Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Eyeverse
Website:	https://eyeverse.world
Platform:	Ethereum
Language:	Solidity
Date:	January 13th, 2023

Table of contents

Introduction

EtherAuthority was contracted by Eyeverse to perform the Security audit of the Eyeverse 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 January 13th, 2023.

The purpose of this audit was to address the following:

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

Project Background

- Eyeverse: A world of darkness and little light, a world of mystery and many realms that fight. Where the Eye Kings are superior beings with roundtables of Eye Lords. Eye Lords that lead armies of Night Watchers and Day Watchers ruling all The Watched.
- Eyeverse Contract is an NFT smart contract, having functions like burn, mint, stake, unStake, claim, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Eyeverse System Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	EyeVerseWrap.sol
File 1 MD5 Hash	B2DEEB2B298EC0BA0BF5E3B84CD056F6
File 2	NFTStaker.sol
File 2 MD5 Hash	D2CF4E659F6BA6AF75F80472FCDA464E
Audit Date	January 13th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 EyeVerseWrap.sol ● Name: EyeVerseWrap ● Symbol: EVW **Ownership Control:** ● Owner can set a new baseURI. ● Owner can set a new extension. **Allowed Operator Control:** ● Operator can set an approval for all addresses. ● Operator can approve all operator addresses. ● Operator can transfer a token from sender address to receiver address.	**YES, This is valid.**
File 2 NFTStaker.sol ● Rewards Per Day: 0.000000000000000003 GS ● Minimum Lock Time: 1 day **Ownership Control:** ● Only the NFT owner can lock tokens. ● Only staker can release tokens. ● Owner can set a new reward Per day. ● Owner can set a minimum lock time.	**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.

Insecure
Poor secured
Secure
Well-secured
You are here

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 1 low and some very low level issues.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
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 contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Eyeverse 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 Eyeverse Protocol.

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

All code parts are not well commented on smart contracts.

Documentation

We were given an Eyeverse smart contract code in the form of a goerli.etherscan.io link. The hash of that code is mentioned above in the table.

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

Another source of information was its website: https://eyeverse.world 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

EyeVerseWrap.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	supportsInterface	read	Passed	No Issue
9	tokenOfOwnerByIndex	read	Passed	No Issue
10	totalSupply	read	Passed	No Issue
11	tokenByIndex	read	Passed	No Issue
12	_beforeTokenTransfer	internal	Passed	No Issue
13	_addTokenToOwnerEnumeration	write	Passed	No Issue
14	_addTokenToAllTokensEnumeration	write	Passed	No Issue
15	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
16	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
17	checkOwnerShipOld	modifier	Passed	No Issue
18	checkOwnerShipNew	modifier	Passed	No Issue
19	singleMintWrap	write	check OwnerShip Old	No Issue
20	multiplrMintWrap	write	Infinite loops possibility	Refer Audit Findings
21	singleUnwrap	write	check OwnerShip New	No Issue
22	multiplrUnWrap	write	Infinite loops possibility	Refer Audit Findings
23	_beforeTokenTransfer	internal	Passed	No Issue
24	supportsInterface	read	Passed	No Issue
25	setBaseURI	write	access only Owner	No Issue
26	setBaseExtension	write	access only Owner	No Issue
27	tokenURI	read	Passed	No Issue
28	setApprovalForAll	write	access only Allowed Operator Approval	No Issue
29	approve	write	access only Allowed Operator Approval	No Issue
30	transferFrom	write	access only Allowed Operator Approval	No Issue

31	safeTransferFrom	write	access only Allowed Operator Approval	No Issue
32	safeTransferFrom	write	Passed	No Issue
33	onlyAllowedOperator	modifier	Passed	No Issue
34	onlyAllowedOperatorApproval	modifier	Passed	No Issue
35	_checkFilterOperator	internal	Passed	No Issue

NFTStaker.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	setMinimumLockTime	external	access only Owner	No Issue
9	stake	write	Infinite loops possibility	Refer Audit Findings
10	multipleStake	write	Passed	No Issue
11	unStake	write	Infinite loops possibility	Refer Audit Findings
12	multipleUnStake	write	Passed	No Issue
13	calculateDays	read	Passed	No Issue
14	getReward	read	Passed	No Issue
15	setRewardPerDay	external	access only Owner	No Issue
16	claim	write	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No critical severity vulnerabilities were found.

High Severity

No high severity vulnerabilities were found.

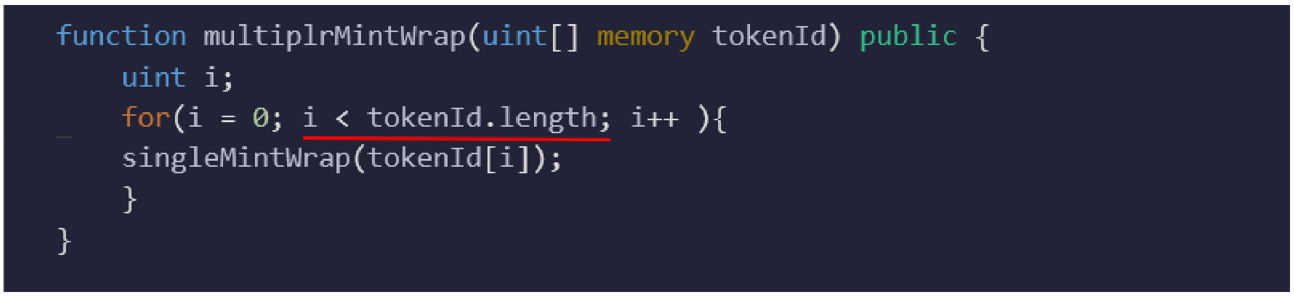
Medium

No medium severity vulnerabilities were found.

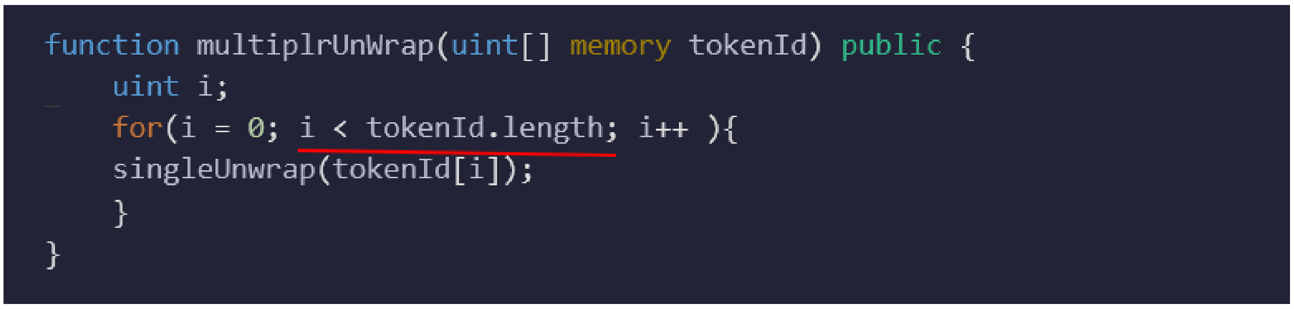
Low

(1) Infinite loops possibility:

EyeVerseWrap.sol

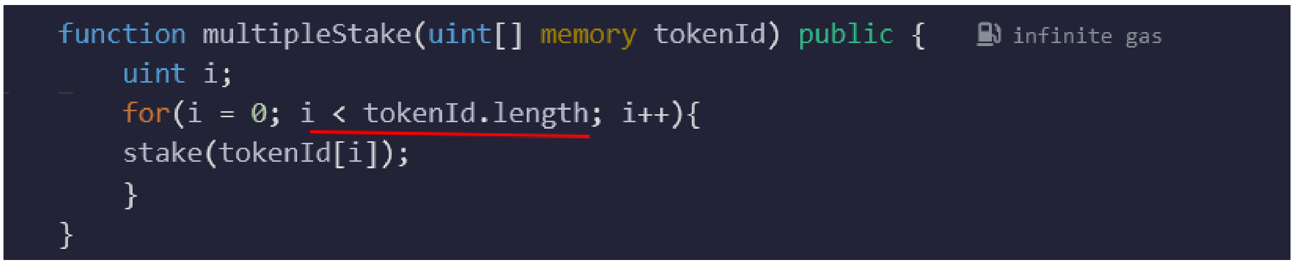


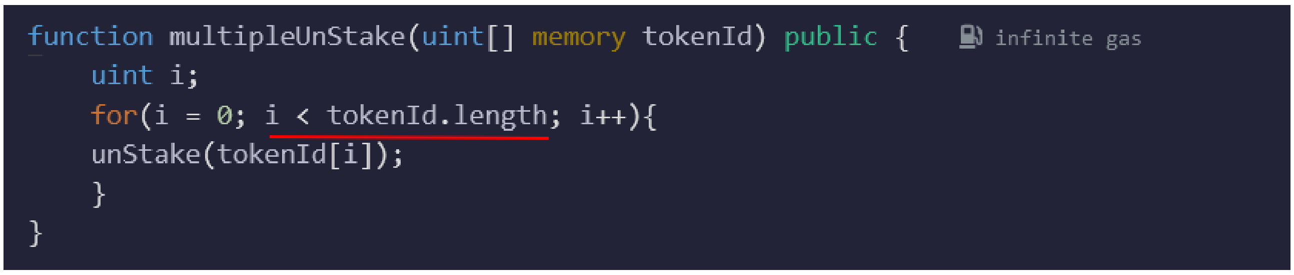
```
function multiplrMintWrap(uint[] memory tokenId) public {
      uint i;
      for(i = 0; i < tokenId.length; i++ ){
      singleMintWrap(tokenId[i]);
      }
}
```



```
function multiplrUnWrap(uint[] memory tokenId) public {
      uint i;
      for(i = 0; i < tokenId.length; i++ ){
      singleUnwrap(tokenId[i]);
      }
}
```

NFTStaker.sol


```
function multipleStake(uint[] memory tokenId) public {    infinite gas
    uint i;
    for(i = 0; i < tokenId.length; i++){
    stake(tokenId[i]);
    }
}
```


```
function multipleUnStake(uint[] memory tokenId) public {    infinite gas
    uint i;
    for(i = 0; i < tokenId.length; i++){
    unStake(tokenId[i]);
    }
}
```

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

EyeVerseWrap.sol

- multiplrMintWrap() - tokenId.length.
- multiplrUnWrap() - tokenId.length.

NFTStaker.sol

- multipleStake() - tokenId.length.
- multipleUnStake() - tokenId.length.

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

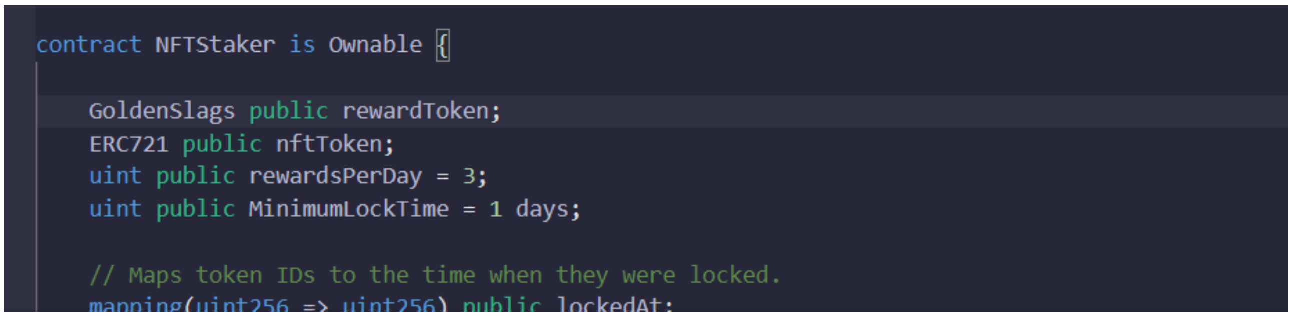
Very Low / Informational / Best practices:

(1) Spelling mistake: **NFTStaker.sol**

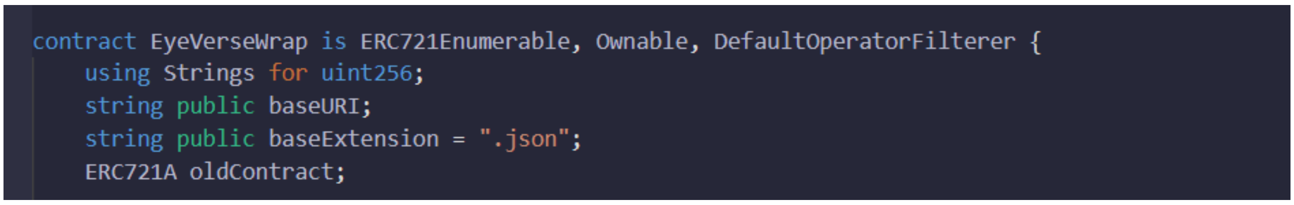
Spelling mistake in constructor comments.
“CHNGE” word should be **“CHANGE”**.

Resolution: Correct the spelling.

(2) Immutable variables:



```
contract NFTStaker is Ownable {

    GoldenSlags public rewardToken;
    ERC721 public nftToken;
    uint public rewardsPerDay = 3;
    uint public MinimumLockTime = 1 days;

    // Maps token IDs to the time when they were locked.
    mapping(uint256 => uint256) public lockedAt;
```



```
contract EyeVerseWrap is ERC721Enumerable, Ownable, DefaultOperatorFilterer {
    using Strings for uint256;
    string public baseURI;
    string public baseExtension = ".json";
    ERC721A oldContract;
```

Some variables are set only in the constructor and then remain unchanged. So those can be defined as immutable.
Variables are:

NFTStaker.sol

- rewardToken
- nftToken

EyeVerseWrap.sol

- oldContract

Resolution: We suggest defining these variables as immutable.

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:

EyeVerseWrap.sol

- singleMintWrap: Old NFT token owner can wrap new token.
- singleUnwrap: New NFT token owner can unwrap old token.
- setBaseURI: EyeVerseWrap owner can set a new base URI.
- setBaseExtension: EyeVerseWrap owner can set a new base extension.
- setApprovalForAll: EyeVerseWrap Operator can set an approval for all addresses.

NFTStaker.sol

- setRewardPerDay: NFTStaker owner can set rewards per day.
- setMinimumLockTime: NFTStaker owner can set minimum lock time for unstake.

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 goerli.etherscan.io link. And we have used all possible tests based on given objects as files. We have not observed any major issue in the smart contracts. **So smart contracts are good to go for the mainnet deployment**.

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

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

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 **"Secure".**

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.

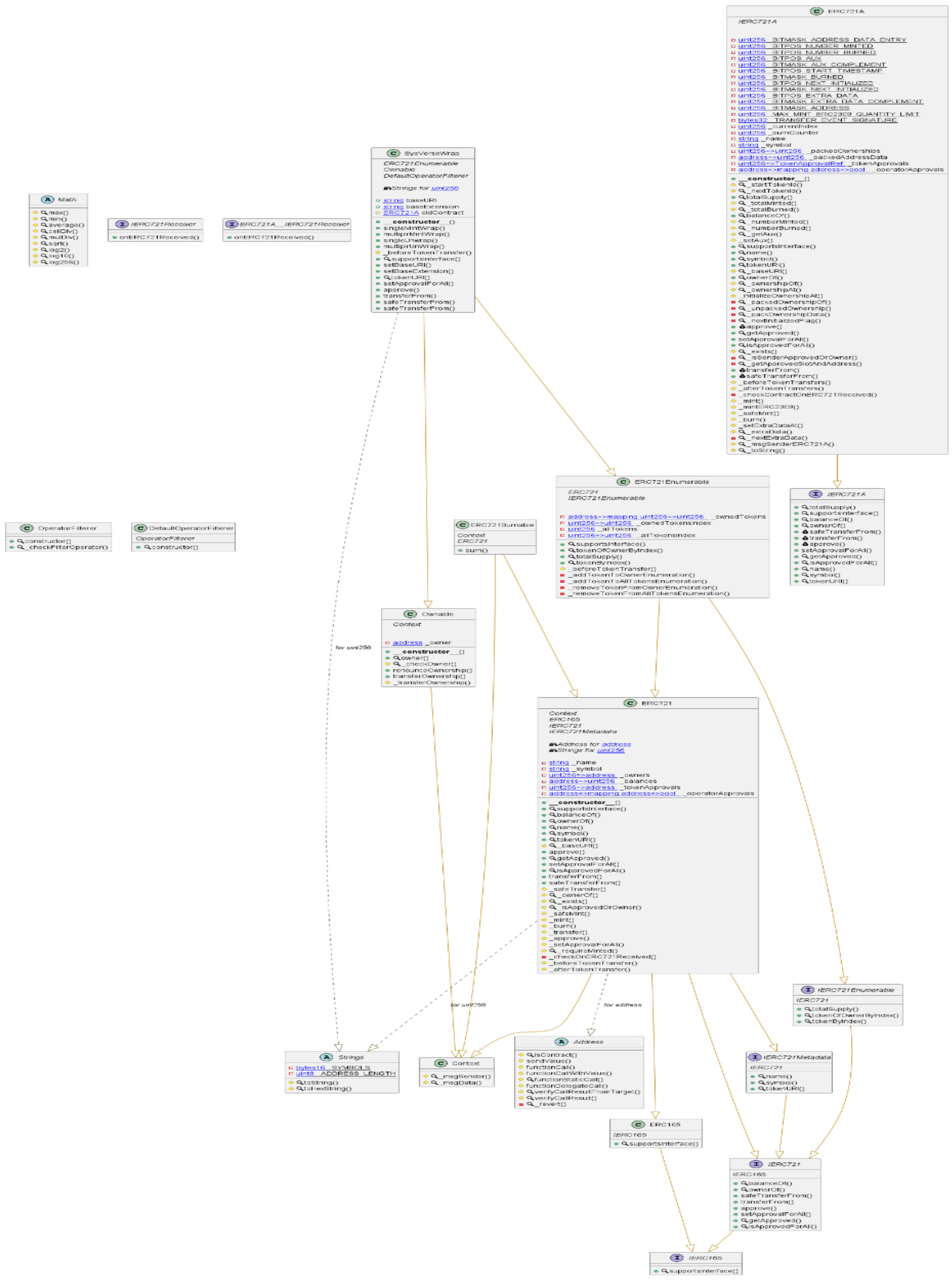
Technical Disclaimer

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.

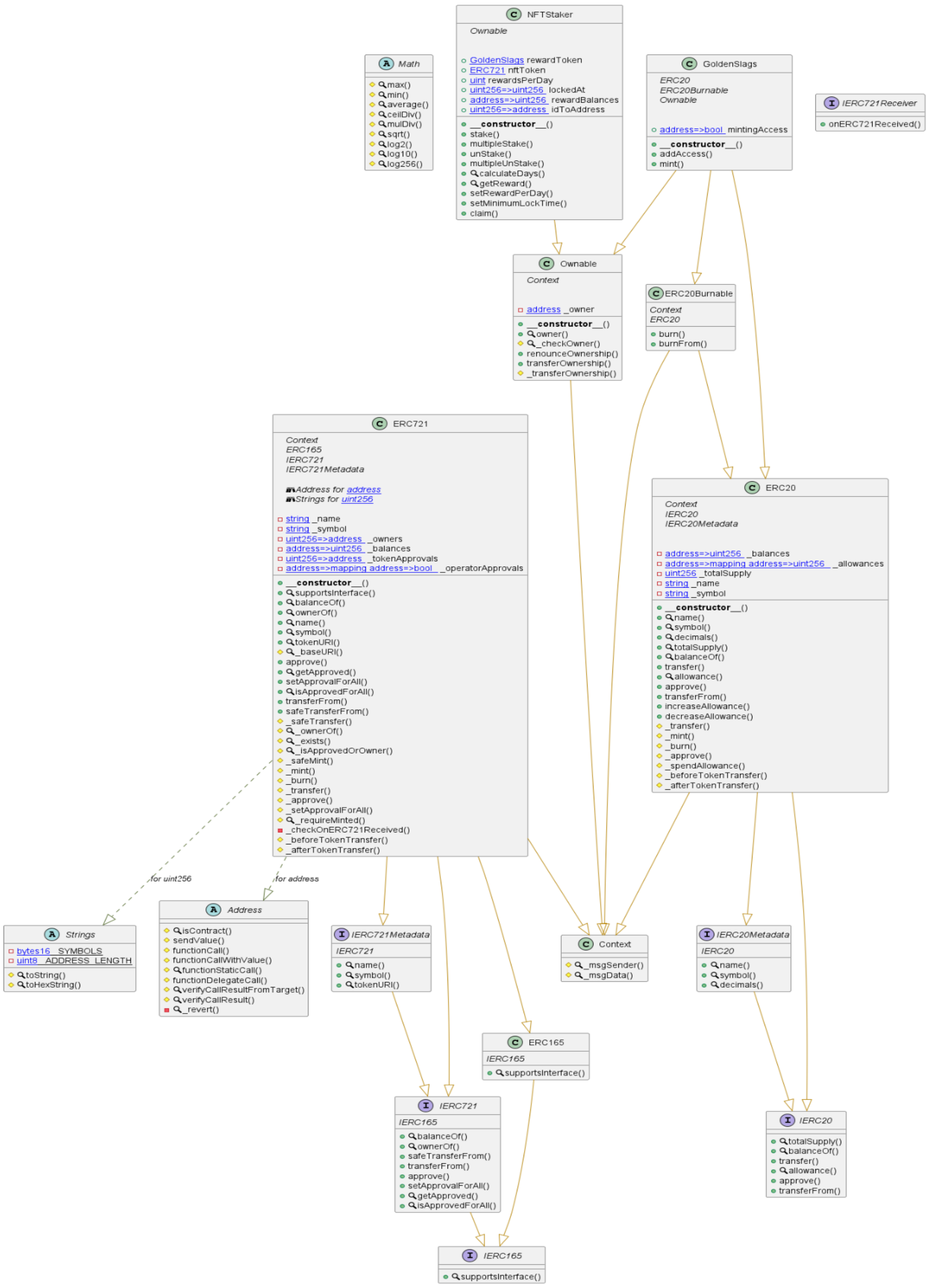
Appendix

Code Flow Diagram - Eyeverse Protocol

EyeVerseWrap Diagram

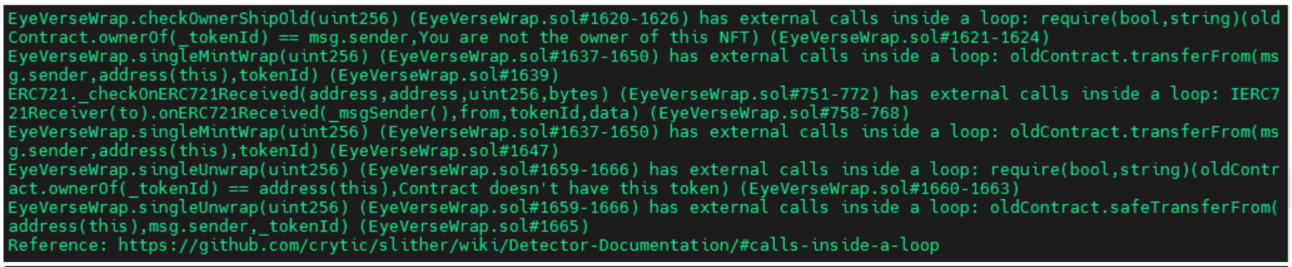


NFTStaker Diagram

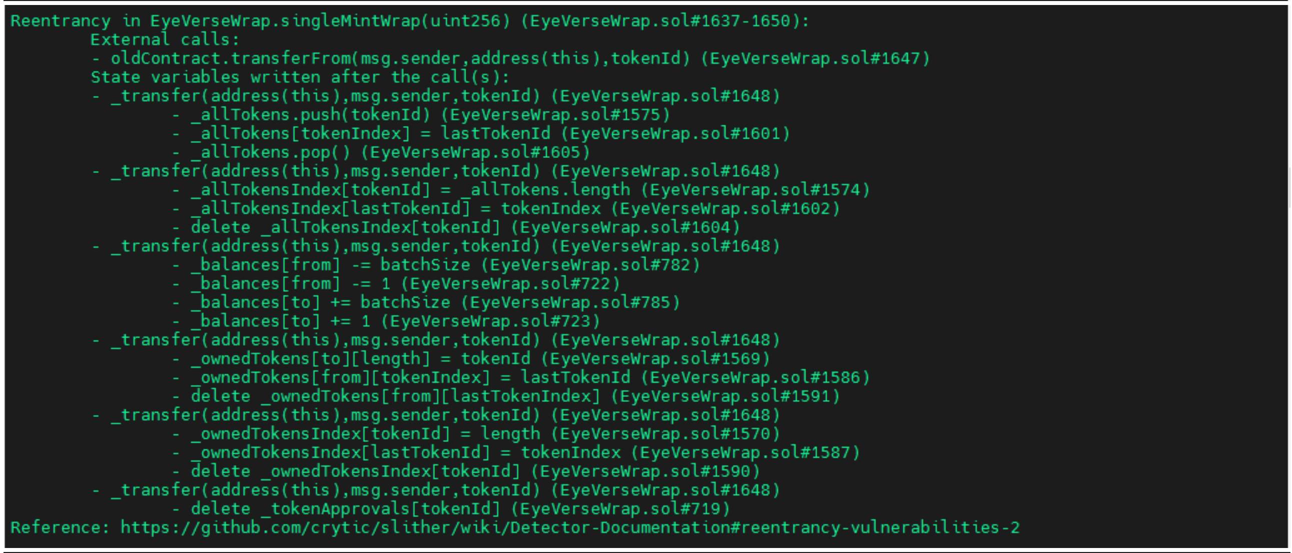


Slither Results Log

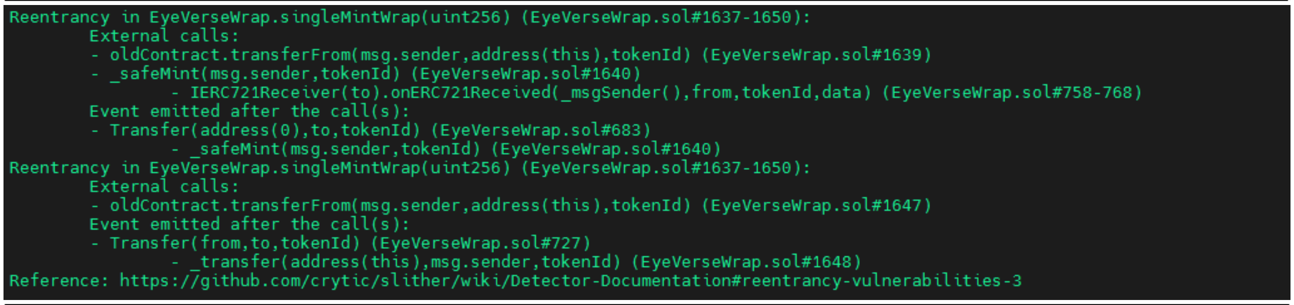
Slither Log >> EyeVerseWrap.sol



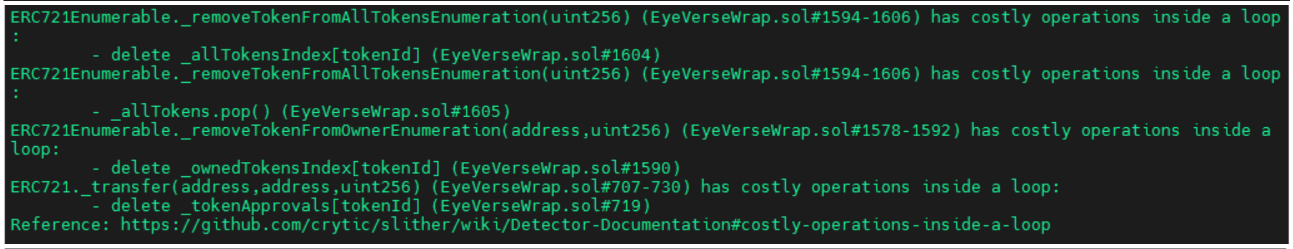
```
EyeVerseWrap.checkOwnerShipOld(uint256) (EyeVerseWrap.sol#1620-1626) has external calls inside a loop: require(bool,string)(old
Contract.ownerOf(_tokenId) == msg.sender,You are not the owner of this NFT) (EyeVerseWrap.sol#1621-1624)
EyeVerseWrap.singleMintWrap(uint256) (EyeVerseWrap.sol#1637-1650) has external calls inside a loop: oldContract.transferFrom(ms
g.sender,address(this),tokenId) (EyeVerseWrap.sol#1639)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (EyeVerseWrap.sol#751-772) has external calls inside a loop: IERC7
21Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (EyeVerseWrap.sol#758-768)
EyeVerseWrap.singleMintWrap(uint256) (EyeVerseWrap.sol#1637-1650) has external calls inside a loop: oldContract.transferFrom(ms
g.sender,address(this),tokenId) (EyeVerseWrap.sol#1647)
EyeVerseWrap.singleUnwrap(uint256) (EyeVerseWrap.sol#1659-1666) has external calls inside a loop: require(bool,string)(oldContr
act.ownerOf(_tokenId) == address(this),Contract doesn't have this token) (EyeVerseWrap.sol#1660-1663)
EyeVerseWrap.singleUnwrap(uint256) (EyeVerseWrap.sol#1659-1666) has external calls inside a loop: oldContract.safeTransferFrom(
address(this),msg.sender,_tokenId) (EyeVerseWrap.sol#1665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```



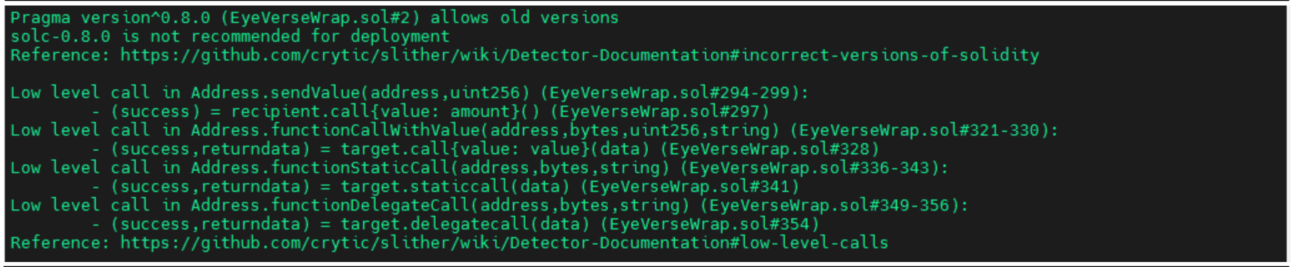
```
Reentrancy in EyeVerseWrap.singleMintWrap(uint256) (EyeVerseWrap.sol#1637-1650):
        External calls:
        - oldContract.transferFrom(msg.sender,address(this),tokenId) (EyeVerseWrap.sol#1647)
        State variables written after the call(s):
        - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
                - _allTokens.push(tokenId) (EyeVerseWrap.sol#1575)
                - _allTokens[tokenIndex] = lastTokenId (EyeVerseWrap.sol#1601)
                - _allTokens.pop() (EyeVerseWrap.sol#1605)
        - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
                - _allTokensIndex[tokenId] = _allTokens.length (EyeVerseWrap.sol#1574)
                - _allTokensIndex[lastTokenId] = tokenIndex (EyeVerseWrap.sol#1602)
                - delete _allTokensIndex[tokenId] (EyeVerseWrap.sol#1604)
        - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
                - _balances[from] -= batchSize (EyeVerseWrap.sol#782)
                - _balances[from] -= 1 (EyeVerseWrap.sol#722)
                - _balances[to] += batchSize (EyeVerseWrap.sol#785)
                - _balances[to] += 1 (EyeVerseWrap.sol#723)
        - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
                - _ownedTokens[to][length] = tokenId (EyeVerseWrap.sol#1569)
                - _ownedTokens[from][tokenIndex] = lastTokenId (EyeVerseWrap.sol#1586)
                - delete _ownedTokens[from][lastTokenIndex] (EyeVerseWrap.sol#1591)
        - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
                - _ownedTokensIndex[tokenId] = length (EyeVerseWrap.sol#1570)
                - _ownedTokensIndex[lastTokenId] = tokenIndex (EyeVerseWrap.sol#1587)
                - delete _ownedTokensIndex[tokenId] (EyeVerseWrap.sol#1590)
        - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
                - delete _tokenApprovals[tokenId] (EyeVerseWrap.sol#719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```



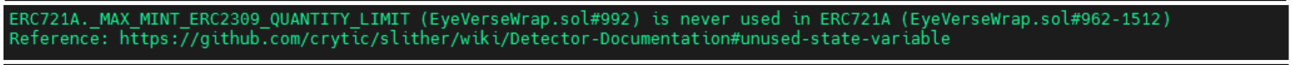
```
Reentrancy in EyeVerseWrap.singleMintWrap(uint256) (EyeVerseWrap.sol#1637-1650):
        External calls:
        - oldContract.transferFrom(msg.sender,address(this),tokenId) (EyeVerseWrap.sol#1639)
        - _safeMint(msg.sender,tokenId) (EyeVerseWrap.sol#1640)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (EyeVerseWrap.sol#758-768)
        Event emitted after the call(s):
        - Transfer(address(0),to,tokenId) (EyeVerseWrap.sol#683)
                - _safeMint(msg.sender,tokenId) (EyeVerseWrap.sol#1640)
Reentrancy in EyeVerseWrap.singleMintWrap(uint256) (EyeVerseWrap.sol#1637-1650):
        External calls:
        - oldContract.transferFrom(msg.sender,address(this),tokenId) (EyeVerseWrap.sol#1647)
        Event emitted after the call(s):
        - Transfer(from,to,tokenId) (EyeVerseWrap.sol#727)
                - _transfer(address(this),msg.sender,tokenId) (EyeVerseWrap.sol#1648)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



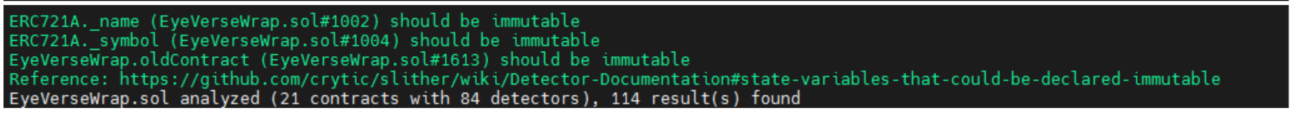
```
ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (EyeVerseWrap.sol#1594-1606) has costly operations inside a loop
:
        - delete _allTokensIndex[tokenId] (EyeVerseWrap.sol#1604)
ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (EyeVerseWrap.sol#1594-1606) has costly operations inside a loop
:
        - _allTokens.pop() (EyeVerseWrap.sol#1605)
ERC721Enumerable._removeTokenFromOwnerEnumeration(address,uint256) (EyeVerseWrap.sol#1578-1592) has costly operations inside a
loop:
        - delete _ownedTokensIndex[tokenId] (EyeVerseWrap.sol#1590)
ERC721._transfer(address,address,uint256) (EyeVerseWrap.sol#707-730) has costly operations inside a loop:
        - delete _tokenApprovals[tokenId] (EyeVerseWrap.sol#719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```



```
Pragma version^0.8.0 (EyeVerseWrap.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (EyeVerseWrap.sol#294-299):
        - (success) = recipient.call{value: amount}() (EyeVerseWrap.sol#297)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (EyeVerseWrap.sol#321-330):
        - (success,returndata) = target.call{value: value}(data) (EyeVerseWrap.sol#328)
Low level call in Address.functionStaticCall(address,bytes,string) (EyeVerseWrap.sol#336-343):
        - (success,returndata) = target.staticcall(data) (EyeVerseWrap.sol#341)
Low level call in Address.functionDelegateCall(address,bytes,string) (EyeVerseWrap.sol#349-356):
        - (success,returndata) = target.delegatecall(data) (EyeVerseWrap.sol#354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

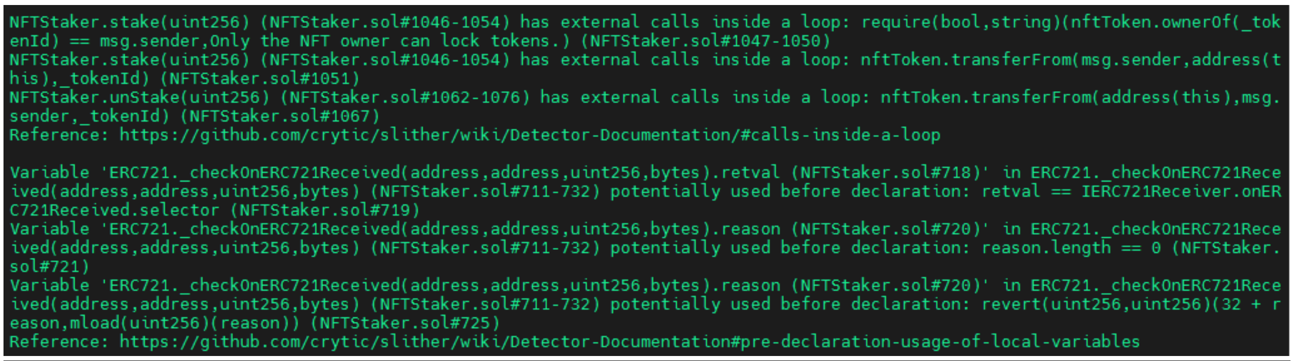


```
ERC721A._MAX_MINT_ERC2309_QUANTITY_LIMIT (EyeVerseWrap.sol#992) is never used in ERC721A (EyeVerseWrap.sol#962-1512)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

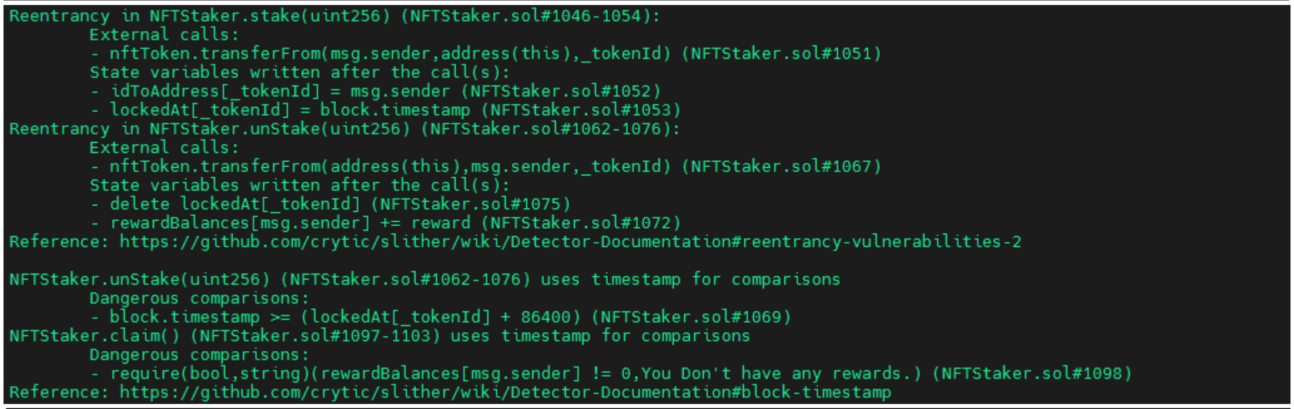


```
ERC721A._name (EyeVerseWrap.sol#1002) should be immutable
ERC721A._symbol (EyeVerseWrap.sol#1004) should be immutable
EyeVerseWrap.oldContract (EyeVerseWrap.sol#1613) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
EyeVerseWrap.sol analyzed (21 contracts with 84 detectors), 114 result(s) found
```

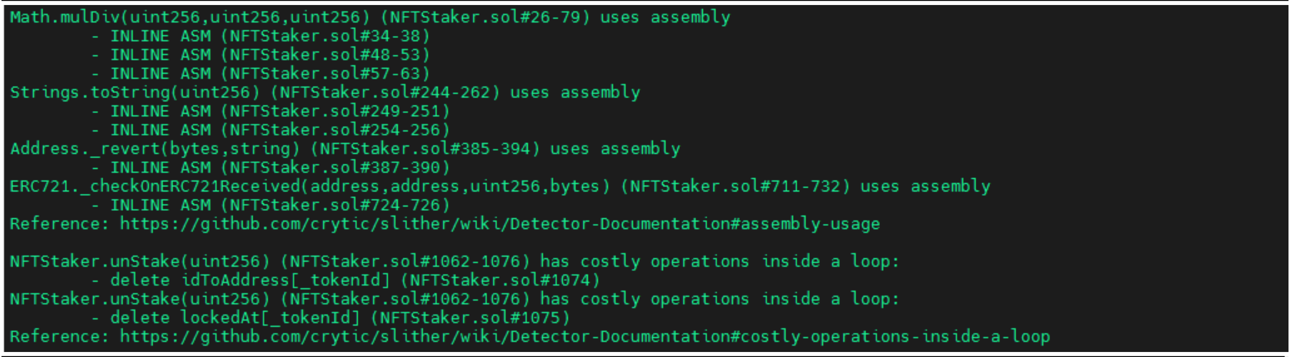
Slither Log >> NFTStaker.sol



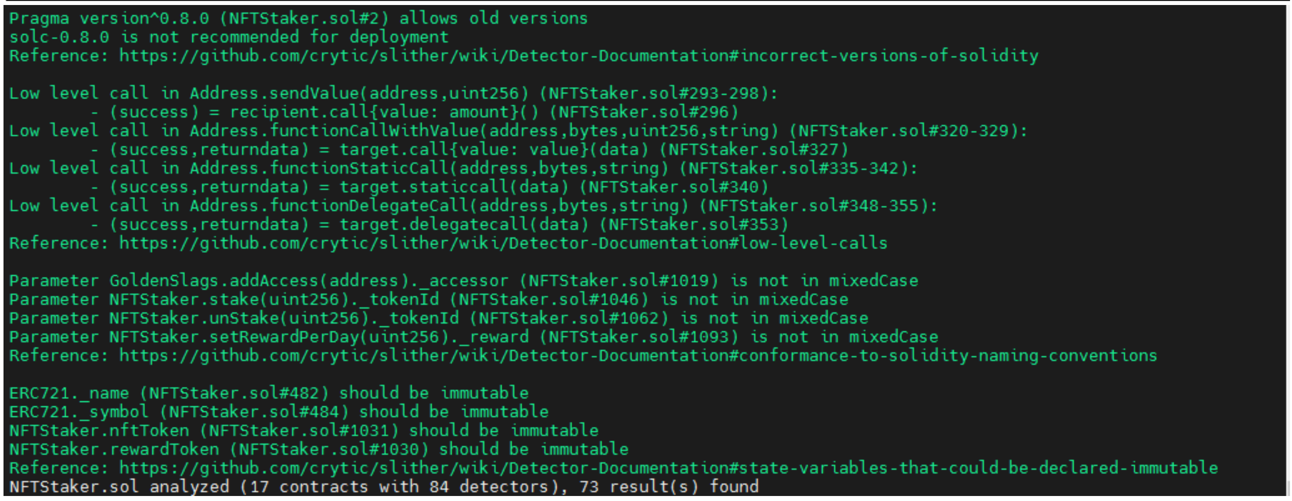
```
NFTStaker.stake(uint256) (NFTStaker.sol#1046-1054) has external calls inside a loop: require(bool,string)(nftToken.ownerOf(_tok
enId) == msg.sender,Only the NFT owner can lock tokens.) (NFTStaker.sol#1047-1050)
NFTStaker.stake(uint256) (NFTStaker.sol#1046-1054) has external calls inside a loop: nftToken.transferFrom(msg.sender,address(t
his),_tokenId) (NFTStaker.sol#1051)
NFTStaker.unStake(uint256) (NFTStaker.sol#1062-1076) has external calls inside a loop: nftToken.transferFrom(address(this),msg.
sender,_tokenId) (NFTStaker.sol#1067)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (NFTStaker.sol#718)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (NFTStaker.sol#711-732) potentially used before declaration: retval == IERC721Receiver.onER
C721Received.selector (NFTStaker.sol#719)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (NFTStaker.sol#720)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (NFTStaker.sol#711-732) potentially used before declaration: reason.length == 0 (NFTStaker.
sol#721)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (NFTStaker.sol#720)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (NFTStaker.sol#711-732) potentially used before declaration: revert(uint256,uint256)(32 + r
eason,mload(uint256)(reason)) (NFTStaker.sol#725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```



```
Reentrancy in NFTStaker.stake(uint256) (NFTStaker.sol#1046-1054):
        External calls:
        - nftToken.transferFrom(msg.sender,address(this),_tokenId) (NFTStaker.sol#1051)
        State variables written after the call(s):
        - idToAddress[_tokenId] = msg.sender (NFTStaker.sol#1052)
        - lockedAt[_tokenId] = block.timestamp (NFTStaker.sol#1053)
Reentrancy in NFTStaker.unStake(uint256) (NFTStaker.sol#1062-1076):
        External calls:
        - nftToken.transferFrom(address(this),msg.sender,_tokenId) (NFTStaker.sol#1067)
        State variables written after the call(s):
        - delete lockedAt[_tokenId] (NFTStaker.sol#1075)
        - rewardBalances[msg.sender] += reward (NFTStaker.sol#1072)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

NFTStaker.unStake(uint256) (NFTStaker.sol#1062-1076) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= (lockedAt[_tokenId] + 86400) (NFTStaker.sol#1069)
NFTStaker.claim() (NFTStaker.sol#1097-1103) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(rewardBalances[msg.sender] != 0,You Don't have any rewards.) (NFTStaker.sol#1098)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```



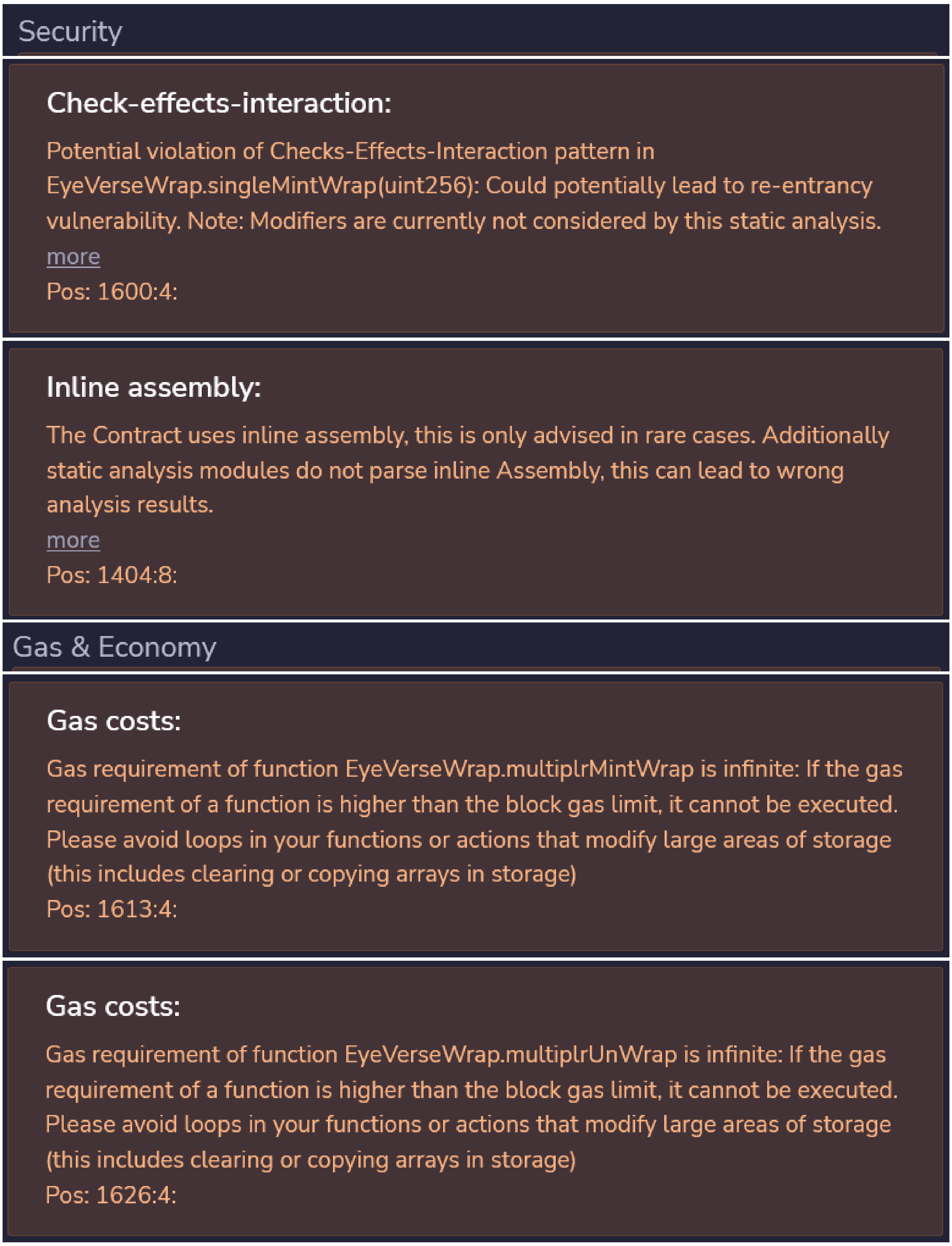
```
Math.mulDiv(uint256,uint256,uint256) (NFTStaker.sol#26-79) uses assembly
        - INLINE ASM (NFTStaker.sol#34-38)
        - INLINE ASM (NFTStaker.sol#48-53)
        - INLINE ASM (NFTStaker.sol#57-63)
Strings.toString(uint256) (NFTStaker.sol#244-262) uses assembly
        - INLINE ASM (NFTStaker.sol#249-251)
        - INLINE ASM (NFTStaker.sol#254-256)
Address._revert(bytes,string) (NFTStaker.sol#385-394) uses assembly
        - INLINE ASM (NFTStaker.sol#387-390)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (NFTStaker.sol#711-732) uses assembly
        - INLINE ASM (NFTStaker.sol#724-726)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

NFTStaker.unStake(uint256) (NFTStaker.sol#1062-1076) has costly operations inside a loop:
        - delete idToAddress[_tokenId] (NFTStaker.sol#1074)
NFTStaker.unStake(uint256) (NFTStaker.sol#1062-1076) has costly operations inside a loop:
        - delete lockedAt[_tokenId] (NFTStaker.sol#1075)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```



```
Pragma version^0.8.0 (NFTStaker.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (NFTStaker.sol#293-298):
        - (success) = recipient.call{value: amount}() (NFTStaker.sol#296)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (NFTStaker.sol#320-329):
        - (success,returndata) = target.call{value: value}(data) (NFTStaker.sol#327)
Low level call in Address.functionStaticCall(address,bytes,string) (NFTStaker.sol#335-342):
        - (success,returndata) = target.staticcall(data) (NFTStaker.sol#340)
Low level call in Address.functionDelegateCall(address,bytes,string) (NFTStaker.sol#348-355):
        - (success,returndata) = target.delegatecall(data) (NFTStaker.sol#353)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter GoldenSlags.addAccess(address)._accessor (NFTStaker.sol#1019) is not in mixedCase
Parameter NFTStaker.stake(uint256)._tokenId (NFTStaker.sol#1046) is not in mixedCase
Parameter NFTStaker.unStake(uint256)._tokenId (NFTStaker.sol#1062) is not in mixedCase
Parameter NFTStaker.setRewardPerDay(uint256)._reward (NFTStaker.sol#1093) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ERC721._name (NFTStaker.sol#482) should be immutable
ERC721._symbol (NFTStaker.sol#484) should be immutable
NFTStaker.nftToken (NFTStaker.sol#1031) should be immutable
NFTStaker.rewardToken (NFTStaker.sol#1030) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
NFTStaker.sol analyzed (17 contracts with 84 detectors), 73 result(s) found
```

Solidity Static Analysis

EyeVerseWrap.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in EyeVerseWrap.singleMintWrap(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1600:4:

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: 1404:8:

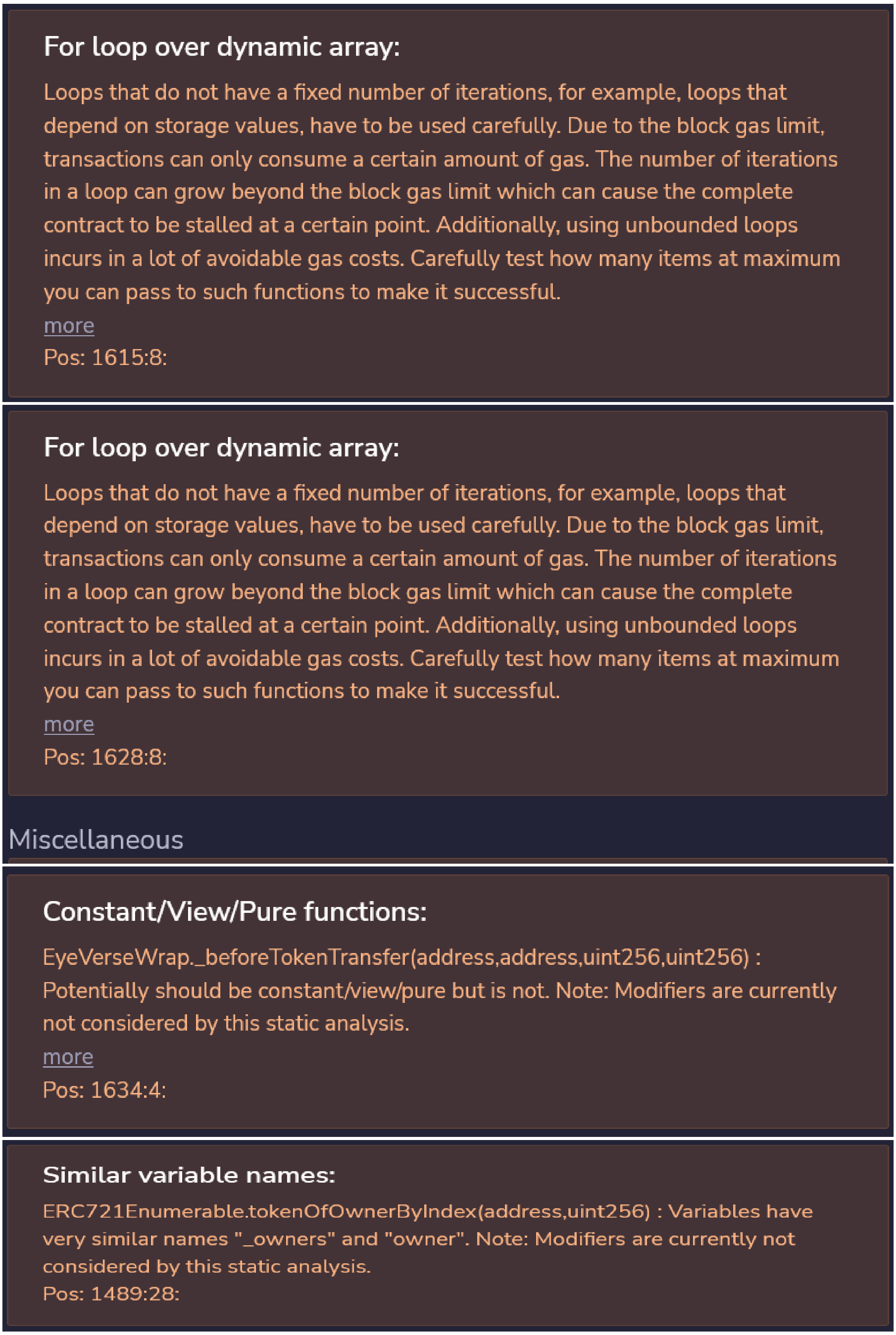
Gas & Economy

Gas costs:

Gas requirement of function EyeVerseWrap.multiplrMintWrap 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: 1613:4:

Gas costs:

Gas requirement of function EyeVerseWrap.multiplrUnWrap 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: 1626:4:



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: 1615:8:

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: 1628:8:

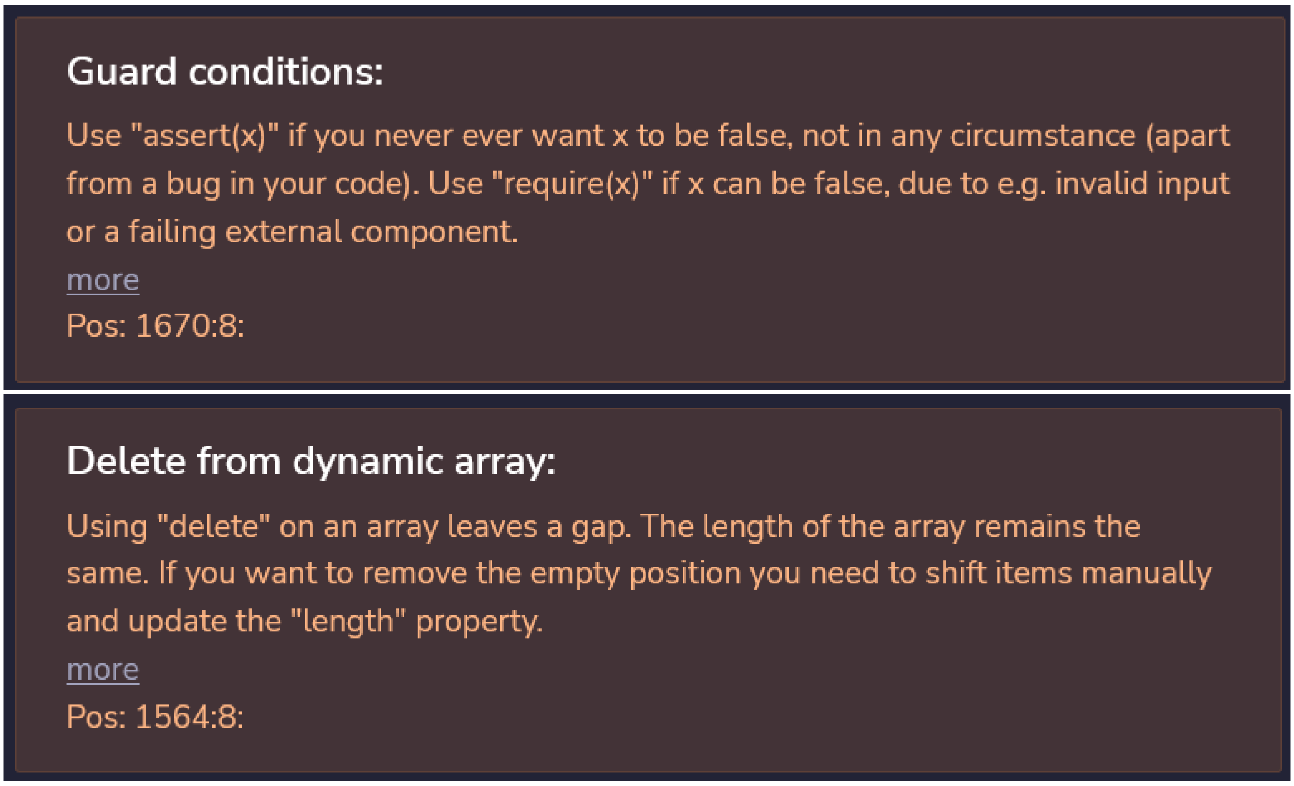
Miscellaneous

Constant/View/Pure functions:

EyeVerseWrap._beforeTokenTransfer(address,address,uint256,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1634:4:

Similar variable names:

ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) : Variables have very similar names "_owners" and "owner". Note: Modifiers are currently not considered by this static analysis.
Pos: 1489:28:



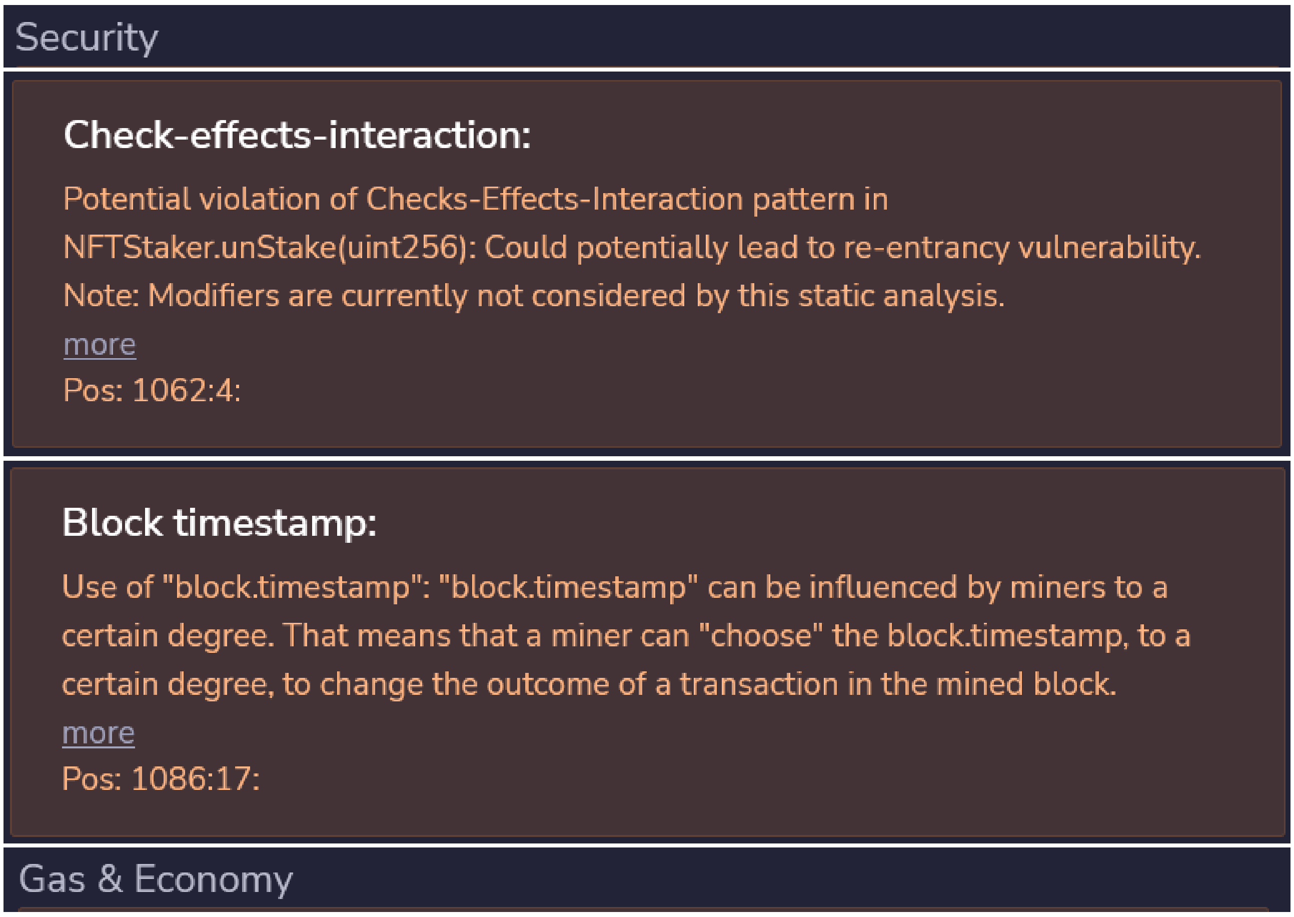
Guard conditions:

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: 1670:8:

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: 1564:8:

NFTStaker.sol



Security

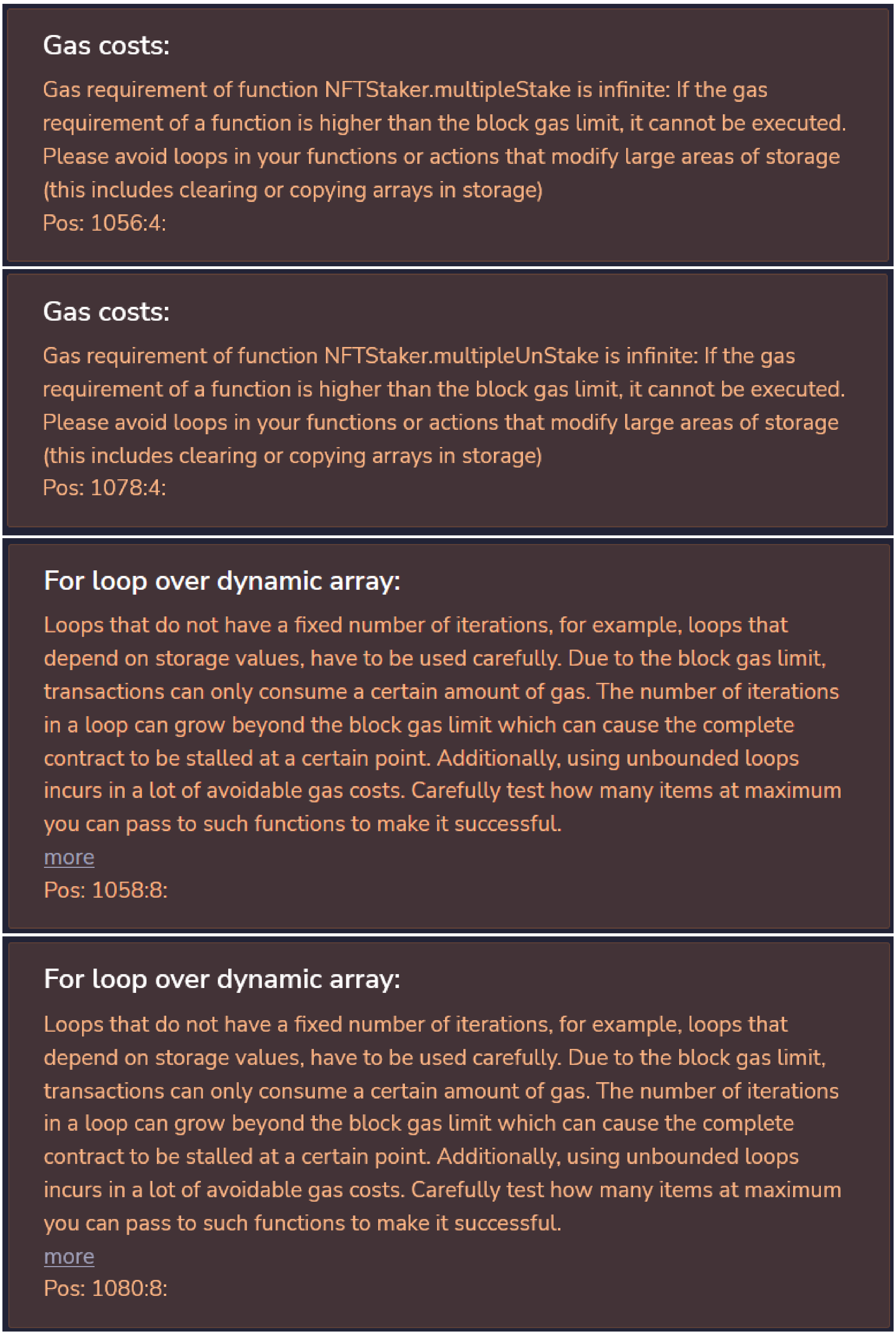
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in NFTStaker.unStake(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1062:4:

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: 1086:17:

Gas & Economy



Gas costs:

Gas requirement of function NFTStaker.multipleStake 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: 1056:4:

Gas costs:

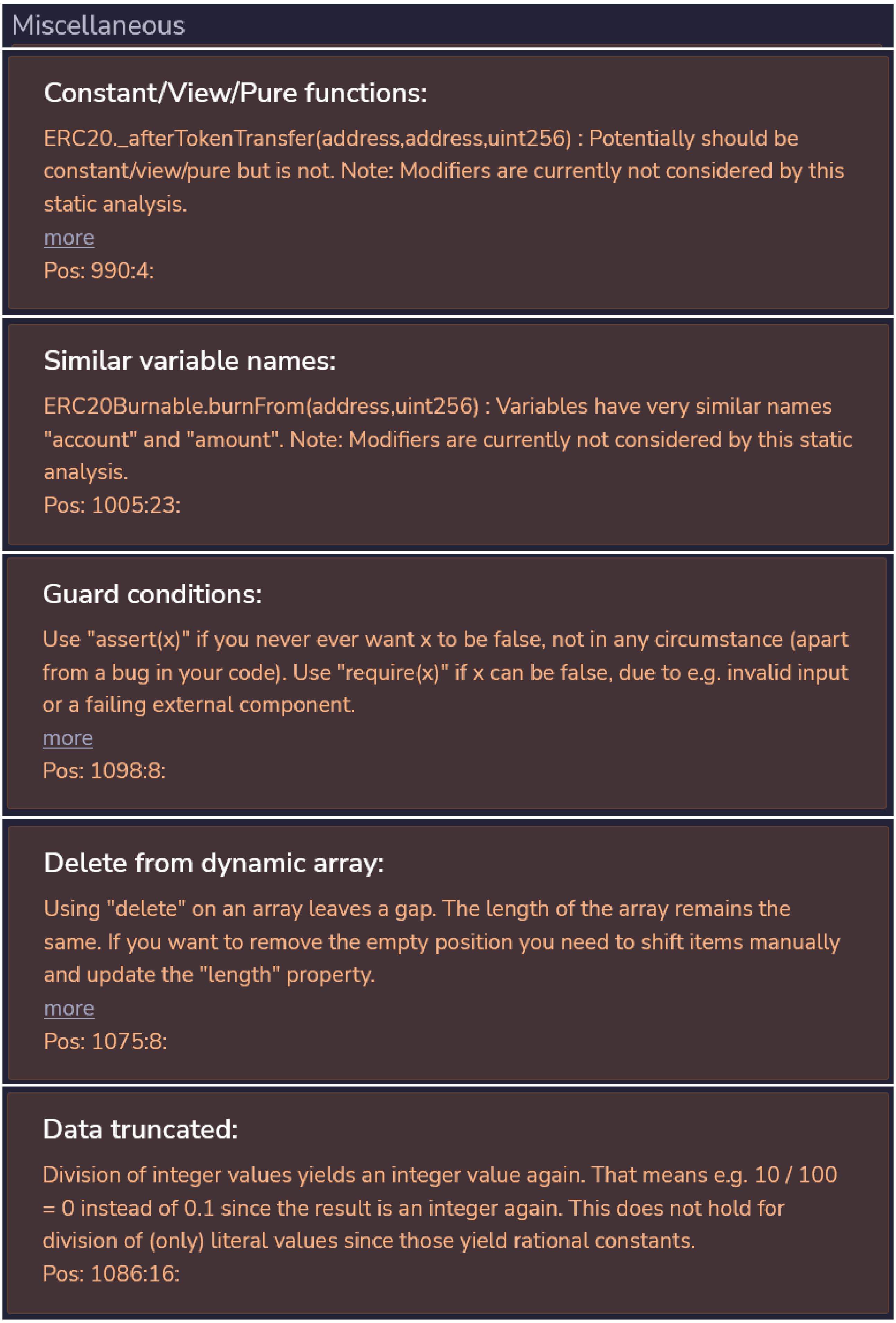
Gas requirement of function NFTStaker.multipleUnStake is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1078:4:

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: 1058:8:

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: 1080:8:



Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 990:4:

Similar variable names:

ERC20Burnable.burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1005:23:

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: 1098:8:

Delete from dynamic array:

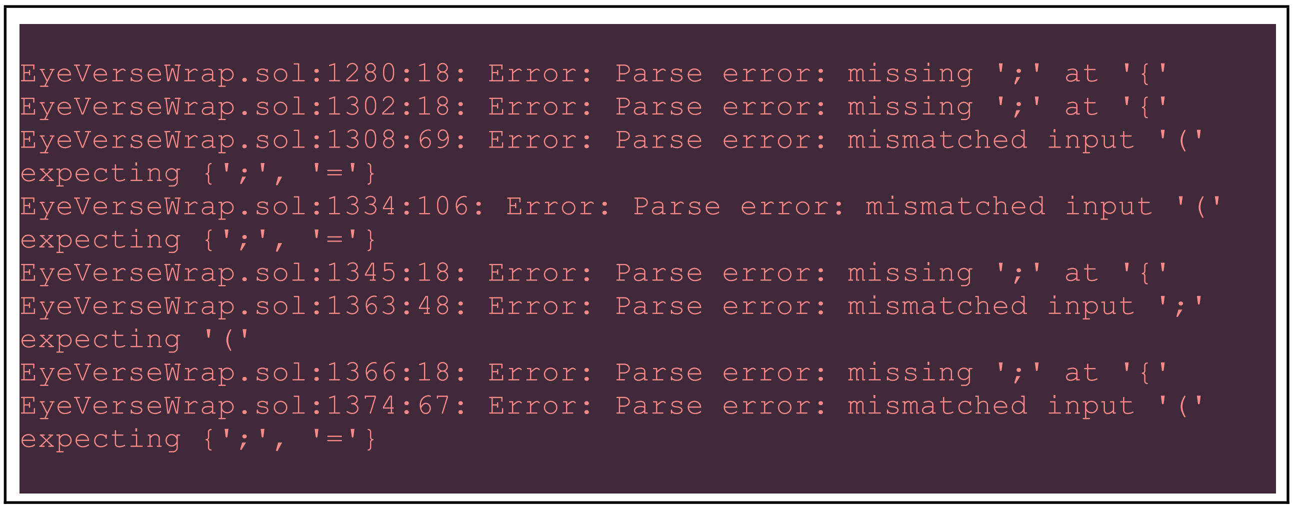
Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1075:8:

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: 1086:16:

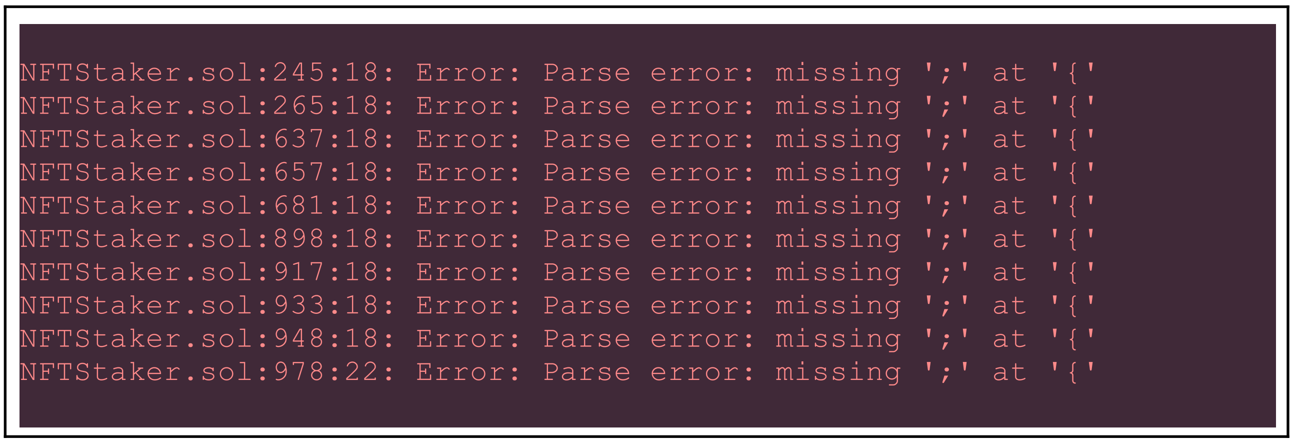
Solhint Linter

EyeVerseWrap.sol



```
EyeVerseWrap.sol:1280:18: Error: Parse error: missing ';' at '{'
EyeVerseWrap.sol:1302:18: Error: Parse error: missing ';' at '{'
EyeVerseWrap.sol:1308:69: Error: Parse error: mismatched input '('
expecting {';', '='}
EyeVerseWrap.sol:1334:106: Error: Parse error: mismatched input '('
expecting {';', '='}
EyeVerseWrap.sol:1345:18: Error: Parse error: missing ';' at '{'
EyeVerseWrap.sol:1363:48: Error: Parse error: mismatched input ';'
expecting '('
EyeVerseWrap.sol:1366:18: Error: Parse error: missing ';' at '{'
EyeVerseWrap.sol:1374:67: Error: Parse error: mismatched input '('
expecting {';', '='}
```

NFTStaker.sol



```
NFTStaker.sol:245:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:265:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:637:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:657:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:681:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:898:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:917:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:933:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:948:18: Error: Parse error: missing ';' at '{'
NFTStaker.sol:978:22: 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.