AntiHACK.me

SMART CONTRACT

Security Audit Report

Project:	Morpheus Labs MITx Staking Campaign
Platform:	Ethereum or Polygon
Language:	Solidity
Date:	November 25th, 2021

Table of contents

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

Introduction

AntiHACK.me was contracted by the Morpheus Labs team to perform the Security audit of the Morpheus Labs Staking Campaign 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 November 25th, 2021.

The purpose of this audit was to address the following:

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

Project Background

Morpheus Labs Staking Campaign is a simple staking smart contract which can be used for multiple campaigns. Users can stake tokens defined by campaigns for a specific time and then claim their tokens with rewards. This audit only considers Morpheus Labs Staking Campaign smart contracts, and does not cover any other smart contracts in the platform.

Audit scope

The code for the audit was taken from the following official links:

https://github.com/Morpheuslabs-io/seed-staking-sc-v1/tree/main/contracts-v1

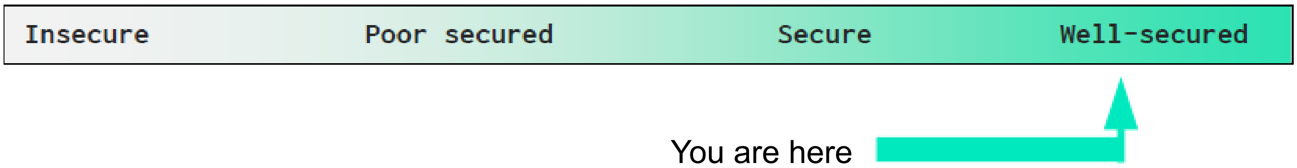
Name	Code Review and Security Analysis Report for Morpheus Labs Staking Campaign Smart Contracts
Platform	**Ethereum or Polygon / Solidity**
File 1	StakingCampaign.sol
File 1 MD5 Hash	BF278FA31972E0E50DA440D8FA5E24C7
File 2	StakingCampaignManager.sol
File 2 MD5 Hash	6FEF47FC6B7B6936BC5348AC0F9DE823
Audit Date	November 25th, 2021

Claimed Smart Contracts Features

Claimed Feature Detail	Our Observation
File 1: StakingCampaign.sol ● The admin can set the staking campaign name, total day of stake,annual percentage rate, expiration time of the campaign, etc.	**YES, This is valid.**
File 2: StakingCampaignManager.sol ● The admin can access to enable or disable, add, clear Authorization, add or clear AuthorizedBatch addresses, etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Well Secured"**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.



We used various tools like MythX, Slither 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 the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 2 low and some very low-level issues.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Random number generation/use vulnerability	Passed
	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
	Unnecessary code	Moderated
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 2 smart contracts. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts.

The libraries in the Morpheus Labs Staking Campaign contracts 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 Morpheus Labs Staking Campaign contracts.

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

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

Documentation

We were given a Morpheus Labs Staking Campaign smart contract code in the form of a github web link code. The details of that code are mentioned above in the table.

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

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well-known industry standard open-source projects. And their core code blocks are written well.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

StakingCampaign.sol

(1) Interface

(a) IERC20

(2) Struct

(a) StackingInfo

(3) Events

(a) event Deposited (address indexed sender, uint seq, uint amount, uint256 timestamp);

(b) event Claimed (address indexed sender, uint seq, uint amount, uint reward, uint256 timestamp);

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	deposit	external	Passed	No Issue
3	claim	write	Passed	No Issue
4	claimRemainingReward	write	onlyAdmin	No Issue
5	getClaimableRemainingReward	read	Passed	No Issue
6	getStakings	read	Passed	No Issue
7	getCampaignInfo	read	Passed	No Issue
8	transferOwnership	write	onlyAdmin	No Issue

StakingCampaignManager.sol

(1) Interface

(a) IERC20

(2) Inherited contracts

(a) EIP712MetaTransaction

(b) EIP712Base

(c) StakingCampaign

(3) Struct

(a) StackingInfo

(b) EIP712Domain

(c) MetaTransaction

(d) AddressBlockNum

(4) Events

(a) event Deposited (address indexed sender, uint seq, uint amount,uint256 timestamp);

(b) event Claimed (address indexed sender,uint seq, uint amount, uint reward, uint256 timestamp);

(c) event MetaTransactionExecuted(address userAddress,address payable relayerAddress,bytes functionSignature);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	onlyAdmin	modifier	Passed	No Issue
3	isAuthorized	modifier	Passed	No Issue
4	enableAuthorization	write	access only Admin	No Issue
5	disableAuthorization	write	access only Admin	No Issue
6	addAuthorized	write	access only Admin	No Issue

7	addAuthorizedBatch	write	Infinite loop possibility	Refer Audit Findings
8	clearAuthorized	write	access only Admin	No Issue
9	clearAuthorizedBatch	write	Infinite loop possibility	Refer Audit Findings
10	checkAuthorized	read	Passed	No Issue
11	getCampaignAddressBlockNumListCount	external	Passed	No Issue
12	getCampaignAddressBlockNumAtIndex	external	Passed	No Issue
13	deployCampaign	external	Critical operation lacks event log	Refer Audit Findings
14	deposit	external	Passed	No Issue
15	claim	external	Passed	No Issue
16	claimRemainingReward	external	Critical operation lacks event log	Refer Audit Findings
17	getClaimableRemainningReward	external	Passed	No Issue
18	getStakings	external	Passed	No Issue
19	getCampaignInfo	external	Passed	No Issue
20	executeMetaTransaction	write	Passed	No Issue
21	hashMetaTransaction	internal	Passed	No Issue
22	msgSender	internal	Passed	No Issue
23	getNonce	read	Passed	No Issue
24	verify	internal	Passed	No Issue
25	getDomainSeparator	read	Passed	No Issue
26	toTypedMessageHash	internal	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log: **-StakingCampaignManager.sol**

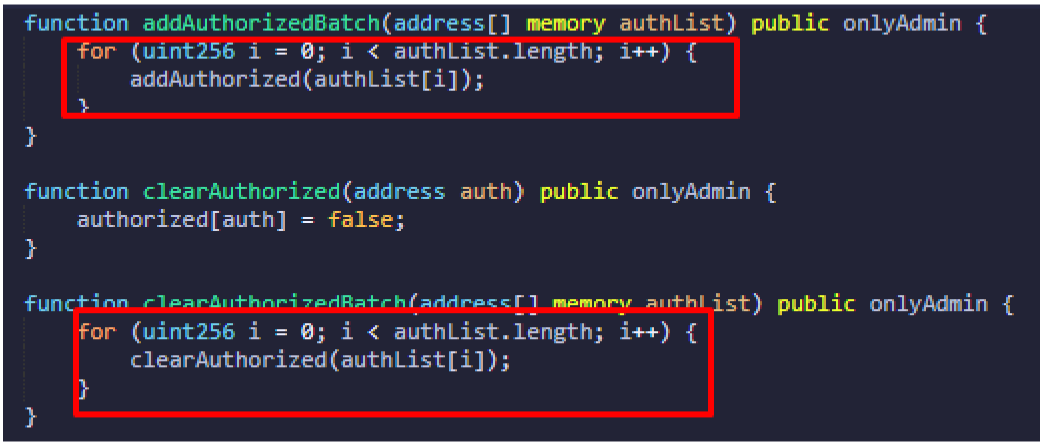
Some functions should log for events.

Functions are:

- deployCampaign
- addAuthorizedBatch
- clearAuthorizedBatch
- claimRemainingReward

Resolution: We suggest adding a log for these functions.

(2) Infinite loop possibility:**-StakingCampaignManager.sol**



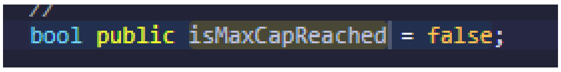
```
function addAuthorizedBatch(address[] memory authList) public onlyAdmin {
    for (uint256 i = 0; i < authList.length; i++) {
        addAuthorized(authList[i]);
    }
}

function clearAuthorized(address auth) public onlyAdmin {
    authorized[auth] = false;
}

function clearAuthorizedBatch(address[] memory authList) public onlyAdmin {
    for (uint256 i = 0; i < authList.length; i++) {
        clearAuthorized(authList[i]);
    }
}
```

In these functions, large array lengths can be the cause of high gas issues.

Resolution: We suggest checking array length before executing the for loop.

Very Low / Discussion / Best practices:

(1) Unused variable:**-StakingCampaign.sol**



```
//
bool public isMaxCapReached = false;
```

This variable has been defined and set in the deposit function. But not used anywhere in code.

Resolution: We suggest removing this unused variable.

Centralization

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

- claimRemainingReward: The StakingCampaign Admin owner can claim remaining rewards.
- enableAuthorization: StakingCampaignManager admin can enable authorization status true.
- disableAuthorization: StakingCampaignManager admin can enable authorization status false.
- addAuthorized: StakingCampaignManager admin can add a new authorized address.
- addAuthorizedBatch: StakingCampaignManager admin can add a new authorized address list.
- clearAuthorized: StakingCampaignManager admin can clear authorized addresses.
- clearAuthorizedBatch: StakingCampaignManager admin can clear authorized lists.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues, but they are not critical. So, **it's good to go to production**.

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

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

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

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

AntiHACK.me Disclaimer

AntiHACK.me 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).

Because 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.

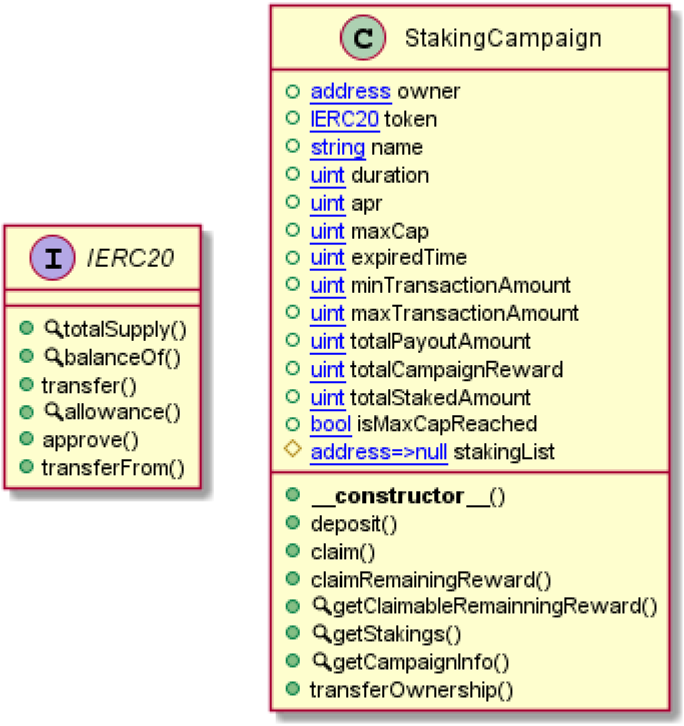
Technical Disclaimer

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.

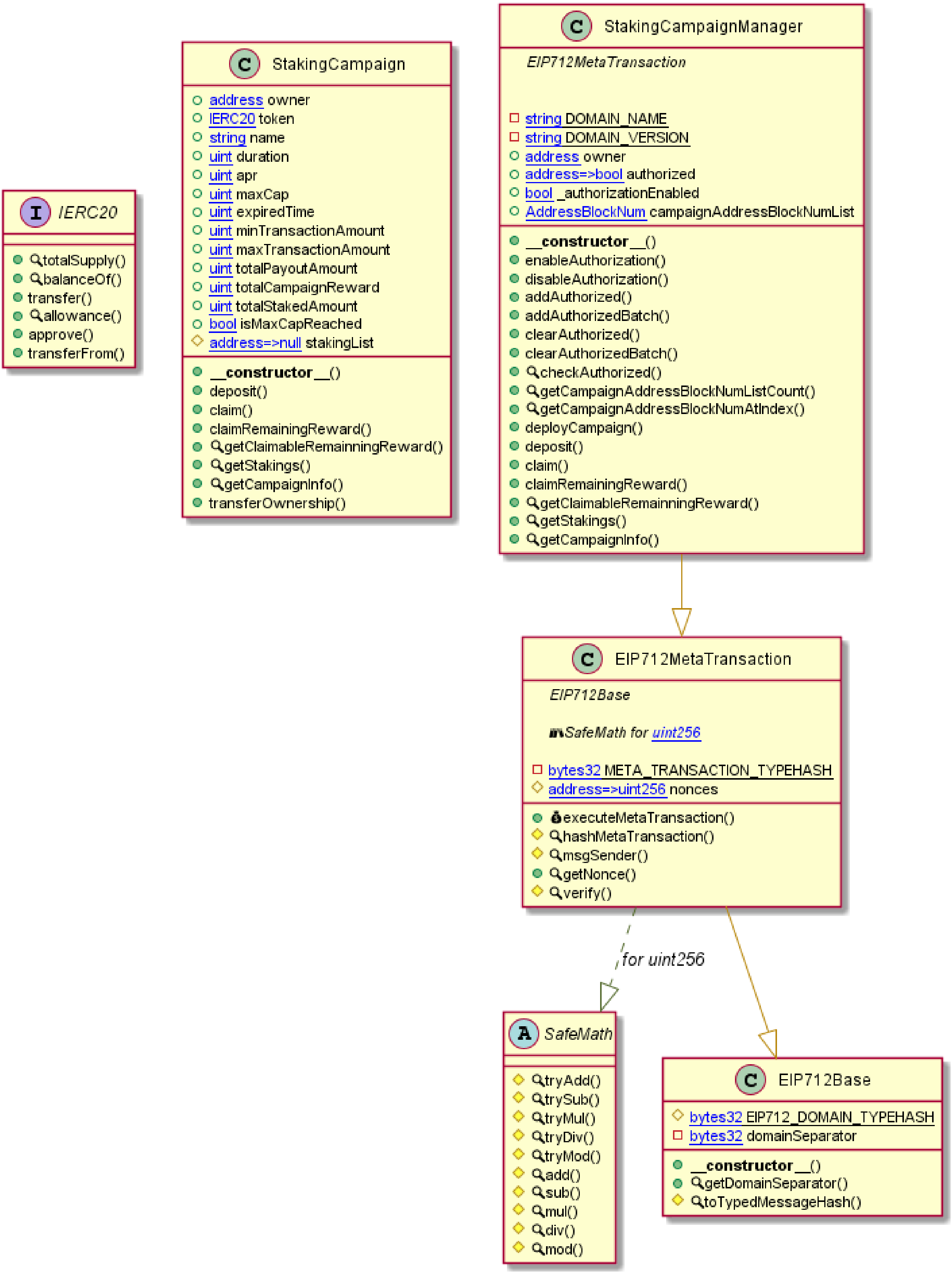
Appendix

Code Flow Diagram - Morpheus Labs Staking Campaign Diagram

StakingCampaign Diagram

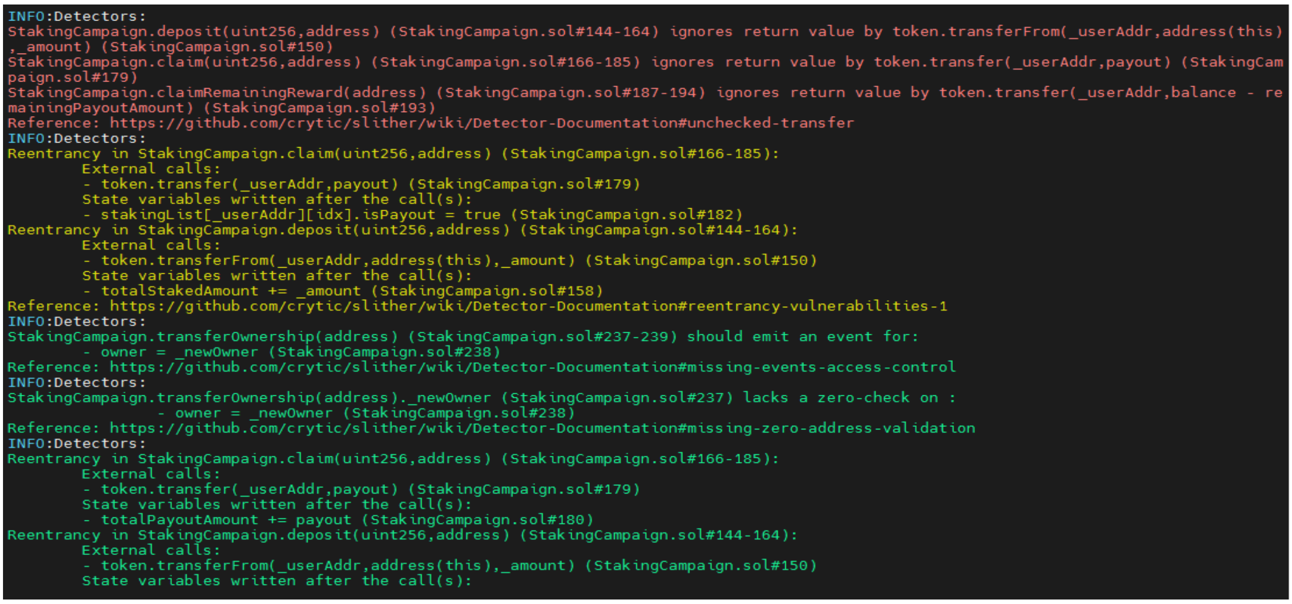


StakingCampaignManager Diagram

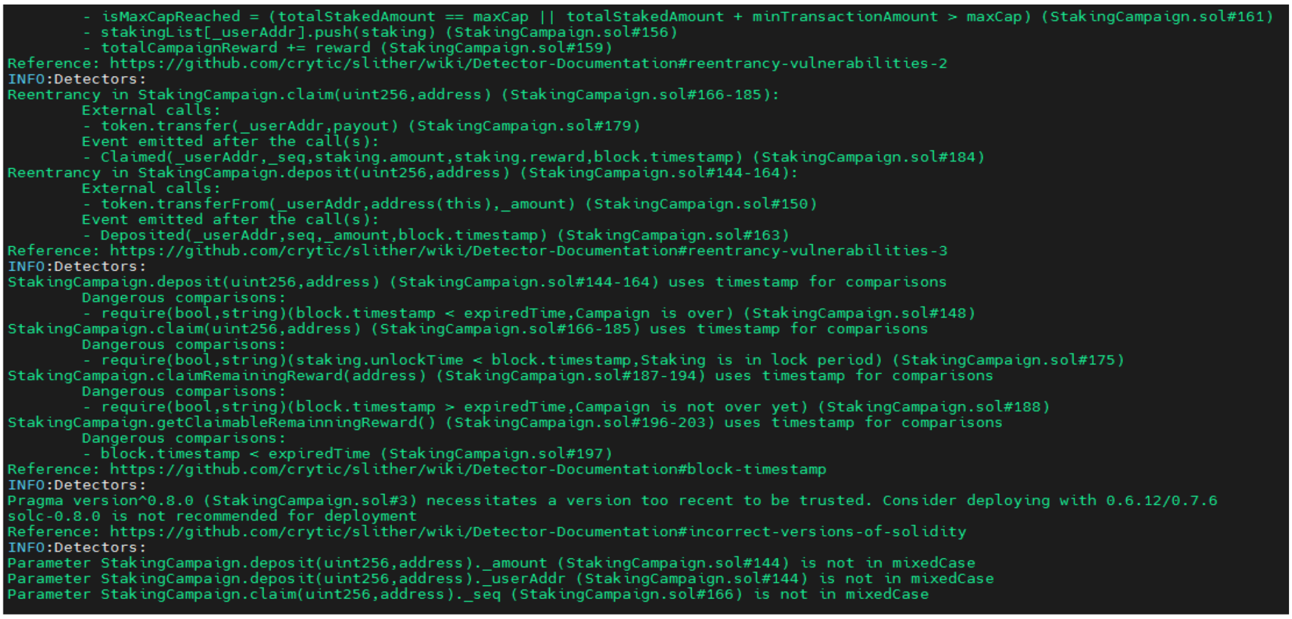


Slither Results Log

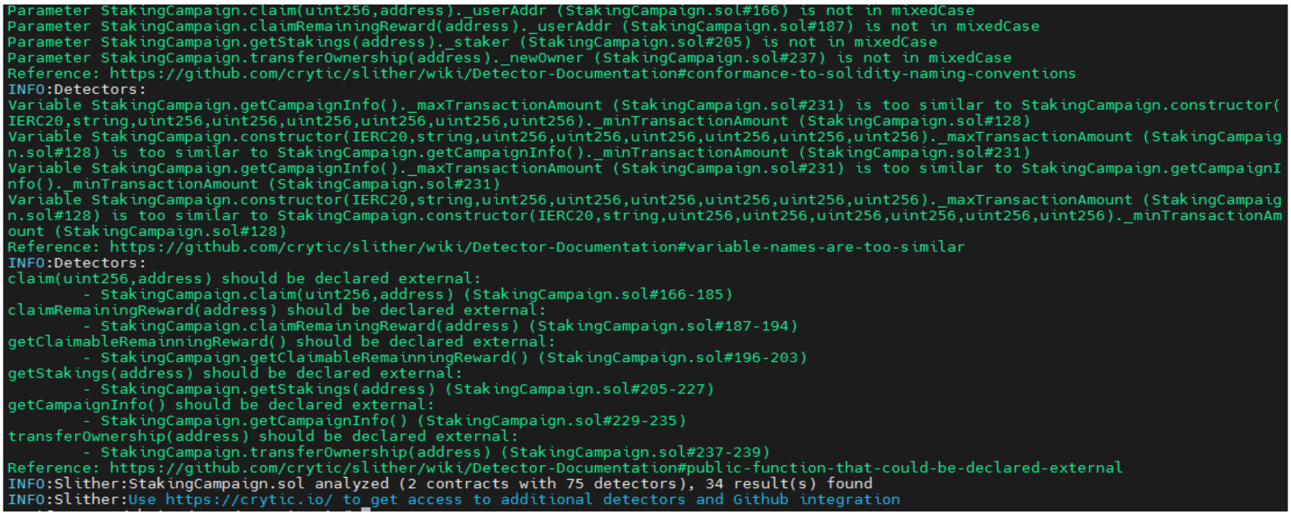
Slither log >> StakingCampaign.sol



```
INFO:Detectors:
StakingCampaign.deposit(uint256,address) (StakingCampaign.sol#144-164) ignores return value by token.transferFrom(_userAddr,address(this)
,_amount) (StakingCampaign.sol#150)
StakingCampaign.claim(uint256,address) (StakingCampaign.sol#166-185) ignores return value by token.transfer(_userAddr,payout) (StakingCam
paign.sol#179)
StakingCampaign.claimRemainingReward(address) (StakingCampaign.sol#187-194) ignores return value by token.transfer(_userAddr,balance - re
mainingPayoutAmount) (StakingCampaign.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reentrancy in StakingCampaign.claim(uint256,address) (StakingCampaign.sol#166-185):
        External calls:
        - token.transfer(_userAddr,payout) (StakingCampaign.sol#179)
        State variables written after the call(s):
        - stakingList[_userAddr][idx].isPayout = true (StakingCampaign.sol#182)
Reentrancy in StakingCampaign.deposit(uint256,address) (StakingCampaign.sol#144-164):
        External calls:
        - token.transferFrom(_userAddr,address(this),_amount) (StakingCampaign.sol#150)
        State variables written after the call(s):
        - totalStakedAmount += _amount (StakingCampaign.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StakingCampaign.transferOwnership(address) (StakingCampaign.sol#237-239) should emit an event for:
        - owner = _newOwner (StakingCampaign.sol#238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StakingCampaign.transferOwnership(address)._newOwner (StakingCampaign.sol#237) lacks a zero-check on :
                - owner = _newOwner (StakingCampaign.sol#238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in StakingCampaign.claim(uint256,address) (StakingCampaign.sol#166-185):
        External calls:
        - token.transfer(_userAddr,payout) (StakingCampaign.sol#179)
        State variables written after the call(s):
        - totalPayoutAmount += payout (StakingCampaign.sol#180)
Reentrancy in StakingCampaign.deposit(uint256,address) (StakingCampaign.sol#144-164):
        External calls:
        - token.transferFrom(_userAddr,address(this),_amount) (StakingCampaign.sol#150)
        State variables written after the call(s):
```

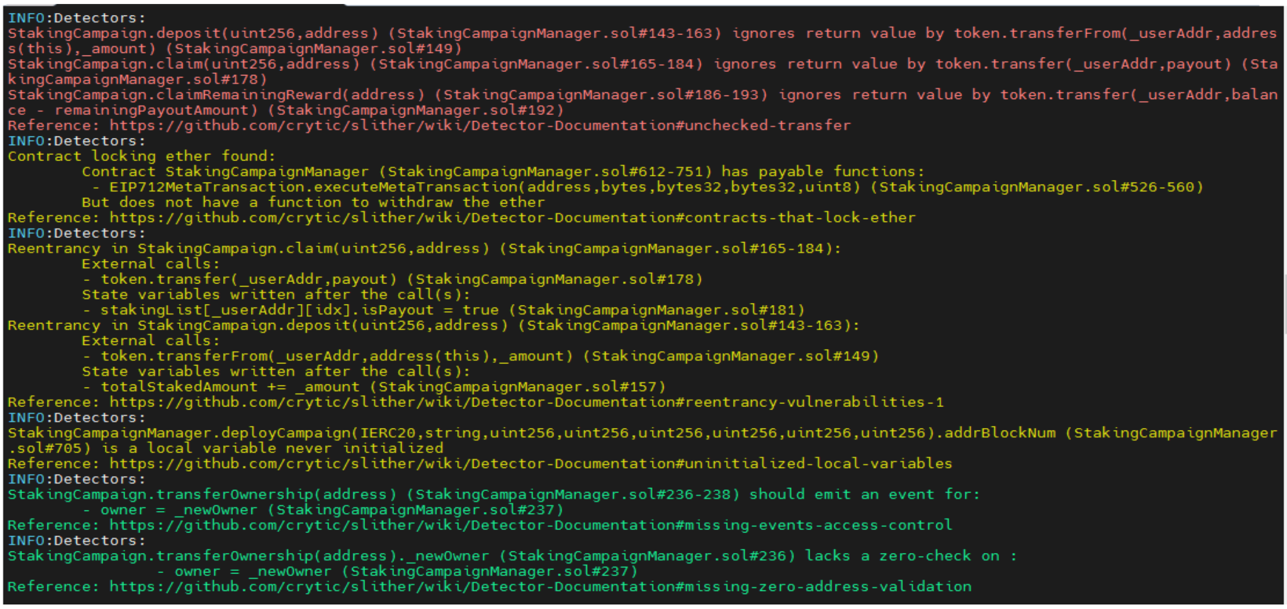


```
        - isMaxCapReached = (totalStakedAmount == maxCap || totalStakedAmount + minTransactionAmount > maxCap) (StakingCampaign.sol#161)
        - stakingList[_userAddr].push(staking) (StakingCampaign.sol#156)
        - totalCampaignReward += reward (StakingCampaign.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in StakingCampaign.claim(uint256,address) (StakingCampaign.sol#166-185):
        External calls:
        - token.transfer(_userAddr,payout) (StakingCampaign.sol#179)
        Event emitted after the call(s):
        - Claimed(_userAddr,_seq,staking.amount,staking.reward,block.timestamp) (StakingCampaign.sol#184)
Reentrancy in StakingCampaign.deposit(uint256,address) (StakingCampaign.sol#144-164):
        External calls:
        - token.transferFrom(_userAddr,address(this),_amount) (StakingCampaign.sol#150)
        Event emitted after the call(s):
        - Deposited(_userAddr,seq,_amount,block.timestamp) (StakingCampaign.sol#163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingCampaign.deposit(uint256,address) (StakingCampaign.sol#144-164) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < expiredTime,Campaign is over) (StakingCampaign.sol#148)
StakingCampaign.claim(uint256,address) (StakingCampaign.sol#166-185) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(staking.unlockTime < block.timestamp,Staking is in lock period) (StakingCampaign.sol#175)
StakingCampaign.claimRemainingReward(address) (StakingCampaign.sol#187-194) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > expiredTime,Campaign is not over yet) (StakingCampaign.sol#188)
StakingCampaign.getClaimableRemainningReward() (StakingCampaign.sol#196-203) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < expiredTime (StakingCampaign.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version^0.8.0 (StakingCampaign.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter StakingCampaign.deposit(uint256,address)._amount (StakingCampaign.sol#144) is not in mixedCase
Parameter StakingCampaign.deposit(uint256,address)._userAddr (StakingCampaign.sol#144) is not in mixedCase
Parameter StakingCampaign.claim(uint256,address)._seq (StakingCampaign.sol#166) is not in mixedCase
```

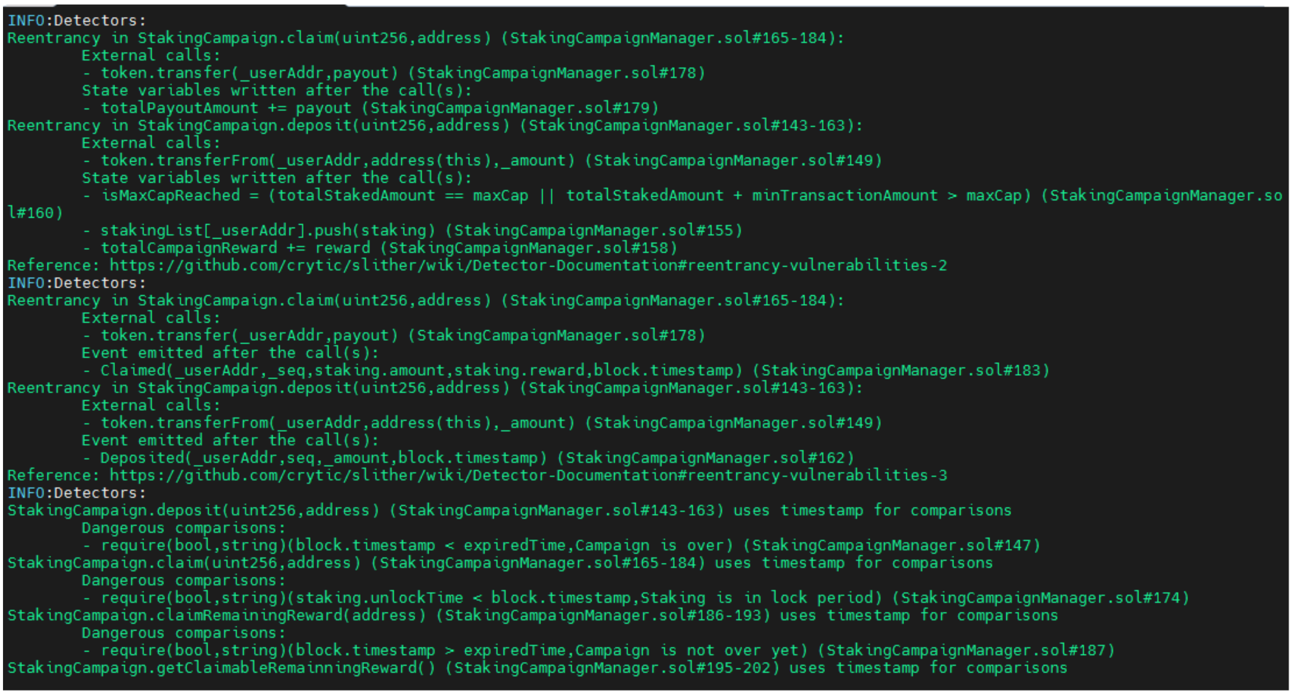


```
Parameter StakingCampaign.claim(uint256,address)._userAddr (StakingCampaign.sol#166) is not in mixedCase
Parameter StakingCampaign.claimRemainingReward(address)._userAddr (StakingCampaign.sol#187) is not in mixedCase
Parameter StakingCampaign.getStakings(address)._staker (StakingCampaign.sol#205) is not in mixedCase
Parameter StakingCampaign.transferOwnership(address)._newOwner (StakingCampaign.sol#237) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable StakingCampaign.getCampaignInfo()._maxTransactionAmount (StakingCampaign.sol#231) is too similar to StakingCampaign.constructor(
IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._minTransactionAmount (StakingCampaign.sol#128)
Variable StakingCampaign.constructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (StakingCampaig
n.sol#128) is too similar to StakingCampaign.getCampaignInfo()._minTransactionAmount (StakingCampaign.sol#231)
Variable StakingCampaign.getCampaignInfo()._maxTransactionAmount (StakingCampaign.sol#231) is too similar to StakingCampaign.getCampaignI
nfo()._minTransactionAmount (StakingCampaign.sol#231)
Variable StakingCampaign.constructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (StakingCampaig
n.sol#128) is too similar to StakingCampaign.constructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._minTransactionAm
ount (StakingCampaign.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
claim(uint256,address) should be declared external:
        - StakingCampaign.claim(uint256,address) (StakingCampaign.sol#166-185)
claimRemainingReward(address) should be declared external:
        - StakingCampaign.claimRemainingReward(address) (StakingCampaign.sol#187-194)
getClaimableRemainningReward() should be declared external:
        - StakingCampaign.getClaimableRemainningReward() (StakingCampaign.sol#196-203)
getStakings(address) should be declared external:
        - StakingCampaign.getStakings(address) (StakingCampaign.sol#205-227)
getCampaignInfo() should be declared external:
        - StakingCampaign.getCampaignInfo() (StakingCampaign.sol#229-235)
transferOwnership(address) should be declared external:
        - StakingCampaign.transferOwnership(address) (StakingCampaign.sol#237-239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StakingCampaign.sol analyzed (2 contracts with 75 detectors), 34 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

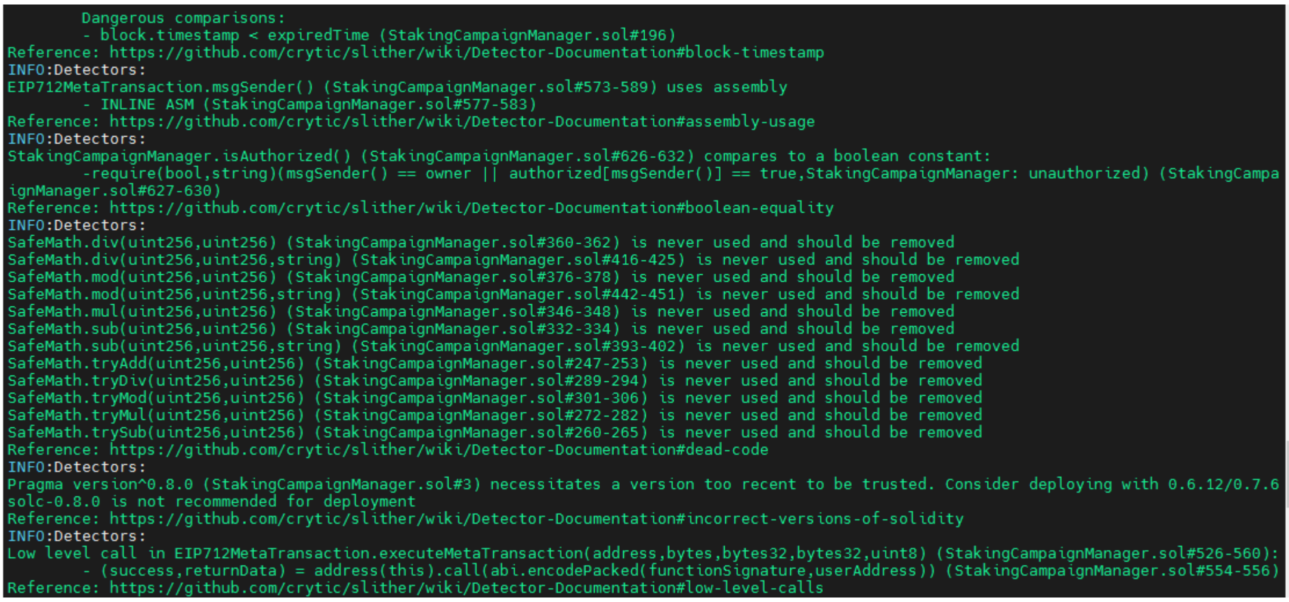
Slither log >> StakingCampaignManager.sol



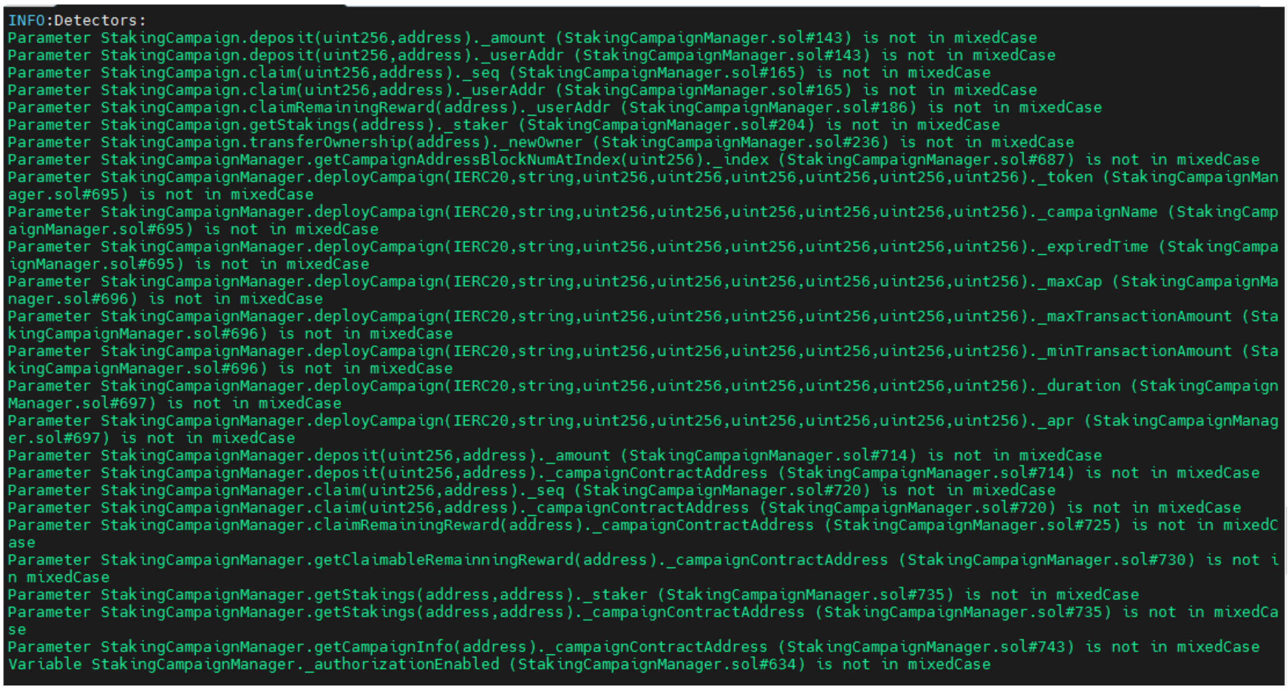
```
INFO:Detectors:
StakingCampaign.deposit(uint256,address) (StakingCampaignManager.sol#143-163) ignores return value by token.transferFrom(_userAddr,addres
s(this),_amount) (StakingCampaignManager.sol#149)
StakingCampaign.claim(uint256,address) (StakingCampaignManager.sol#165-184) ignores return value by token.transfer(_userAddr,payout) (Sta
kingCampaignManager.sol#178)
StakingCampaign.claimRemainingReward(address) (StakingCampaignManager.sol#186-193) ignores return value by token.transfer(_userAddr,balan
ce - remainingPayoutAmount) (StakingCampaignManager.sol#192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Contract locking ether found:
        Contract StakingCampaignManager (StakingCampaignManager.sol#612-751) has payable functions:
         - EIP712MetaTransaction.executeMetaTransaction(address,bytes,bytes32,bytes32,uint8) (StakingCampaignManager.sol#526-560)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
Reentrancy in StakingCampaign.claim(uint256,address) (StakingCampaignManager.sol#165-184):
        External calls:
        - token.transfer(_userAddr,payout) (StakingCampaignManager.sol#178)
        State variables written after the call(s):
        - stakingList[_userAddr][idx].isPayout = true (StakingCampaignManager.sol#181)
Reentrancy in StakingCampaign.deposit(uint256,address) (StakingCampaignManager.sol#143-163):
        External calls:
        - token.transferFrom(_userAddr,address(this),_amount) (StakingCampaignManager.sol#149)
        State variables written after the call(s):
        - totalStakedAmount += _amount (StakingCampaignManager.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256).addrBlockNum (StakingCampaignManager
.sol#705) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
StakingCampaign.transferOwnership(address) (StakingCampaignManager.sol#236-238) should emit an event for:
        - owner = _newOwner (StakingCampaignManager.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StakingCampaign.transferOwnership(address)._newOwner (StakingCampaignManager.sol#236) lacks a zero-check on :
                - owner = _newOwner (StakingCampaignManager.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



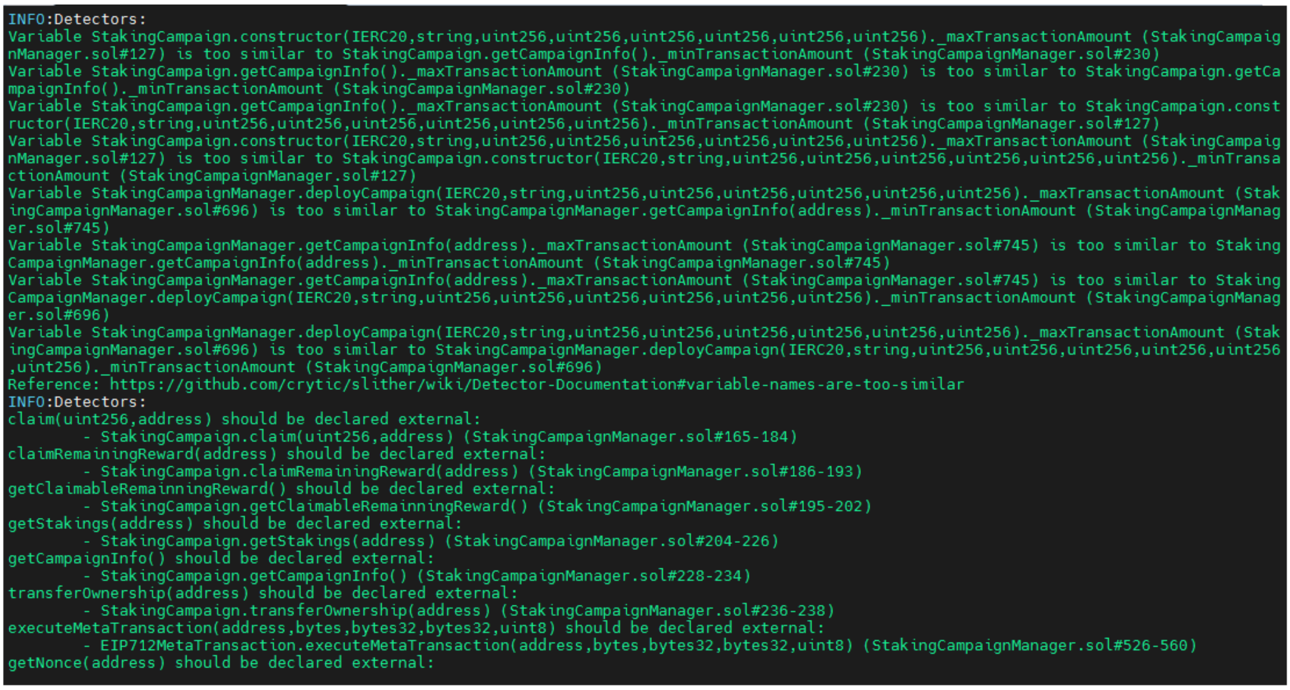
```
INFO:Detectors:
Reentrancy in StakingCampaign.claim(uint256,address) (StakingCampaignManager.sol#165-184):
        External calls:
        - token.transfer(_userAddr,payout) (StakingCampaignManager.sol#178)
        State variables written after the call(s):
        - totalPayoutAmount += payout (StakingCampaignManager.sol#179)
Reentrancy in StakingCampaign.deposit(uint256,address) (StakingCampaignManager.sol#143-163):
        External calls:
        - token.transferFrom(_userAddr,address(this),_amount) (StakingCampaignManager.sol#149)
        State variables written after the call(s):
        - isMaxCapReached = (totalStakedAmount == maxCap || totalStakedAmount + minTransactionAmount > maxCap) (StakingCampaignManager.so
l#160)
        - stakingList[_userAddr].push(staking) (StakingCampaignManager.sol#155)
        - totalCampaignReward += reward (StakingCampaignManager.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in StakingCampaign.claim(uint256,address) (StakingCampaignManager.sol#165-184):
        External calls:
        - token.transfer(_userAddr,payout) (StakingCampaignManager.sol#178)
        Event emitted after the call(s):
        - Claimed(_userAddr,_seq,staking.amount,staking.reward,block.timestamp) (StakingCampaignManager.sol#183)
Reentrancy in StakingCampaign.deposit(uint256,address) (StakingCampaignManager.sol#143-163):
        External calls:
        - token.transferFrom(_userAddr,address(this),_amount) (StakingCampaignManager.sol#149)
        Event emitted after the call(s):
        - Deposited(_userAddr,seq,_amount,block.timestamp) (StakingCampaignManager.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingCampaign.deposit(uint256,address) (StakingCampaignManager.sol#143-163) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < expiredTime,Campaign is over) (StakingCampaignManager.sol#147)
StakingCampaign.claim(uint256,address) (StakingCampaignManager.sol#165-184) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(staking.unlockTime < block.timestamp,Staking is in lock period) (StakingCampaignManager.sol#174)
StakingCampaign.claimRemainingReward(address) (StakingCampaignManager.sol#186-193) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > expiredTime,Campaign is not over yet) (StakingCampaignManager.sol#187)
StakingCampaign.getClaimableRemainningReward() (StakingCampaignManager.sol#195-202) uses timestamp for comparisons
```



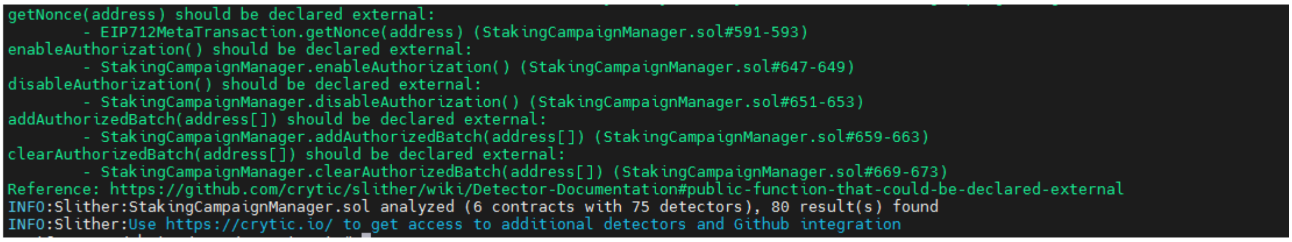
```
        Dangerous comparisons:
        - block.timestamp < expiredTime (StakingCampaignManager.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EIP712MetaTransaction.msgSender() (StakingCampaignManager.sol#573-589) uses assembly
        - INLINE ASM (StakingCampaignManager.sol#577-583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
StakingCampaignManager.isAuthorized() (StakingCampaignManager.sol#626-632) compares to a boolean constant:
        -require(bool,string)(msgSender() == owner || authorized[msgSender()] == true,StakingCampaignManager: unauthorized) (StakingCampa
ignManager.sol#627-630)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
SafeMath.div(uint256,uint256) (StakingCampaignManager.sol#360-362) is never used and should be removed
SafeMath.div(uint256,uint256,string) (StakingCampaignManager.sol#416-425) is never used and should be removed
SafeMath.mod(uint256,uint256) (StakingCampaignManager.sol#376-378) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (StakingCampaignManager.sol#442-451) is never used and should be removed
SafeMath.mul(uint256,uint256) (StakingCampaignManager.sol#346-348) is never used and should be removed
SafeMath.sub(uint256,uint256) (StakingCampaignManager.sol#332-334) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (StakingCampaignManager.sol#393-402) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (StakingCampaignManager.sol#247-253) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (StakingCampaignManager.sol#289-294) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (StakingCampaignManager.sol#301-306) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (StakingCampaignManager.sol#272-282) is never used and should be removed
SafeMath.trySub(uint256,uint256) (StakingCampaignManager.sol#260-265) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (StakingCampaignManager.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in EIP712MetaTransaction.executeMetaTransaction(address,bytes,bytes32,bytes32,uint8) (StakingCampaignManager.sol#526-560):
        - (success,returnData) = address(this).call(abi.encodePacked(functionSignature,userAddress)) (StakingCampaignManager.sol#554-556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```



```
INFO:Detectors:
Parameter StakingCampaign.deposit(uint256,address)._amount (StakingCampaignManager.sol#143) is not in mixedCase
Parameter StakingCampaign.deposit(uint256,address)._userAddr (StakingCampaignManager.sol#143) is not in mixedCase
Parameter StakingCampaign.claim(uint256,address)._seq (StakingCampaignManager.sol#165) is not in mixedCase
Parameter StakingCampaign.claim(uint256,address)._userAddr (StakingCampaignManager.sol#165) is not in mixedCase
Parameter StakingCampaign.claimRemainingReward(address)._userAddr (StakingCampaignManager.sol#186) is not in mixedCase
Parameter StakingCampaign.getStakings(address)._staker (StakingCampaignManager.sol#204) is not in mixedCase
Parameter StakingCampaign.transferOwnership(address)._newOwner (StakingCampaignManager.sol#236) is not in mixedCase
Parameter StakingCampaignManager.getCampaignAddressBlockNumAtIndex(uint256)._index (StakingCampaignManager.sol#687) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._token (StakingCampaignMan
ager.sol#695) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._campaignName (StakingCamp
aignManager.sol#695) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._expiredTime (StakingCampa
ignManager.sol#695) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxCap (StakingCampaignMa
nager.sol#696) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (Sta
kingCampaignManager.sol#696) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._minTransactionAmount (Sta
kingCampaignManager.sol#696) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._duration (StakingCampaign
Manager.sol#697) is not in mixedCase
Parameter StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._apr (StakingCampaignManag
er.sol#697) is not in mixedCase
Parameter StakingCampaignManager.deposit(uint256,address)._amount (StakingCampaignManager.sol#714) is not in mixedCase
Parameter StakingCampaignManager.deposit(uint256,address)._campaignContractAddress (StakingCampaignManager.sol#714) is not in mixedCase
Parameter StakingCampaignManager.claim(uint256,address)._seq (StakingCampaignManager.sol#720) is not in mixedCase
Parameter StakingCampaignManager.claim(uint256,address)._campaignContractAddress (StakingCampaignManager.sol#720) is not in mixedCase
Parameter StakingCampaignManager.claimRemainingReward(address)._campaignContractAddress (StakingCampaignManager.sol#725) is not in mixedC
ase
Parameter StakingCampaignManager.getClaimableRemainningReward(address)._campaignContractAddress (StakingCampaignManager.sol#730) is not i
n mixedCase
Parameter StakingCampaignManager.getStakings(address,address)._staker (StakingCampaignManager.sol#735) is not in mixedCase
Parameter StakingCampaignManager.getStakings(address,address)._campaignContractAddress (StakingCampaignManager.sol#735) is not in mixedCa
se
Parameter StakingCampaignManager.getCampaignInfo(address)._campaignContractAddress (StakingCampaignManager.sol#743) is not in mixedCase
Variable StakingCampaignManager._authorizationEnabled (StakingCampaignManager.sol#634) is not in mixedCase
```



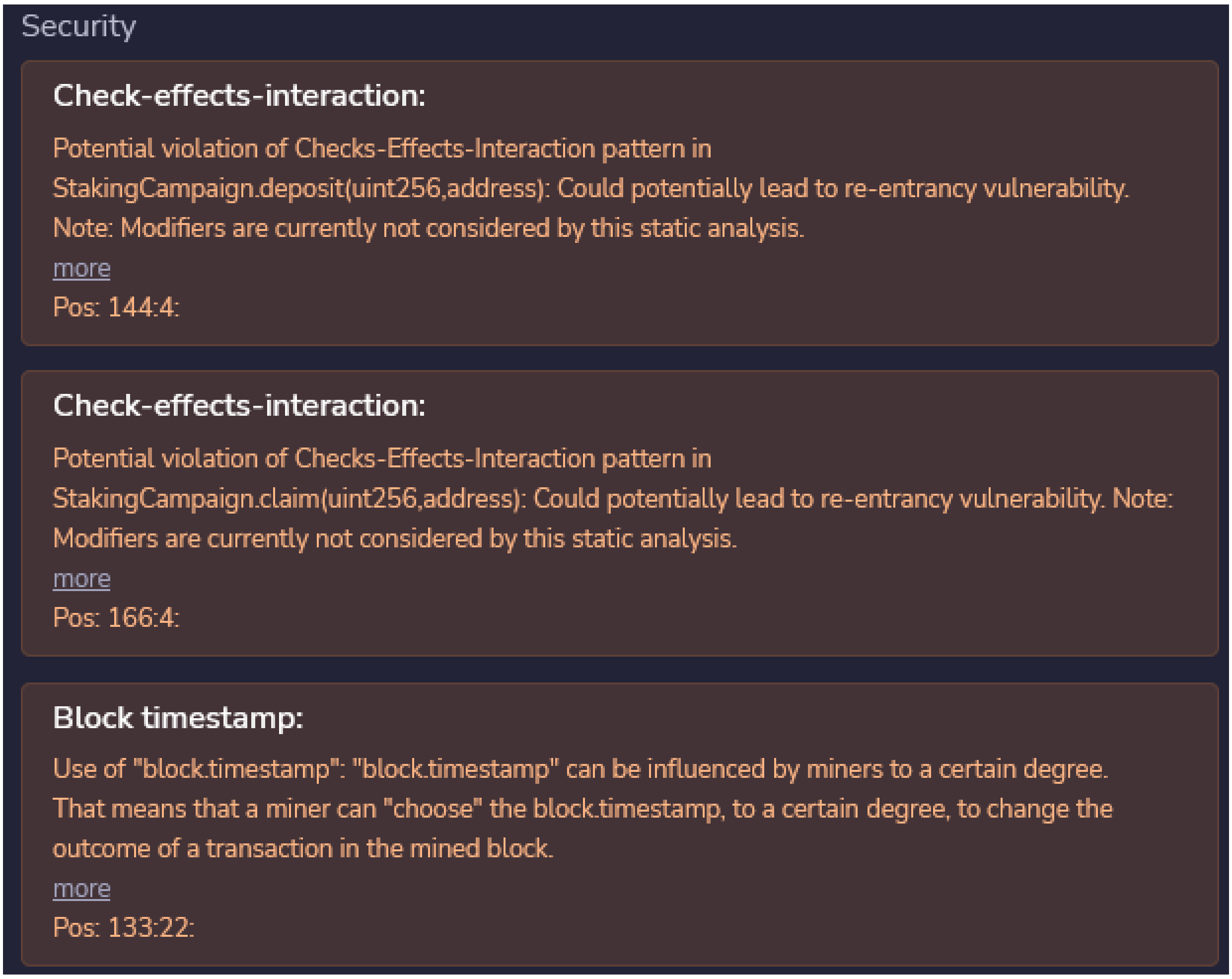
```
INFO:Detectors:
Variable StakingCampaign.constructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (StakingCampaig
nManager.sol#127) is too similar to StakingCampaign.getCampaignInfo()._minTransactionAmount (StakingCampaignManager.sol#230)
Variable StakingCampaign.getCampaignInfo()._maxTransactionAmount (StakingCampaignManager.sol#230) is too similar to StakingCampaign.getCa
mpaignInfo()._minTransactionAmount (StakingCampaignManager.sol#230)
Variable StakingCampaign.getCampaignInfo()._maxTransactionAmount (StakingCampaignManager.sol#230) is too similar to StakingCampaign.const
ructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._minTransactionAmount (StakingCampaignManager.sol#127)
Variable StakingCampaign.constructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (StakingCampaig
nManager.sol#127) is too similar to StakingCampaign.constructor(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._minTransa
ctionAmount (StakingCampaignManager.sol#127)
Variable StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (Stak
ingCampaignManager.sol#696) is too similar to StakingCampaignManager.getCampaignInfo(address)._minTransactionAmount (StakingCampaignManag
er.sol#745)
Variable StakingCampaignManager.getCampaignInfo(address)._maxTransactionAmount (StakingCampaignManager.sol#745) is too similar to Staking
CampaignManager.getCampaignInfo(address)._minTransactionAmount (StakingCampaignManager.sol#745)
Variable StakingCampaignManager.getCampaignInfo(address)._maxTransactionAmount (StakingCampaignManager.sol#745) is too similar to Staking
CampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._minTransactionAmount (StakingCampaignManag
er.sol#696)
Variable StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256)._maxTransactionAmount (Stak
ingCampaignManager.sol#696) is too similar to StakingCampaignManager.deployCampaign(IERC20,string,uint256,uint256,uint256,uint256,uint256
,uint256)._minTransactionAmount (StakingCampaignManager.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
claim(uint256,address) should be declared external:
        - StakingCampaign.claim(uint256,address) (StakingCampaignManager.sol#165-184)
claimRemainingReward(address) should be declared external:
        - StakingCampaign.claimRemainingReward(address) (StakingCampaignManager.sol#186-193)
getClaimableRemainningReward() should be declared external:
        - StakingCampaign.getClaimableRemainningReward() (StakingCampaignManager.sol#195-202)
getStakings(address) should be declared external:
        - StakingCampaign.getStakings(address) (StakingCampaignManager.sol#204-226)
getCampaignInfo() should be declared external:
        - StakingCampaign.getCampaignInfo() (StakingCampaignManager.sol#228-234)
transferOwnership(address) should be declared external:
        - StakingCampaign.transferOwnership(address) (StakingCampaignManager.sol#236-238)
executeMetaTransaction(address,bytes,bytes32,bytes32,uint8) should be declared external:
        - EIP712MetaTransaction.executeMetaTransaction(address,bytes,bytes32,bytes32,uint8) (StakingCampaignManager.sol#526-560)
getNonce(address) should be declared external:
```



```
getNonce(address) should be declared external:
        - EIP712MetaTransaction.getNonce(address) (StakingCampaignManager.sol#591-593)
enableAuthorization() should be declared external:
        - StakingCampaignManager.enableAuthorization() (StakingCampaignManager.sol#647-649)
disableAuthorization() should be declared external:
        - StakingCampaignManager.disableAuthorization() (StakingCampaignManager.sol#651-653)
addAuthorizedBatch(address[]) should be declared external:
        - StakingCampaignManager.addAuthorizedBatch(address[]) (StakingCampaignManager.sol#659-663)
clearAuthorizedBatch(address[]) should be declared external:
        - StakingCampaignManager.clearAuthorizedBatch(address[]) (StakingCampaignManager.sol#669-673)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StakingCampaignManager.sol analyzed (6 contracts with 75 detectors), 80 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity static analysis

StakingCampaign.sol



Security

Check-effects-interaction:

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

more

Pos: 144:4:

Check-effects-interaction:

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

more

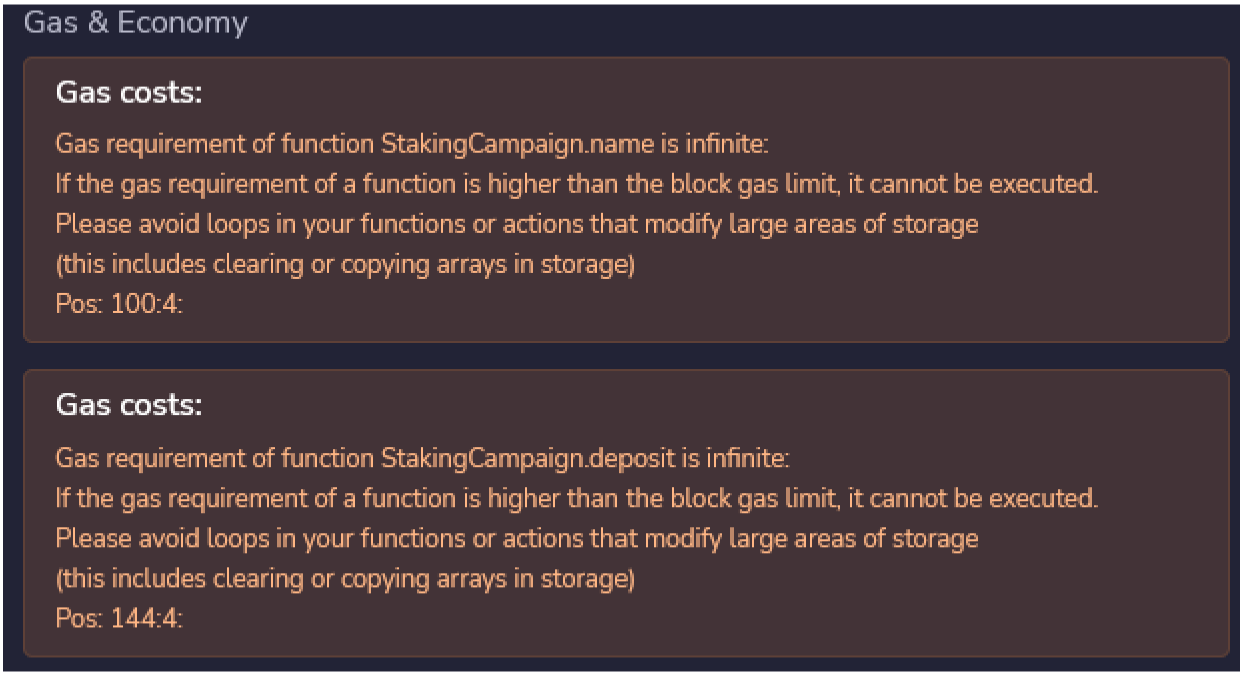
Pos: 166:4:

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: 133:22:



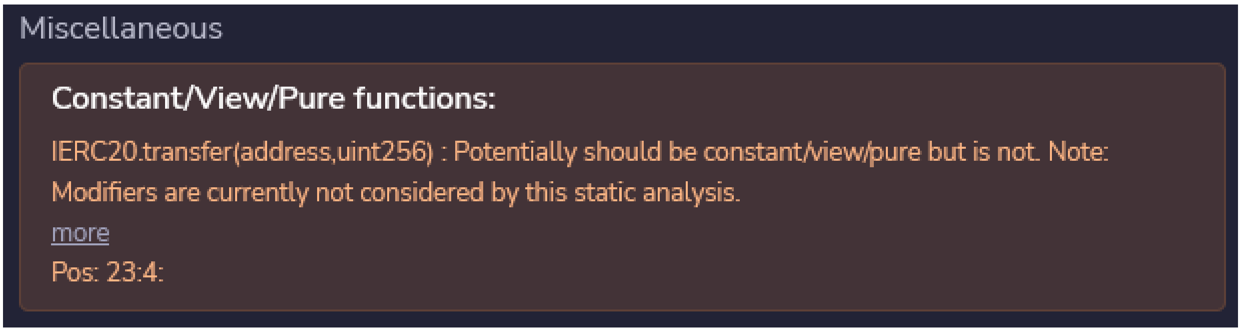
Gas & Economy

Gas costs:

Gas requirement of function StakingCampaign.name 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: 100:4:

Gas costs:

Gas requirement of function StakingCampaign.deposit 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: 144:4:



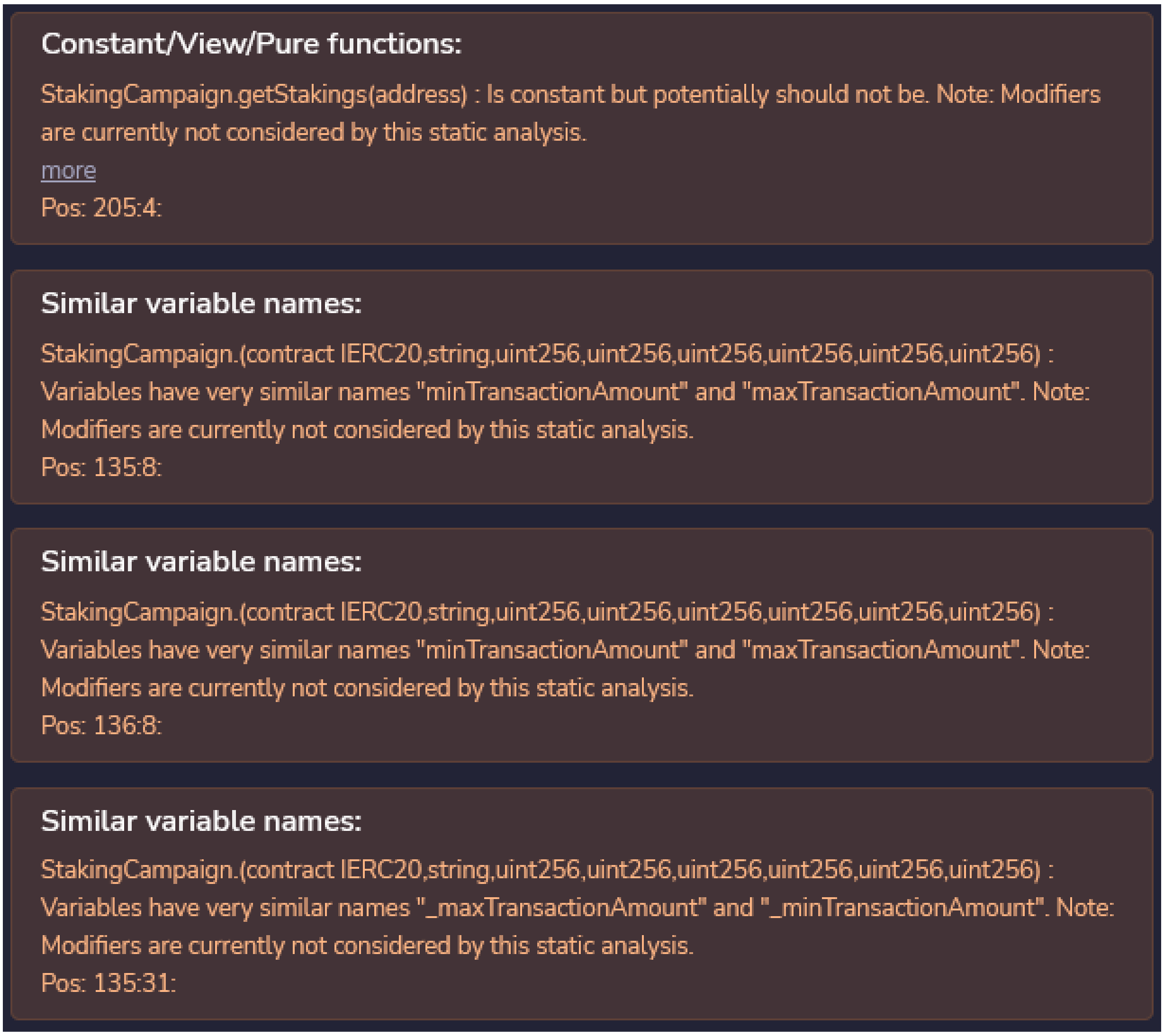
Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 23:4:



Constant/View/Pure functions:

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

more

Pos: 205:4:

Similar variable names:

StakingCampaign.(contract IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256) : Variables have very similar names "minTransactionAmount" and "maxTransactionAmount". Note: Modifiers are currently not considered by this static analysis.

Pos: 135:8:

Similar variable names:

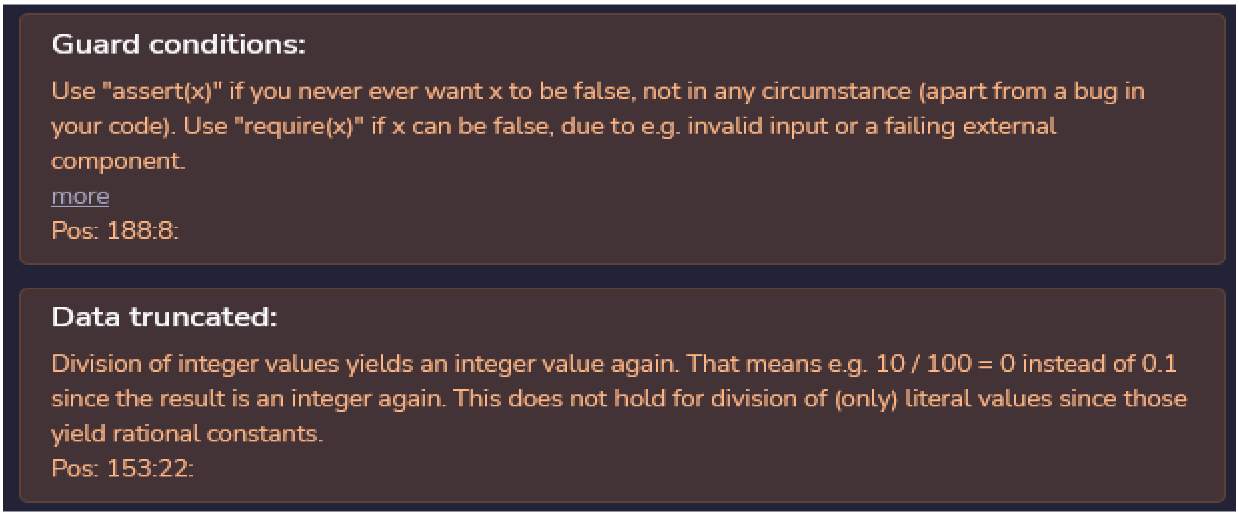
StakingCampaign.(contract IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256) : Variables have very similar names "minTransactionAmount" and "maxTransactionAmount". Note: Modifiers are currently not considered by this static analysis.

Pos: 136:8:

Similar variable names:

StakingCampaign.(contract IERC20,string,uint256,uint256,uint256,uint256,uint256,uint256) : Variables have very similar names "_maxTransactionAmount" and "_minTransactionAmount". Note: Modifiers are currently not considered by this static analysis.

Pos: 135:31:



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

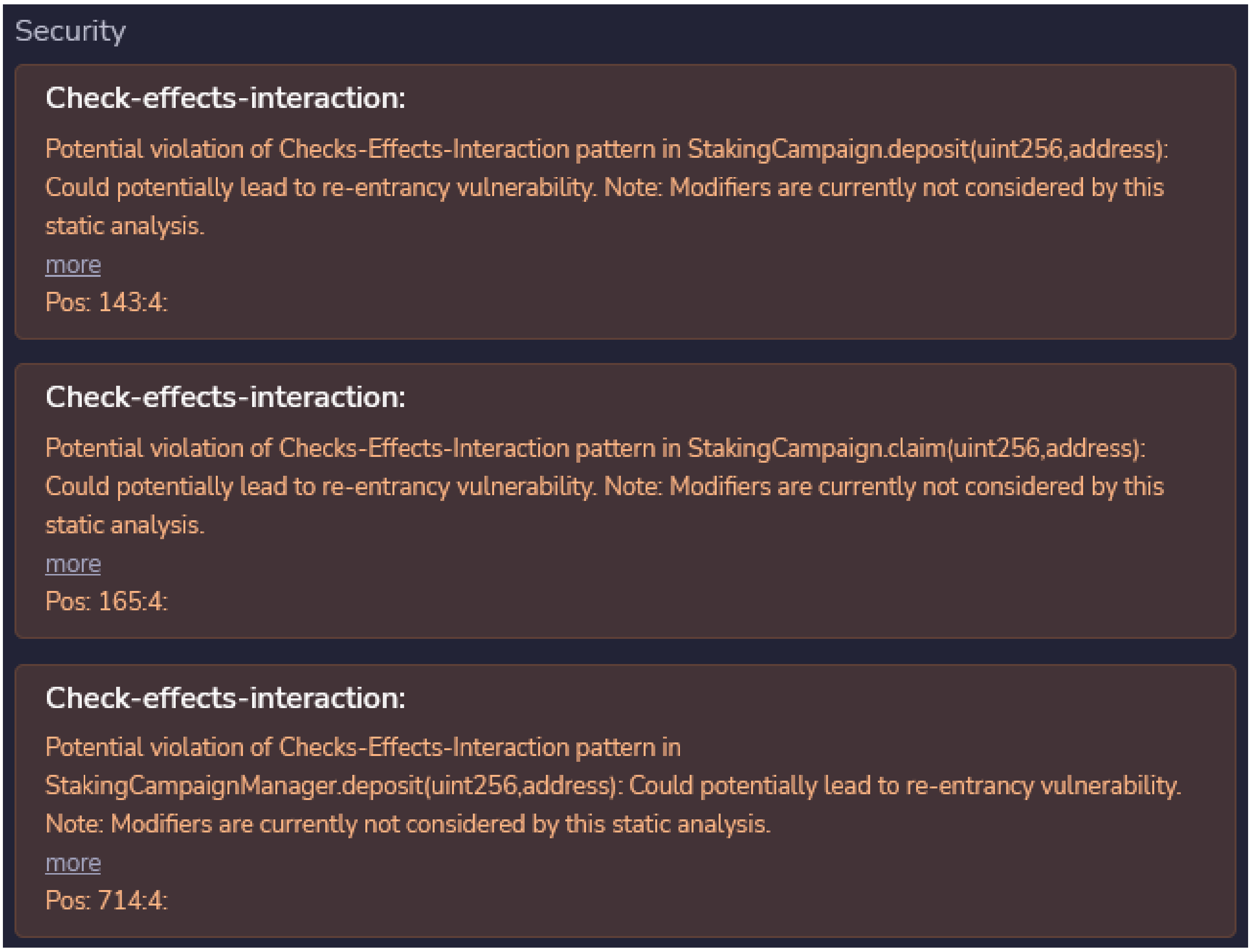
Pos: 188:8:

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: 153:22:

StakingCampaignManager.sol



Security

Check-effects-interaction:

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

more

Pos: 143:4:

Check-effects-interaction:

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

more

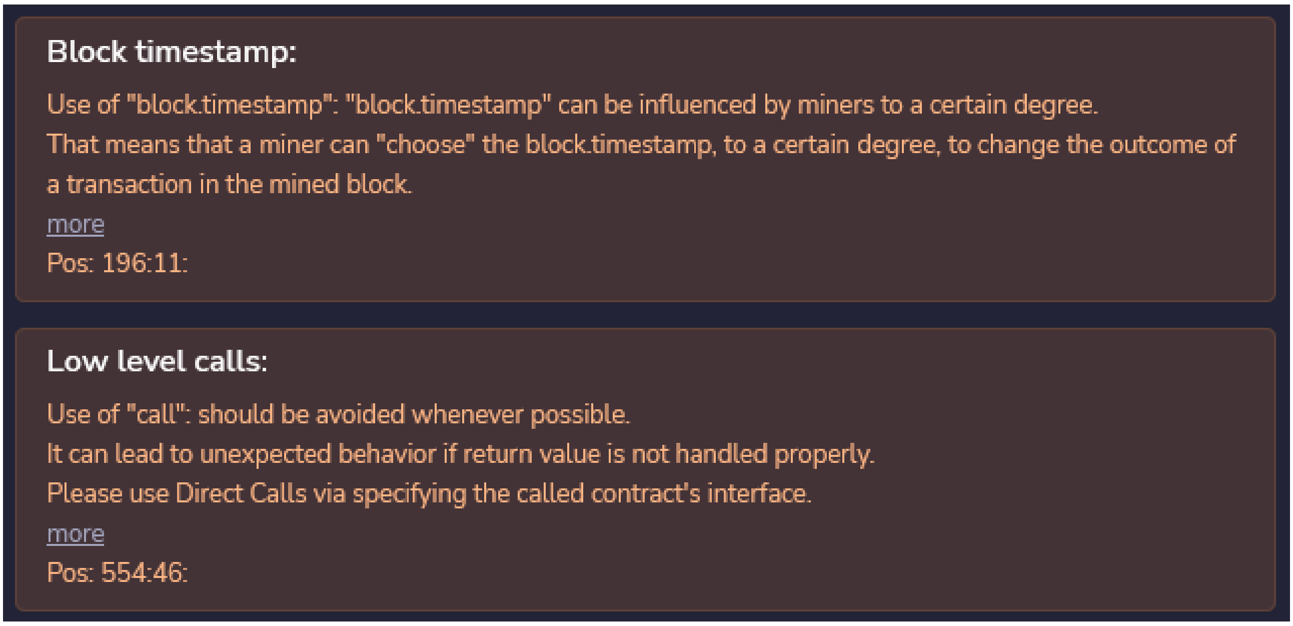
Pos: 165:4:

Check-effects-interaction:

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

more

Pos: 714:4:



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

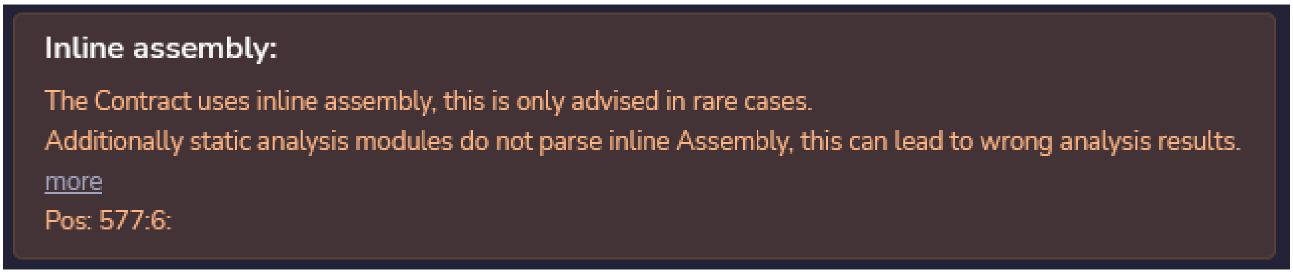
Pos: 196:11:

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: 554:46:

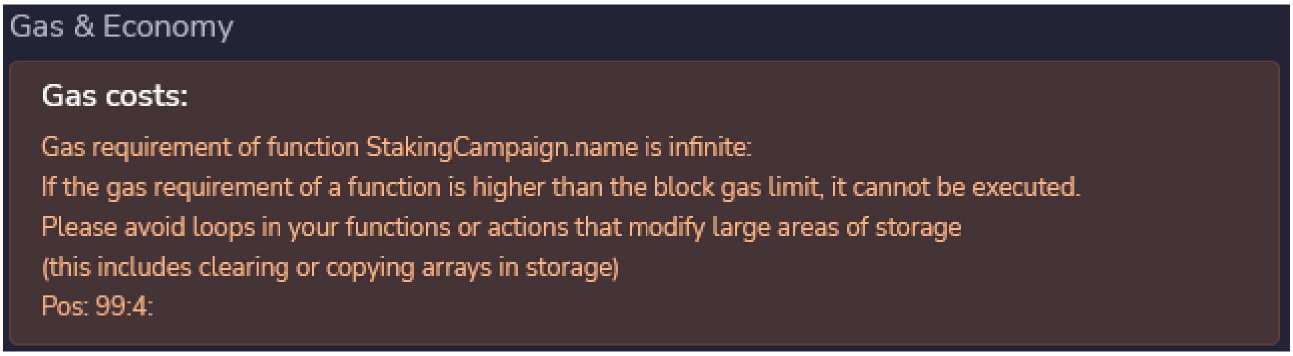


Inline assembly:

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

more

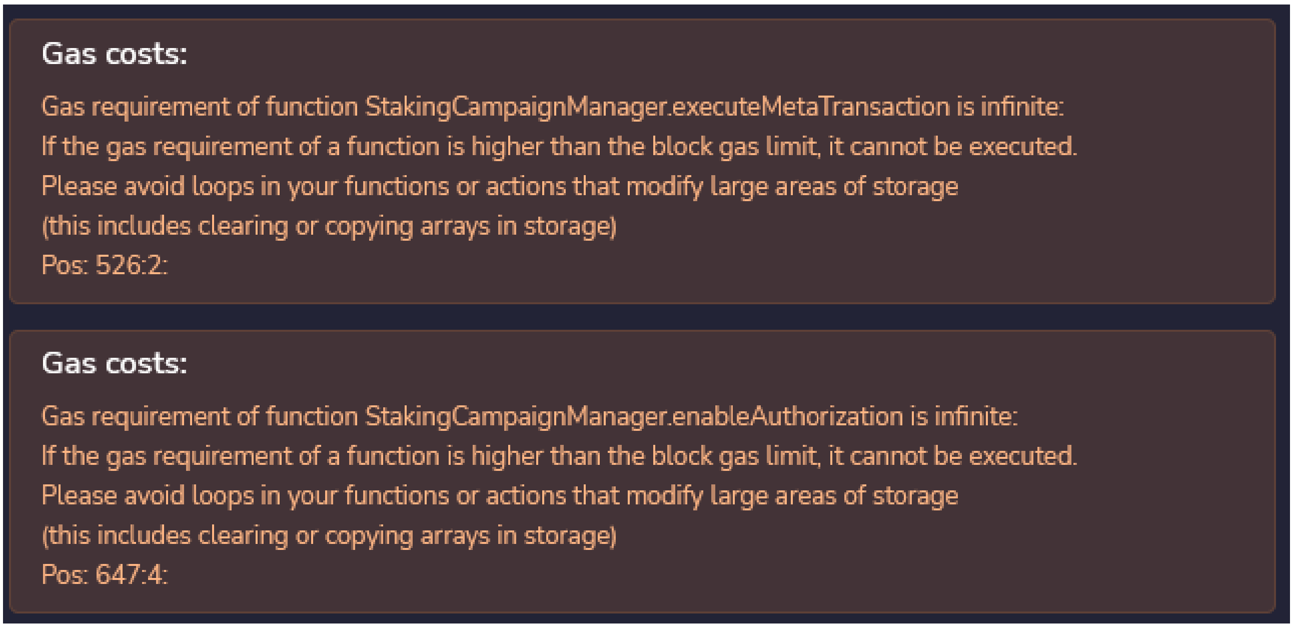
Pos: 577:6:



Gas & Economy

Gas costs:

Gas requirement of function StakingCampaign.name 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: 99:4:

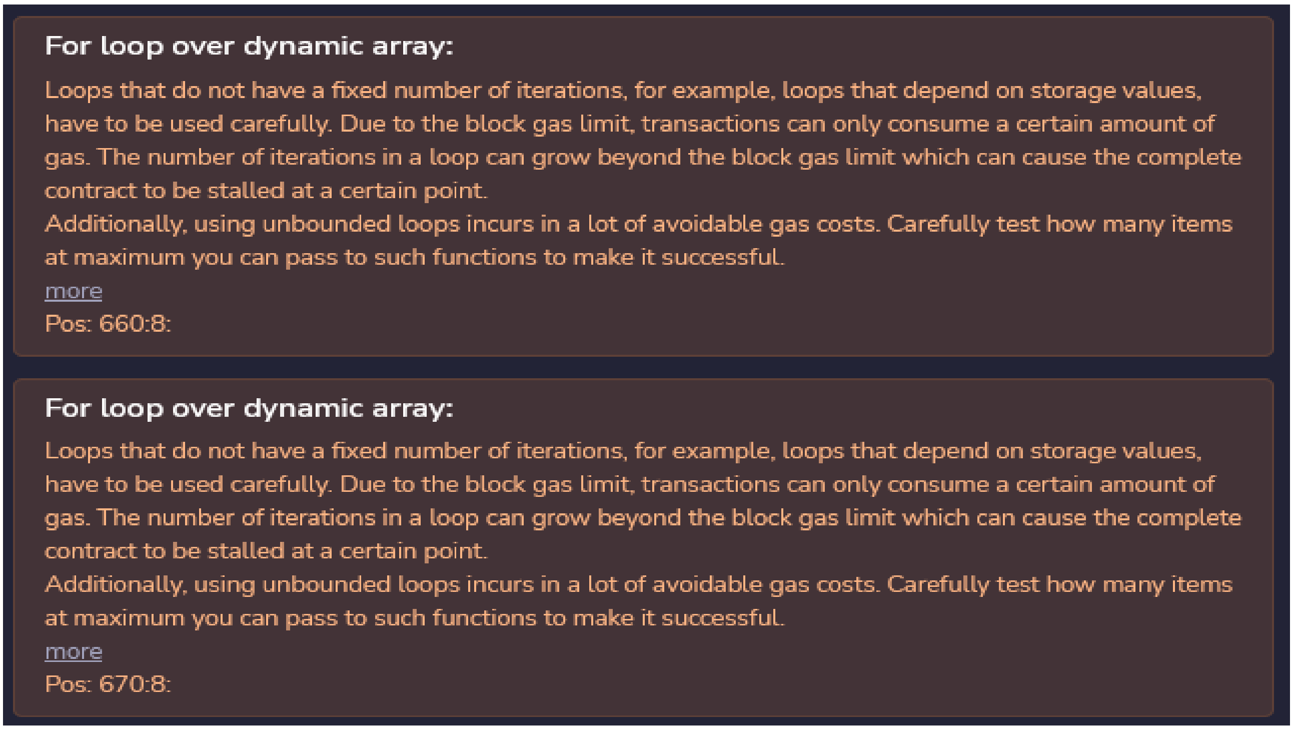


Gas costs:

Gas requirement of function StakingCampaignManager.executeMetaTransaction 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: 526:2:

Gas costs:

Gas requirement of function StakingCampaignManager.enableAuthorization 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: 647:4:

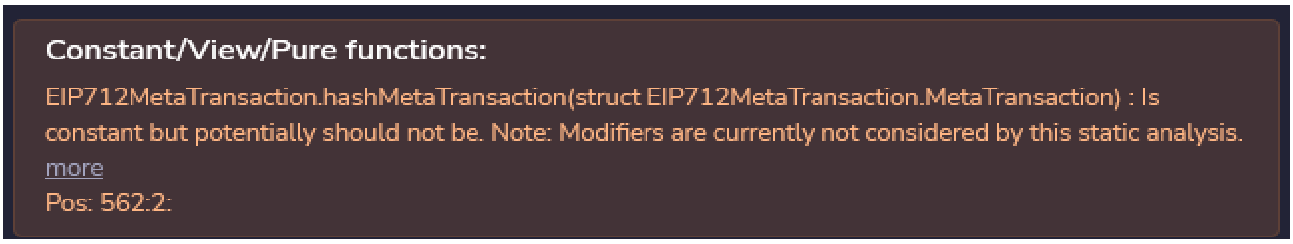


For loop over dynamic array:

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: 660:8:

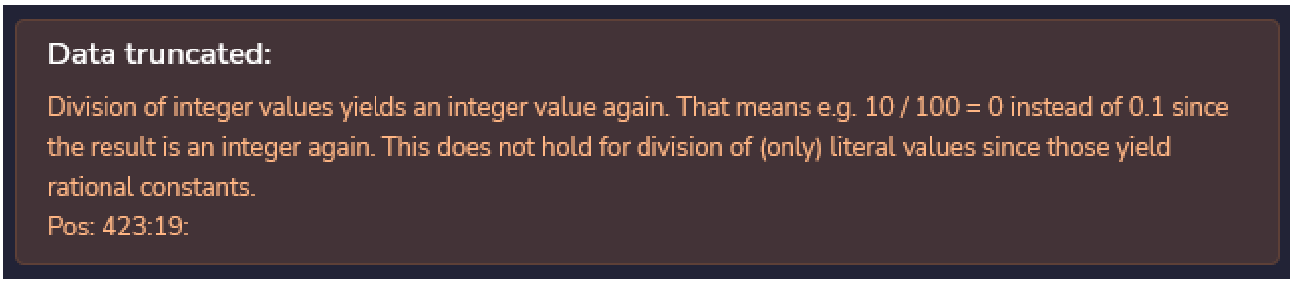
For loop over dynamic array:

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: 670:8:



Constant/View/Pure functions:

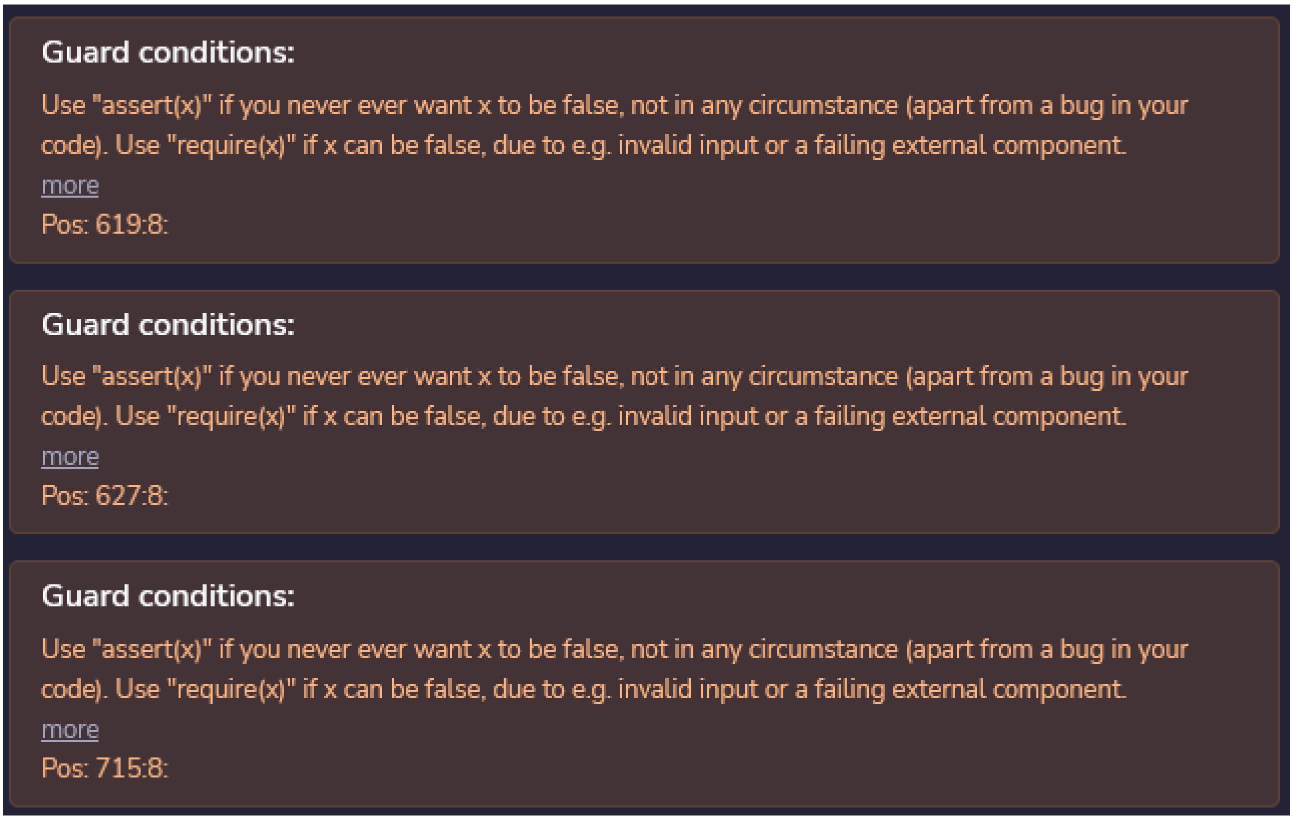
EIP712MetaTransaction.hashMetaTransaction(struct EIP712MetaTransaction.MetaTransaction) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 562:2:



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: 423:19:



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: 619:8:

Guard conditions:

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

more

Pos: 627:8:

Guard conditions:

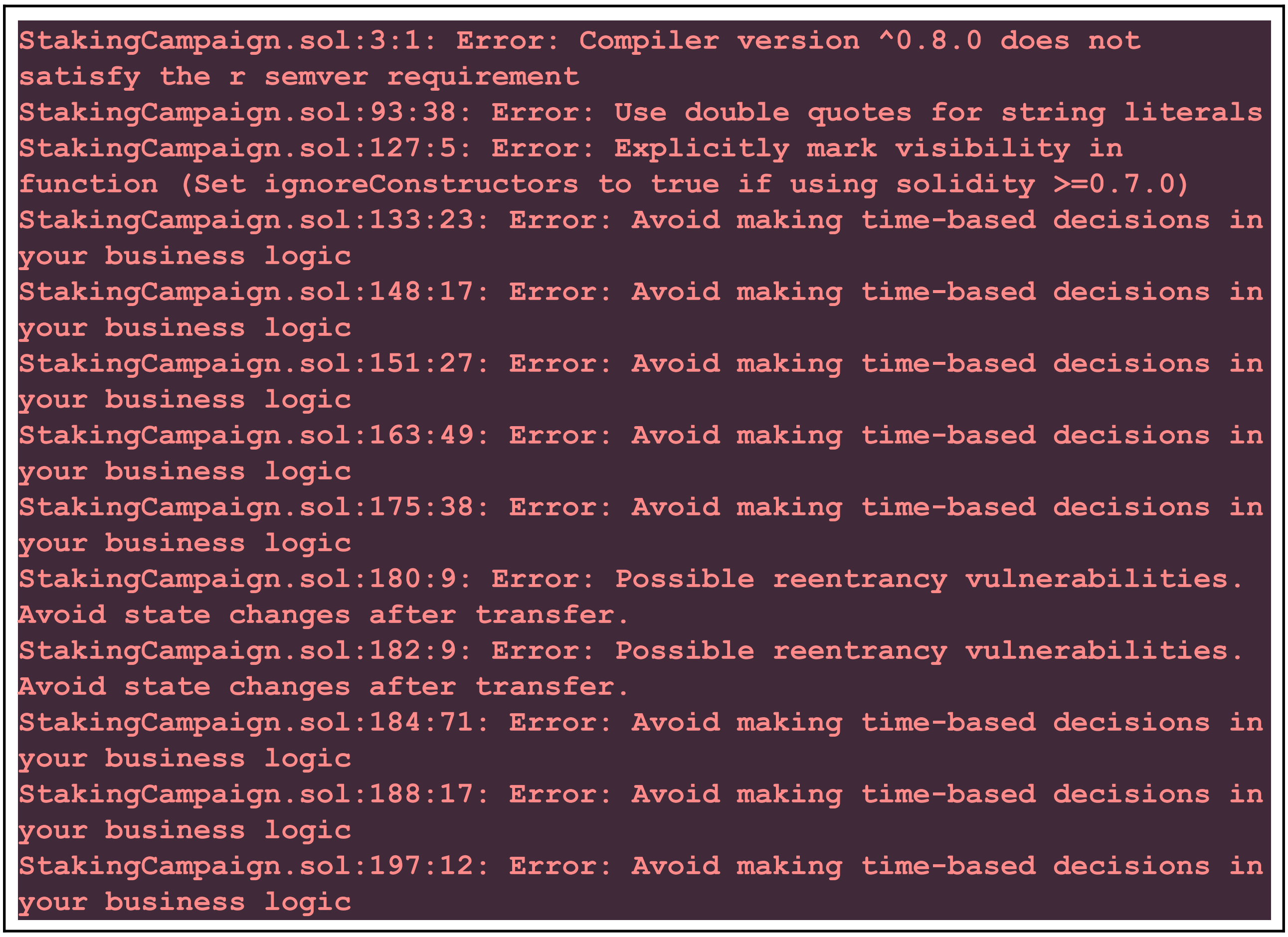
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 715:8:

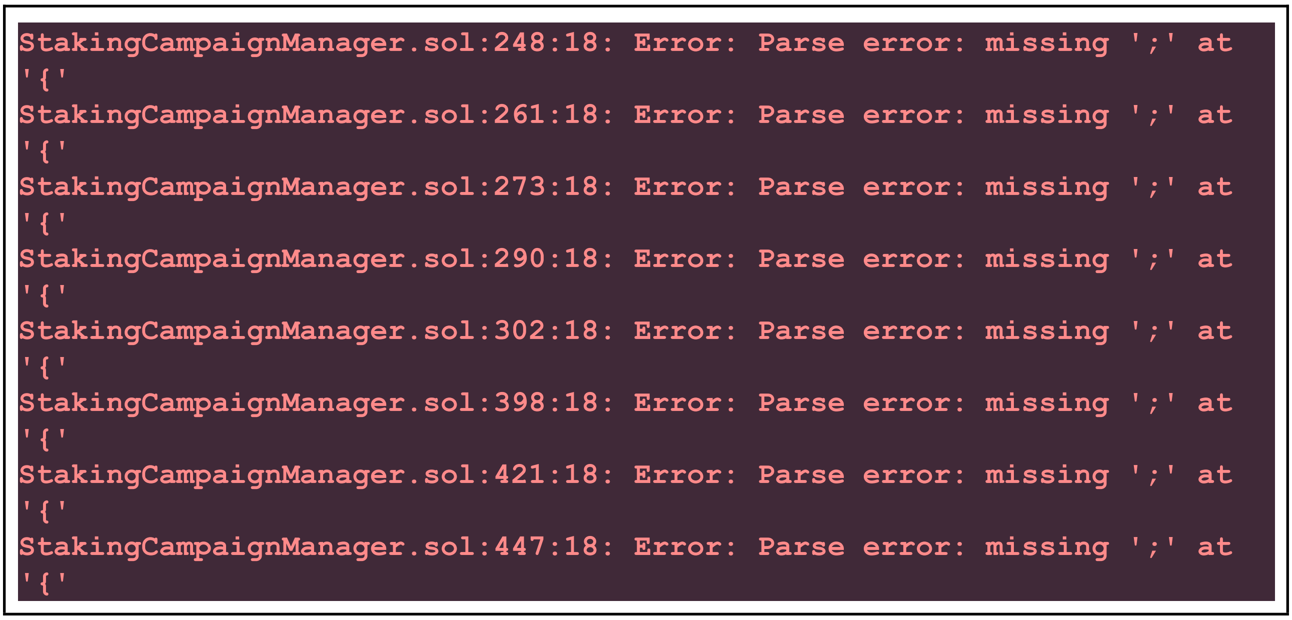
Solhint Linter

StakingCampaign.sol



```
StakingCampaign.sol:3:1: Error: Compiler version ^0.8.0 does not
satisfy the r semver requirement
StakingCampaign.sol:93:38: Error: Use double quotes for string literals
StakingCampaign.sol:127:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
StakingCampaign.sol:133:23: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:148:17: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:151:27: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:163:49: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:175:38: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:180:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakingCampaign.sol:182:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
StakingCampaign.sol:184:71: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:188:17: Error: Avoid making time-based decisions in
your business logic
StakingCampaign.sol:197:12: Error: Avoid making time-based decisions in
your business logic
```

StakingCampaignManager.sol



```
StakingCampaignManager.sol:248:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:261:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:273:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:290:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:302:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:398:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:421:18: Error: Parse error: missing ';' at
'{'
StakingCampaignManager.sol:447:18: 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.