EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	JustCause Finance
Website:	www.justcause.finance
Platform:	Polygon Network
Language:	Solidity
Date:	July 19th, 2022

Table of contents

Introduction

EtherAuthority was contracted by JustCause Protocol to perform the Security audit of the JustCause Protocol 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 July 19th, 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

JustCause is a crowdfunding platform having functionalities like pool creation, deposit, withdraw, and claim. The JustCauseProtocol contract inherits the ERC721URIStorageUpgradeable, Initializable, ReentrancyGuard, Clones 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

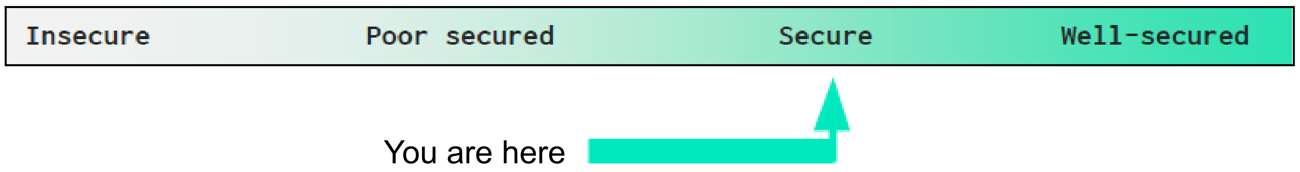
Name	**Code Review and Security Analysis Report for JustCause Protocol Smart Contracts**
Platform	**Polygon / Solidity**
File 1	JCDepositorERC721.sol
File 1 MD5 Hash	3A66C8A87CE303E274DA0F8271582DD8
File 2	JustCausePool.sol
File 2 MD5 Hash	830D9093E6932775E4809C4774452269
Updated File 2 MD5 Hash	7645BE6D2F6B5789F993C4BF5193F9DA
File 3	PoolTracker.sol
File 3 MD5 Hash	7BE2DE8EEC1F7E3E543326F72103CA9F
Updated File 3 MD5 Hash	10E88A24059B3275A99983BEA3D0AA6B
Audit Date	July 19th,2022
Revise Audit Date	July 20th,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 JCDepositorERC721.sol ● Name: JCP Contributor Token ● Symbol: JCPC ● Owner can add new Fund Tokens.	**YES, This is valid.**
File 2 JustCausePool.sol ● Owner can set a new reference for the pool. ● Owner can set a meta URI.	**YES, This is valid.**
File 3 PoolTracker.sol ● PoolTracker has functions like: addDeposit, withdrawDeposit, claimInterest, etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"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 0 critical, 0 high, 0 medium and 1 low and some very low level issues.
All the issues have been resolved in the revised code.

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	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 3 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the JustCause Protocol 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 JustCause Protocol.

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

Some code parts are well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given a JustCause Protocol 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. 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://www.justcause.finance/ which provided rich information about the project architecture.

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

JCDepositorERC721.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__ERC721URIStorage_init	internal	access only Initializing	No Issue
3	__ERC721URIStorage_init_unchained	internal	access only Initializing	No Issue
4	tokenURI	read	Passed	No Issue
5	_setTokenURI	internal	Passed	No Issue
6	_burn	internal	Passed	No Issue
7	onlyPoolTracker	modifier	Passed	No Issue
8	initialize	write	access only Initializer	No Issue
9	addFunds	write	access only Pool Tracker	No Issue
10	withdrawFunds	external	access only Pool Tracker	No Issue
11	getDepositInfo	read	Passed	No Issue
12	getUserBalance	read	Passed	No Issue
13	getUserTokens	external	Passed	No Issue
14	getPool	read	Passed	No Issue
15	_beforeTokenTransfer	internal	Passed	No Issue
16	tokenURI	read	Passed	No Issue

JustCausePool.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	onlyAllowedTokens	modifier	Passed	No Issue
7	onlyReceiver	modifier	Passed	No Issue
8	onlyPoolTracker	modifier	Passed	No Issue
9	strLength	modifier	Passed	No Issue
10	initialize	external	Passed	No Issue
11	deposit	external	access only Pool Tracker	No Issue
12	withdraw	external	access only Pool Tracker	No Issue

13	withdrawDonations	external	access only Pool Tracker	No Issue
14	calcSplit	internal	Passed	No Issue
15	setAbout	external	access only Receiver	No Issue
16	setMetaUri	external	access only Receiver	No Issue
17	getAcceptedTokens	external	Passed	No Issue
18	getName	external	Passed	No Issue
19	getAbout	external	Passed	No Issue
20	getPicHash	external	Passed	No Issue
21	getMetaUri	external	Passed	No Issue
22	getIsVerified	external	Passed	No Issue
23	getRecipient	external	Passed	No Issue
24	getERC721Address	external	Passed	No Issue
25	getPoolInfo	external	Passed	No Issue
26	getATokenAddress	read	Passed	No Issue
27	getTotalDeposits	read	Passed	No Issue
28	getUnclaimedInterest	read	Passed	No Issue
29	getClaimedInterest	read	Passed	No Issue
30	getATokenBalance	read	Passed	No Issue
31	getReserveNormalizedIncome	read	Passed	No Issue
32	getAaveLiquidityIndex	read	Passed	No Issue
33	getPoolTokenInfo	external	Passed	No Issue

PoolTracker.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	_nonReentrantBefore	write	Passed	No Issue
4	_nonReentrantAfter	write	Passed	No Issue
5	onlyPools	modifier	Passed	No Issue
6	onlyAcceptedTokens	modifier	Passed	No Issue
7	onlyAcceptedToken	modifier	Passed	No Issue
8	onlyMultiSig	modifier	Passed	No Issue
9	addDeposit	external	access only Pools	No Issue
10	withdrawDeposit	external	access only Pools	No Issue
11	claimInterest	external	access only Pools	No Issue
12	createJCPoolClone	external	access only Accepted Tokens	No Issue
13	setBpFee	write	access only MultiSig	No Issue
14	getBpFee	read	Passed	No Issue
15	getTVL	read	Passed	No Issue
16	getTotalDonated	read	Passed	No Issue
17	getDepositorERC721Address	read	Passed	No Issue

18	getReceiverPools	read	Passed	No Issue
19	getMultiSig	read	Passed	No Issue
20	getContributions	read	Passed	No Issue
21	getPoolAddr	read	Passed	No Issue
22	getReservesList	read	Passed	No Issue
23	getBaseJCPoolAddress	read	Passed	No Issue
24	getVerifiedPools	read	Passed	No Issue
25	checkPool	read	Passed	No Issue
26	getAddressFromName	external	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

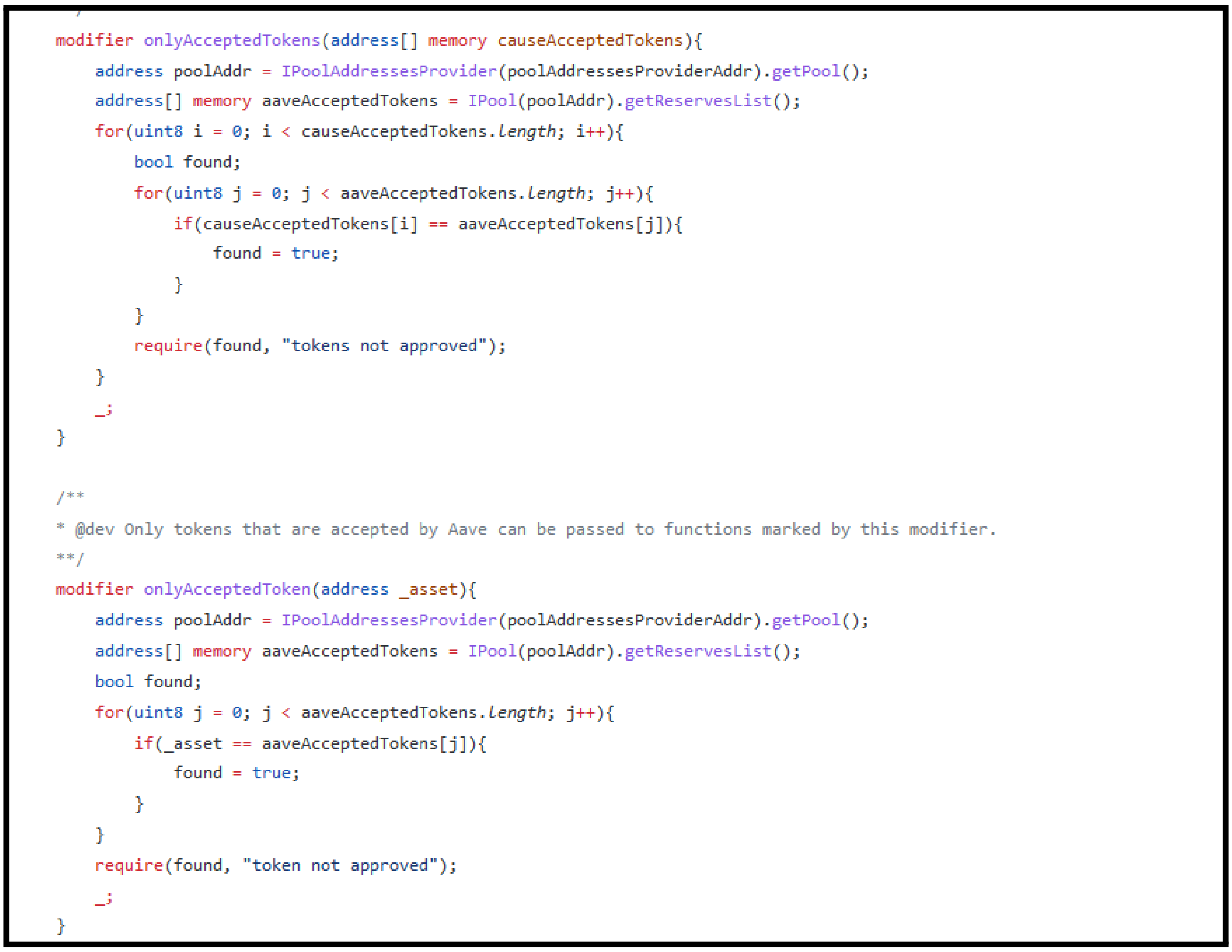
No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) High gas consuming loops: **PoolTracker.sol**



```
modifier onlyAcceptedTokens(address[] memory causeAcceptedTokens){
    address poolAddr = IPoolAddressesProvider(poolAddressesProviderAddr).getPool();
    address[] memory aaveAcceptedTokens = IPool(poolAddr).getReservesList();
    for(uint8 i = 0; i < causeAcceptedTokens.length; i++){
        bool found;
        for(uint8 j = 0; j < aaveAcceptedTokens.length; j++){
            if(causeAcceptedTokens[i] == aaveAcceptedTokens[j]){
                found = true;
            }
        }
        require(found, "tokens not approved");
    }
    _;
}

/**
* @dev Only tokens that are accepted by Aave can be passed to functions marked by this modifier.
**/
modifier onlyAcceptedToken(address _asset){
    address poolAddr = IPoolAddressesProvider(poolAddressesProviderAddr).getPool();
    address[] memory aaveAcceptedTokens = IPool(poolAddr).getReservesList();
    bool found;
    for(uint8 j = 0; j < aaveAcceptedTokens.length; j++){
        if(_asset == aaveAcceptedTokens[j]){
            found = true;
        }
    }
    require(found, "token not approved");
    _;
}
```

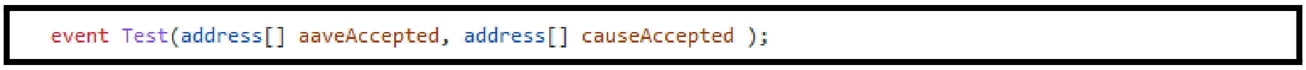
The modifiers onlyAcceptedTokens and onlyAcceptedToken have an unbound loop. This does not create a major security or logical vulnerability, but it may hit the block's gas limit if there are high numbers of entries used in the loop.

Resolution: The best practice is to set a limit on the number of entries that are expected. On another hand, this can be safely acknowledged that only a limited number of tokens will be minted in a batch.

Status: Fixed

Very Low / Informational / Best practices:

(1) Unused event: **PoolTracker.sol**



```
event Test(address[] aaveAccepted, address[] causeAccepted );
```

Test event is defined but not used.

Resolution: We suggest removing unused events.

Status: Fixed

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:

- setBpFee: PoolTracker owner can set the fixed rate of fees.
- setAbout: JustCausePool Receiver owner can set new reference for pool.
- setMetaUri: JustCausePool Receiver owner can update metaUri reference for pool.

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 Github Weblink. And we have used all possible tests based on given objects as files. We have not observed some issues in the smart contracts and those issues have been resolved in the revised code. **So, the smart contracts are ready 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.

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

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

Our Methodology

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

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

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

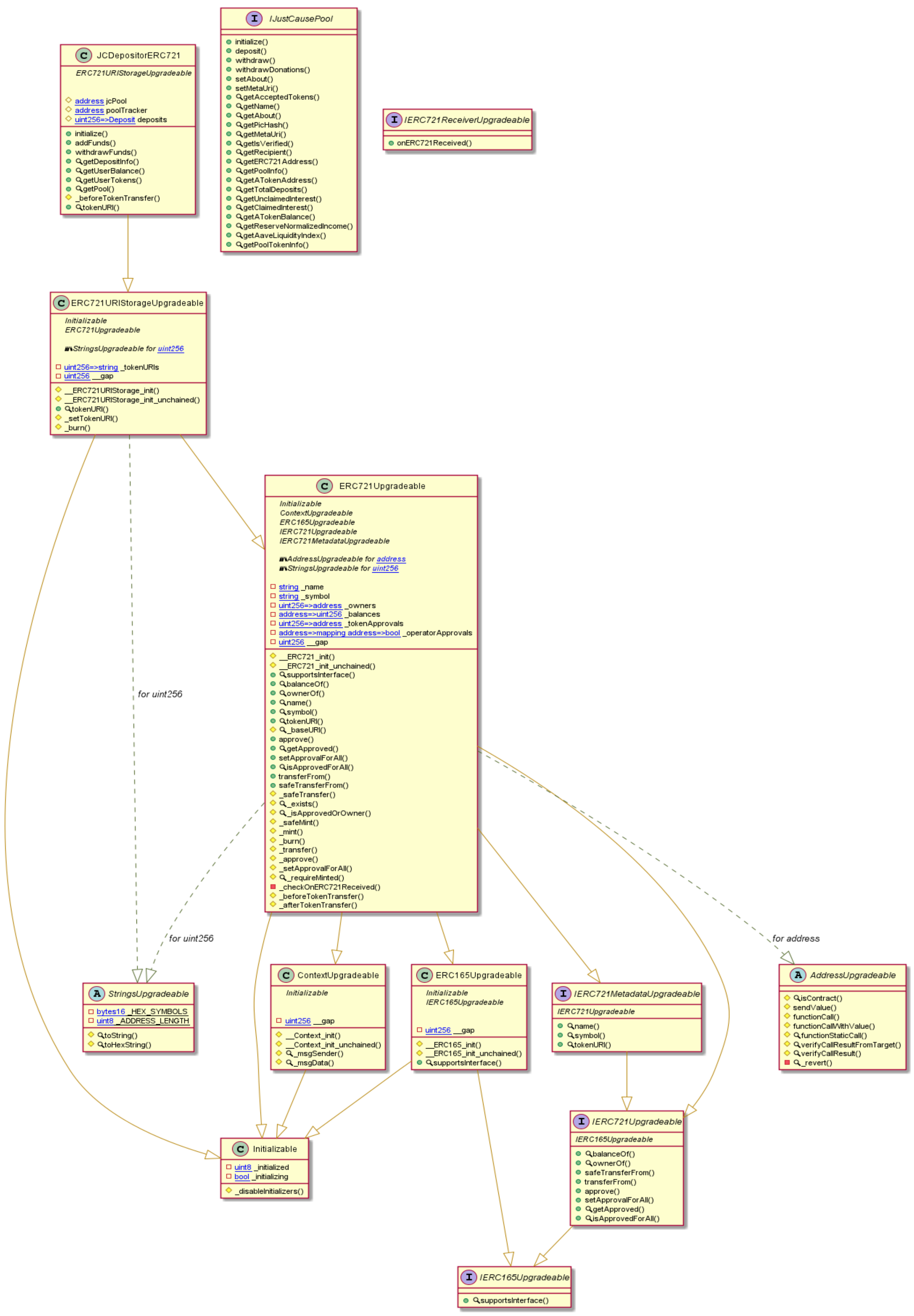
Technical Disclaimer

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

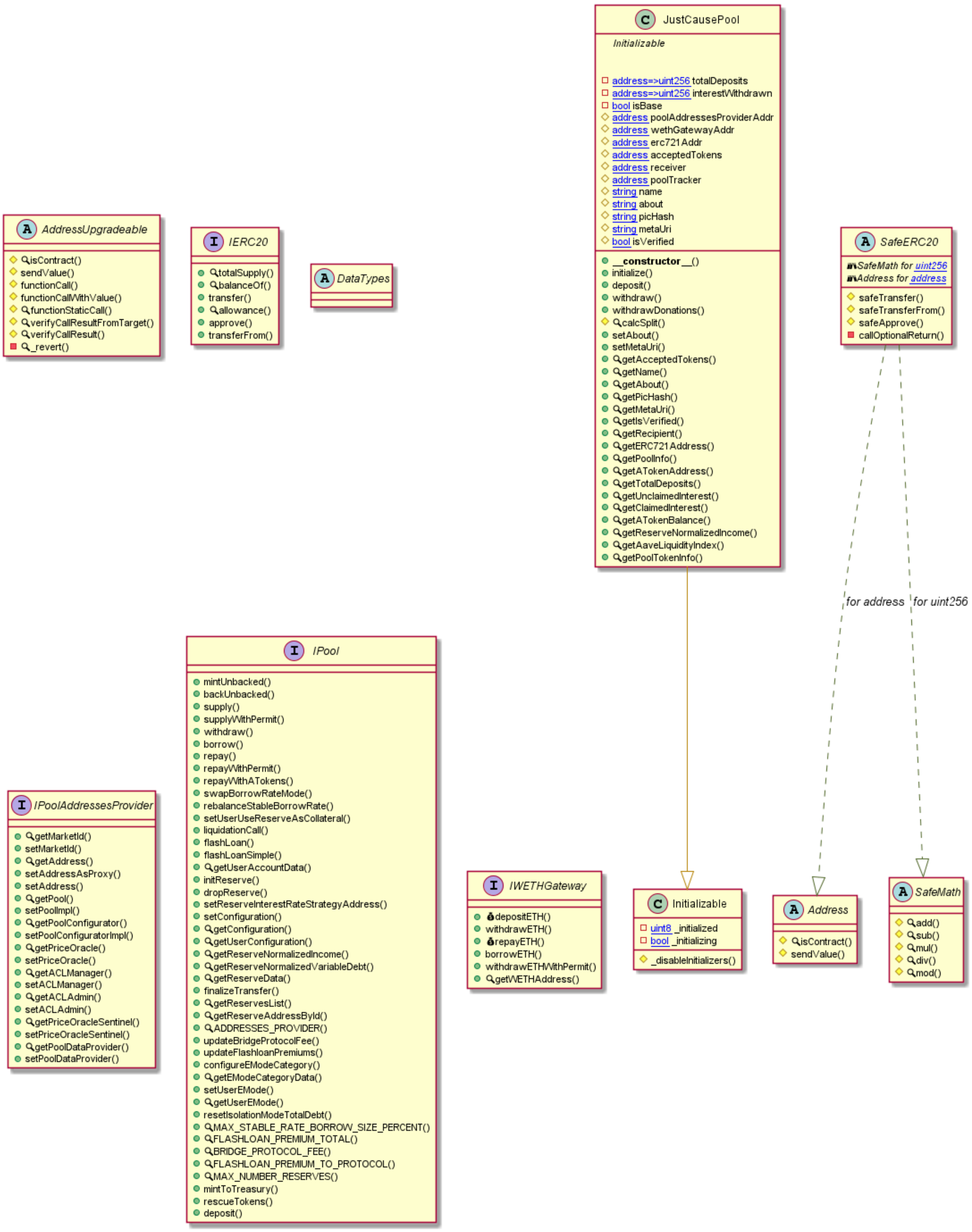
Appendix

Code Flow Diagram - JustCause Protocol

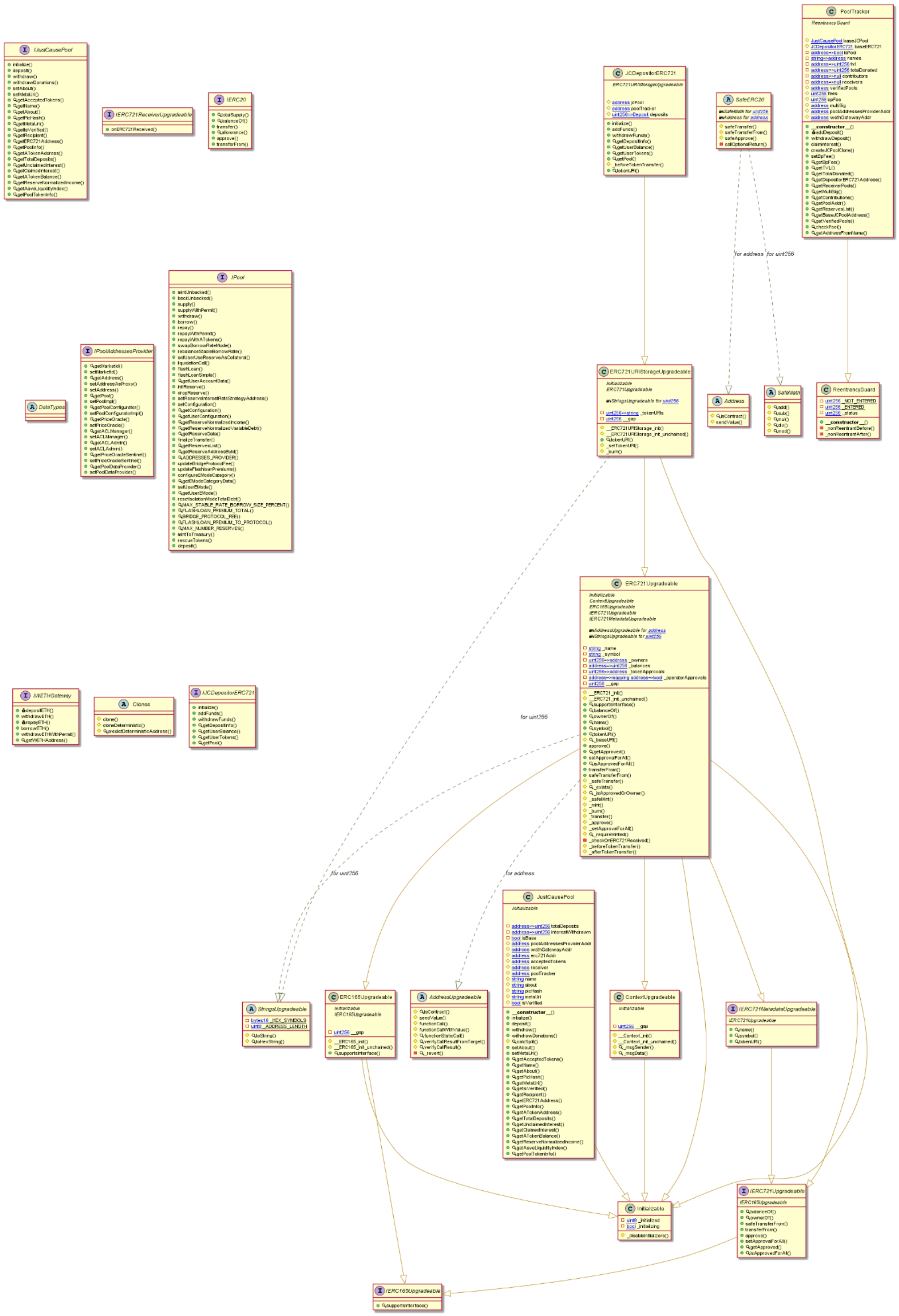
JCDepositorERC721 Diagram



JustCausePool Diagram

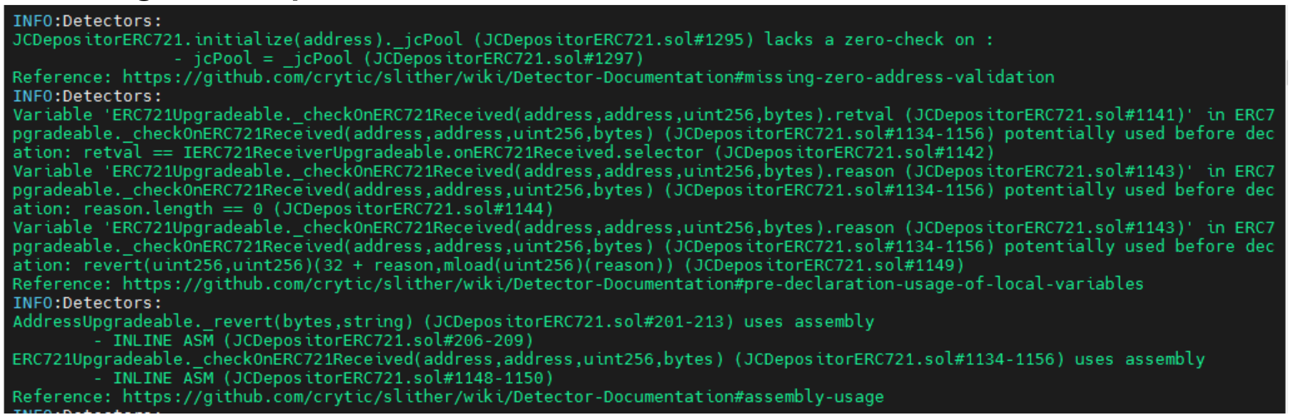


PoolTracker Diagram

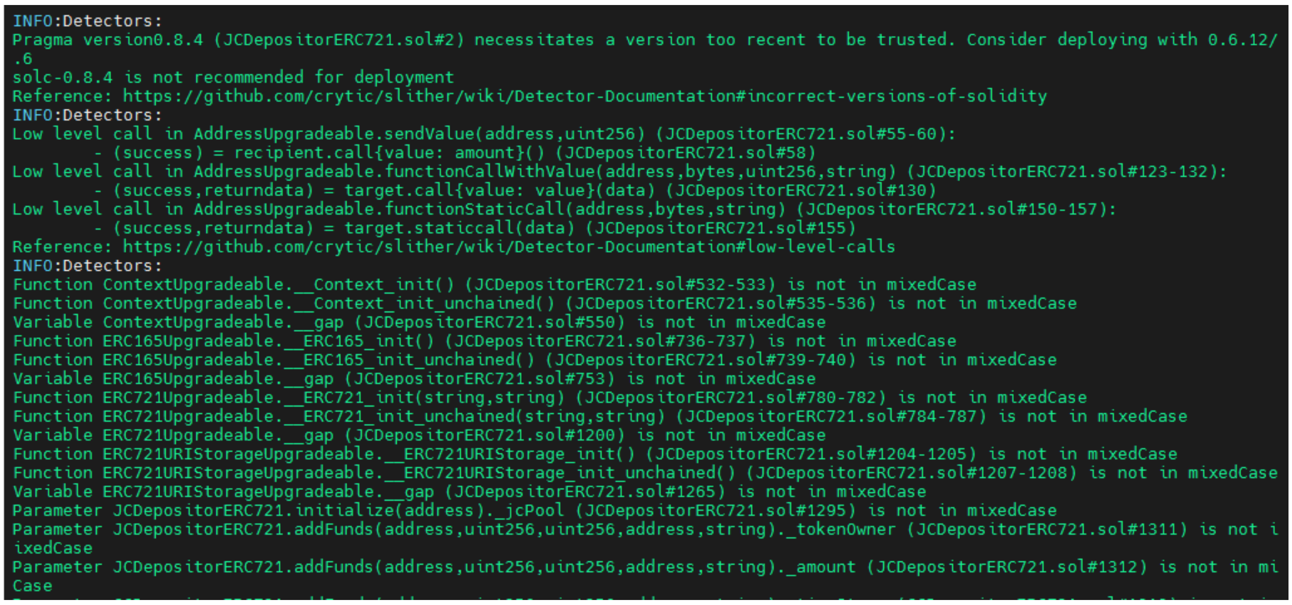


Slither Results Log

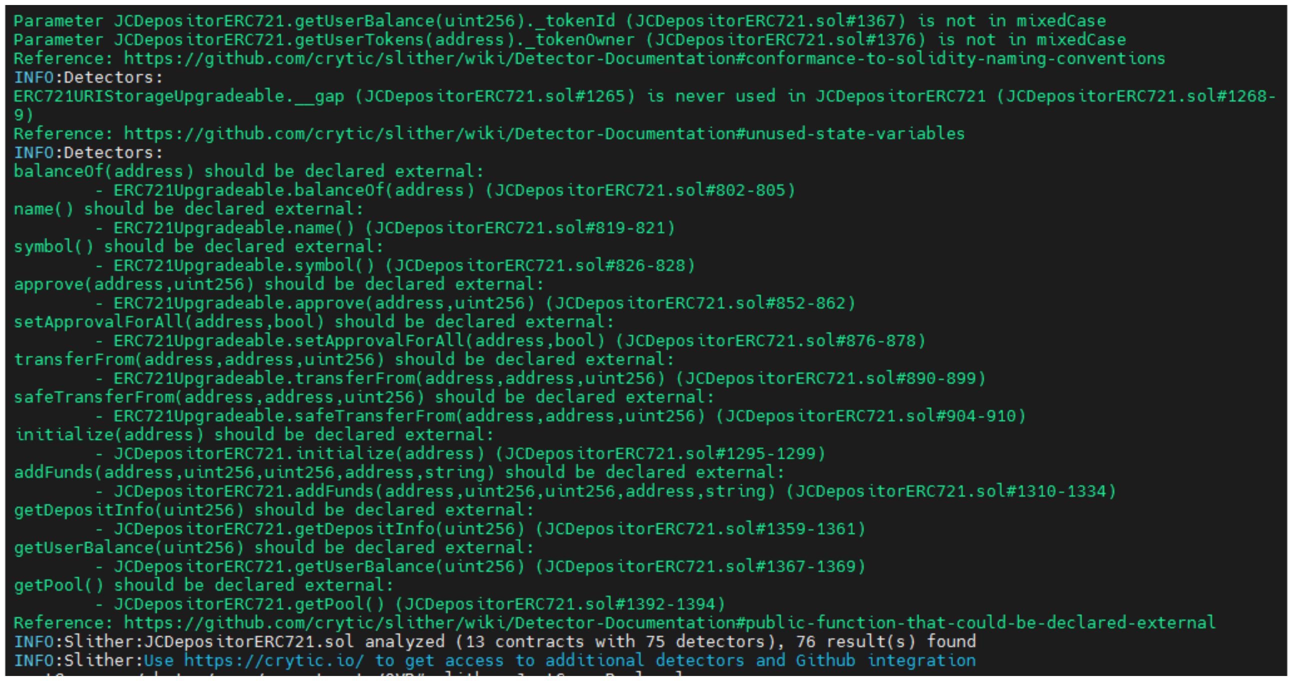
Slither log >> JCDepositorERC721.sol



```
INFO:Detectors:
JCDepositorERC721.initialize(address)._jcPool (JCDepositorERC721.sol#1295) lacks a zero-check on :
                - jcPool = _jcPool (JCDepositorERC721.sol#1297)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).retval (JCDepositorERC721.sol#1141)' in ERC7
pgradeable._checkOnERC721Received(address,address,uint256,bytes) (JCDepositorERC721.sol#1134-1156) potentially used before dec
ation: retval == IERC721ReceiverUpgradeable.onERC721Received.selector (JCDepositorERC721.sol#1142)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (JCDepositorERC721.sol#1143)' in ERC7
pgradeable._checkOnERC721Received(address,address,uint256,bytes) (JCDepositorERC721.sol#1134-1156) potentially used before dec
ation: reason.length == 0 (JCDepositorERC721.sol#1144)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (JCDepositorERC721.sol#1143)' in ERC7
pgradeable._checkOnERC721Received(address,address,uint256,bytes) (JCDepositorERC721.sol#1134-1156) potentially used before dec
ation: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (JCDepositorERC721.sol#1149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (JCDepositorERC721.sol#201-213) uses assembly
        - INLINE ASM (JCDepositorERC721.sol#206-209)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (JCDepositorERC721.sol#1134-1156) uses assembly
        - INLINE ASM (JCDepositorERC721.sol#1148-1150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

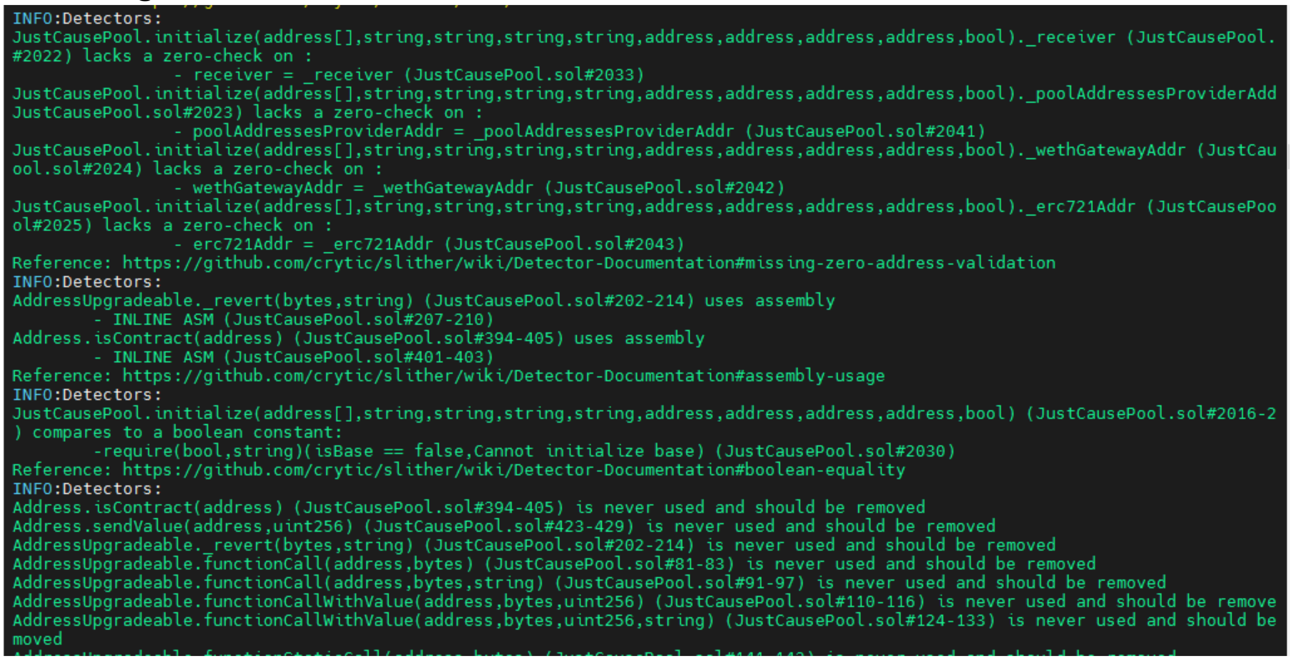


```
INFO:Detectors:
Pragma version0.8.4 (JCDepositorERC721.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (JCDepositorERC721.sol#55-60):
        - (success) = recipient.call{value: amount}() (JCDepositorERC721.sol#58)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (JCDepositorERC721.sol#123-132):
        - (success,returndata) = target.call{value: value}(data) (JCDepositorERC721.sol#130)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (JCDepositorERC721.sol#150-157):
        - (success,returndata) = target.staticcall(data) (JCDepositorERC721.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (JCDepositorERC721.sol#532-533) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (JCDepositorERC721.sol#535-536) is not in mixedCase
Variable ContextUpgradeable.__gap (JCDepositorERC721.sol#550) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (JCDepositorERC721.sol#736-737) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (JCDepositorERC721.sol#739-740) is not in mixedCase
Variable ERC165Upgradeable.__gap (JCDepositorERC721.sol#753) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (JCDepositorERC721.sol#780-782) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (JCDepositorERC721.sol#784-787) is not in mixedCase
Variable ERC721Upgradeable.__gap (JCDepositorERC721.sol#1200) is not in mixedCase
Function ERC721URIStorageUpgradeable.__ERC721URIStorage_init() (JCDepositorERC721.sol#1204-1205) is not in mixedCase
Function ERC721URIStorageUpgradeable.__ERC721URIStorage_init_unchained() (JCDepositorERC721.sol#1207-1208) is not in mixedCase
Variable ERC721URIStorageUpgradeable.__gap (JCDepositorERC721.sol#1265) is not in mixedCase
Parameter JCDepositorERC721.initialize(address)._jcPool (JCDepositorERC721.sol#1295) is not in mixedCase
Parameter JCDepositorERC721.addFunds(address,uint256,uint256,address,string)._tokenOwner (JCDepositorERC721.sol#1311) is not i
ixedCase
Parameter JCDepositorERC721.addFunds(address,uint256,uint256,address,string)._amount (JCDepositorERC721.sol#1312) is not in mi
Case
```

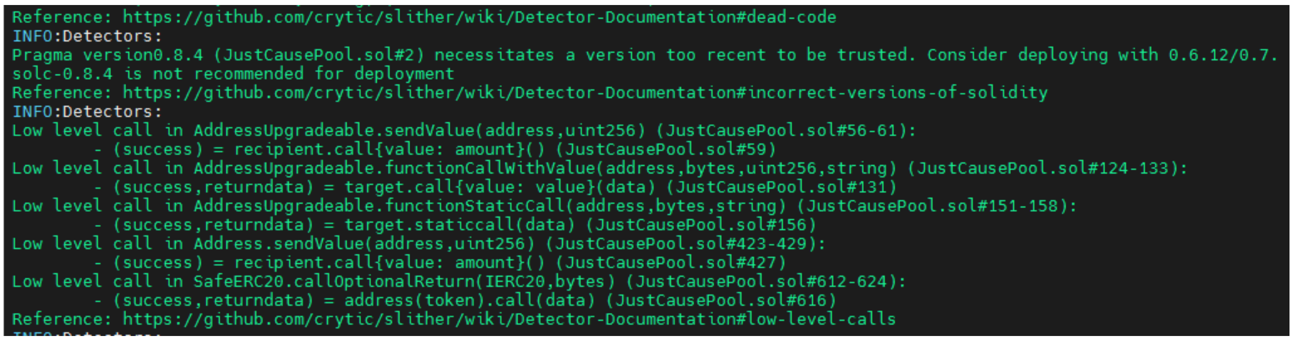


```
Parameter JCDepositorERC721.getUserBalance(uint256)._tokenId (JCDepositorERC721.sol#1367) is not in mixedCase
Parameter JCDepositorERC721.getUserTokens(address)._tokenOwner (JCDepositorERC721.sol#1376) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ERC721URIStorageUpgradeable.__gap (JCDepositorERC721.sol#1265) is never used in JCDepositorERC721 (JCDepositorERC721.sol#1268-
9)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
balanceOf(address) should be declared external:
        - ERC721Upgradeable.balanceOf(address) (JCDepositorERC721.sol#802-805)
name() should be declared external:
        - ERC721Upgradeable.name() (JCDepositorERC721.sol#819-821)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (JCDepositorERC721.sol#826-828)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (JCDepositorERC721.sol#852-862)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (JCDepositorERC721.sol#876-878)
transferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.transferFrom(address,address,uint256) (JCDepositorERC721.sol#890-899)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (JCDepositorERC721.sol#904-910)
initialize(address) should be declared external:
        - JCDepositorERC721.initialize(address) (JCDepositorERC721.sol#1295-1299)
addFunds(address,uint256,uint256,address,string) should be declared external:
        - JCDepositorERC721.addFunds(address,uint256,uint256,address,string) (JCDepositorERC721.sol#1310-1334)
getDepositInfo(uint256) should be declared external:
        - JCDepositorERC721.getDepositInfo(uint256) (JCDepositorERC721.sol#1359-1361)
getUserBalance(uint256) should be declared external:
        - JCDepositorERC721.getUserBalance(uint256) (JCDepositorERC721.sol#1367-1369)
getPool() should be declared external:
        - JCDepositorERC721.getPool() (JCDepositorERC721.sol#1392-1394)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:JCDepositorERC721.sol analyzed (13 contracts with 75 detectors), 76 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

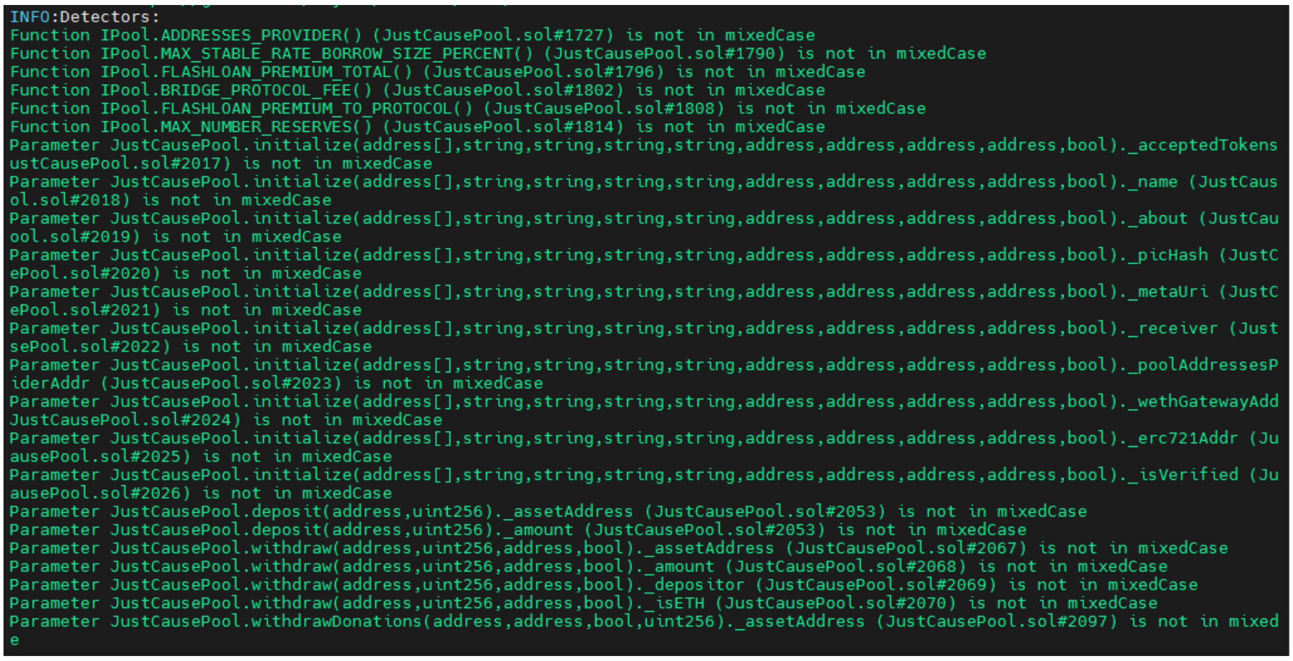
Slither log >> JustCausePool.sol



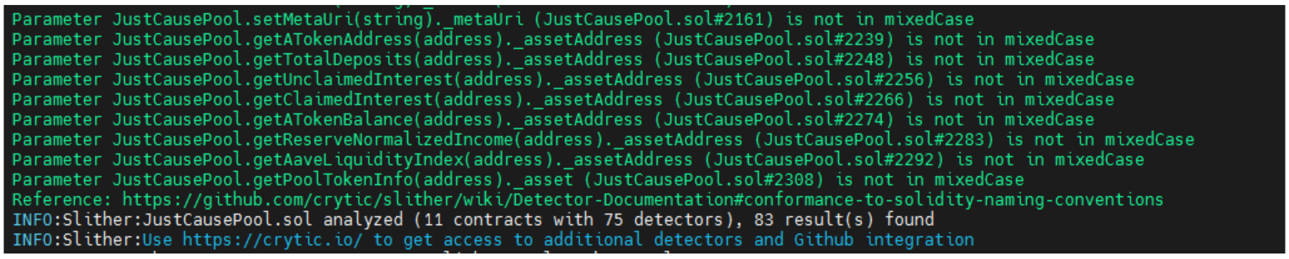
```
INFO:Detectors:
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._receiver (JustCausePool.
#2022) lacks a zero-check on :
                - receiver = _receiver (JustCausePool.sol#2033)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._poolAddressesProviderAdd
JustCausePool.sol#2023) lacks a zero-check on :
                - poolAddressesProviderAddr = _poolAddressesProviderAddr (JustCausePool.sol#2041)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._wethGatewayAddr (JustCau
ool.sol#2024) lacks a zero-check on :
                - wethGatewayAddr = _wethGatewayAddr (JustCausePool.sol#2042)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._erc721Addr (JustCausePoo
ol#2025) lacks a zero-check on :
                - erc721Addr = _erc721Addr (JustCausePool.sol#2043)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (JustCausePool.sol#202-214) uses assembly
        - INLINE ASM (JustCausePool.sol#207-210)
Address.isContract(address) (JustCausePool.sol#394-405) uses assembly
        - INLINE ASM (JustCausePool.sol#401-403)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool) (JustCausePool.sol#2016-2
) compares to a boolean constant:
        -require(bool,string)(isBase == false,Cannot initialize base) (JustCausePool.sol#2030)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.isContract(address) (JustCausePool.sol#394-405) is never used and should be removed
Address.sendValue(address,uint256) (JustCausePool.sol#423-429) is never used and should be removed
AddressUpgradeable._revert(bytes,string) (JustCausePool.sol#202-214) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (JustCausePool.sol#81-83) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (JustCausePool.sol#91-97) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (JustCausePool.sol#110-116) is never used and should be remove
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (JustCausePool.sol#124-133) is never used and should be
moved
AddressUpgradeable.functionStaticCall(address,bytes) (JustCausePool.sol#141-143) is never used and should be removed
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (JustCausePool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (JustCausePool.sol#56-61):
        - (success) = recipient.call{value: amount}() (JustCausePool.sol#59)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (JustCausePool.sol#124-133):
        - (success,returndata) = target.call{value: value}(data) (JustCausePool.sol#131)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (JustCausePool.sol#151-158):
        - (success,returndata) = target.staticcall(data) (JustCausePool.sol#156)
Low level call in Address.sendValue(address,uint256) (JustCausePool.sol#423-429):
        - (success) = recipient.call{value: amount}() (JustCausePool.sol#427)
Low level call in SafeERC20.callOptionalReturn(IERC20,bytes) (JustCausePool.sol#612-624):
        - (success,returndata) = address(token).call(data) (JustCausePool.sol#616)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

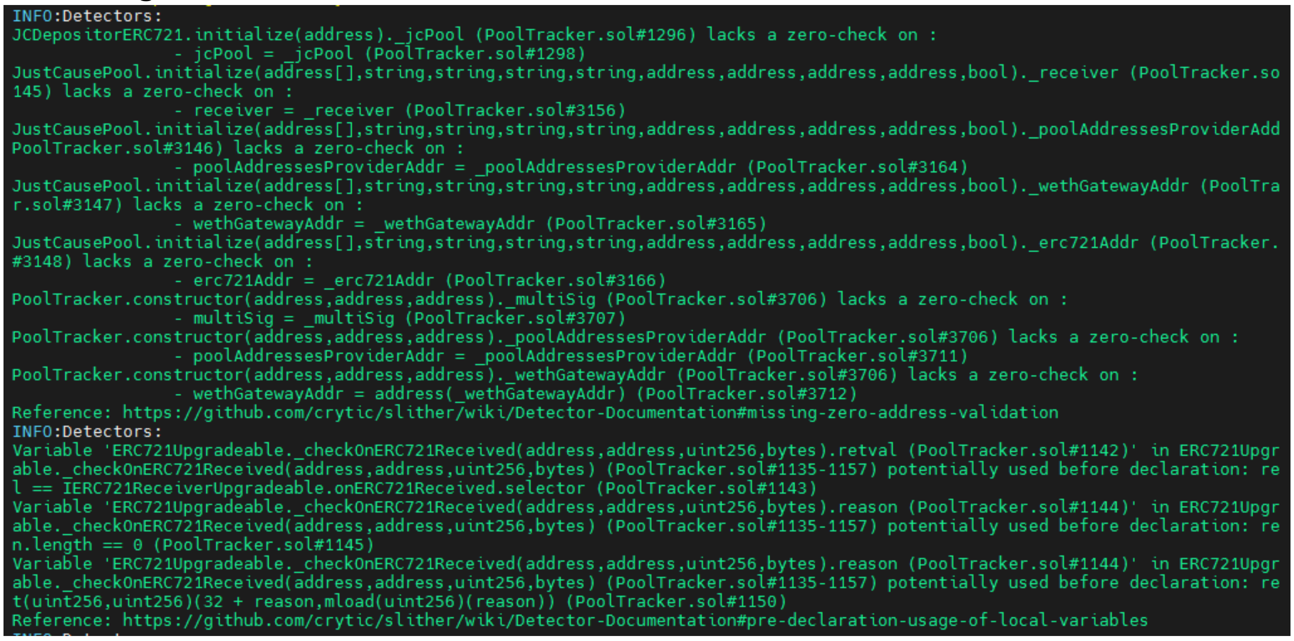


```
INFO:Detectors:
Function IPool.ADDRESSES_PROVIDER() (JustCausePool.sol#1727) is not in mixedCase
Function IPool.MAX_STABLE_RATE_BORROW_SIZE_PERCENT() (JustCausePool.sol#1790) is not in mixedCase
Function IPool.FLASHLOAN_PREMIUM_TOTAL() (JustCausePool.sol#1796) is not in mixedCase
Function IPool.BRIDGE_PROTOCOL_FEE() (JustCausePool.sol#1802) is not in mixedCase
Function IPool.FLASHLOAN_PREMIUM_TO_PROTOCOL() (JustCausePool.sol#1808) is not in mixedCase
Function IPool.MAX_NUMBER_RESERVES() (JustCausePool.sol#1814) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._acceptedTokens
ustCausePool.sol#2017) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._name (JustCaus
ol.sol#2018) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._about (JustCau
ool.sol#2019) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._picHash (JustC
ePool.sol#2020) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._metaUri (JustC
ePool.sol#2021) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._receiver (Just
sePool.sol#2022) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._poolAddressesP
iderAddr (JustCausePool.sol#2023) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._wethGatewayAdd
JustCausePool.sol#2024) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._erc721Addr (Ju
ausePool.sol#2025) is not in mixedCase
Parameter JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._isVerified (Ju
ausePool.sol#2026) is not in mixedCase
Parameter JustCausePool.deposit(address,uint256)._assetAddress (JustCausePool.sol#2053) is not in mixedCase
Parameter JustCausePool.deposit(address,uint256)._amount (JustCausePool.sol#2053) is not in mixedCase
Parameter JustCausePool.withdraw(address,uint256,address,bool)._assetAddress (JustCausePool.sol#2067) is not in mixedCase
Parameter JustCausePool.withdraw(address,uint256,address,bool)._amount (JustCausePool.sol#2068) is not in mixedCase
Parameter JustCausePool.withdraw(address,uint256,address,bool)._depositor (JustCausePool.sol#2069) is not in mixedCase
Parameter JustCausePool.withdraw(address,uint256,address,bool)._isETH (JustCausePool.sol#2070) is not in mixedCase
Parameter JustCausePool.withdrawDonations(address,address,bool,uint256)._assetAddress (JustCausePool.sol#2097) is not in mixed
e
```

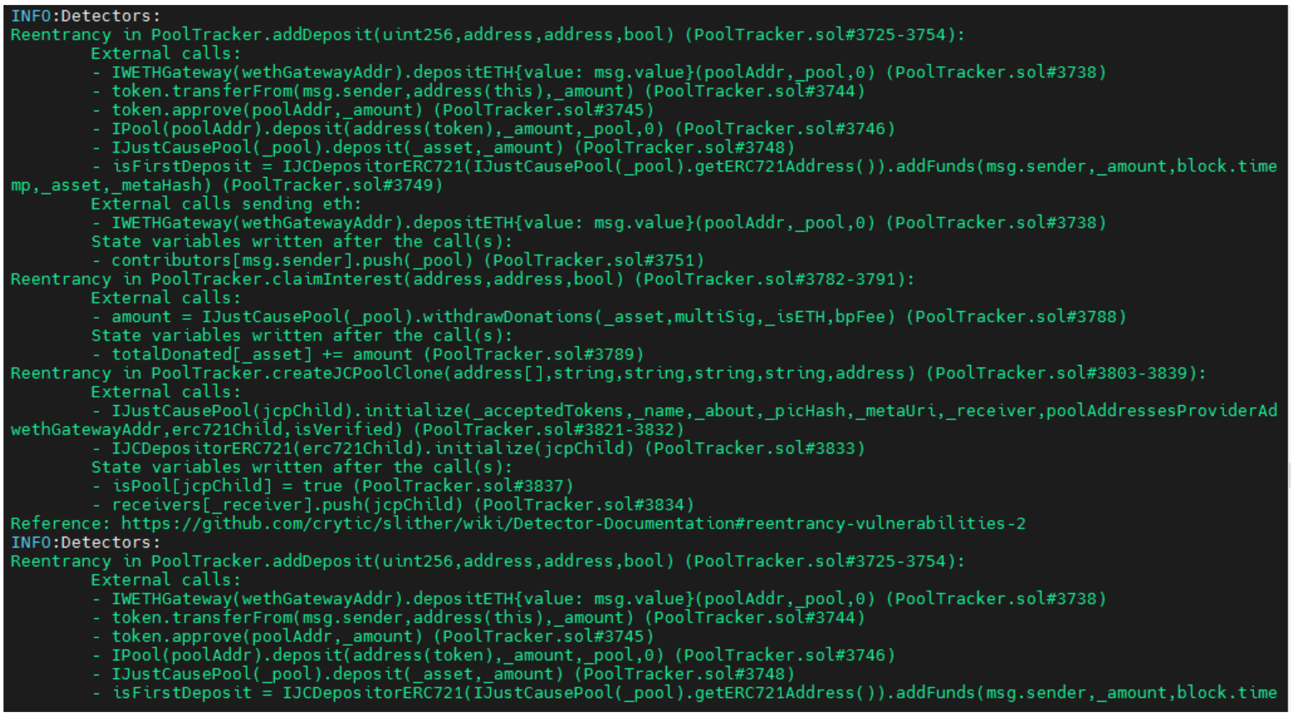


```
Parameter JustCausePool.setMetaUri(string)._metaUri (JustCausePool.sol#2161) is not in mixedCase
Parameter JustCausePool.getATokenAddress(address)._assetAddress (JustCausePool.sol#2239) is not in mixedCase
Parameter JustCausePool.getTotalDeposits(address)._assetAddress (JustCausePool.sol#2248) is not in mixedCase
Parameter JustCausePool.getUnclaimedInterest(address)._assetAddress (JustCausePool.sol#2256) is not in mixedCase
Parameter JustCausePool.getClaimedInterest(address)._assetAddress (JustCausePool.sol#2266) is not in mixedCase
Parameter JustCausePool.getATokenBalance(address)._assetAddress (JustCausePool.sol#2274) is not in mixedCase
Parameter JustCausePool.getReserveNormalizedIncome(address)._assetAddress (JustCausePool.sol#2283) is not in mixedCase
Parameter JustCausePool.getAaveLiquidityIndex(address)._assetAddress (JustCausePool.sol#2292) is not in mixedCase
Parameter JustCausePool.getPoolTokenInfo(address)._asset (JustCausePool.sol#2308) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:JustCausePool.sol analyzed (11 contracts with 75 detectors), 83 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

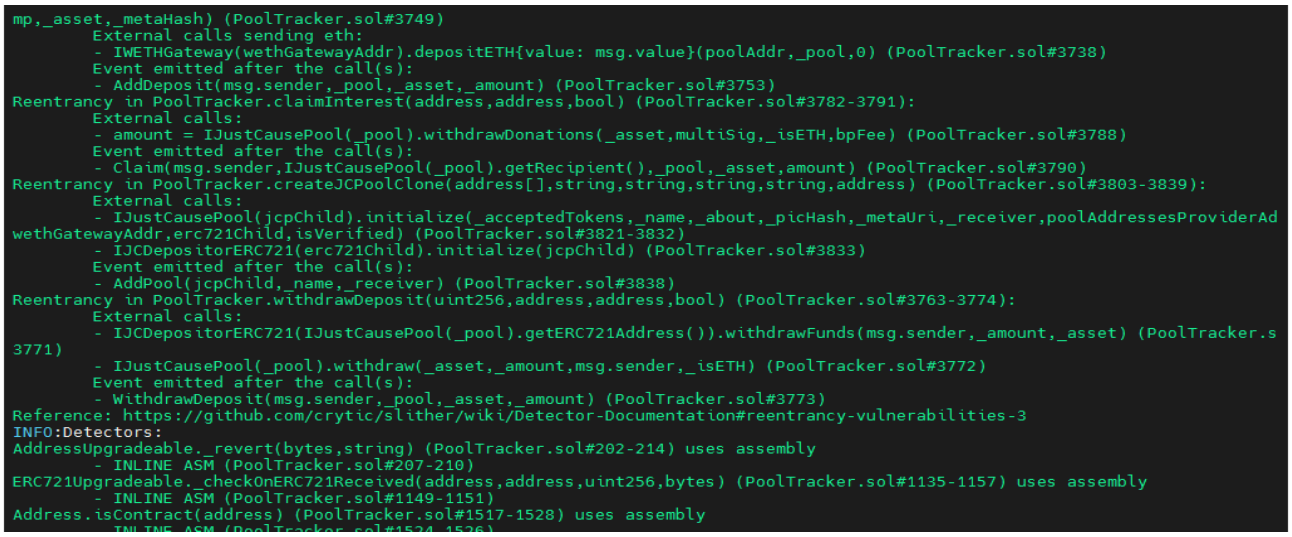
Slither log >> PoolTracker.sol



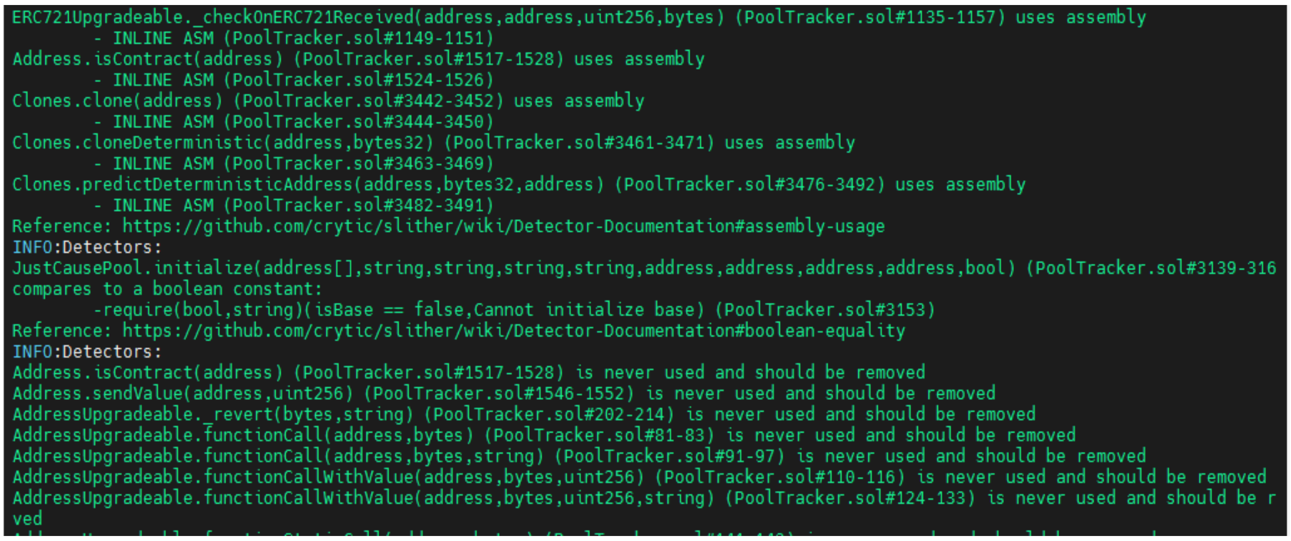
```
INFO:Detectors:
JCDepositorERC721.initialize(address)._jcPool (PoolTracker.sol#1296) lacks a zero-check on :
                - jcPool = _jcPool (PoolTracker.sol#1298)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._receiver (PoolTracker.so
145) lacks a zero-check on :
                - receiver = _receiver (PoolTracker.sol#3156)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._poolAddressesProviderAdd
PoolTracker.sol#3146) lacks a zero-check on :
                - poolAddressesProviderAddr = _poolAddressesProviderAddr (PoolTracker.sol#3164)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._wethGatewayAddr (PoolTra
r.sol#3147) lacks a zero-check on :
                - wethGatewayAddr = _wethGatewayAddr (PoolTracker.sol#3165)
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool)._erc721Addr (PoolTracker.
#3148) lacks a zero-check on :
                - erc721Addr = _erc721Addr (PoolTracker.sol#3166)
PoolTracker.constructor(address,address,address)._multiSig (PoolTracker.sol#3706) lacks a zero-check on :
                - multiSig = _multiSig (PoolTracker.sol#3707)
PoolTracker.constructor(address,address,address)._poolAddressesProviderAddr (PoolTracker.sol#3706) lacks a zero-check on :
                - poolAddressesProviderAddr = _poolAddressesProviderAddr (PoolTracker.sol#3711)
PoolTracker.constructor(address,address,address)._wethGatewayAddr (PoolTracker.sol#3706) lacks a zero-check on :
                - wethGatewayAddr = address(_wethGatewayAddr) (PoolTracker.sol#3712)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).retval (PoolTracker.sol#1142)' in ERC721Upgr
able._checkOnERC721Received(address,address,uint256,bytes) (PoolTracker.sol#1135-1157) potentially used before declaration: re
l == IERC721ReceiverUpgradeable.onERC721Received.selector (PoolTracker.sol#1143)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (PoolTracker.sol#1144)' in ERC721Upgr
able._checkOnERC721Received(address,address,uint256,bytes) (PoolTracker.sol#1135-1157) potentially used before declaration: re
n.length == 0 (PoolTracker.sol#1145)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (PoolTracker.sol#1144)' in ERC721Upgr
able._checkOnERC721Received(address,address,uint256,bytes) (PoolTracker.sol#1135-1157) potentially used before declaration: re
t(uint256,uint256)(32 + reason,mload(uint256)(reason)) (PoolTracker.sol#1150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```



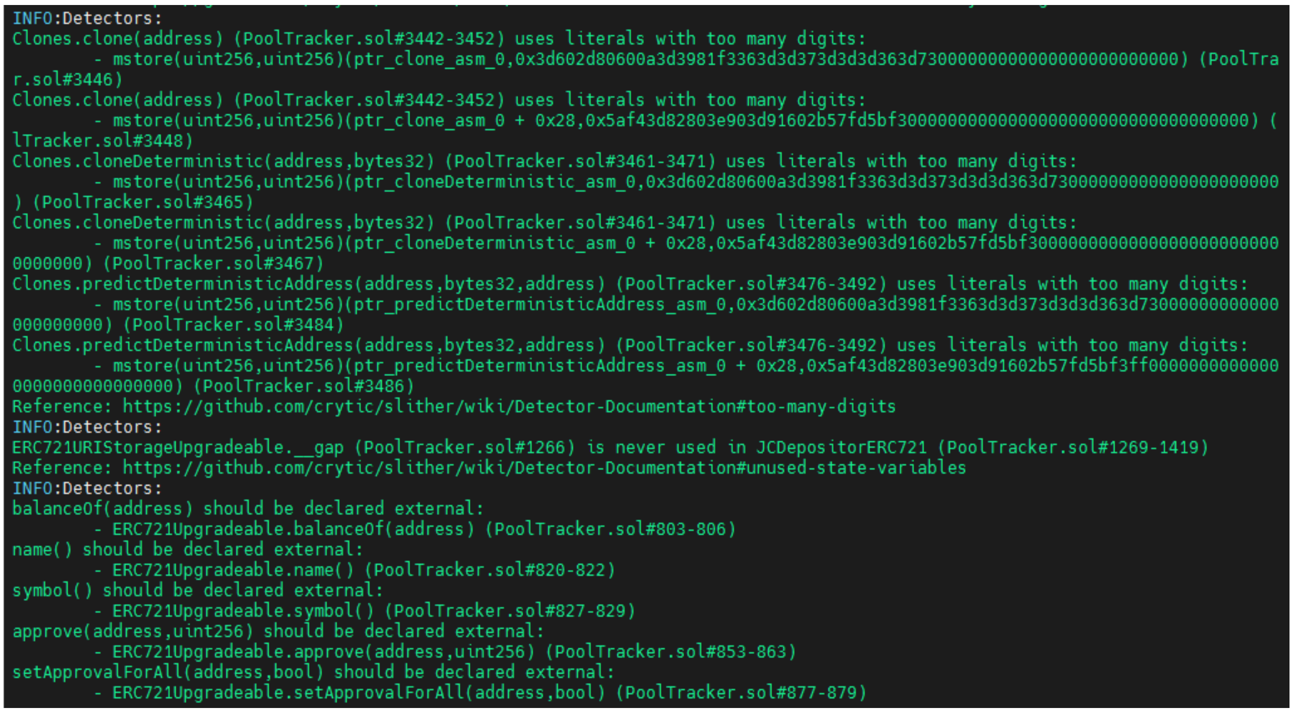
```
INFO:Detectors:
Reentrancy in PoolTracker.addDeposit(uint256,address,address,bool) (PoolTracker.sol#3725-3754):
        External calls:
        - IWETHGateway(wethGatewayAddr).depositETH{value: msg.value}(poolAddr,_pool,0) (PoolTracker.sol#3738)
        - token.transferFrom(msg.sender,address(this),_amount) (PoolTracker.sol#3744)
        - token.approve(poolAddr,_amount) (PoolTracker.sol#3745)
        - IPool(poolAddr).deposit(address(token),_amount,_pool,0) (PoolTracker.sol#3746)
        - IJustCausePool(_pool).deposit(_asset,_amount) (PoolTracker.sol#3748)
        - isFirstDeposit = IJCDepositorERC721(IJustCausePool(_pool).getERC721Address()).addFunds(msg.sender,_amount,block.time
mp,_asset,_metaHash) (PoolTracker.sol#3749)
        External calls sending eth:
        - IWETHGateway(wethGatewayAddr).depositETH{value: msg.value}(poolAddr,_pool,0) (PoolTracker.sol#3738)
        State variables written after the call(s):
        - contributors[msg.sender].push(_pool) (PoolTracker.sol#3751)
Reentrancy in PoolTracker.claimInterest(address,address,bool) (PoolTracker.sol#3782-3791):
        External calls:
        - amount = IJustCausePool(_pool).withdrawDonations(_asset,multiSig,_isETH,bpFee) (PoolTracker.sol#3788)
        State variables written after the call(s):
        - totalDonated[_asset] += amount (PoolTracker.sol#3789)
Reentrancy in PoolTracker.createJCPoolClone(address[],string,string,string,string,address) (PoolTracker.sol#3803-3839):
        External calls:
        - IJustCausePool(jcpChild).initialize(_acceptedTokens,_name,_about,_picHash,_metaUri,_receiver,poolAddressesProviderAd
wethGatewayAddr,erc721Child,isVerified) (PoolTracker.sol#3821-3832)
        - IJCDepositorERC721(erc721Child).initialize(jcpChild) (PoolTracker.sol#3833)
        State variables written after the call(s):
        - isPool[jcpChild] = true (PoolTracker.sol#3837)
        - receivers[_receiver].push(jcpChild) (PoolTracker.sol#3834)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in PoolTracker.addDeposit(uint256,address,address,bool) (PoolTracker.sol#3725-3754):
        External calls:
        - IWETHGateway(wethGatewayAddr).depositETH{value: msg.value}(poolAddr,_pool,0) (PoolTracker.sol#3738)
        - token.transferFrom(msg.sender,address(this),_amount) (PoolTracker.sol#3744)
        - token.approve(poolAddr,_amount) (PoolTracker.sol#3745)
        - IPool(poolAddr).deposit(address(token),_amount,_pool,0) (PoolTracker.sol#3746)
        - IJustCausePool(_pool).deposit(_asset,_amount) (PoolTracker.sol#3748)
        - isFirstDeposit = IJCDepositorERC721(IJustCausePool(_pool).getERC721Address()).addFunds(msg.sender,_amount,block.time
```



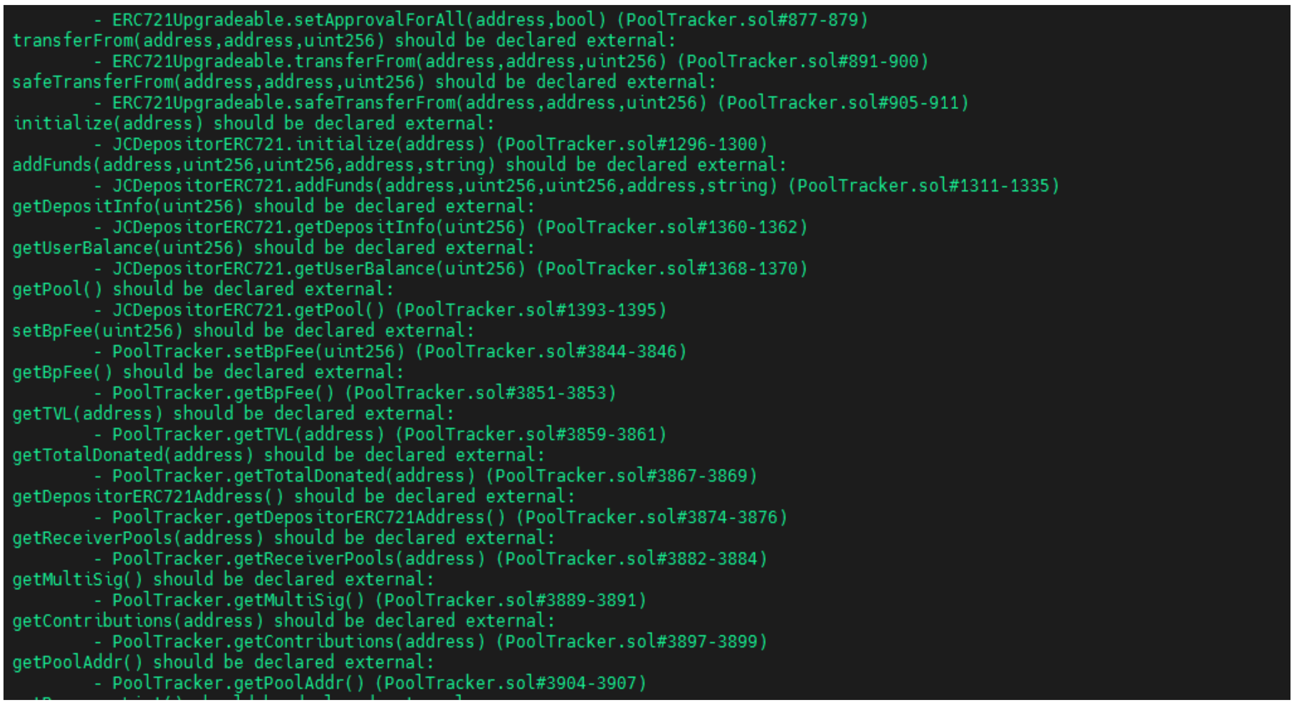
```
mp,_asset,_metaHash) (PoolTracker.sol#3749)
        External calls sending eth:
        - IWETHGateway(wethGatewayAddr).depositETH{value: msg.value}(poolAddr,_pool,0) (PoolTracker.sol#3738)
        Event emitted after the call(s):
        - AddDeposit(msg.sender,_pool,_asset,_amount) (PoolTracker.sol#3753)
Reentrancy in PoolTracker.claimInterest(address,address,bool) (PoolTracker.sol#3782-3791):
        External calls:
        - amount = IJustCausePool(_pool).withdrawDonations(_asset,multiSig,_isETH,bpFee) (PoolTracker.sol#3788)
        Event emitted after the call(s):
        - Claim(msg.sender,IJustCausePool(_pool).getRecipient(),_pool,_asset,amount) (PoolTracker.sol#3790)
Reentrancy in PoolTracker.createJCPoolClone(address[],string,string,string,string,address) (PoolTracker.sol#3803-3839):
        External calls:
        - IJustCausePool(jcpChild).initialize(_acceptedTokens,_name,_about,_picHash,_metaUri,_receiver,poolAddressesProviderAd
wethGatewayAddr,erc721Child,isVerified) (PoolTracker.sol#3821-3832)
        - IJCDepositorERC721(erc721Child).initialize(jcpChild) (PoolTracker.sol#3833)
        Event emitted after the call(s):
        - AddPool(jcpChild,_name,_receiver) (PoolTracker.sol#3838)
Reentrancy in PoolTracker.withdrawDeposit(uint256,address,address,bool) (PoolTracker.sol#3763-3774):
        External calls:
        - IJCDepositorERC721(IJustCausePool(_pool).getERC721Address()).withdrawFunds(msg.sender,_amount,_asset) (PoolTracker.s
3771)
        - IJustCausePool(_pool).withdraw(_asset,_amount,msg.sender,_isETH) (PoolTracker.sol#3772)
        Event emitted after the call(s):
        - WithdrawDeposit(msg.sender,_pool,_asset,_amount) (PoolTracker.sol#3773)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (PoolTracker.sol#202-214) uses assembly
        - INLINE ASM (PoolTracker.sol#207-210)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (PoolTracker.sol#1135-1157) uses assembly
        - INLINE ASM (PoolTracker.sol#1149-1151)
Address.isContract(address) (PoolTracker.sol#1517-1528) uses assembly
```



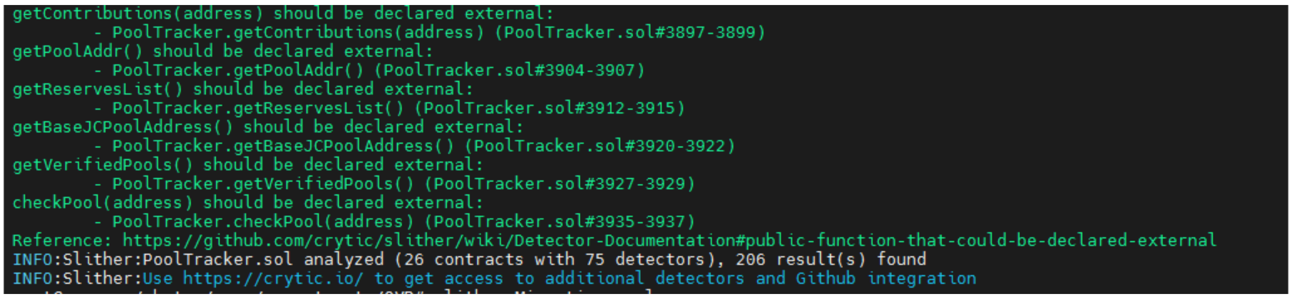
```
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (PoolTracker.sol#1135-1157) uses assembly
        - INLINE ASM (PoolTracker.sol#1149-1151)
Address.isContract(address) (PoolTracker.sol#1517-1528) uses assembly
        - INLINE ASM (PoolTracker.sol#1524-1526)
Clones.clone(address) (PoolTracker.sol#3442-3452) uses assembly
        - INLINE ASM (PoolTracker.sol#3444-3450)
Clones.cloneDeterministic(address,bytes32) (PoolTracker.sol#3461-3471) uses assembly
        - INLINE ASM (PoolTracker.sol#3463-3469)
Clones.predictDeterministicAddress(address,bytes32,address) (PoolTracker.sol#3476-3492) uses assembly
        - INLINE ASM (PoolTracker.sol#3482-3491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
JustCausePool.initialize(address[],string,string,string,string,address,address,address,address,bool) (PoolTracker.sol#3139-316
compares to a boolean constant:
        -require(bool,string)(isBase == false,Cannot initialize base) (PoolTracker.sol#3153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.isContract(address) (PoolTracker.sol#1517-1528) is never used and should be removed
Address.sendValue(address,uint256) (PoolTracker.sol#1546-1552) is never used and should be removed
AddressUpgradeable._revert(bytes,string) (PoolTracker.sol#202-214) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (PoolTracker.sol#81-83) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (PoolTracker.sol#91-97) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (PoolTracker.sol#110-116) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (PoolTracker.sol#124-133) is never used and should be r
ved
```



```
INFO:Detectors:
Clones.clone(address) (PoolTracker.sol#3442-3452) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_clone_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000000000000000000000) (PoolTra
r.sol#3446)
Clones.clone(address) (PoolTracker.sol#3442-3452) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_clone_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000) (
lTracker.sol#3448)
Clones.cloneDeterministic(address,bytes32) (PoolTracker.sol#3461-3471) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000000
) (PoolTracker.sol#3465)
Clones.cloneDeterministic(address,bytes32) (PoolTracker.sol#3461-3471) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000
0000000) (PoolTracker.sol#3467)
Clones.predictDeterministicAddress(address,bytes32,address) (PoolTracker.sol#3476-3492) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000000000
000000000) (PoolTracker.sol#3484)
Clones.predictDeterministicAddress(address,bytes32,address) (PoolTracker.sol#3476-3492) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3ff0000000000000
0000000000000000) (PoolTracker.sol#3486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ERC721URIStorageUpgradeable.__gap (PoolTracker.sol#1266) is never used in JCDepositorERC721 (PoolTracker.sol#1269-1419)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
balanceOf(address) should be declared external:
        - ERC721Upgradeable.balanceOf(address) (PoolTracker.sol#803-806)
name() should be declared external:
        - ERC721Upgradeable.name() (PoolTracker.sol#820-822)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (PoolTracker.sol#827-829)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (PoolTracker.sol#853-863)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (PoolTracker.sol#877-879)
```



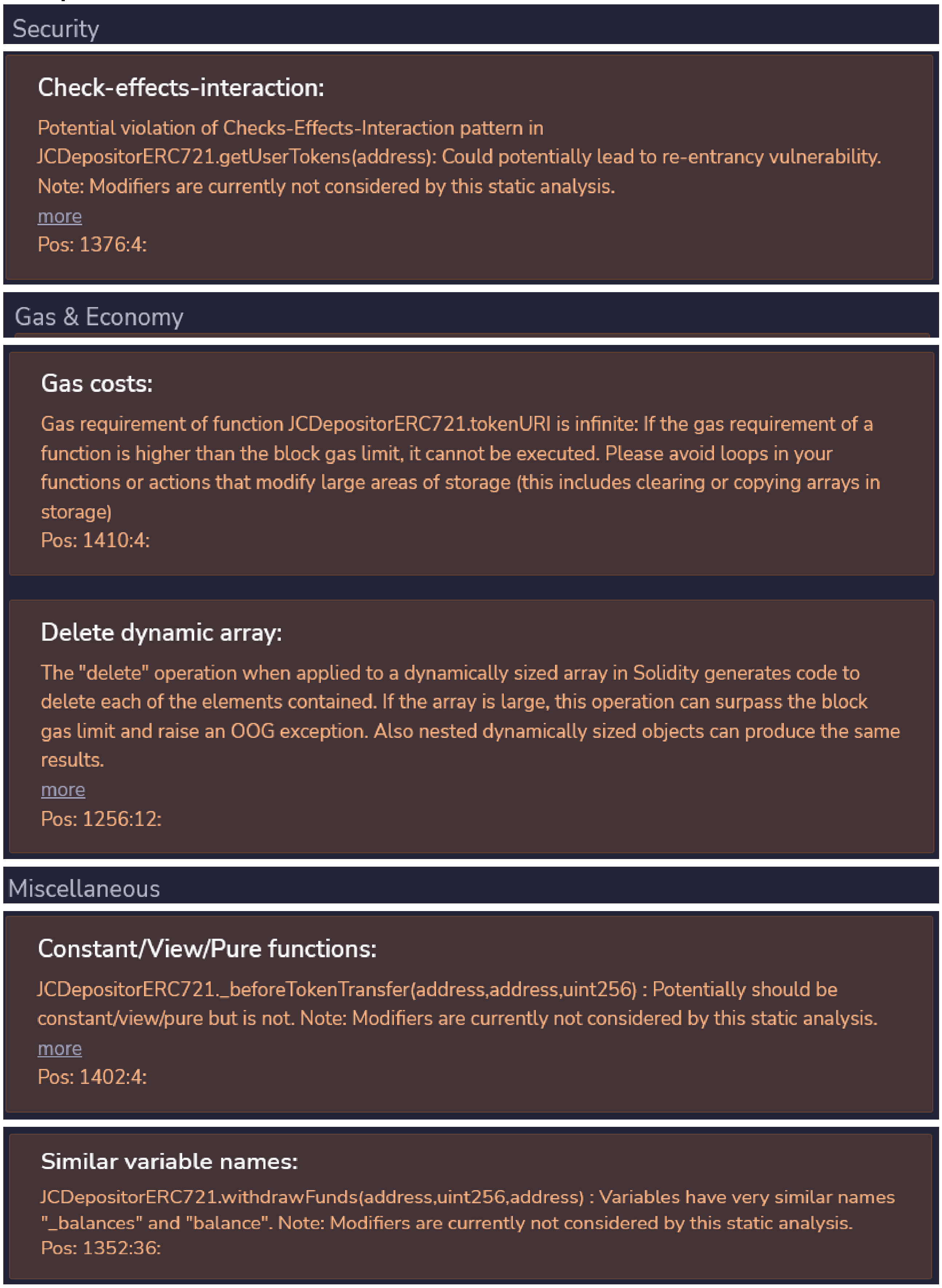
```
        - ERC721Upgradeable.setApprovalForAll(address,bool) (PoolTracker.sol#877-879)
transferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.transferFrom(address,address,uint256) (PoolTracker.sol#891-900)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (PoolTracker.sol#905-911)
initialize(address) should be declared external:
        - JCDepositorERC721.initialize(address) (PoolTracker.sol#1296-1300)
addFunds(address,uint256,uint256,address,string) should be declared external:
        - JCDepositorERC721.addFunds(address,uint256,uint256,address,string) (PoolTracker.sol#1311-1335)
getDepositInfo(uint256) should be declared external:
        - JCDepositorERC721.getDepositInfo(uint256) (PoolTracker.sol#1360-1362)
getUserBalance(uint256) should be declared external:
        - JCDepositorERC721.getUserBalance(uint256) (PoolTracker.sol#1368-1370)
getPool() should be declared external:
        - JCDepositorERC721.getPool() (PoolTracker.sol#1393-1395)
setBpFee(uint256) should be declared external:
        - PoolTracker.setBpFee(uint256) (PoolTracker.sol#3844-3846)
getBpFee() should be declared external:
        - PoolTracker.getBpFee() (PoolTracker.sol#3851-3853)
getTVL(address) should be declared external:
        - PoolTracker.getTVL(address) (PoolTracker.sol#3859-3861)
getTotalDonated(address) should be declared external:
        - PoolTracker.getTotalDonated(address) (PoolTracker.sol#3867-3869)
getDepositorERC721Address() should be declared external:
        - PoolTracker.getDepositorERC721Address() (PoolTracker.sol#3874-3876)
getReceiverPools(address) should be declared external:
        - PoolTracker.getReceiverPools(address) (PoolTracker.sol#3882-3884)
getMultiSig() should be declared external:
        - PoolTracker.getMultiSig() (PoolTracker.sol#3889-3891)
getContributions(address) should be declared external:
        - PoolTracker.getContributions(address) (PoolTracker.sol#3897-3899)
getPoolAddr() should be declared external:
        - PoolTracker.getPoolAddr() (PoolTracker.sol#3904-3907)
```



```
getContributions(address) should be declared external:
        - PoolTracker.getContributions(address) (PoolTracker.sol#3897-3899)
getPoolAddr() should be declared external:
        - PoolTracker.getPoolAddr() (PoolTracker.sol#3904-3907)
getReservesList() should be declared external:
        - PoolTracker.getReservesList() (PoolTracker.sol#3912-3915)
getBaseJCPoolAddress() should be declared external:
        - PoolTracker.getBaseJCPoolAddress() (PoolTracker.sol#3920-3922)
getVerifiedPools() should be declared external:
        - PoolTracker.getVerifiedPools() (PoolTracker.sol#3927-3929)
checkPool(address) should be declared external:
        - PoolTracker.checkPool(address) (PoolTracker.sol#3935-3937)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PoolTracker.sol analyzed (26 contracts with 75 detectors), 206 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

JCDepositorERC721.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in JCDepositorERC721.getUserTokens(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1376:4:

Gas & Economy

Gas costs:

Gas requirement of function JCDepositorERC721.tokenURI 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: 1410:4:

Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.
more
Pos: 1256:12:

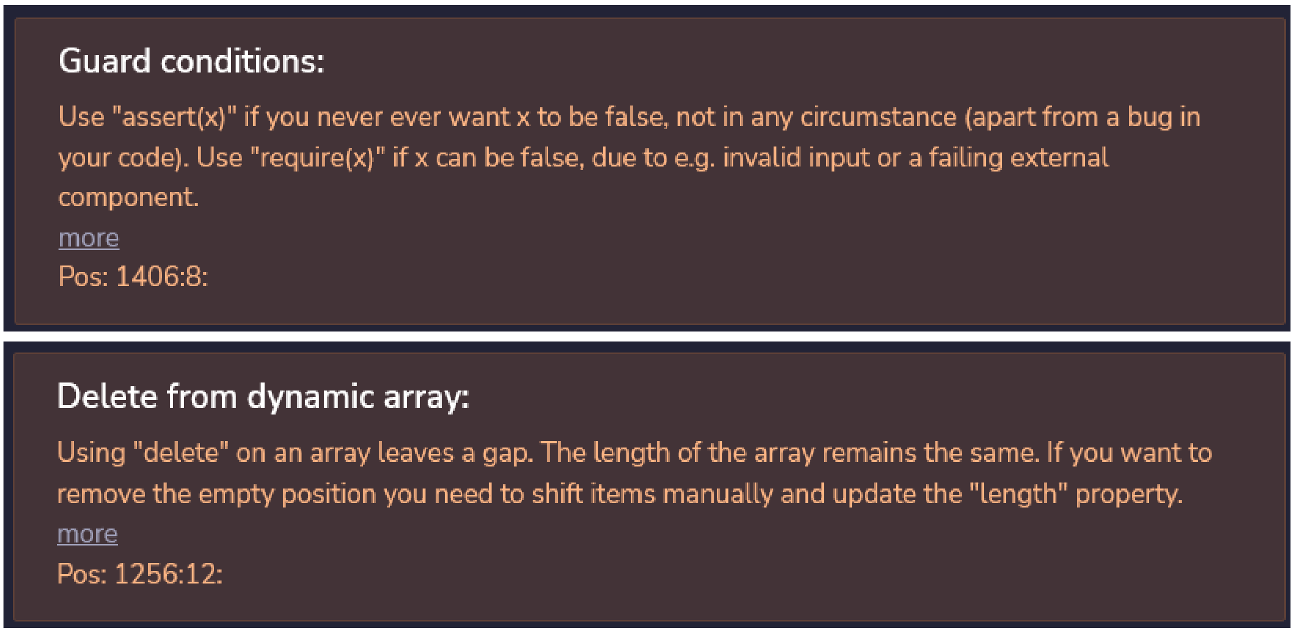
Miscellaneous

Constant/View/Pure functions:

JCDepositorERC721._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1402:4:

Similar variable names:

JCDepositorERC721.withdrawFunds(address,uint256,address) : Variables have very similar names "_balances" and "balance". Note: Modifiers are currently not considered by this static analysis.
Pos: 1352:36:



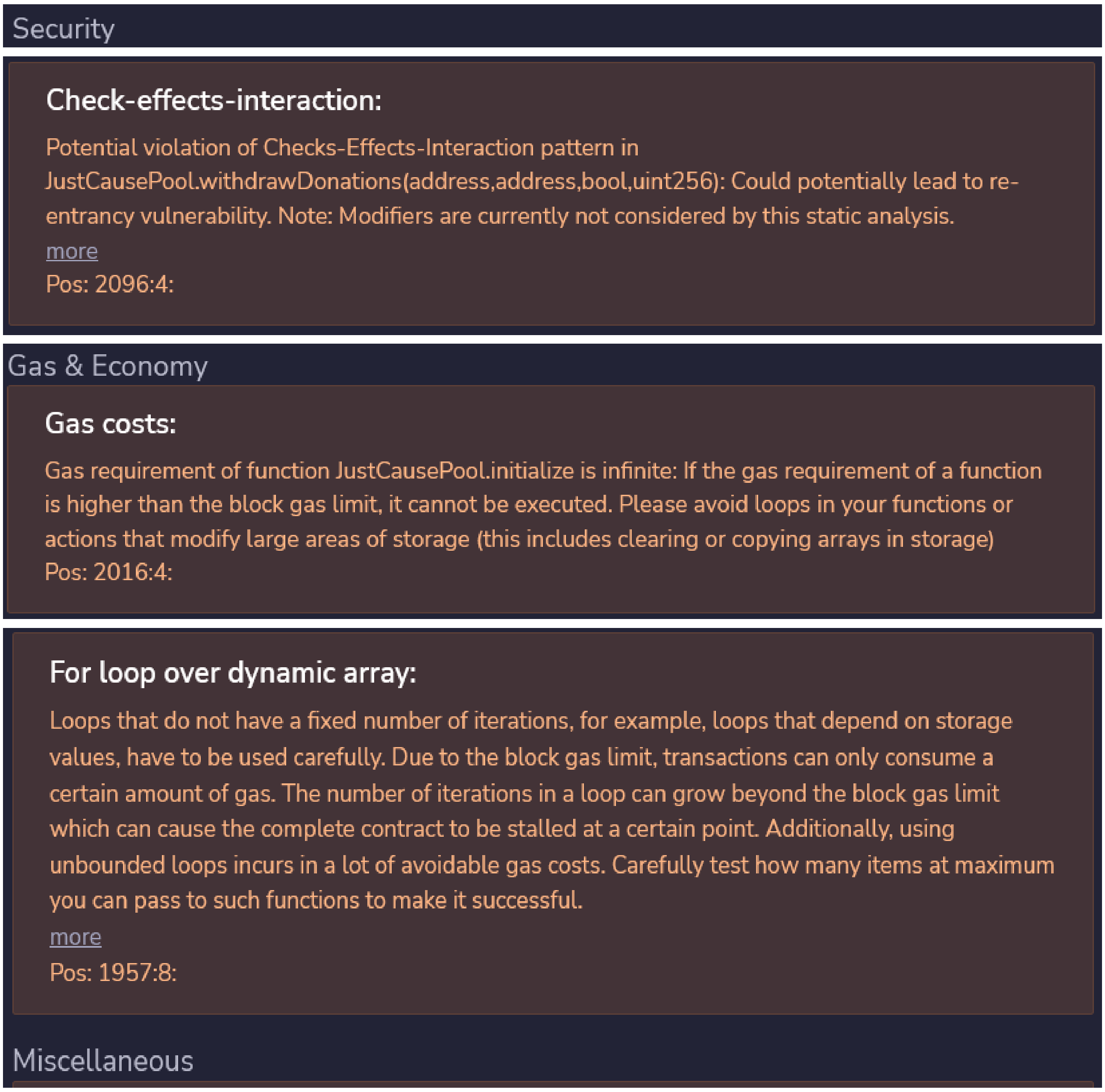
Guard conditions:

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: 1406:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1256:12:

JustCausePool.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in JustCausePool.withdrawDonations(address,address,bool,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2096:4:

Gas & Economy

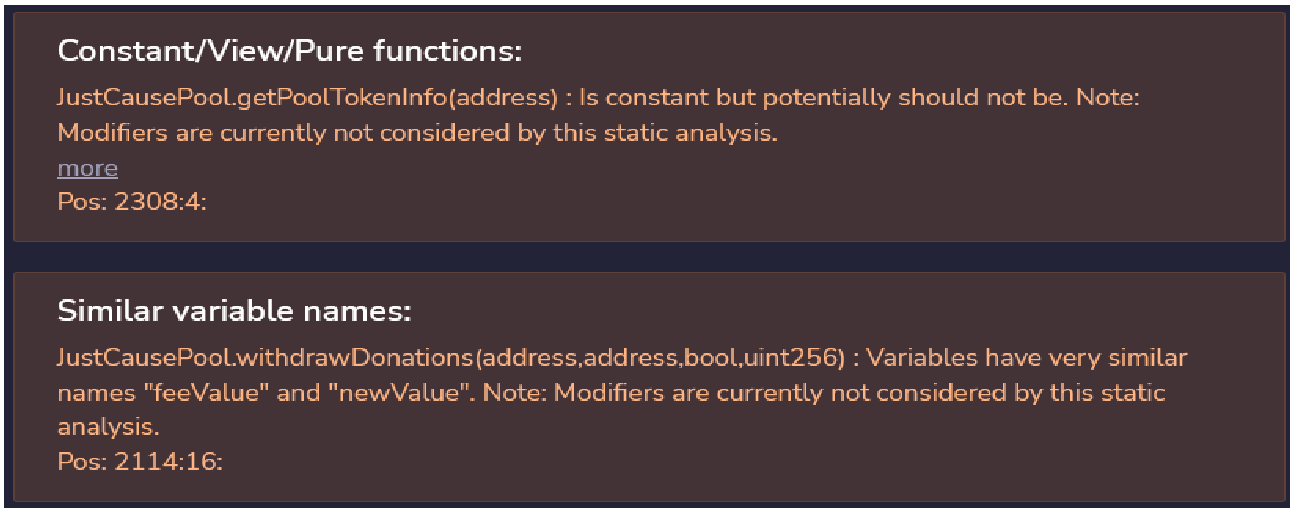
Gas costs:

Gas requirement of function JustCausePool.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: 2016:4:

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: 1957:8:

Miscellaneous

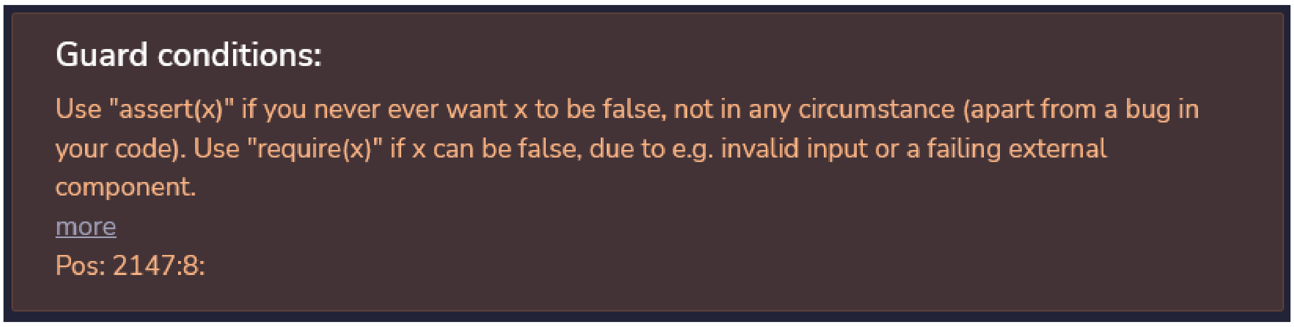


Constant/View/Pure functions:

JustCausePool.getPoolTokenInfo(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2308:4:

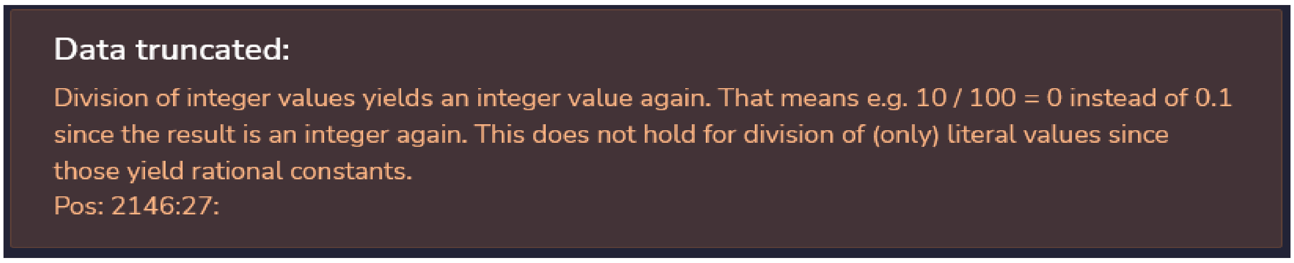
Similar variable names:

JustCausePool.withdrawDonations(address,address,bool,uint256) : Variables have very similar names "feeValue" and "newValue". Note: Modifiers are currently not considered by this static analysis.
Pos: 2114:16:



Guard conditions:

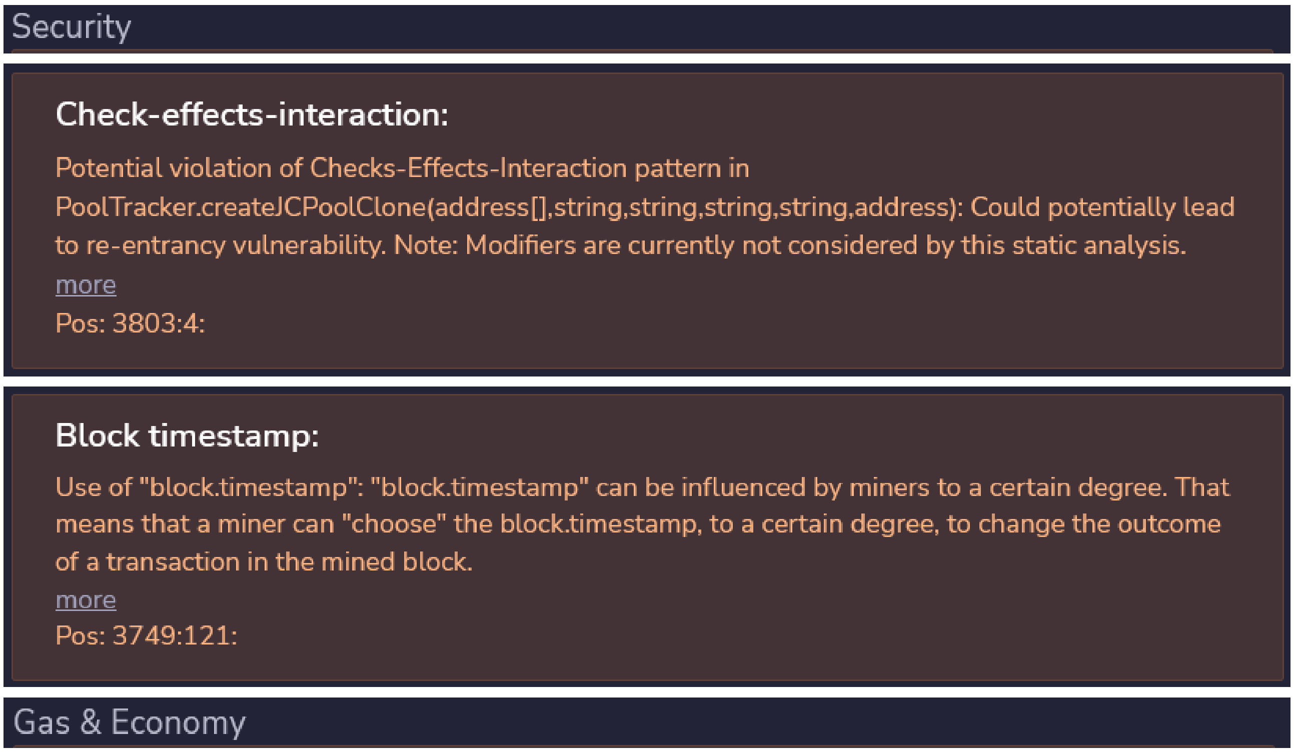
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2147:8:



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: 2146:27:

PoolTracker.sol



Security

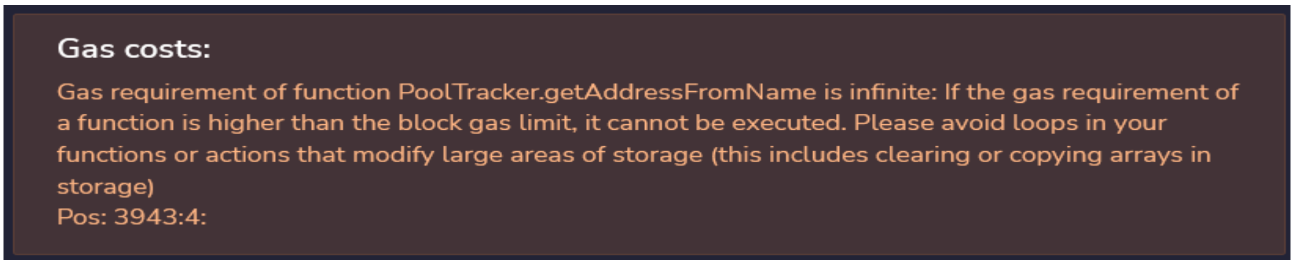
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PoolTracker.createJCPoolClone(address[],string,string,string,string,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3803:4:

Block timestamp:

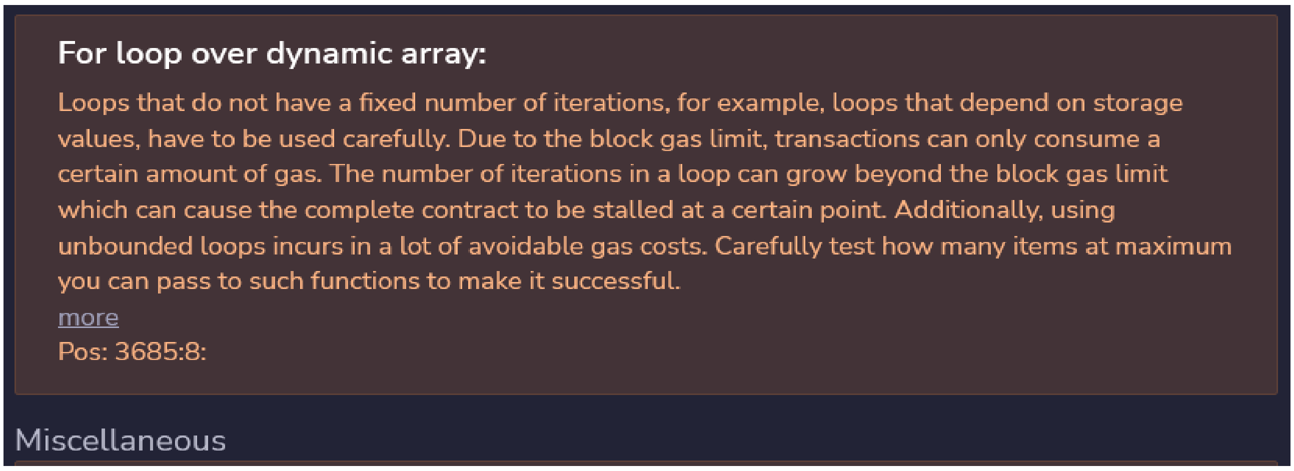
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 3749:121:

Gas & Economy



Gas costs:

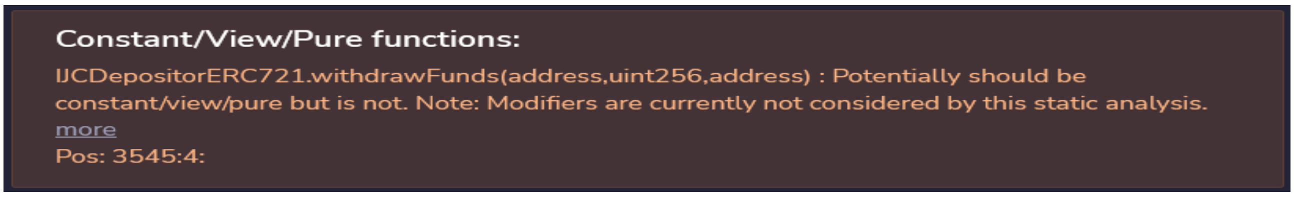
Gas requirement of function PoolTracker.getAddressFromName is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 3943:4:



For loop over dynamic array:

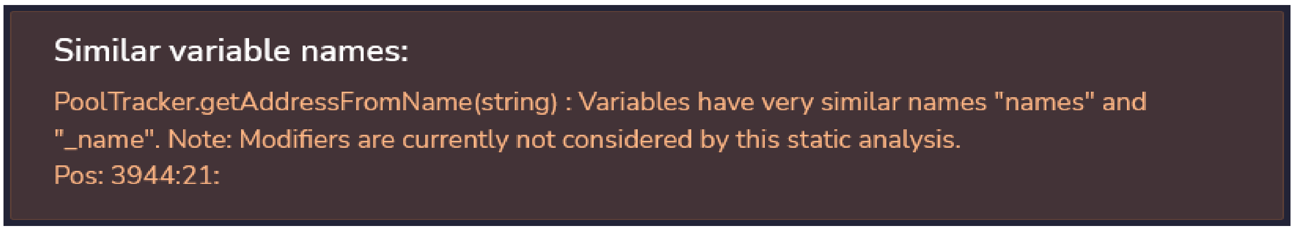
Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the 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: 3685:8:

Miscellaneous



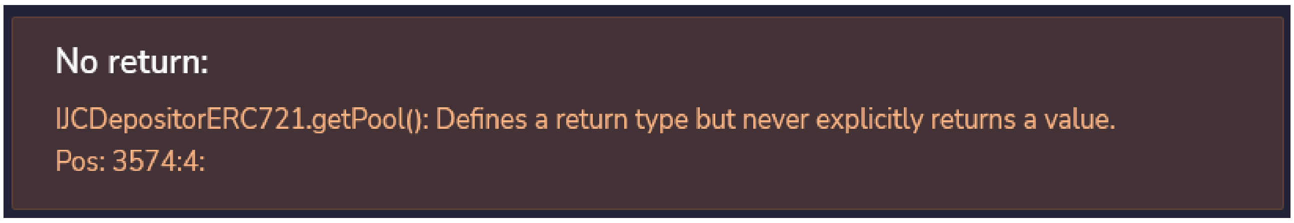
Constant/View/Pure functions:

IJCDepositorERC721.withdrawFunds(address,uint256,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3545:4:



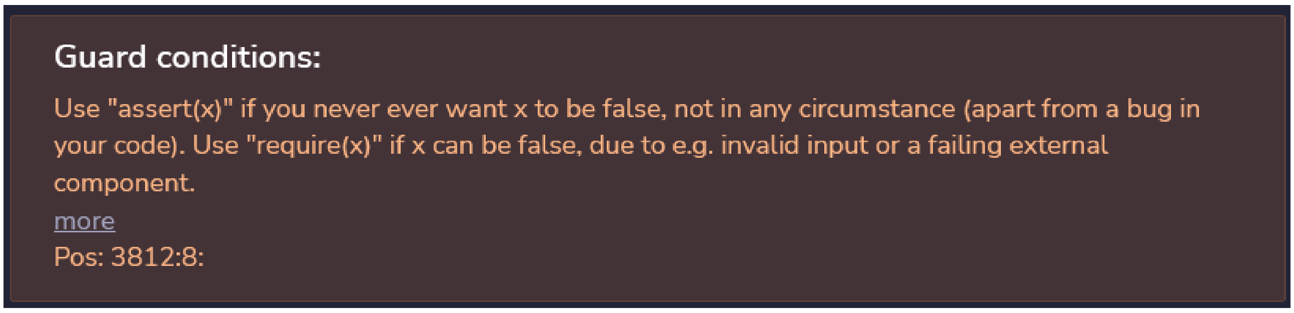
Similar variable names:

PoolTracker.getAddressFromName(string) : Variables have very similar names "names" and "_name". Note: Modifiers are currently not considered by this static analysis.
Pos: 3944:21:



No return:

IJCDepositorERC721.getPool(): Defines a return type but never explicitly returns a value.
Pos: 3574:4:

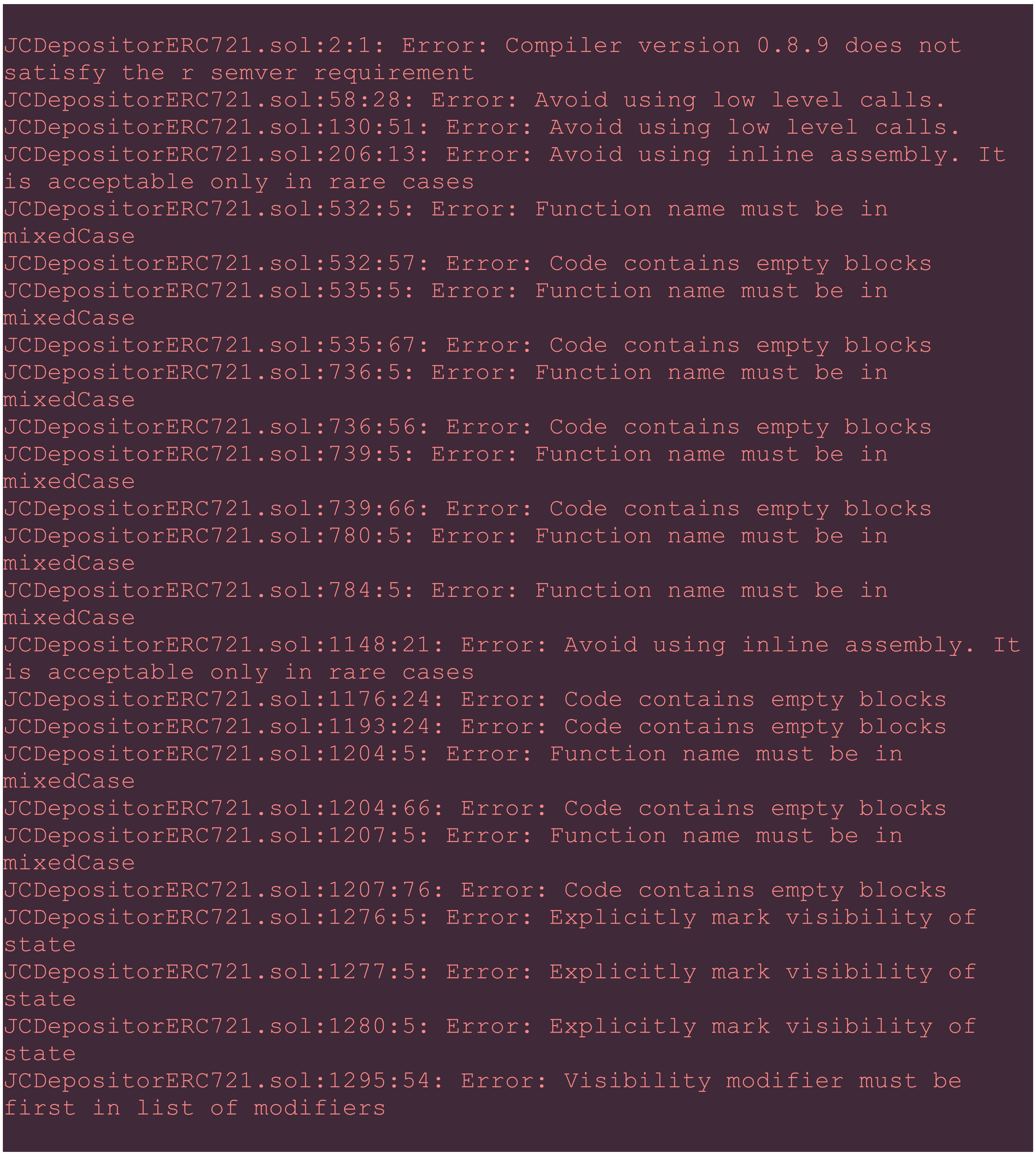


Guard conditions:

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: 3812:8:

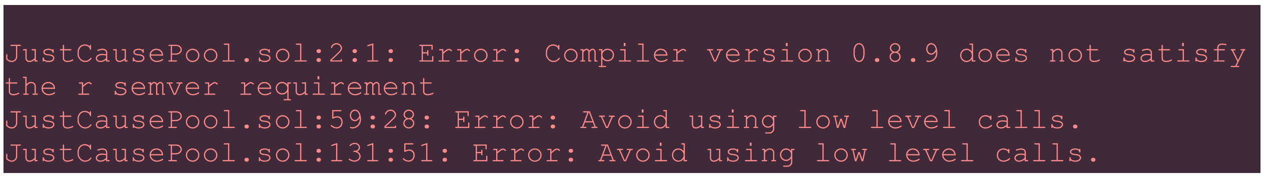
Solhint Linter

JCDepositorERC721.sol

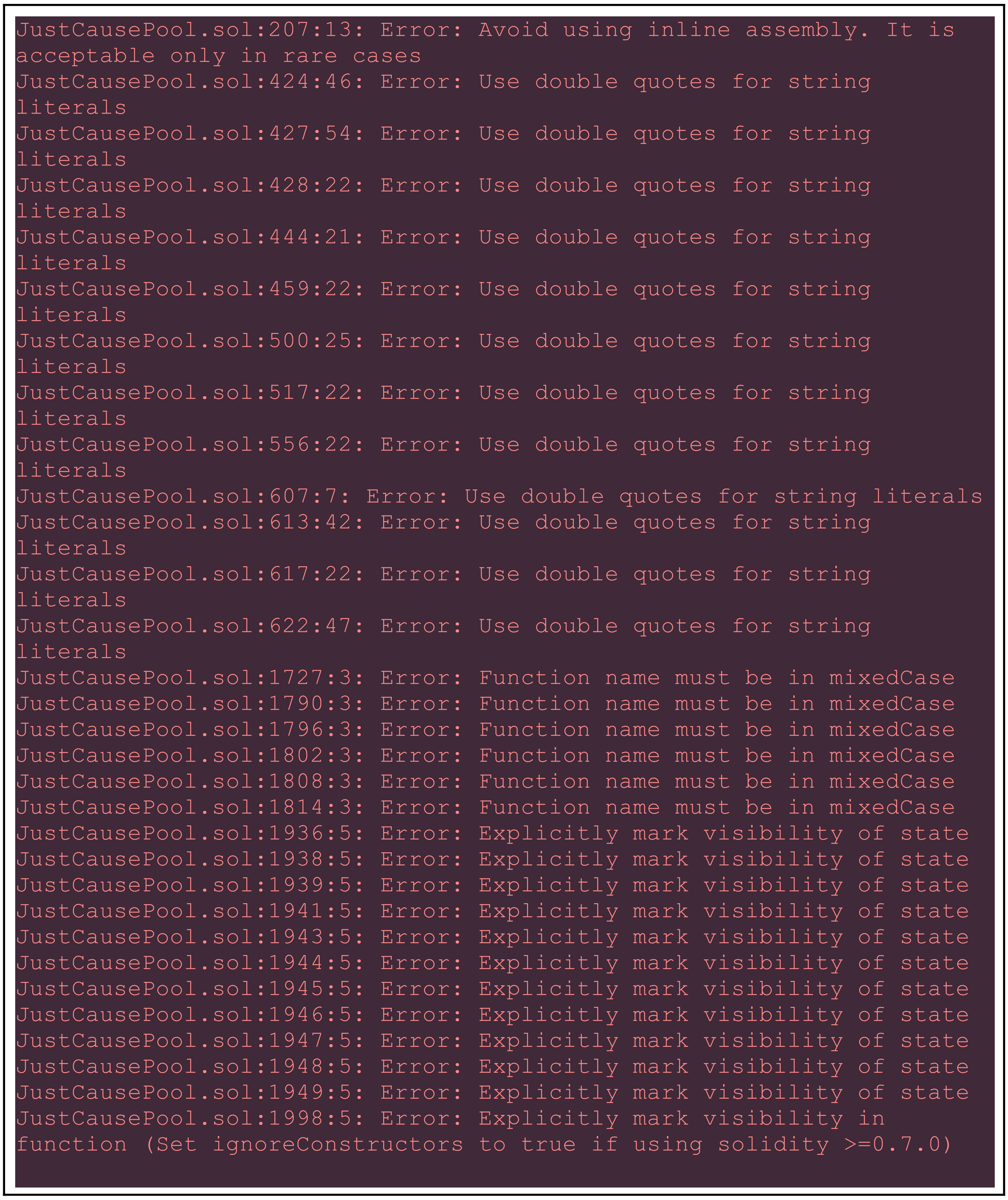


```
JCDepositorERC721.sol:2:1: Error: Compiler version 0.8.9 does not
satisfy the r semver requirement
JCDepositorERC721.sol:58:28: Error: Avoid using low level calls.
JCDepositorERC721.sol:130:51: Error: Avoid using low level calls.
JCDepositorERC721.sol:206:13: Error: Avoid using inline assembly. It
is acceptable only in rare cases
JCDepositorERC721.sol:532:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:532:57: Error: Code contains empty blocks
JCDepositorERC721.sol:535:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:535:67: Error: Code contains empty blocks
JCDepositorERC721.sol:736:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:736:56: Error: Code contains empty blocks
JCDepositorERC721.sol:739:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:739:66: Error: Code contains empty blocks
JCDepositorERC721.sol:780:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:784:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:1148:21: Error: Avoid using inline assembly. It
is acceptable only in rare cases
JCDepositorERC721.sol:1176:24: Error: Code contains empty blocks
JCDepositorERC721.sol:1193:24: Error: Code contains empty blocks
JCDepositorERC721.sol:1204:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:1204:66: Error: Code contains empty blocks
JCDepositorERC721.sol:1207:5: Error: Function name must be in
mixedCase
JCDepositorERC721.sol:1207:76: Error: Code contains empty blocks
JCDepositorERC721.sol:1276:5: Error: Explicitly mark visibility of
state
JCDepositorERC721.sol:1277:5: Error: Explicitly mark visibility of
state
JCDepositorERC721.sol:1280:5: Error: Explicitly mark visibility of
state
JCDepositorERC721.sol:1295:54: Error: Visibility modifier must be
first in list of modifiers
```

JustCausePool.sol

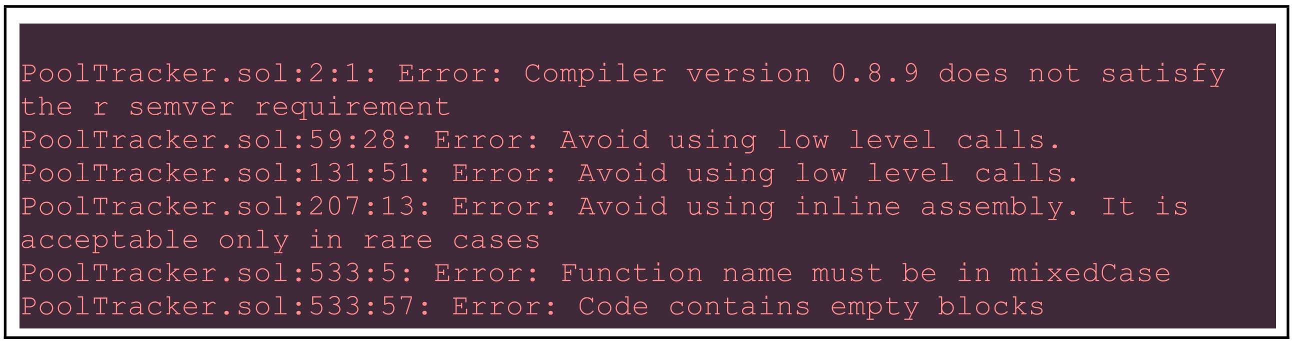


```
JustCausePool.sol:2:1: Error: Compiler version 0.8.9 does not satisfy
the r semver requirement
JustCausePool.sol:59:28: Error: Avoid using low level calls.
JustCausePool.sol:131:51: Error: Avoid using low level calls.
```

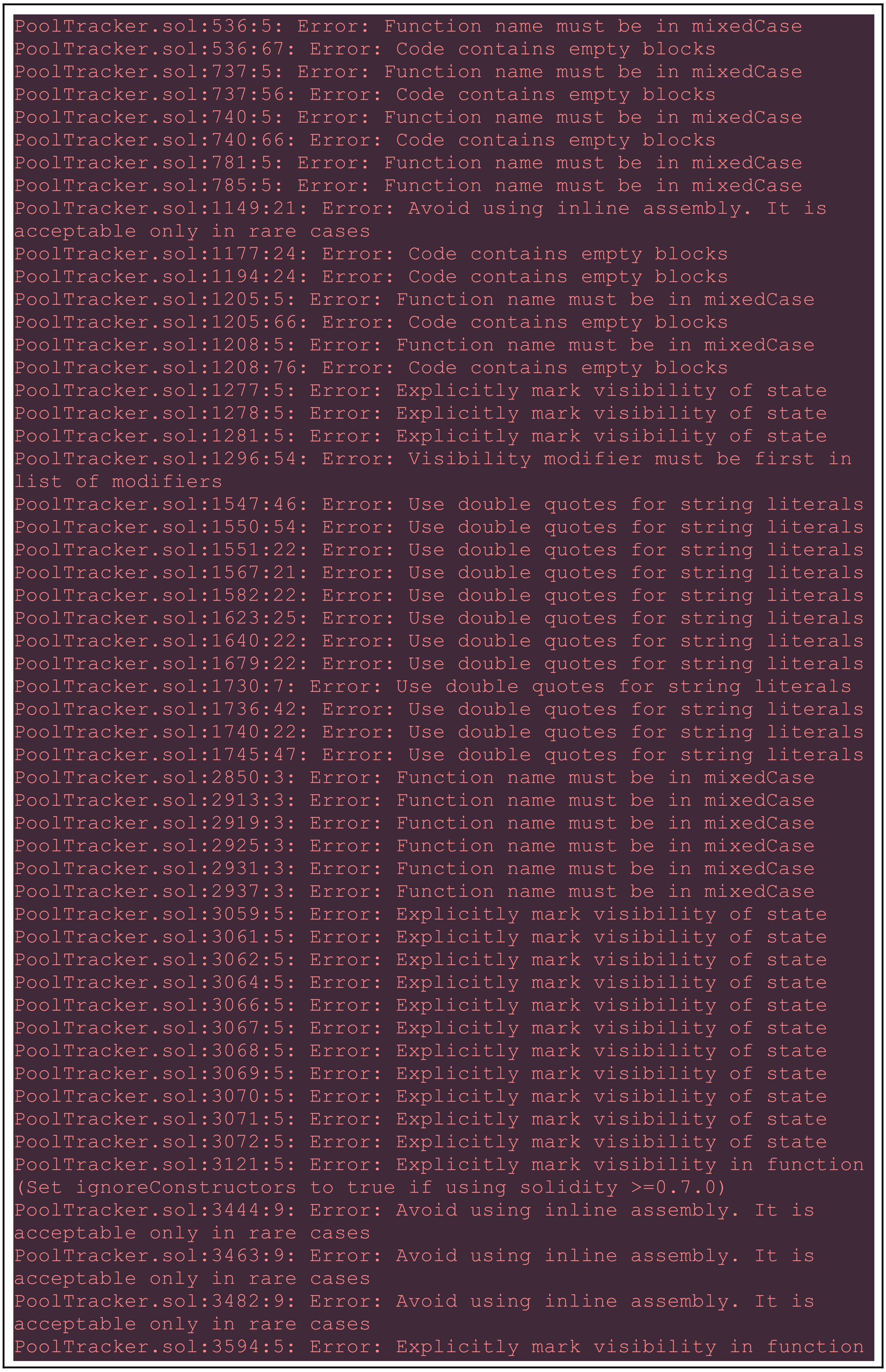


```
JustCausePool.sol:207:13: Error: Avoid using inline assembly. It is
acceptable only in rare cases
JustCausePool.sol:424:46: Error: Use double quotes for string
literals
JustCausePool.sol:427:54: Error: Use double quotes for string
literals
JustCausePool.sol:428:22: Error: Use double quotes for string
literals
JustCausePool.sol:444:21: Error: Use double quotes for string
literals
JustCausePool.sol:459:22: Error: Use double quotes for string
literals
JustCausePool.sol:500:25: Error: Use double quotes for string
literals
JustCausePool.sol:517:22: Error: Use double quotes for string
literals
JustCausePool.sol:556:22: Error: Use double quotes for string
literals
JustCausePool.sol:607:7: Error: Use double quotes for string literals
JustCausePool.sol:613:42: Error: Use double quotes for string
literals
JustCausePool.sol:617:22: Error: Use double quotes for string
literals
JustCausePool.sol:622:47: Error: Use double quotes for string
literals
JustCausePool.sol:1727:3: Error: Function name must be in mixedCase
JustCausePool.sol:1790:3: Error: Function name must be in mixedCase
JustCausePool.sol:1796:3: Error: Function name must be in mixedCase
JustCausePool.sol:1802:3: Error: Function name must be in mixedCase
JustCausePool.sol:1808:3: Error: Function name must be in mixedCase
JustCausePool.sol:1814:3: Error: Function name must be in mixedCase
JustCausePool.sol:1936:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1938:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1939:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1941:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1943:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1944:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1945:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1946:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1947:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1948:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1949:5: Error: Explicitly mark visibility of state
JustCausePool.sol:1998:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

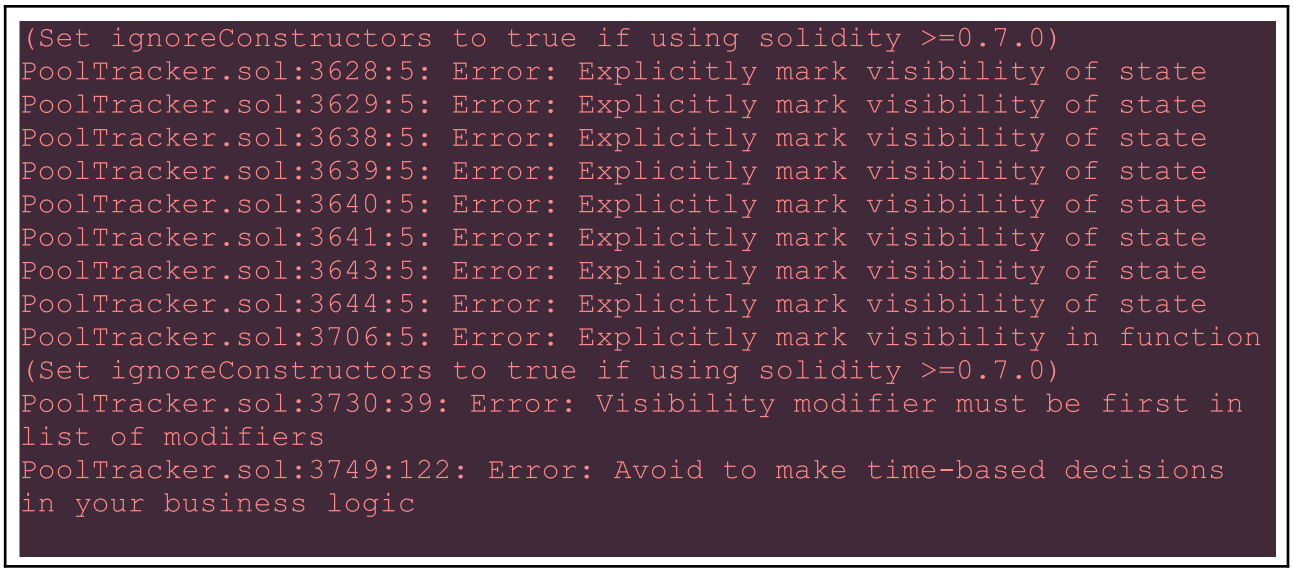
PoolTracker.sol



```
PoolTracker.sol:2:1: Error: Compiler version 0.8.9 does not satisfy
the r semver requirement
PoolTracker.sol:59:28: Error: Avoid using low level calls.
PoolTracker.sol:131:51: Error: Avoid using low level calls.
PoolTracker.sol:207:13: Error: Avoid using inline assembly. It is
acceptable only in rare cases
PoolTracker.sol:533:5: Error: Function name must be in mixedCase
PoolTracker.sol:533:57: Error: Code contains empty blocks
```



```
PoolTracker.sol:536:5: Error: Function name must be in mixedCase
PoolTracker.sol:536:67: Error: Code contains empty blocks
PoolTracker.sol:737:5: Error: Function name must be in mixedCase
PoolTracker.sol:737:56: Error: Code contains empty blocks
PoolTracker.sol:740:5: Error: Function name must be in mixedCase
PoolTracker.sol:740:66: Error: Code contains empty blocks
PoolTracker.sol:781:5: Error: Function name must be in mixedCase
PoolTracker.sol:785:5: Error: Function name must be in mixedCase
PoolTracker.sol:1149:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
PoolTracker.sol:1177:24: Error: Code contains empty blocks
PoolTracker.sol:1194:24: Error: Code contains empty blocks
PoolTracker.sol:1205:5: Error: Function name must be in mixedCase
PoolTracker.sol:1205:66: Error: Code contains empty blocks
PoolTracker.sol:1208:5: Error: Function name must be in mixedCase
PoolTracker.sol:1208:76: Error: Code contains empty blocks
PoolTracker.sol:1277:5: Error: Explicitly mark visibility of state
PoolTracker.sol:1278:5: Error: Explicitly mark visibility of state
PoolTracker.sol:1281:5: Error: Explicitly mark visibility of state
PoolTracker.sol:1296:54: Error: Visibility modifier must be first in
list of modifiers
PoolTracker.sol:1547:46: Error: Use double quotes for string literals
PoolTracker.sol:1550:54: Error: Use double quotes for string literals
PoolTracker.sol:1551:22: Error: Use double quotes for string literals
PoolTracker.sol:1567:21: Error: Use double quotes for string literals
PoolTracker.sol:1582:22: Error: Use double quotes for string literals
PoolTracker.sol:1623:25: Error: Use double quotes for string literals
PoolTracker.sol:1640:22: Error: Use double quotes for string literals
PoolTracker.sol:1679:22: Error: Use double quotes for string literals
PoolTracker.sol:1730:7: Error: Use double quotes for string literals
PoolTracker.sol:1736:42: Error: Use double quotes for string literals
PoolTracker.sol:1740:22: Error: Use double quotes for string literals
PoolTracker.sol:1745:47: Error: Use double quotes for string literals
PoolTracker.sol:2850:3: Error: Function name must be in mixedCase
PoolTracker.sol:2913:3: Error: Function name must be in mixedCase
PoolTracker.sol:2919:3: Error: Function name must be in mixedCase
PoolTracker.sol:2925:3: Error: Function name must be in mixedCase
PoolTracker.sol:2931:3: Error: Function name must be in mixedCase
PoolTracker.sol:2937:3: Error: Function name must be in mixedCase
PoolTracker.sol:3059:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3061:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3062:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3064:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3066:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3067:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3068:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3069:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3070:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3071:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3072:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3121:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
PoolTracker.sol:3444:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
PoolTracker.sol:3463:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
PoolTracker.sol:3482:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
PoolTracker.sol:3594:5: Error: Explicitly mark visibility in function
```



```
(Set ignoreConstructors to true if using solidity >=0.7.0)
PoolTracker.sol:3628:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3629:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3638:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3639:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3640:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3641:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3643:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3644:5: Error: Explicitly mark visibility of state
PoolTracker.sol:3706:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
PoolTracker.sol:3730:39: Error: Visibility modifier must be first in
list of modifiers
PoolTracker.sol:3749:122: Error: Avoid to make time-based decisions
in your business logic
```

Software analysis result:

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

EtherAuthority