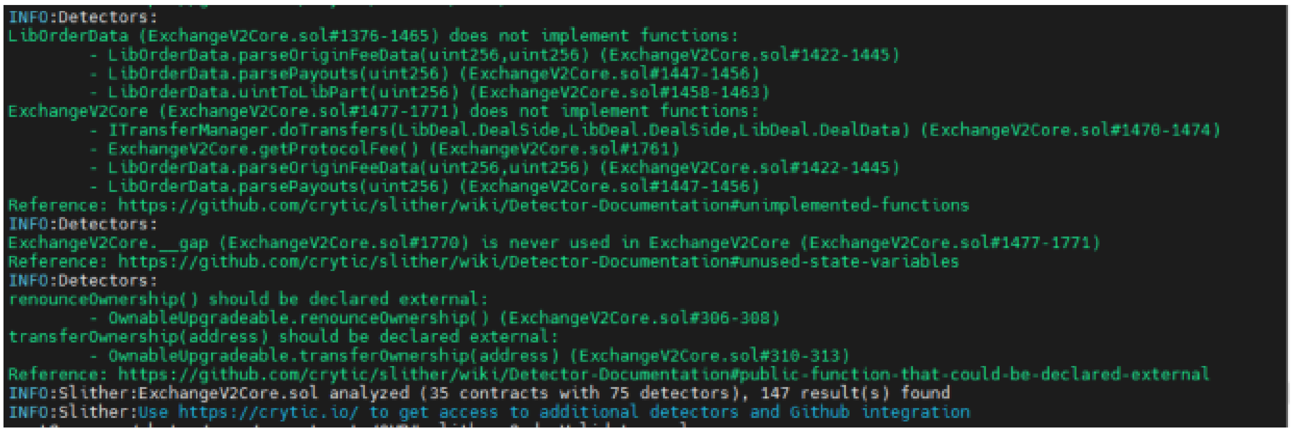


```
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (AssetMatcher.sol#60-65):
        - (success) = recipient.call{value: amount}() (AssetMatcher.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (AssetMatcher.sol#87-96):
        - (success,returndata) = target.call{value: value}(data) (AssetMatcher.sol#94)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (AssetMatcher.sol#102-109):
        - (success,returndata) = target.staticcall(data) (AssetMatcher.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (AssetMatcher.sol#199-200) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (AssetMatcher.sol#202-203) is not in mixedCase
Variable ContextUpgradeable.__gap (AssetMatcher.sol#212) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (AssetMatcher.sol#220-222) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (AssetMatcher.sol#224-226) is not in mixedCase
Variable OwnableUpgradeable.__gap (AssetMatcher.sol#256) is not in mixedCase
Variable AssetMatcher.__gap (AssetMatcher.sol#327) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
AssetMatcher.__gap (AssetMatcher.sol#327) is never used in AssetMatcher (AssetMatcher.sol#260-328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (AssetMatcher.sol#241-243)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (AssetMatcher.sol#245-248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AssetMatcher.sol analyzed (7 contracts with 75 detectors), 40 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ExchangeV2Core.sol

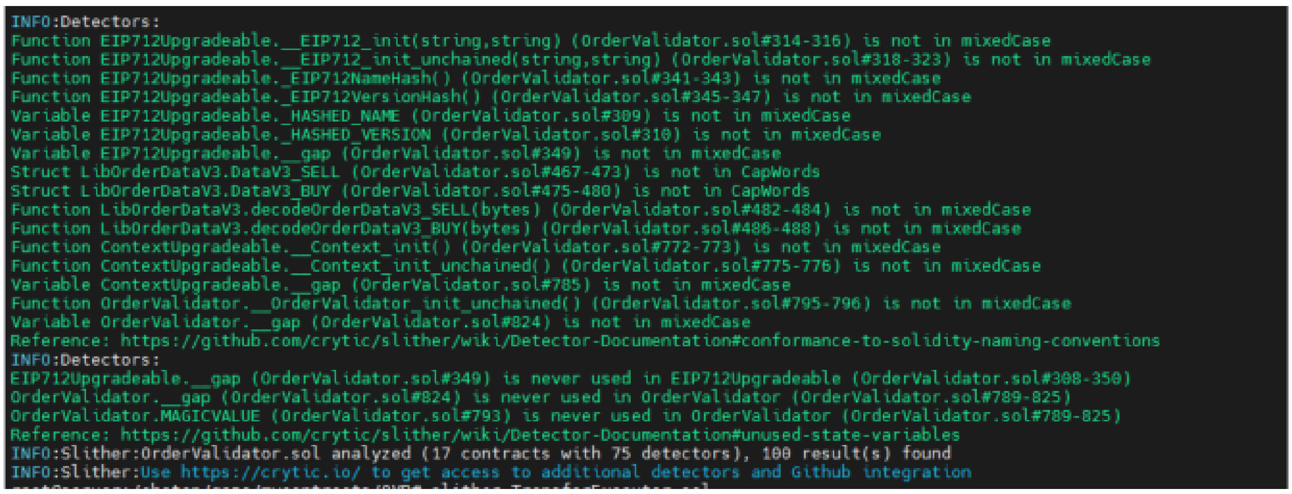


```
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (ExchangeV2Core.sol#204-213) uses assembly
        - INLINE ASM (ExchangeV2Core.sol#206-209)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (ExchangeV2Core.sol#707-760) uses assembly
        - INLINE ASM (ExchangeV2Core.sol#715-719)
        - INLINE ASM (ExchangeV2Core.sol#729-734)
        - INLINE ASM (ExchangeV2Core.sol#738-744)
ECDSAUpgradeable.tryRecover(bytes32,bytes) (ExchangeV2Core.sol#977-991) uses assembly
        - INLINE ASM (ExchangeV2Core.sol#982-986)
LibSignature.recover(bytes32,bytes) (ExchangeV2Core.sol#1105-1125) uses assembly
        - INLINE ASM (ExchangeV2Core.sol#1118-1122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



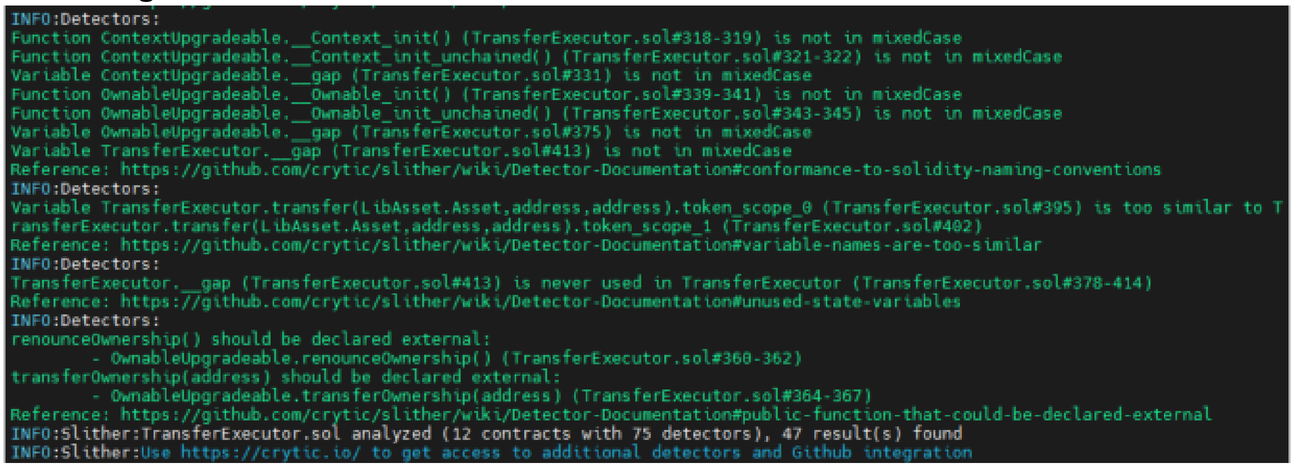
```
INFO:Detectors:
LibOrderData (ExchangeV2Core.sol#1376-1465) does not implement functions:
        - LibOrderData.parseOriginFeeData(uint256,uint256) (ExchangeV2Core.sol#1422-1445)
        - LibOrderData.parsePayouts(uint256) (ExchangeV2Core.sol#1447-1456)
        - LibOrderData.uintToLibPart(uint256) (ExchangeV2Core.sol#1458-1463)
ExchangeV2Core (ExchangeV2Core.sol#1477-1771) does not implement functions:
        - ITransferManager.doTransfers(LibDeal.DealSide,LibDeal.DealSide,LibDeal.DealData) (ExchangeV2Core.sol#1470-1474)
        - ExchangeV2Core.getProtocolFee() (ExchangeV2Core.sol#1761)
        - LibOrderData.parseOriginFeeData(uint256,uint256) (ExchangeV2Core.sol#1422-1445)
        - LibOrderData.parsePayouts(uint256) (ExchangeV2Core.sol#1447-1456)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ExchangeV2Core.__gap (ExchangeV2Core.sol#1770) is never used in ExchangeV2Core (ExchangeV2Core.sol#1477-1771)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (ExchangeV2Core.sol#306-308)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (ExchangeV2Core.sol#310-313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ExchangeV2Core.sol analyzed (35 contracts with 75 detectors), 147 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> OrderValidator.sol



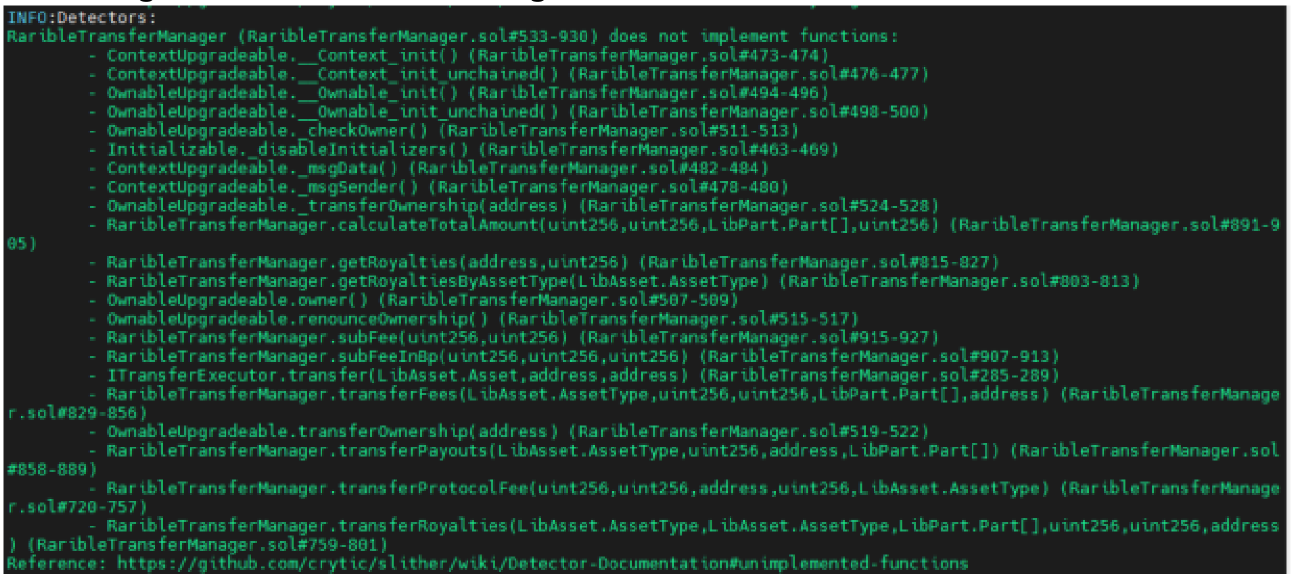
```
INFO:Detectors:
Function EIP712Upgradeable.__EIP712_init(string,string) (OrderValidator.sol#314-316) is not in mixedCase
Function EIP712Upgradeable.__EIP712_init_unchained(string,string) (OrderValidator.sol#318-323) is not in mixedCase
Function EIP712Upgradeable._EIP712NameHash() (OrderValidator.sol#341-343) is not in mixedCase
Function EIP712Upgradeable._EIP712VersionHash() (OrderValidator.sol#345-347) is not in mixedCase
Variable EIP712Upgradeable._HASHED_NAME (OrderValidator.sol#309) is not in mixedCase
Variable EIP712Upgradeable._HASHED_VERSION (OrderValidator.sol#310) is not in mixedCase
Variable EIP712Upgradeable.__gap (OrderValidator.sol#349) is not in mixedCase
Struct LibOrderDataV3.DataV3_SELL (OrderValidator.sol#467-473) is not in CapWords
Struct LibOrderDataV3.DataV3_BUY (OrderValidator.sol#475-480) is not in CapWords
Function LibOrderDataV3.decodeOrderDataV3_SELL(bytes) (OrderValidator.sol#482-484) is not in mixedCase
Function LibOrderDataV3.decodeOrderDataV3_BUY(bytes) (OrderValidator.sol#486-488) is not in mixedCase
Function ContextUpgradeable.__Context_init() (OrderValidator.sol#772-773) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (OrderValidator.sol#775-776) is not in mixedCase
Variable ContextUpgradeable.__gap (OrderValidator.sol#785) is not in mixedCase
Function OrderValidator.__OrderValidator_init_unchained() (OrderValidator.sol#795-796) is not in mixedCase
Variable OrderValidator.__gap (OrderValidator.sol#824) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
EIP712Upgradeable.__gap (OrderValidator.sol#349) is never used in EIP712Upgradeable (OrderValidator.sol#308-350)
OrderValidator.__gap (OrderValidator.sol#824) is never used in OrderValidator (OrderValidator.sol#789-825)
OrderValidator.MAGICVALUE (OrderValidator.sol#793) is never used in OrderValidator (OrderValidator.sol#789-825)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Slither:OrderValidator.sol analyzed (17 contracts with 75 detectors), 100 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> TransferExecutor.sol

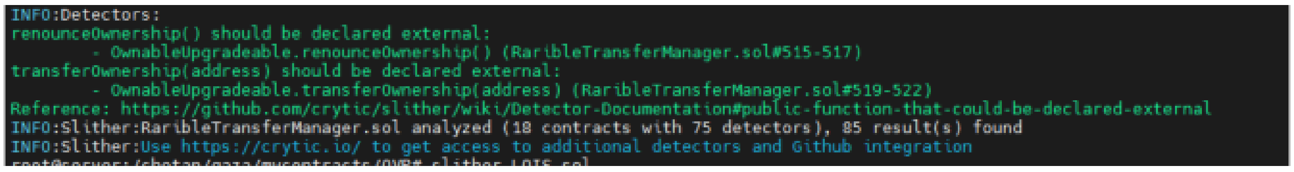


```
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (TransferExecutor.sol#318-319) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (TransferExecutor.sol#321-322) is not in mixedCase
Variable ContextUpgradeable.__gap (TransferExecutor.sol#331) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (TransferExecutor.sol#339-341) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (TransferExecutor.sol#343-345) is not in mixedCase
Variable OwnableUpgradeable.__gap (TransferExecutor.sol#375) is not in mixedCase
Variable TransferExecutor.__gap (TransferExecutor.sol#413) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable TransferExecutor.transfer(LibAsset.Asset,address,address).token_scope_0 (TransferExecutor.sol#395) is too similar to T
ransferExecutor.transfer(LibAsset.Asset,address,address).token_scope_1 (TransferExecutor.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
TransferExecutor.__gap (TransferExecutor.sol#413) is never used in TransferExecutor (TransferExecutor.sol#378-414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (TransferExecutor.sol#360-362)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (TransferExecutor.sol#364-367)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:TransferExecutor.sol analyzed (12 contracts with 75 detectors), 47 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> RaribleTransferManager.sol



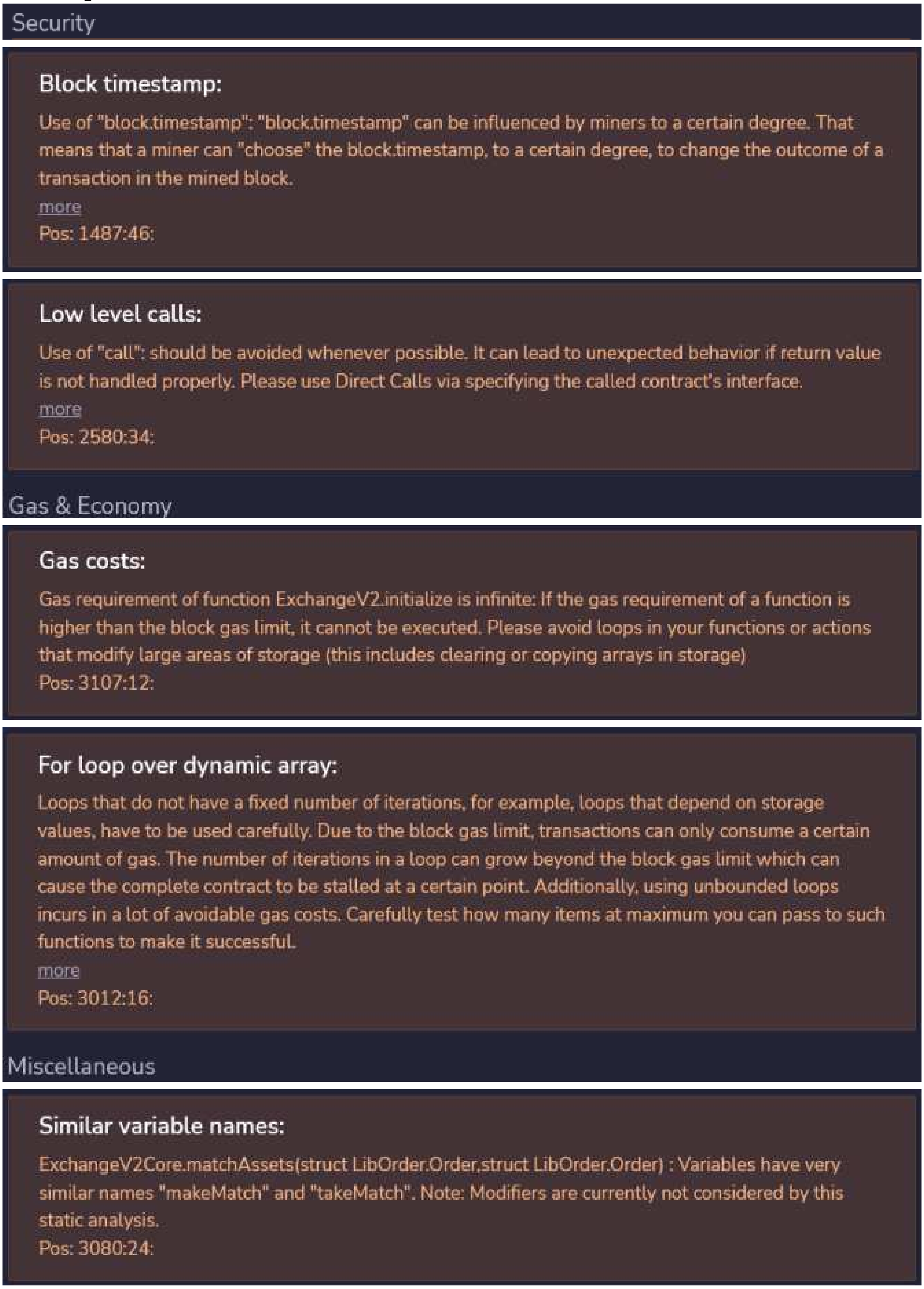
```
INFO:Detectors:
RaribleTransferManager (RaribleTransferManager.sol#533-930) does not implement functions:
        - ContextUpgradeable.__Context_init() (RaribleTransferManager.sol#473-474)
        - ContextUpgradeable.__Context_init_unchained() (RaribleTransferManager.sol#476-477)
        - OwnableUpgradeable.__Ownable_init() (RaribleTransferManager.sol#494-496)
        - OwnableUpgradeable.__Ownable_init_unchained() (RaribleTransferManager.sol#498-500)
        - OwnableUpgradeable._checkOwner() (RaribleTransferManager.sol#511-513)
        - Initializable._disableInitializers() (RaribleTransferManager.sol#463-469)
        - ContextUpgradeable._msgData() (RaribleTransferManager.sol#482-484)
        - ContextUpgradeable._msgSender() (RaribleTransferManager.sol#478-480)
        - OwnableUpgradeable._transferOwnership(address) (RaribleTransferManager.sol#524-528)
        - RaribleTransferManager.calculateTotalAmount(uint256,uint256,LibPart.Part[],uint256) (RaribleTransferManager.sol#891-9
05)
        - RaribleTransferManager.getRoyalties(address,uint256) (RaribleTransferManager.sol#815-827)
        - RaribleTransferManager.getRoyaltiesByAssetType(LibAsset.AssetType) (RaribleTransferManager.sol#803-813)
        - OwnableUpgradeable.owner() (RaribleTransferManager.sol#507-509)
        - OwnableUpgradeable.renounceOwnership() (RaribleTransferManager.sol#515-517)
        - RaribleTransferManager.subFee(uint256,uint256) (RaribleTransferManager.sol#915-927)
        - RaribleTransferManager.subFeeInBp(uint256,uint256,uint256) (RaribleTransferManager.sol#907-913)
        - ITransferExecutor.transfer(LibAsset.Asset,address,address) (RaribleTransferManager.sol#285-289)
        - RaribleTransferManager.transferFees(LibAsset.AssetType,uint256,uint256,LibPart.Part[],address) (RaribleTransferManage
r.sol#829-856)
        - OwnableUpgradeable.transferOwnership(address) (RaribleTransferManager.sol#519-522)
        - RaribleTransferManager.transferPayouts(LibAsset.AssetType,uint256,address,LibPart.Part[]) (RaribleTransferManager.sol
#858-889)
        - RaribleTransferManager.transferProtocolFee(uint256,uint256,address,uint256,LibAsset.AssetType) (RaribleTransferManage
r.sol#720-757)
        - RaribleTransferManager.transferRoyalties(LibAsset.AssetType,LibAsset.AssetType,LibPart.Part[],uint256,uint256,address
) (RaribleTransferManager.sol#759-801)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```



```
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (RaribleTransferManager.sol#515-517)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (RaribleTransferManager.sol#519-522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:RaribleTransferManager.sol analyzed (18 contracts with 75 detectors), 85 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ExchangeV2.sol



Security

Block timestamp:

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

more

Pos: 1487:46:

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: 2580:34:

Gas & Economy

Gas costs:

Gas requirement of function ExchangeV2.initialize 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: 3107:12:

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

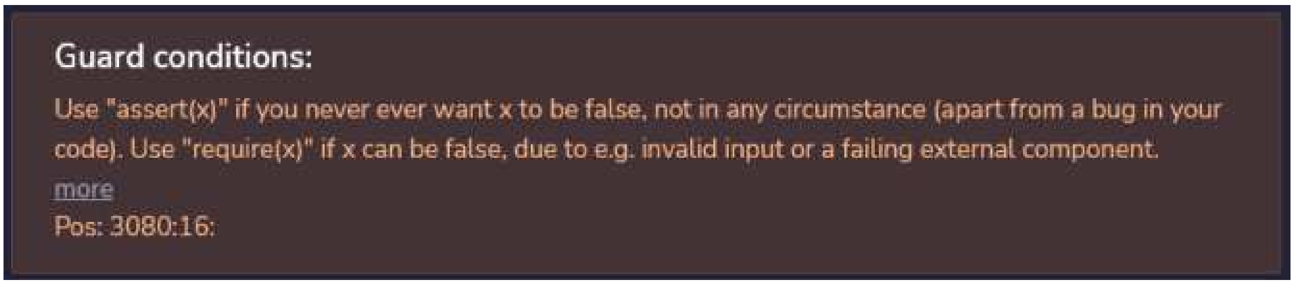
Pos: 3012:16:

Miscellaneous

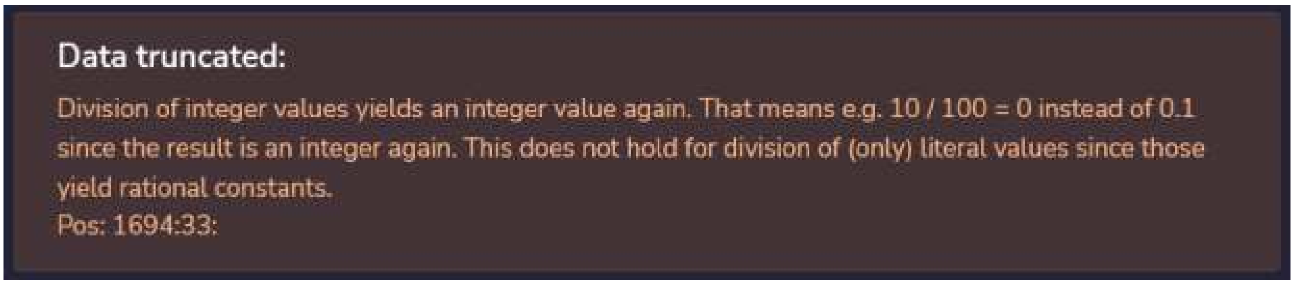
Similar variable names:

ExchangeV2Core.matchAssets(struct LibOrder.Order,struct LibOrder.Order) : Variables have very similar names "makeMatch" and "takeMatch". Note: Modifiers are currently not considered by this static analysis.

Pos: 3080:24:

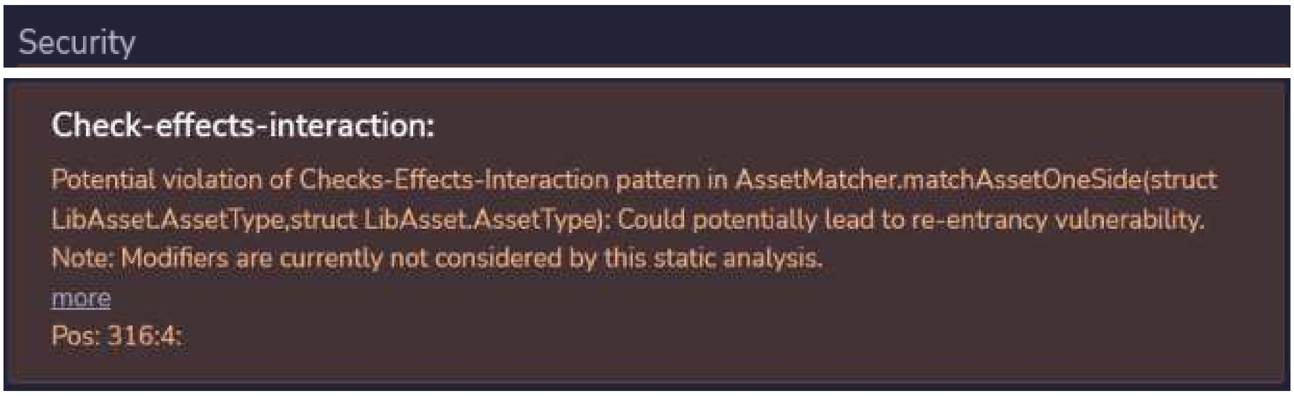

Guard conditions:

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: 3080:16:


Data truncated:

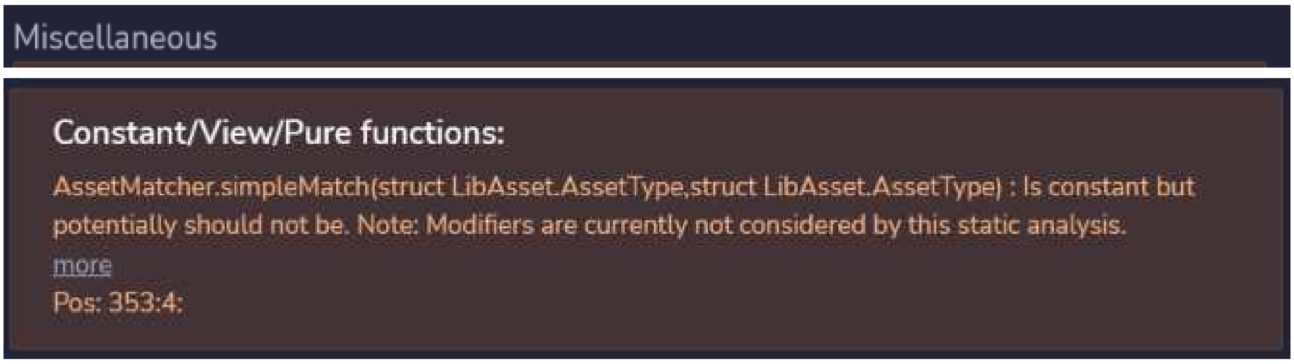
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1694:33:

AssetMatcher.sol


Security

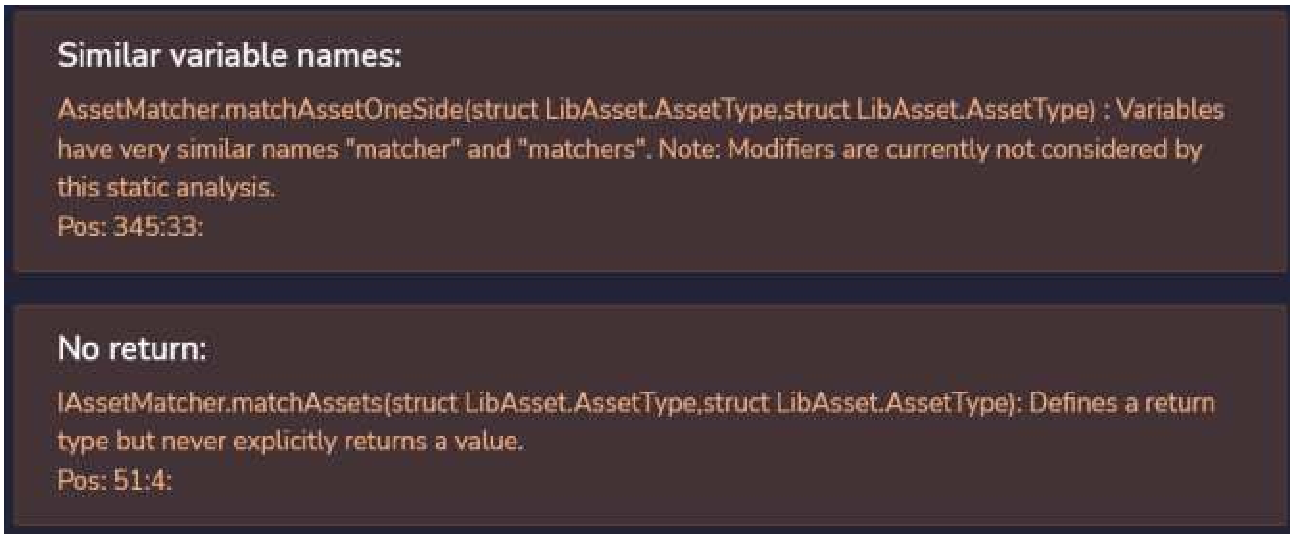
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AssetMatcher.matchAssetOneSide(struct LibAsset.AssetType,struct LibAsset.AssetType): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 316:4:


Miscellaneous

Constant/View/Pure functions:

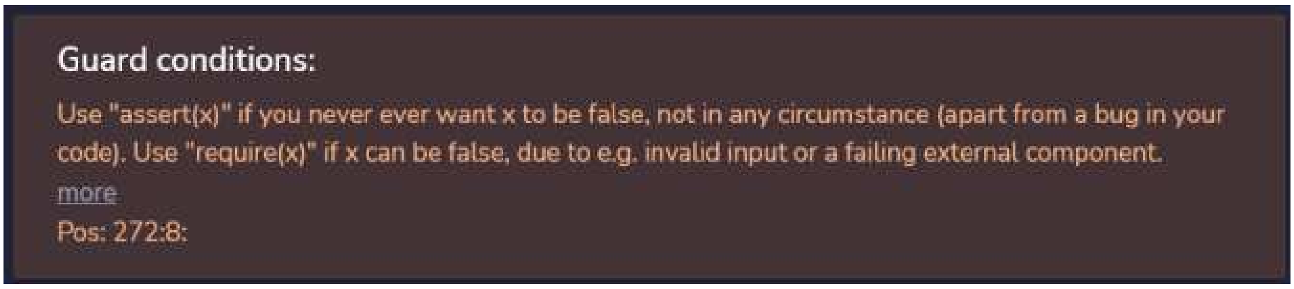
AssetMatcher.simpleMatch(struct LibAsset.AssetType,struct LibAsset.AssetType) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 353:4:


Similar variable names:

AssetMatcher.matchAssetOneSide(struct LibAsset.AssetType,struct LibAsset.AssetType) : Variables have very similar names "matcher" and "matchers". Note: Modifiers are currently not considered by this static analysis.
Pos: 345:33:

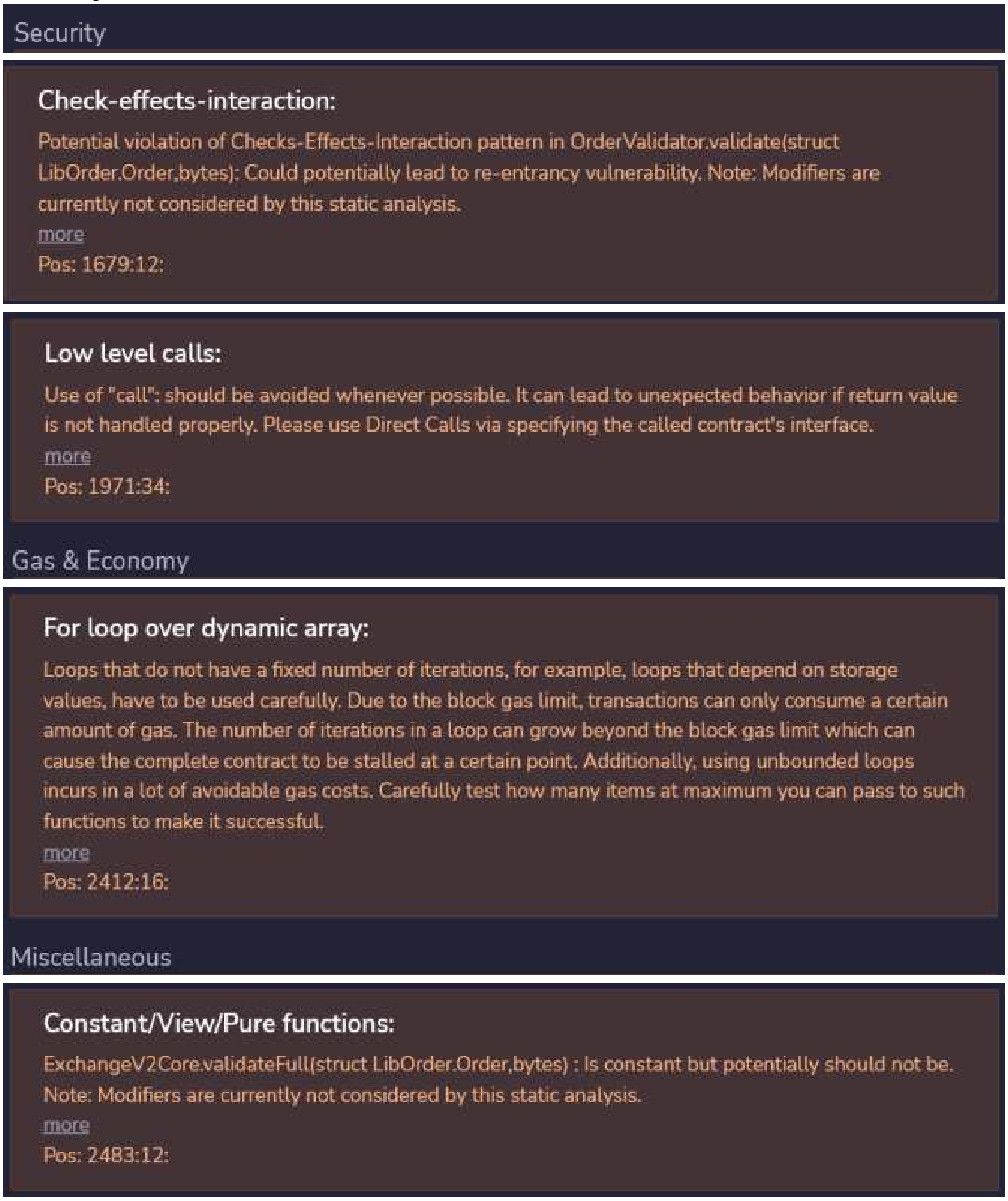
No return:

IAssetMatcher.matchAssets(struct LibAsset.AssetType,struct LibAsset.AssetType): Defines a return type but never explicitly returns a value.
Pos: 51:4:


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: 272:8:

ExchangeV2Core.sol


Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in OrderValidator.validate(struct LibOrder.Order,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1679:12:

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: 1971:34:

Gas & Economy

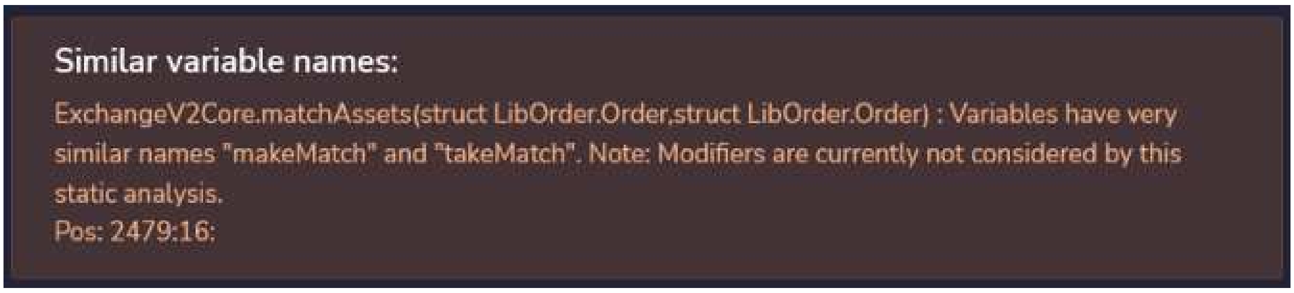
For loop over dynamic array:

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: 2412:16:

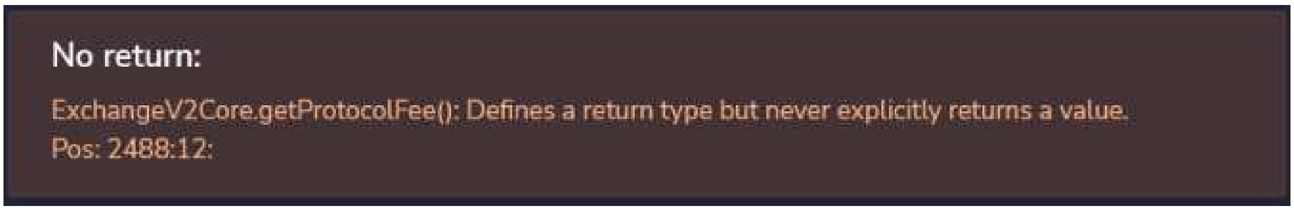
Miscellaneous

Constant/View/Pure functions:

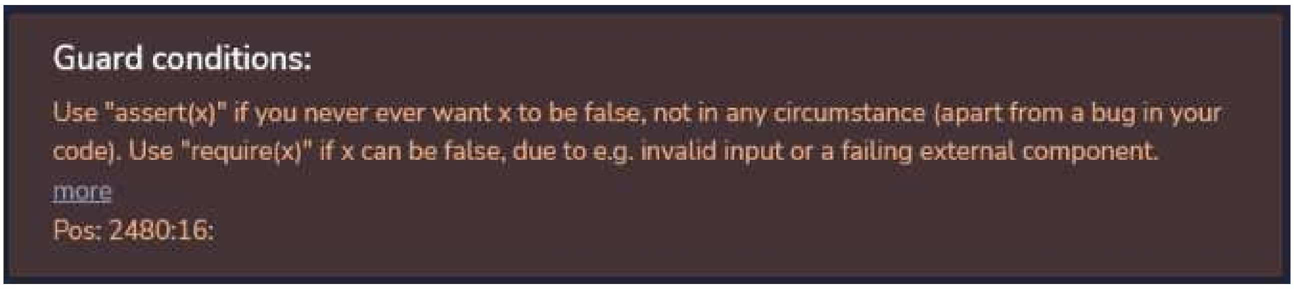
ExchangeV2Core.validateFull(struct LibOrder.Order,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2483:12:


Similar variable names:

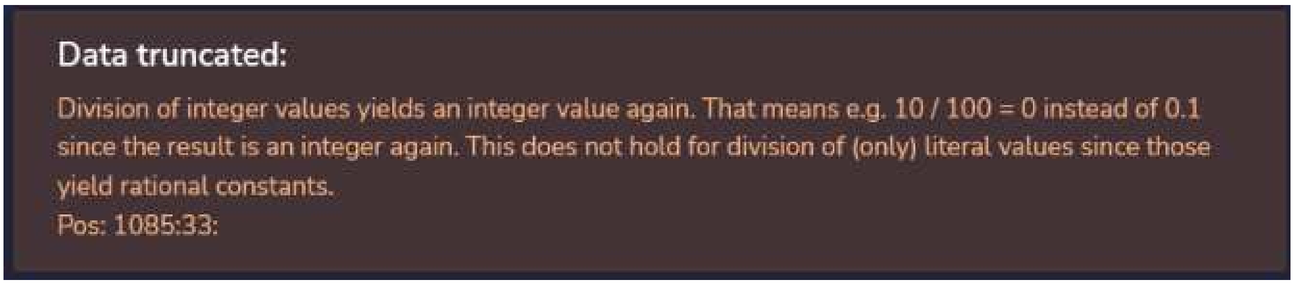
ExchangeV2Core.matchAssets(struct LibOrder.Order,struct LibOrder.Order) : Variables have very similar names "makeMatch" and "takeMatch". Note: Modifiers are currently not considered by this static analysis.
Pos: 2479:16:


No return:

ExchangeV2Core.getProtocolFee(): Defines a return type but never explicitly returns a value.
Pos: 2488:12:

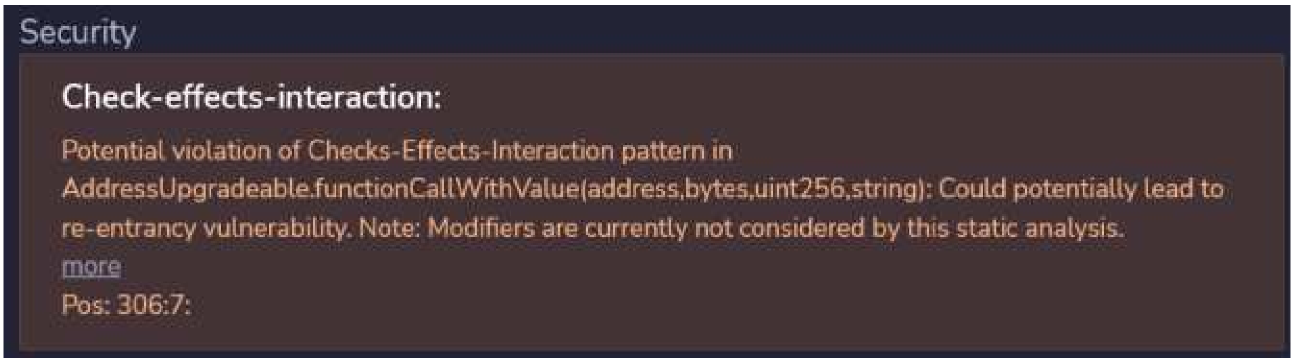

Guard conditions:

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: 2480:16:


Data truncated:

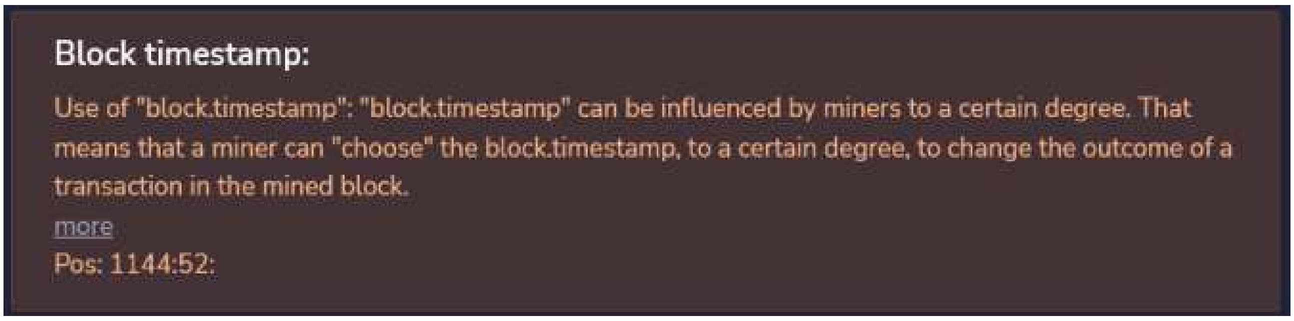
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1085:33:

OrderValidator.sol


Security

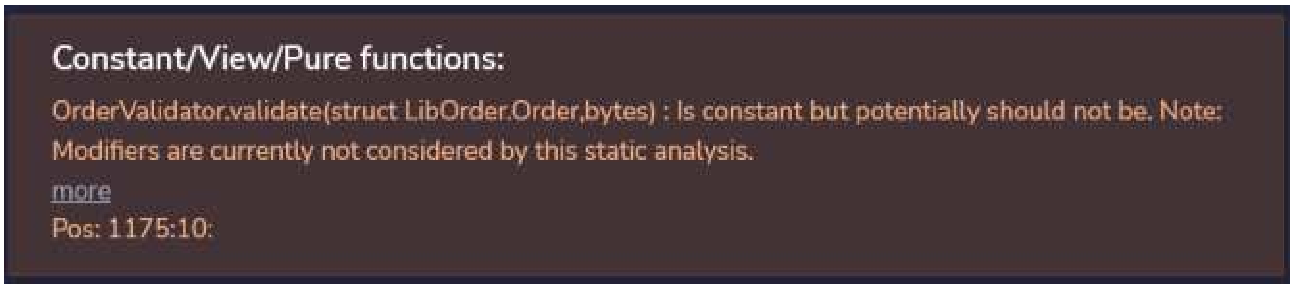
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 306:7:


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: 1144:52:

Miscellaneous

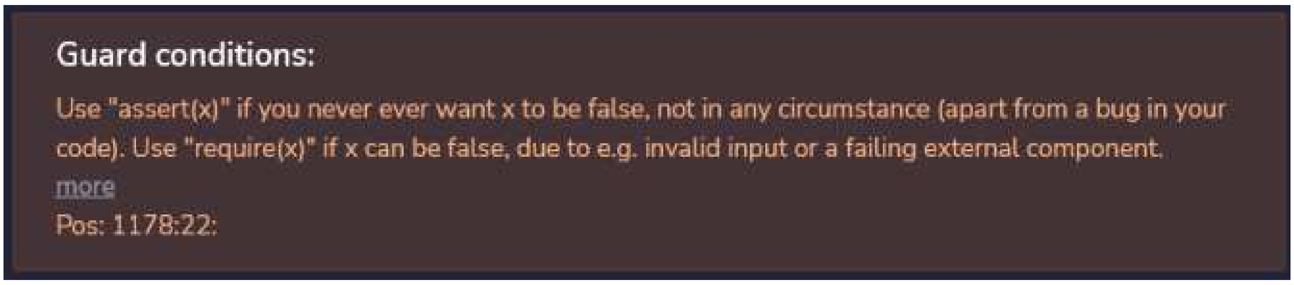


Constant/View/Pure functions:

OrderValidator.validate(struct LibOrder.Order,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1175:10:



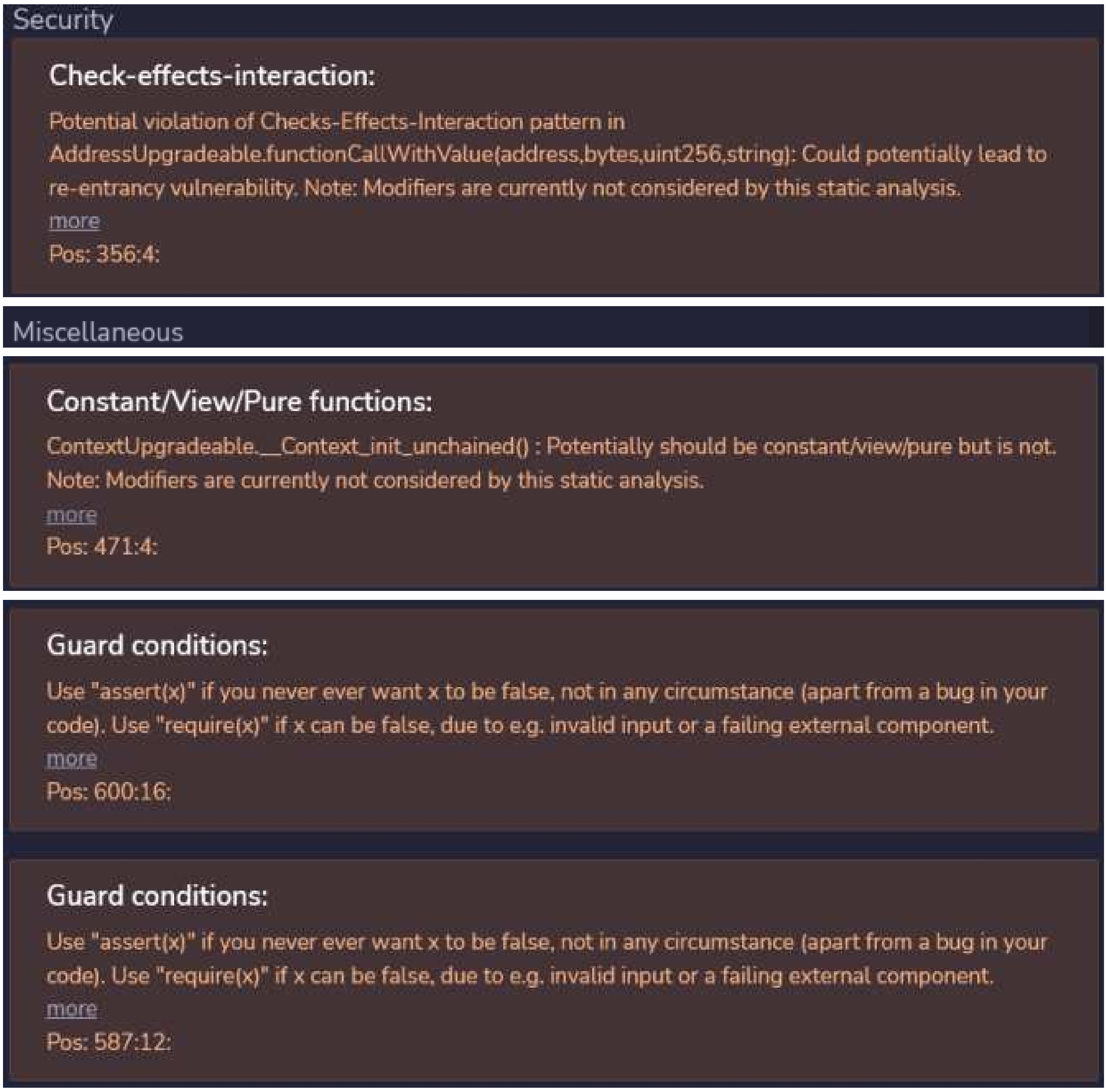
Guard conditions:

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: 1178:22:

TransferExecutor.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 356:4:

Miscellaneous

Constant/View/Pure functions:

ContextUpgradeable.__Context_init_unchained() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 471:4:

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: 600:16:

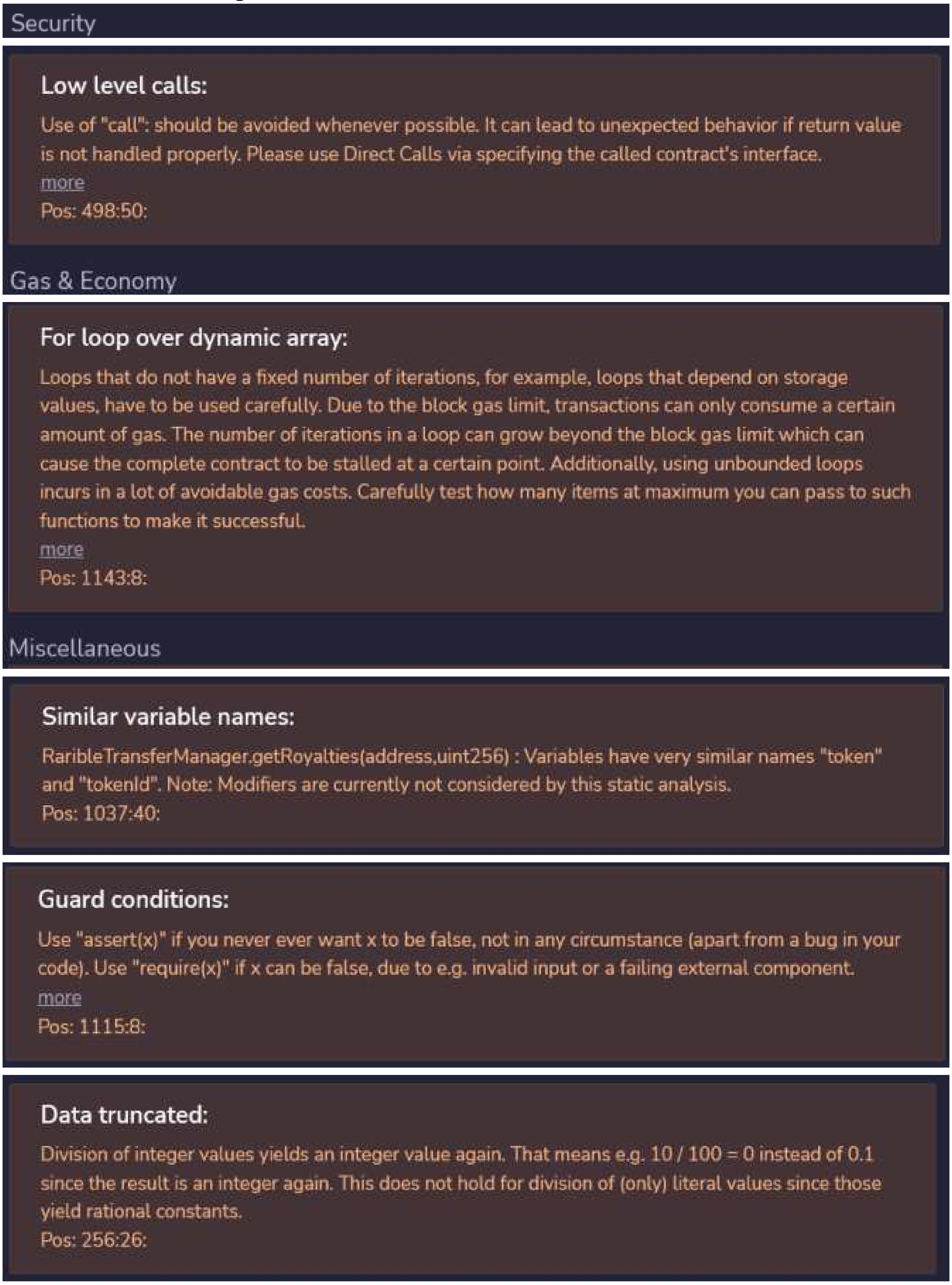
Guard conditions:

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: 587:12:

RaribleTransferManager.sol



Security

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: 498:50:

Gas & Economy

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: 1143:8:

Miscellaneous

Similar variable names:

RaribleTransferManager.getRoyalties(address,uint256) : Variables have very similar names "token" and "tokenId". Note: Modifiers are currently not considered by this static analysis.
Pos: 1037:40:

Guard conditions:

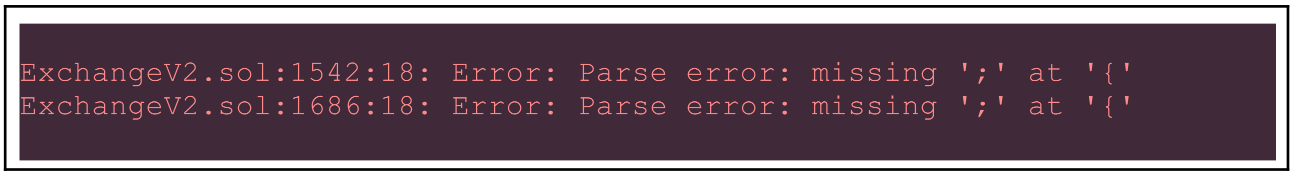
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1115:8:

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: 256:26:

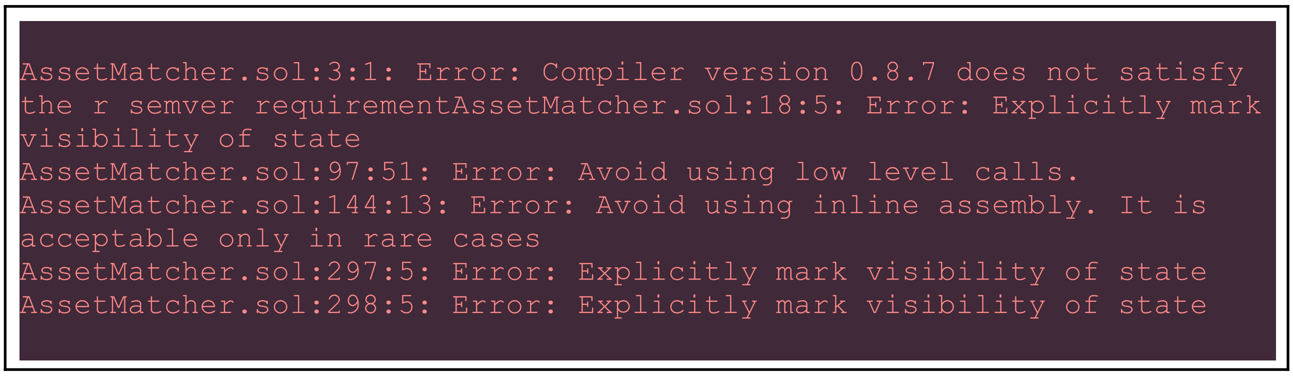
Solhint Linter

ExchangeV2.sol



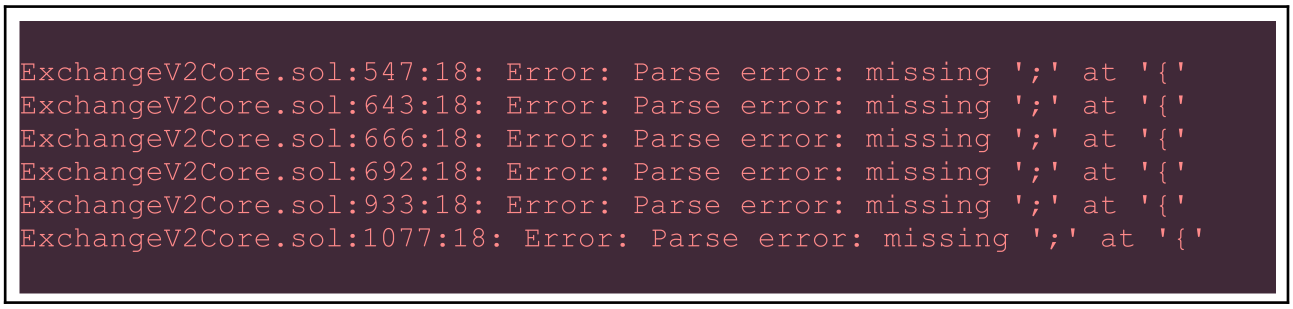
```
ExchangeV2.sol:1542:18: Error: Parse error: missing ';' at '{'
ExchangeV2.sol:1686:18: Error: Parse error: missing ';' at '{'
```

AssetMatcher.sol



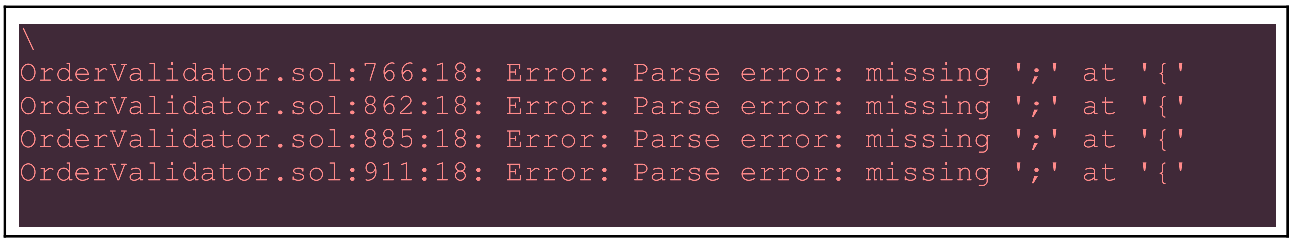
```
AssetMatcher.sol:3:1: Error: Compiler version 0.8.7 does not satisfy
the r semver requirementAssetMatcher.sol:18:5: Error: Explicitly mark
visibility of state
AssetMatcher.sol:97:51: Error: Avoid using low level calls.
AssetMatcher.sol:144:13: Error: Avoid using inline assembly. It is
acceptable only in rare cases
AssetMatcher.sol:297:5: Error: Explicitly mark visibility of state
AssetMatcher.sol:298:5: Error: Explicitly mark visibility of state
```

ExchangeV2Core.sol



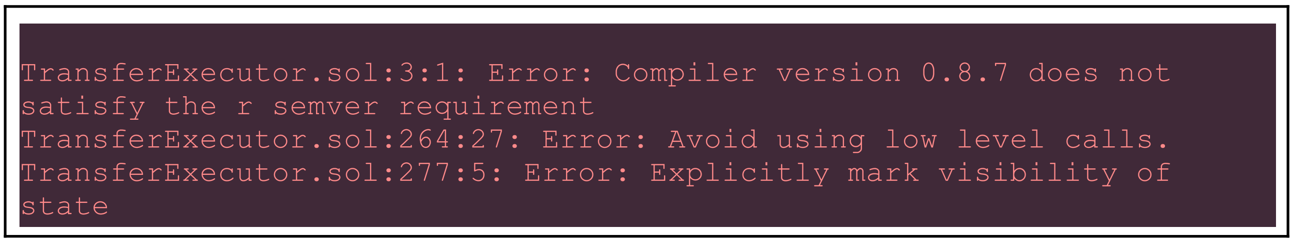
```
ExchangeV2Core.sol:547:18: Error: Parse error: missing ';' at '{'
ExchangeV2Core.sol:643:18: Error: Parse error: missing ';' at '{'
ExchangeV2Core.sol:666:18: Error: Parse error: missing ';' at '{'
ExchangeV2Core.sol:692:18: Error: Parse error: missing ';' at '{'
ExchangeV2Core.sol:933:18: Error: Parse error: missing ';' at '{'
ExchangeV2Core.sol:1077:18: Error: Parse error: missing ';' at '{'
```

OrderValidator.sol

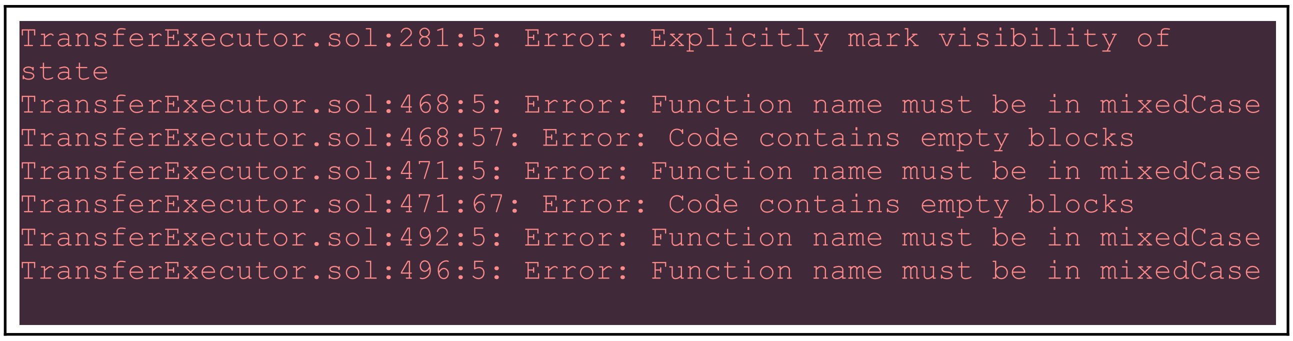


```
\
OrderValidator.sol:766:18: Error: Parse error: missing ';' at '{'
OrderValidator.sol:862:18: Error: Parse error: missing ';' at '{'
OrderValidator.sol:885:18: Error: Parse error: missing ';' at '{'
OrderValidator.sol:911:18: Error: Parse error: missing ';' at '{'
```

TransferExecutor.sol

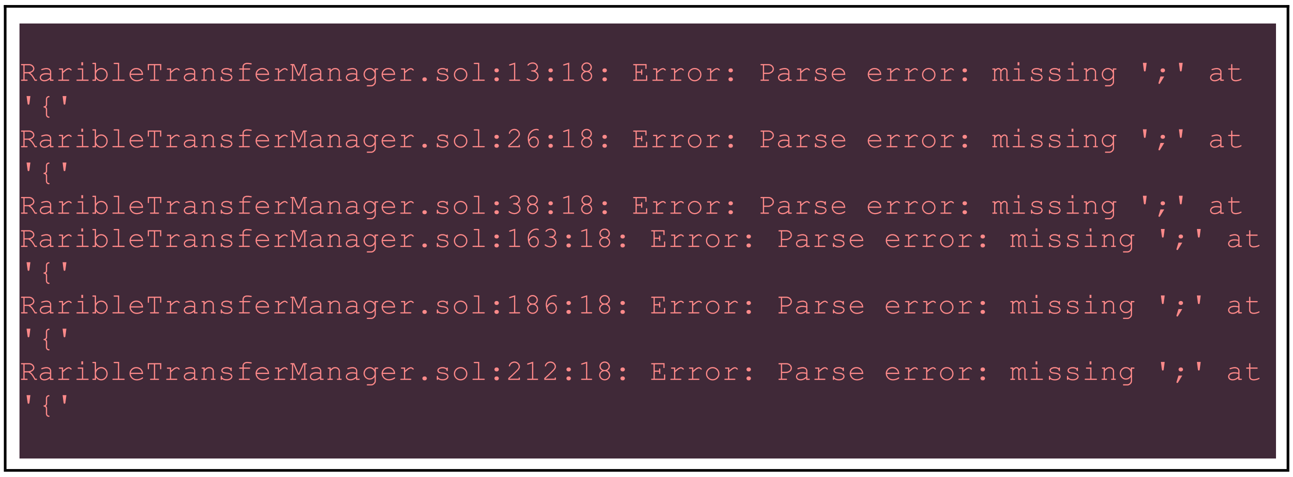


```
TransferExecutor.sol:3:1: Error: Compiler version 0.8.7 does not
satisfy the r semver requirement
TransferExecutor.sol:264:27: Error: Avoid using low level calls.
TransferExecutor.sol:277:5: Error: Explicitly mark visibility of
state
```



```
TransferExecutor.sol:281:5: Error: Explicitly mark visibility of
state
TransferExecutor.sol:468:5: Error: Function name must be in mixedCase
TransferExecutor.sol:468:57: Error: Code contains empty blocks
TransferExecutor.sol:471:5: Error: Function name must be in mixedCase
TransferExecutor.sol:471:67: Error: Code contains empty blocks
TransferExecutor.sol:492:5: Error: Function name must be in mixedCase
TransferExecutor.sol:496:5: Error: Function name must be in mixedCase
```

RaribleTransferManager.sol



```
RaribleTransferManager.sol:13:18: Error: Parse error: missing ';' at
'{'
RaribleTransferManager.sol:26:18: Error: Parse error: missing ';' at
'{'
RaribleTransferManager.sol:38:18: Error: Parse error: missing ';' at
RaribleTransferManager.sol:163:18: Error: Parse error: missing ';' at
'{'
RaribleTransferManager.sol:186:18: Error: Parse error: missing ';' at
'{'
RaribleTransferManager.sol:212:18: 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.

Email: audit@EtherAuthority.io