Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	ArtArmy Protocol
Platform:	Polygon
Website:	https://art.army
Language:	Solidity
Date:	November 19th, 2021

Table of contents

Introduction

EtherAuthority was contracted by the ArtArmy team to perform the Security audit of the ArtArmy Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on November 19th, 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

ArtArmy is a platform to integrate artists, gallery owners, curators, investors and fans revolving around the new technology of art.

Users can auction, exchange, buy and sell pieces of digital art and virtual spaces in the metaverse through blockchain and NFT technology.

Audit scope

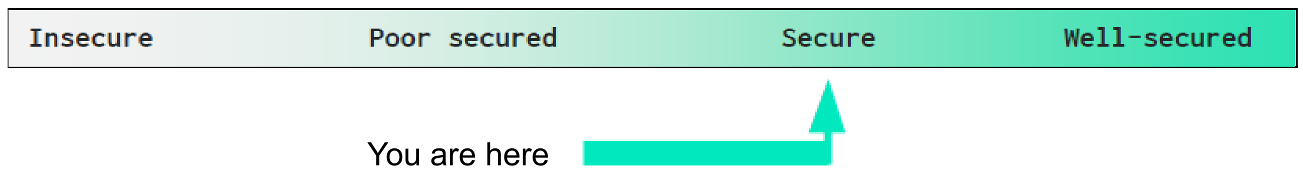
Name	**Code Review and Security Analysis Report for ArtArmy Protocol Smart Contracts**
Platform	**Polygon / Solidity**
File 1	ArtArmyArtwork.sol
File 1 MD5 Hash	636AC770B5EF86171B546892EA8A18EE
File 2	ArtArmySeller.sol
File 2 MD5 Hash	B853C486748982B20744800F2B56F6E5
File 3	ArtArmyHoldersStake.sol
File 3 MD5 Hash	EE6AB053AAA58382C8FC5D50D70FD686
Audit Date	November 19th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 ArtArmyArtwork.sol ● The AetArmyArtwork can access functionality like: Set the auction contract , Set the fixed sales contract address, etc.	**YES, This is valid.**
File 2 ArtArmySeller.sol ● The AetArmySeller can access functionality like:Returns the stake address, Returns the Holders Fee in basis points, Returns the Token BEP20 used for the transactions, etc.	**YES, This is valid.**
File 3 ArtArmyHoldersStake.sol ● The ArtArmyStakeHolders can access functionality like: add and remove new stake, removeInvestor, etc.	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 3 low and some very low level issues. These issues are not critical ones.

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
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	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 3 smart contract files. Smart contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in ArtArmy 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 ArtArmy Protocol.

The ArtArmy Protocol 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 ArtArmy Protocol smart contracts code in the form of a github web link.The hash of that code is 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.

Another source of information was its official website https://art.army which provided rich information about the project architecture and tokenomics.

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

ArtArmyArtwork.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAuctionContract	read	Passed	No Issue
3	getSellerContract	read	Passed	No Issue
4	setAuctionContract	external	Passed	No Issue
5	setSellerContract	external	Passed	No Issue
6	_baseURI	internal	Passed	No Issue
7	getMaximumRoyaltyInBasisPoints	read	Passed	No Issue
8	setMaximumRoyaltyInBasisPoints	external	Passed	No Issue
9	mint	write	Passed	No Issue
10	pause	write	Passed	No Issue
11	unpause	write	Passed	No Issue
12	_beforeTokenTransfer	internal	Passed	No Issue
13	supportsInterface	read	Passed	No Issue
14	_setTokenArtist	internal	Passed	No Issue
15	getArtistWallet	read	Passed	No Issue
16	getArtistRoyalty	read	Passed	No Issue
17	tokenURI	read	Passed	No Issue
18	_setTokenURI	internal	Passed	No Issue
19	_burn	internal	Passed	No Issue
20	isApprovedForAll	read	Passed	No Issue
21	supportsInterface	read	Passed	No Issue
22	getRoleMember	read	Passed	No Issue
23	getRoleMemberCount	read	Passed	No Issue
24	_grantRole	internal	Passed	No Issue
25	_revokeRole	internal	Passed	No Issue
26	supportsInterface	read	Passed	No Issue
27	tokenOfOwnerByIndex	read	Passed	No Issue
28	totalSupply	read	Passed	No Issue
29	tokenByIndex	read	Passed	No Issue
30	_beforeTokenTransfer	internal	Passed	No Issue
31	_addTokenToOwnerEnumeration	write	Passed	No Issue
32	_addTokenToAllTokensEnumeration	write	Passed	No Issue
33	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
34	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
35	burn	write	Passed	No Issue

36	_beforeTokenTransfer	internal	Passed	No Issue
37	tokenURI	read	Passed	No Issue
38	_setTokenURI	internal	Passed	No Issue
39	_burn	internal	Passed	No Issue
40	owner	read	Passed	No Issue
41	onlyOwner	modifier	Passed	No Issue
42	renounceOwnership	write	access only Owner	No Issue
43	transferOwnership	write	access only Owner	No Issue
44	_transferOwnership	internal	Passed	No Issue

ArtArmySeller.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOnSale	read	Passed	No Issue
3	getSalePrice	read	Passed	No Issue
4	getSaleCuratorWallet	read	Passed	No Issue
5	getSaleCuratorFeeInBasisPoints	read	Passed	No Issue
6	getSaleSellerWallet	read	Passed	No Issue
7	getContractName	read	Passed	No Issue
8	getNftContract	read	Passed	No Issue
9	getArtArmyTreasury	read	Passed	No Issue
10	getArtArmyFee	read	Passed	No Issue
11	getHoldersStakeContract	read	Passed	No Issue
12	getHoldersFee	read	Passed	No Issue
13	getCurrency	read	Passed	No Issue
14	getMaximumCuratorFeeInBasisPoints	read	Passed	No Issue
15	setArtArmyFee	external	Passed	No Issue
16	setHoldersFee	external	Passed	No Issue
17	setCurrency	external	Passed	No Issue
18	setMaximumCuratorFeeInBasisPoints	external	Passed	No Issue
19	_tokenExists	internal	Passed	No Issue
20	launchSale	external	Passed	No Issue
21	emergencyTransferNft	external	Passed	No Issue
22	makePurchase	external	Passed	No Issue
23	getBenefit	internal	Passed	No Issue
24	endSale	internal	Passed	No Issue
25	sendViaCall	write	Unused local variable	Refer Audit Findings
26	owner	read	Passed	No Issue
27	onlyOwner	modifier	Passed	No Issue

28	renounceOwnership	write	access only Owner	No Issue
29	transferOwnership	write	access only Owner	No Issue
30	_transferOwnership	internal	Passed	No Issue
31	onERC721Received	write	Passed	No Issue

ArtArmyHoldersStake.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	getContractName	read	Passed	No Issue
3	getInvestorAmount	external	Passed	No Issue
4	getInvestorEarns	external	Passed	No Issue
5	addStake	external	Passed	No Issue
6	removeStake	external	Passed	No Issue
7	removeInvestor	write	Critical operation lacks event log, Infinite loop possibility	Refer Audit Findings
8	distributeEarns	write	Critical operation lacks event log, Infinite loop possibility	Refer Audit Findings

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Balance must be greater than 0: **- ArtArmyHoldersStake.sol**

In the distributeEarns function, contractBalance has been used to distribute tokens among holders.

Resolution: We suggest checking contractBalance before executing the for loop.

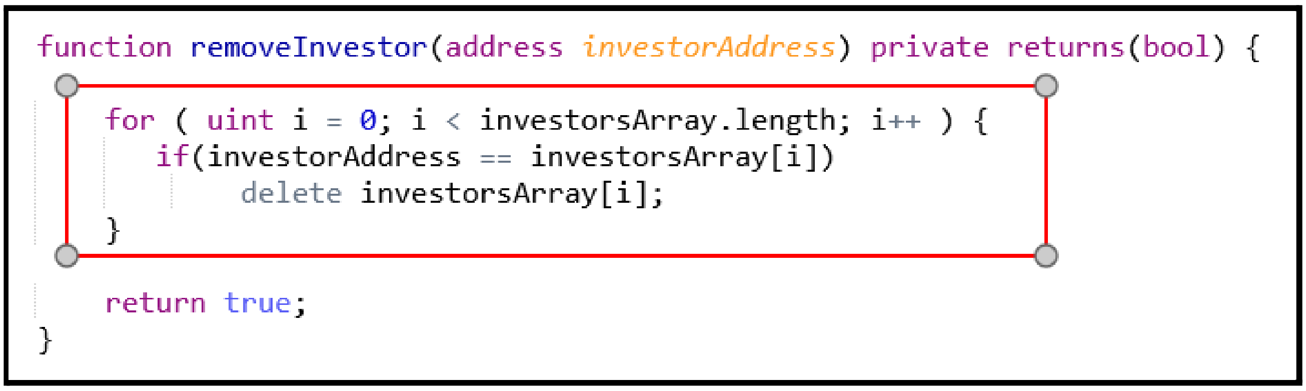
(2) Critical operation lacks event log: **- ArtArmyHoldersStake.sol**

Some functions need to log events.

Resolution: We suggest adding log for functions listed below:

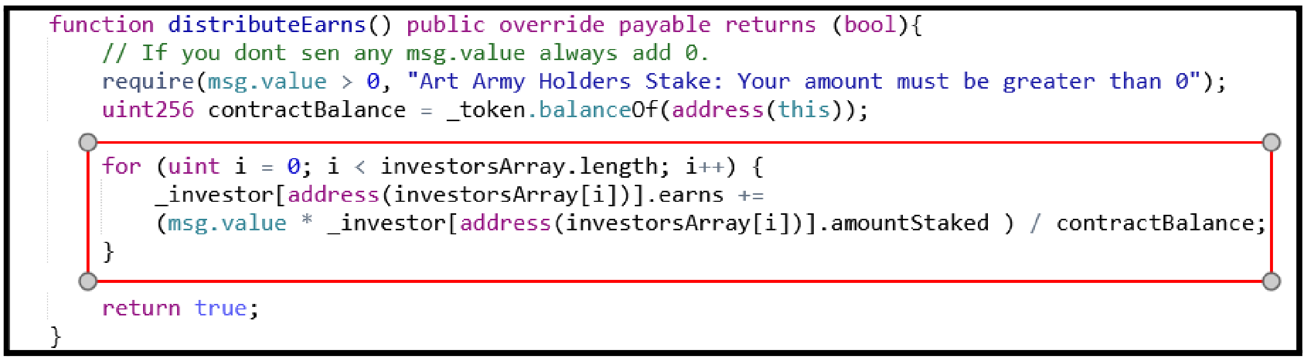
- removeInvestor
- distributeEarns.

(3) Infinite loop possibility: **- ArtArmyHoldersStake.sol**



```
function removeInvestor(address investorAddress) private returns(bool) {
    for ( uint i = 0; i < investorsArray.length; i++ ) {
       if(investorAddress == investorsArray[i])
           delete investorsArray[i];
    }

    return true;
}
```



```
function distributeEarns() public override payable returns (bool){
    // If you dont sen any msg.value always add 0.
    require(msg.value > 0, "Art Army Holders Stake: Your amount must be greater than 0");
    uint256 contractBalance = _token.balanceOf(address(this));

    for (uint i = 0; i < investorsArray.length; i++) {
        _investor[address(investorsArray[i])].earns +=
        (msg.value * _investor[address(investorsArray[i])].amountStaked ) / contractBalance;
    }

    return true;
}
```

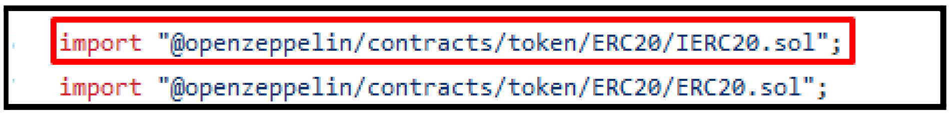
In these functions, array length can be more and thus it gets reverted because of high gas issues.

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

Very Low / Informational / Best practices:

(1) Import the same file:

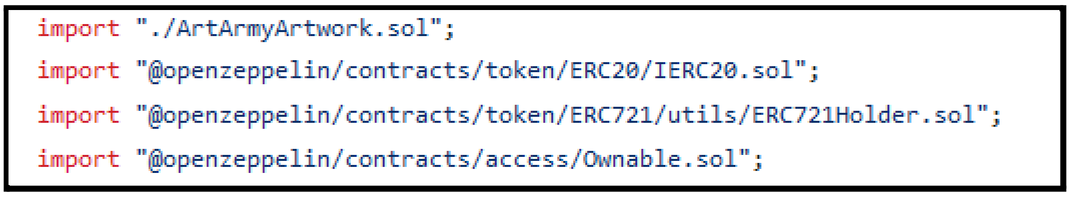
ArtArmyHoldersStake.sol



```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
```

Here IERC20.sol and ERC20.sol are imported. but IERC20.sol is already imported in ERC20.sol.

ArtArmySeller.sol

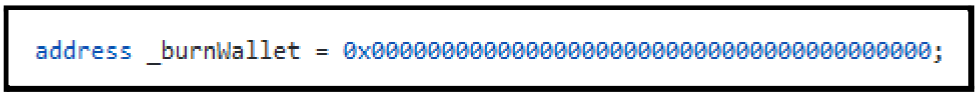


```
import "./ArtArmyArtwork.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
```

Here Ownable.sol is imported. but Ownable.sol is already imported in ArtArmyArtwork.sol.

Resolution: We suggest removing IERC20.sol import from ArtArmyHoldersStake.sol and Ownable.sol import from ArtArmySeller.sol.

(2) Make variable constant:- **ArtArmySeller.sol**

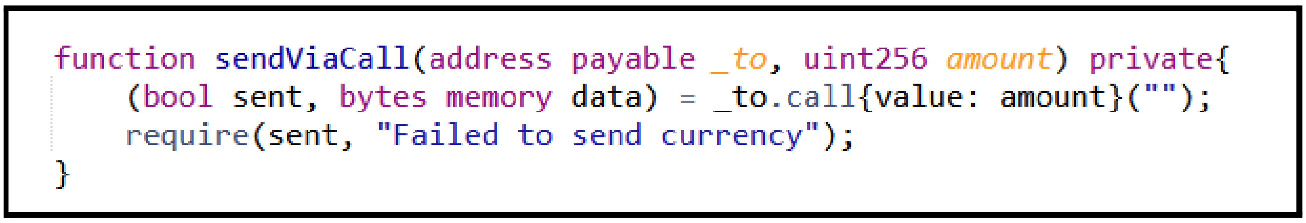


```
address _burnWallet = 0x0000000000000000000000000000000000000000;
```

This variable will not be changed anytime.

Resolution: We suggest declaring this variable as constant to save some gas.

(4) Unused local variable:- **ArtArmySeller.sol**



```
function sendViaCall(address payable _to, uint256 amount) private{
    (bool sent, bytes memory data) = _to.call{value: amount}("");
    require(sent, "Failed to send currency");
}
```

Data variable has been set but not used in the sendViaCall function.

Resolution: We suggest removing unused variables.

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:

- setAuctionContract: The ArtArmyArtwork owner can set the auction contract address.
- setSellerContract: The ArtArmyArtwork owner can set the fixed sales contract address.
- setMaximumRoyaltyInBasisPoints: The ArtArmyArtwork owner can set the maximum Royalty in Basis Points that an artist can set in basis points.
- setArtArmyFee: The ArtArmySeller owner can set the Art Army Fees.
- setHoldersFee: The ArtArmySeller owner can set the Holders Fees.
- setCurrency: The ArtArmySeller owner can set the Token BEP20 used for the transactions.
- setMaximumCuratorFeeInBasisPoints: The ArtArmySeller owner can set the maximum curator's fee in basis points.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but they are not critical ones. 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 **"Secured".**

Our Methodology

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

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

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

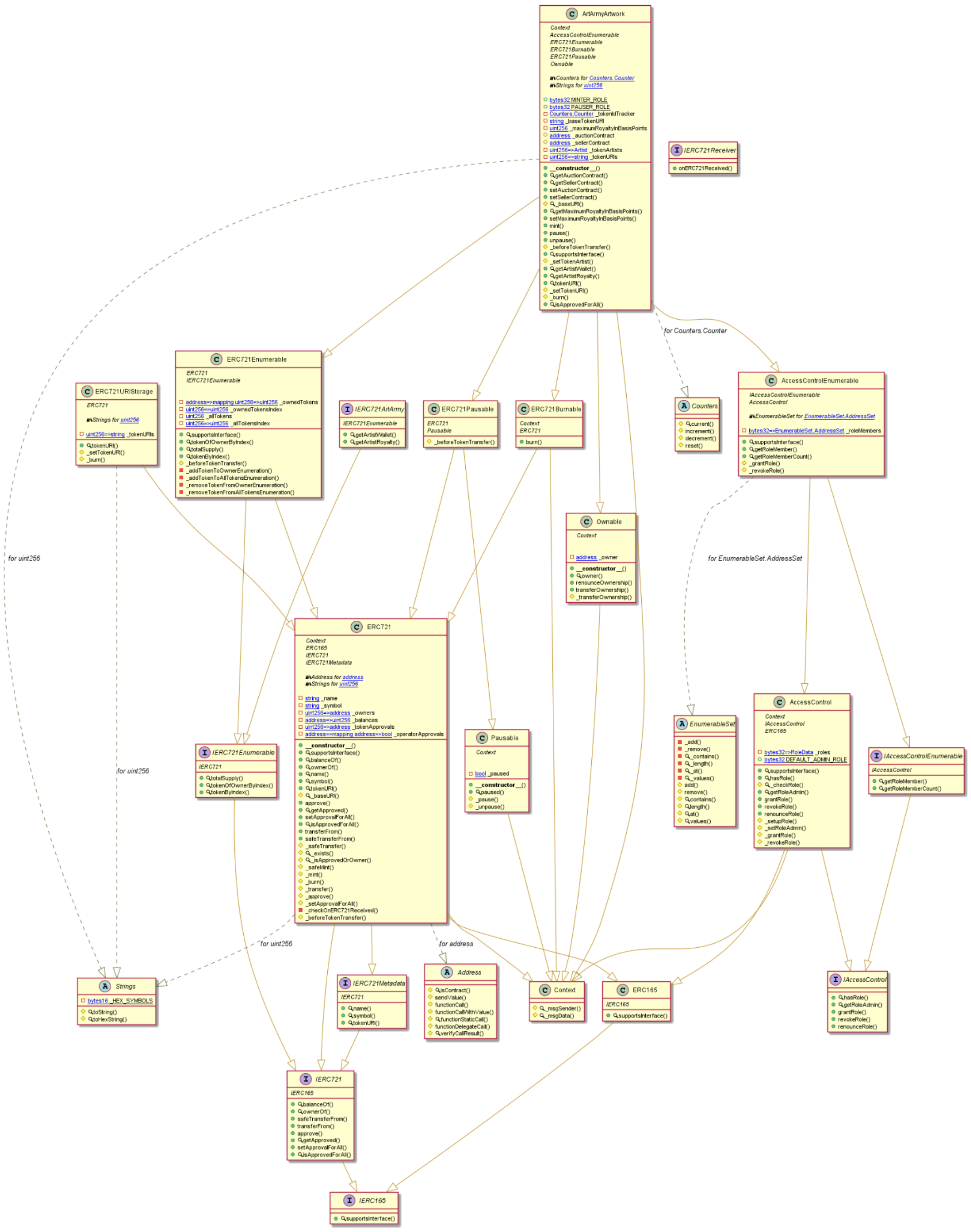
Technical Disclaimer

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

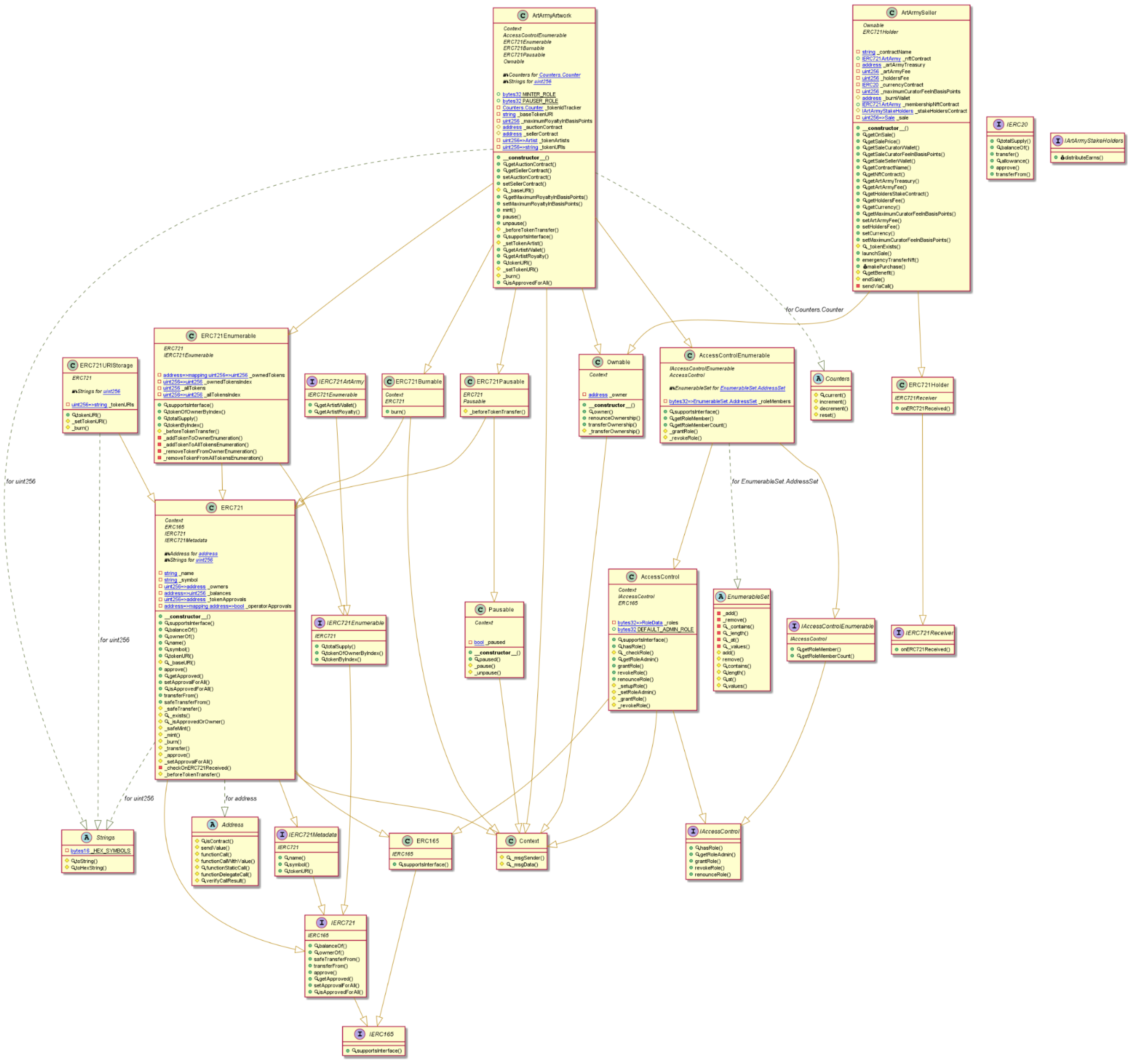
Appendix

Code Flow Diagram - ArtArmy Protocol

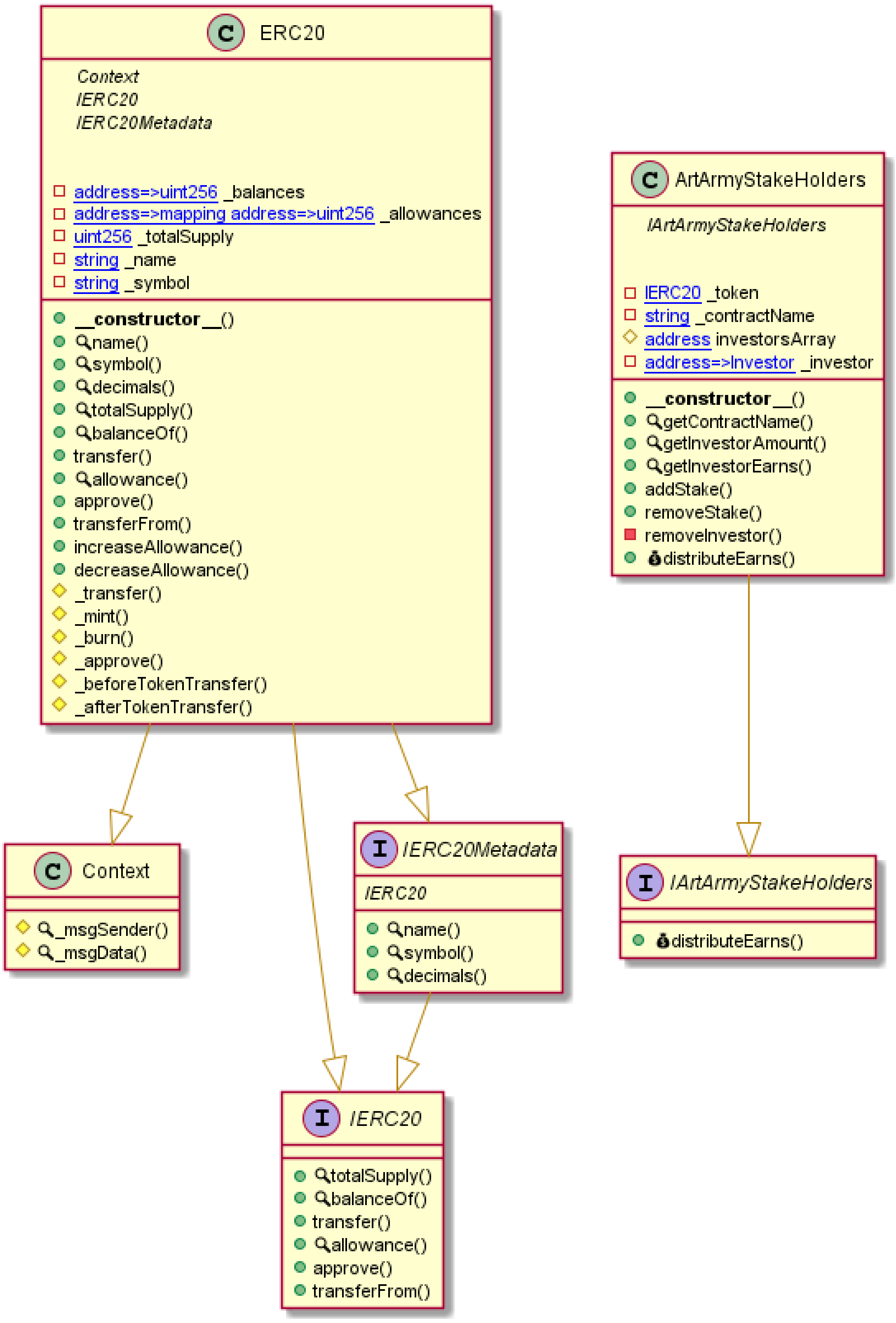
ArtArmyArtwork Diagram



ArtArmySeller Diagram

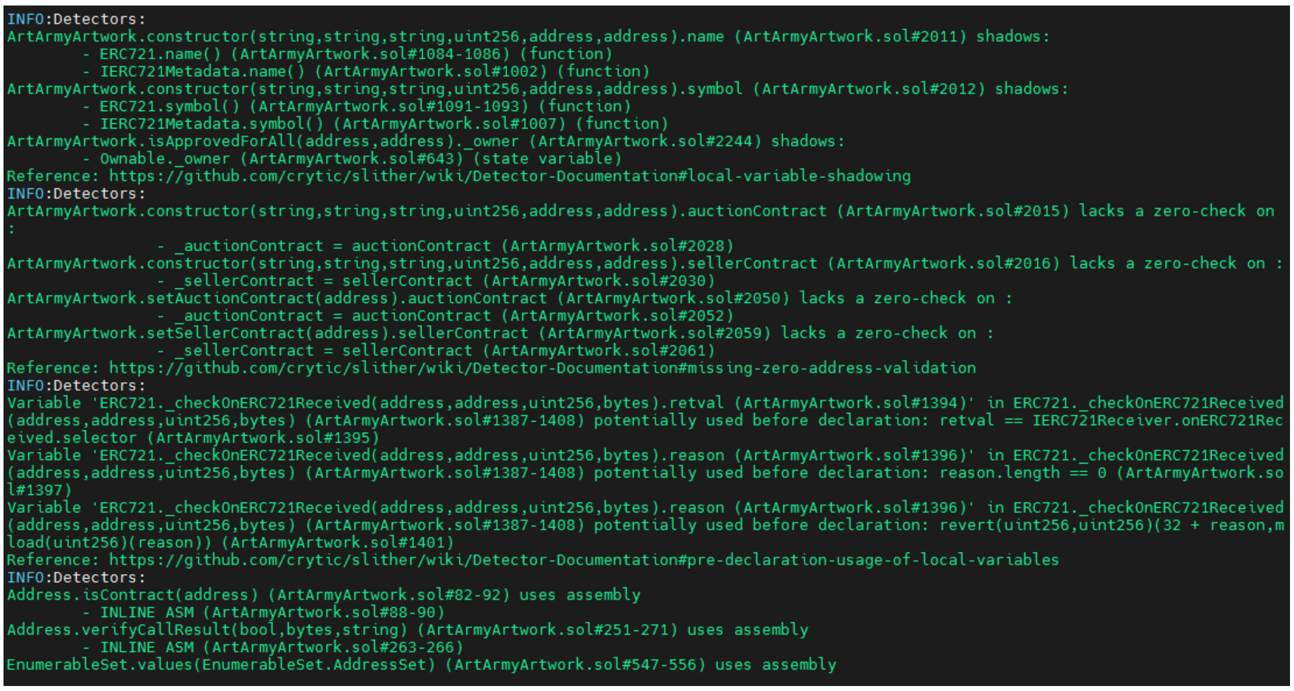


ArtArmyHoldersStake Diagram

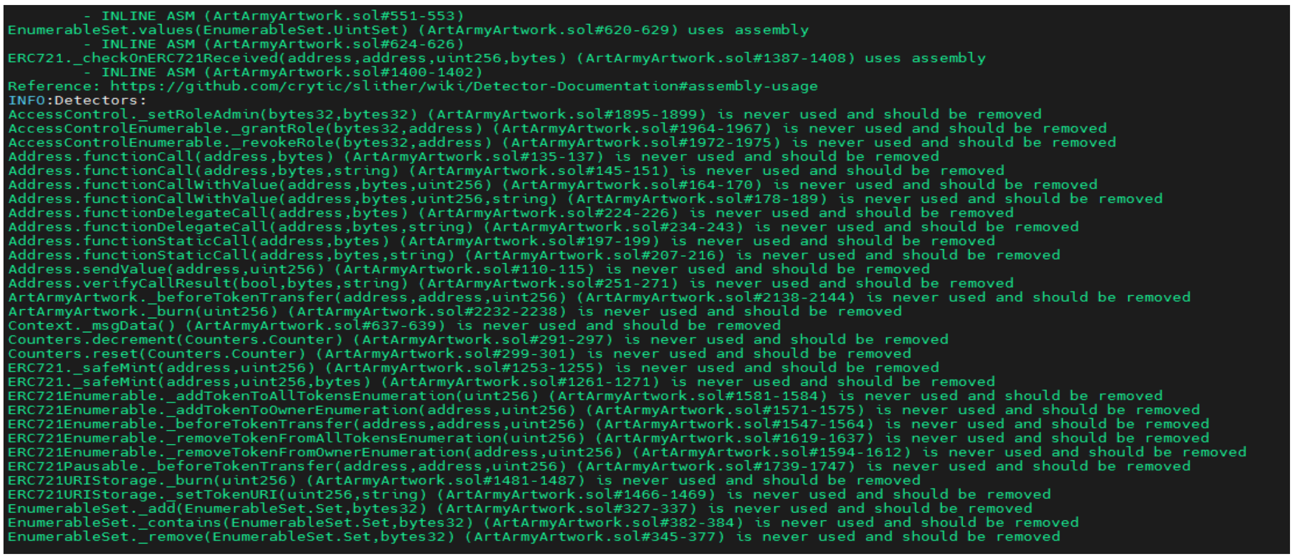


Slither Results Log

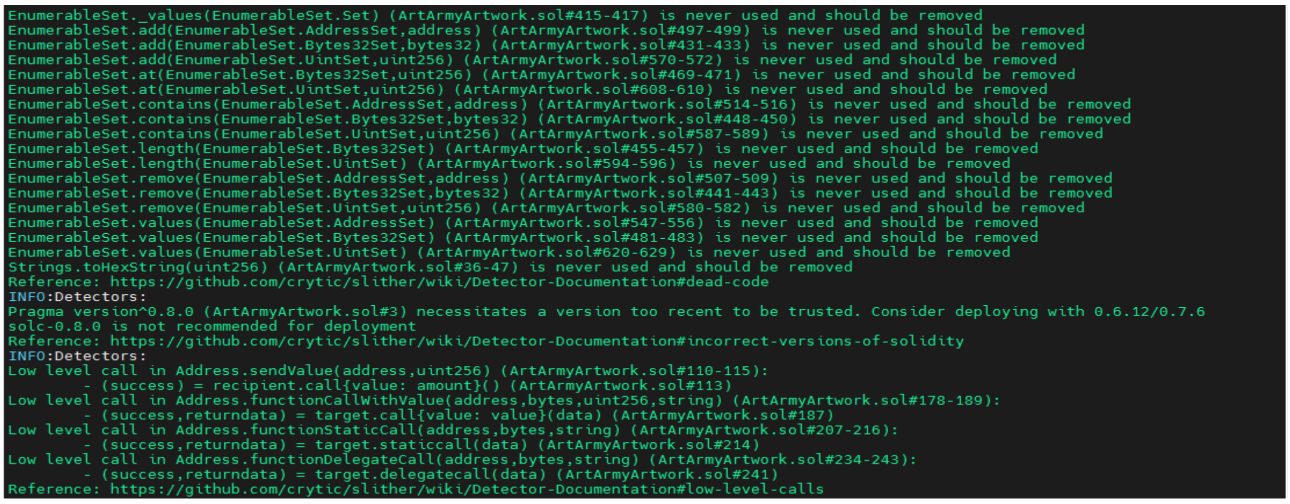
Slither log >> ArtArmyArtwork.sol



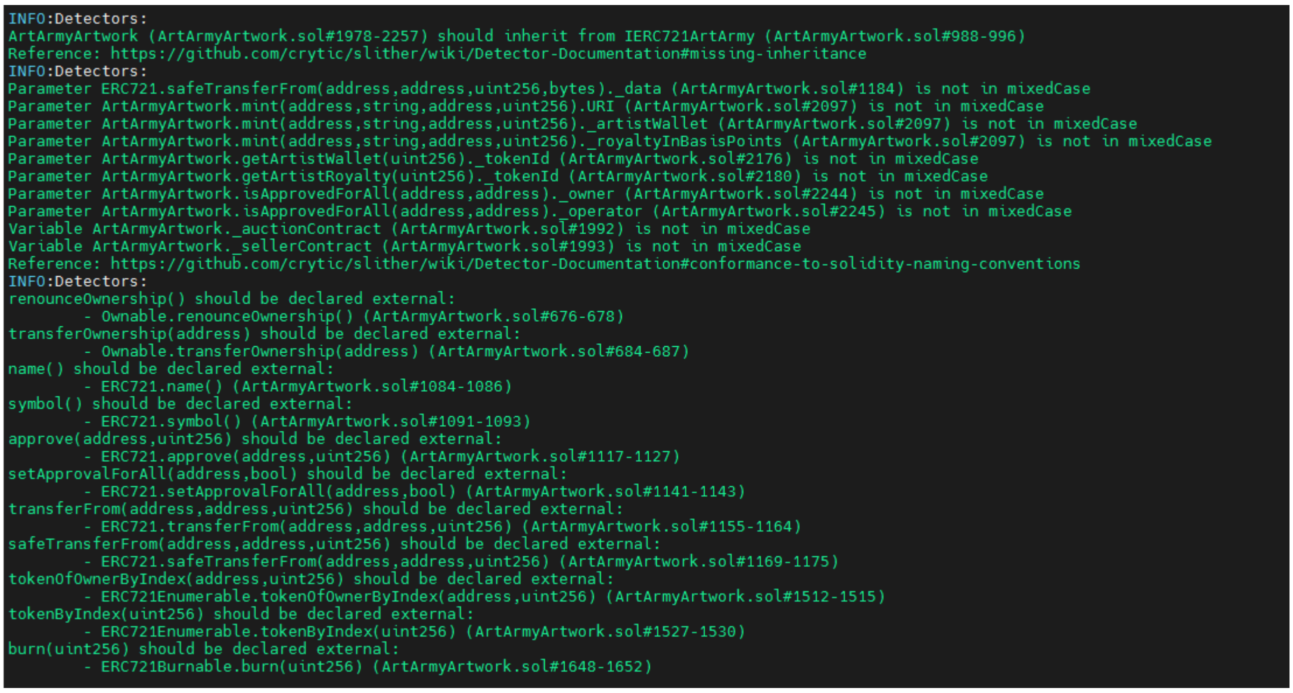
```
INFO:Detectors:
ArtArmyArtwork.constructor(string,string,string,uint256,address,address).name (ArtArmyArtwork.sol#2011) shadows:
        - ERC721.name() (ArtArmyArtwork.sol#1084-1086) (function)
        - IERC721Metadata.name() (ArtArmyArtwork.sol#1002) (function)
ArtArmyArtwork.constructor(string,string,string,uint256,address,address).symbol (ArtArmyArtwork.sol#2012) shadows:
        - ERC721.symbol() (ArtArmyArtwork.sol#1091-1093) (function)
        - IERC721Metadata.symbol() (ArtArmyArtwork.sol#1007) (function)
ArtArmyArtwork.isApprovedForAll(address,address)._owner (ArtArmyArtwork.sol#2244) shadows:
        - Ownable._owner (ArtArmyArtwork.sol#643) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
ArtArmyArtwork.constructor(string,string,string,uint256,address,address).auctionContract (ArtArmyArtwork.sol#2015) lacks a zero-check on
:
                - _auctionContract = auctionContract (ArtArmyArtwork.sol#2028)
ArtArmyArtwork.constructor(string,string,string,uint256,address,address).sellerContract (ArtArmyArtwork.sol#2016) lacks a zero-check on :
                - _sellerContract = sellerContract (ArtArmyArtwork.sol#2030)
ArtArmyArtwork.setAuctionContract(address).auctionContract (ArtArmyArtwork.sol#2050) lacks a zero-check on :
                - _auctionContract = auctionContract (ArtArmyArtwork.sol#2052)
ArtArmyArtwork.setSellerContract(address).sellerContract (ArtArmyArtwork.sol#2059) lacks a zero-check on :
                - _sellerContract = sellerContract (ArtArmyArtwork.sol#2061)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (ArtArmyArtwork.sol#1394)' in ERC721._checkOnERC721Received
(address,address,uint256,bytes) (ArtArmyArtwork.sol#1387-1408) potentially used before declaration: retval == IERC721Receiver.onERC721Rec
eived.selector (ArtArmyArtwork.sol#1395)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (ArtArmyArtwork.sol#1396)' in ERC721._checkOnERC721Received
(address,address,uint256,bytes) (ArtArmyArtwork.sol#1387-1408) potentially used before declaration: reason.length == 0 (ArtArmyArtwork.so
l#1397)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (ArtArmyArtwork.sol#1396)' in ERC721._checkOnERC721Received
(address,address,uint256,bytes) (ArtArmyArtwork.sol#1387-1408) potentially used before declaration: revert(uint256,uint256)(32 + reason,m
load(uint256)(reason)) (ArtArmyArtwork.sol#1401)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Address.isContract(address) (ArtArmyArtwork.sol#82-92) uses assembly
        - INLINE ASM (ArtArmyArtwork.sol#88-90)
Address.verifyCallResult(bool,bytes,string) (ArtArmyArtwork.sol#251-271) uses assembly
        - INLINE ASM (ArtArmyArtwork.sol#263-266)
EnumerableSet.values(EnumerableSet.AddressSet) (ArtArmyArtwork.sol#547-556) uses assembly
```



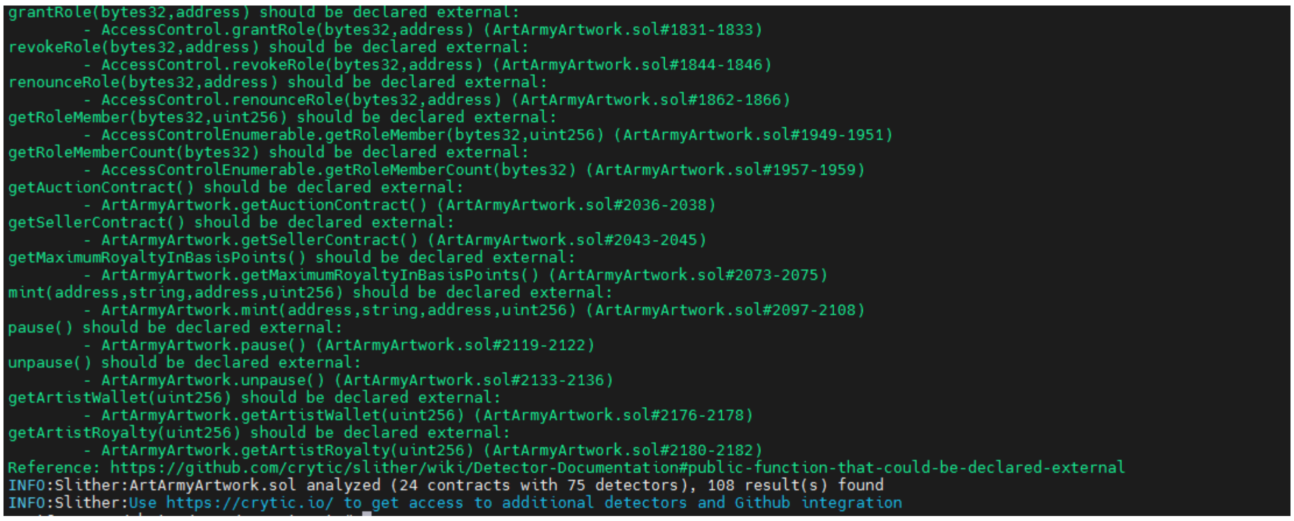
```
        - INLINE ASM (ArtArmyArtwork.sol#551-553)
EnumerableSet.values(EnumerableSet.UintSet) (ArtArmyArtwork.sol#620-629) uses assembly
        - INLINE ASM (ArtArmyArtwork.sol#624-626)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (ArtArmyArtwork.sol#1387-1408) uses assembly
        - INLINE ASM (ArtArmyArtwork.sol#1400-1402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (ArtArmyArtwork.sol#1895-1899) is never used and should be removed
AccessControlEnumerable._grantRole(bytes32,address) (ArtArmyArtwork.sol#1964-1967) is never used and should be removed
AccessControlEnumerable._revokeRole(bytes32,address) (ArtArmyArtwork.sol#1972-1975) is never used and should be removed
Address.functionCall(address,bytes) (ArtArmyArtwork.sol#135-137) is never used and should be removed
Address.functionCall(address,bytes,string) (ArtArmyArtwork.sol#145-151) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ArtArmyArtwork.sol#164-170) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ArtArmyArtwork.sol#178-189) is never used and should be removed
Address.functionDelegateCall(address,bytes) (ArtArmyArtwork.sol#224-226) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (ArtArmyArtwork.sol#234-243) is never used and should be removed
Address.functionStaticCall(address,bytes) (ArtArmyArtwork.sol#197-199) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (ArtArmyArtwork.sol#207-216) is never used and should be removed
Address.sendValue(address,uint256) (ArtArmyArtwork.sol#110-115) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (ArtArmyArtwork.sol#251-271) is never used and should be removed
ArtArmyArtwork._beforeTokenTransfer(address,address,uint256) (ArtArmyArtwork.sol#2138-2144) is never used and should be removed
ArtArmyArtwork._burn(uint256) (ArtArmyArtwork.sol#2232-2238) is never used and should be removed
Context._msgData() (ArtArmyArtwork.sol#637-639) is never used and should be removed
Counters.decrement(Counters.Counter) (ArtArmyArtwork.sol#291-297) is never used and should be removed
Counters.reset(Counters.Counter) (ArtArmyArtwork.sol#299-301) is never used and should be removed
ERC721._safeMint(address,uint256) (ArtArmyArtwork.sol#1253-1255) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (ArtArmyArtwork.sol#1261-1271) is never used and should be removed
ERC721Enumerable._addTokenToAllTokensEnumeration(uint256) (ArtArmyArtwork.sol#1581-1584) is never used and should be removed
ERC721Enumerable._addTokenToOwnerEnumeration(address,uint256) (ArtArmyArtwork.sol#1571-1575) is never used and should be removed
ERC721Enumerable._beforeTokenTransfer(address,address,uint256) (ArtArmyArtwork.sol#1547-1564) is never used and should be removed
ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (ArtArmyArtwork.sol#1619-1637) is never used and should be removed
ERC721Enumerable._removeTokenFromOwnerEnumeration(address,uint256) (ArtArmyArtwork.sol#1594-1612) is never used and should be removed
ERC721Pausable._beforeTokenTransfer(address,address,uint256) (ArtArmyArtwork.sol#1739-1747) is never used and should be removed
ERC721URIStorage._burn(uint256) (ArtArmyArtwork.sol#1481-1487) is never used and should be removed
ERC721URIStorage._setTokenURI(uint256,string) (ArtArmyArtwork.sol#1466-1469) is never used and should be removed
EnumerableSet._add(EnumerableSet.Set,bytes32) (ArtArmyArtwork.sol#327-337) is never used and should be removed
EnumerableSet._contains(EnumerableSet.Set,bytes32) (ArtArmyArtwork.sol#382-384) is never used and should be removed
EnumerableSet._remove(EnumerableSet.Set,bytes32) (ArtArmyArtwork.sol#345-377) is never used and should be removed
```



```
EnumerableSet._values(EnumerableSet.Set) (ArtArmyArtwork.sol#415-417) is never used and should be removed
EnumerableSet.add(EnumerableSet.AddressSet,address) (ArtArmyArtwork.sol#497-499) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (ArtArmyArtwork.sol#431-433) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (ArtArmyArtwork.sol#570-572) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (ArtArmyArtwork.sol#469-471) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (ArtArmyArtwork.sol#608-610) is never used and should be removed
EnumerableSet.contains(EnumerableSet.AddressSet,address) (ArtArmyArtwork.sol#514-516) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (ArtArmyArtwork.sol#448-450) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (ArtArmyArtwork.sol#587-589) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (ArtArmyArtwork.sol#455-457) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (ArtArmyArtwork.sol#594-596) is never used and should be removed
EnumerableSet.remove(EnumerableSet.AddressSet,address) (ArtArmyArtwork.sol#507-509) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (ArtArmyArtwork.sol#441-443) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (ArtArmyArtwork.sol#580-582) is never used and should be removed
EnumerableSet.values(EnumerableSet.AddressSet) (ArtArmyArtwork.sol#547-556) is never used and should be removed
EnumerableSet.values(EnumerableSet.Bytes32Set) (ArtArmyArtwork.sol#481-483) is never used and should be removed
EnumerableSet.values(EnumerableSet.UintSet) (ArtArmyArtwork.sol#620-629) is never used and should be removed
Strings.toHexString(uint256) (ArtArmyArtwork.sol#36-47) 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 (ArtArmyArtwork.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 Address.sendValue(address,uint256) (ArtArmyArtwork.sol#110-115):
        - (success) = recipient.call{value: amount}() (ArtArmyArtwork.sol#113)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ArtArmyArtwork.sol#178-189):
        - (success,returndata) = target.call{value: value}(data) (ArtArmyArtwork.sol#187)
Low level call in Address.functionStaticCall(address,bytes,string) (ArtArmyArtwork.sol#207-216):
        - (success,returndata) = target.staticcall(data) (ArtArmyArtwork.sol#214)
Low level call in Address.functionDelegateCall(address,bytes,string) (ArtArmyArtwork.sol#234-243):
        - (success,returndata) = target.delegatecall(data) (ArtArmyArtwork.sol#241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

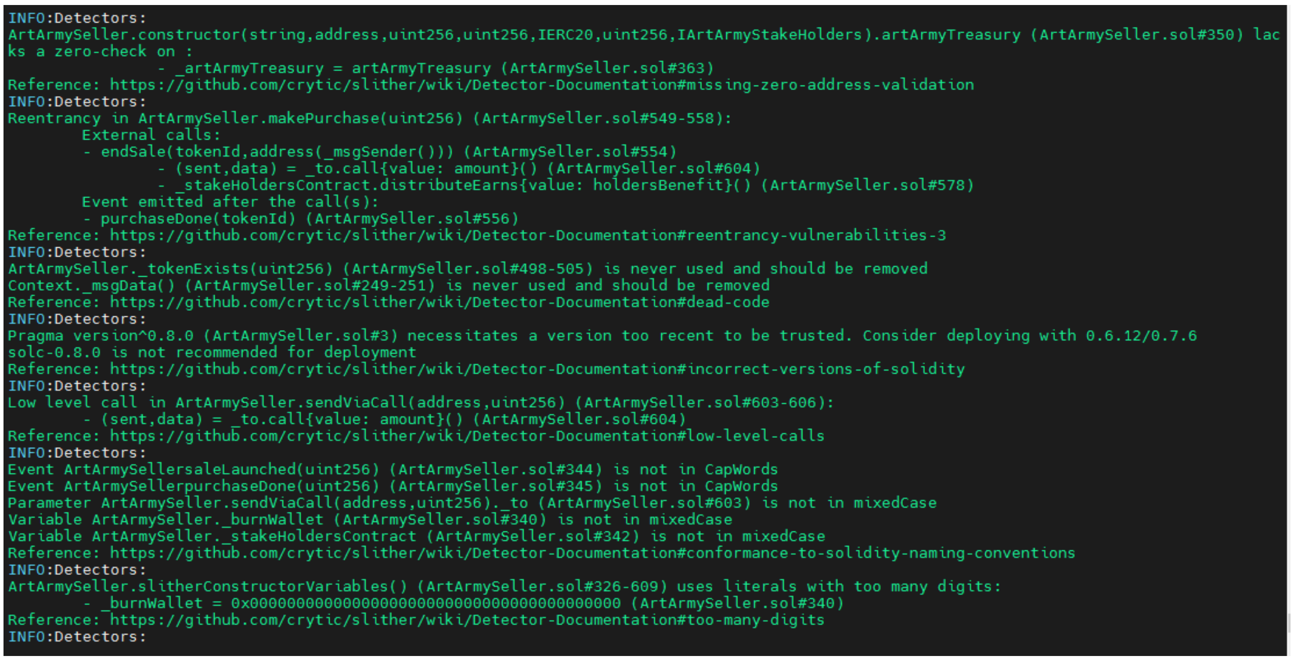


```
INFO:Detectors:
ArtArmyArtwork (ArtArmyArtwork.sol#1978-2257) should inherit from IERC721ArtArmy (ArtArmyArtwork.sol#988-996)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (ArtArmyArtwork.sol#1184) is not in mixedCase
Parameter ArtArmyArtwork.mint(address,string,address,uint256).URI (ArtArmyArtwork.sol#2097) is not in mixedCase
Parameter ArtArmyArtwork.mint(address,string,address,uint256)._artistWallet (ArtArmyArtwork.sol#2097) is not in mixedCase
Parameter ArtArmyArtwork.mint(address,string,address,uint256)._royaltyInBasisPoints (ArtArmyArtwork.sol#2097) is not in mixedCase
Parameter ArtArmyArtwork.getArtistWallet(uint256)._tokenId (ArtArmyArtwork.sol#2176) is not in mixedCase
Parameter ArtArmyArtwork.getArtistRoyalty(uint256)._tokenId (ArtArmyArtwork.sol#2180) is not in mixedCase
Parameter ArtArmyArtwork.isApprovedForAll(address,address)._owner (ArtArmyArtwork.sol#2244) is not in mixedCase
Parameter ArtArmyArtwork.isApprovedForAll(address,address)._operator (ArtArmyArtwork.sol#2245) is not in mixedCase
Variable ArtArmyArtwork._auctionContract (ArtArmyArtwork.sol#1992) is not in mixedCase
Variable ArtArmyArtwork._sellerContract (ArtArmyArtwork.sol#1993) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ArtArmyArtwork.sol#676-678)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ArtArmyArtwork.sol#684-687)
name() should be declared external:
        - ERC721.name() (ArtArmyArtwork.sol#1084-1086)
symbol() should be declared external:
        - ERC721.symbol() (ArtArmyArtwork.sol#1091-1093)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (ArtArmyArtwork.sol#1117-1127)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (ArtArmyArtwork.sol#1141-1143)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (ArtArmyArtwork.sol#1155-1164)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (ArtArmyArtwork.sol#1169-1175)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (ArtArmyArtwork.sol#1512-1515)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (ArtArmyArtwork.sol#1527-1530)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (ArtArmyArtwork.sol#1648-1652)
```

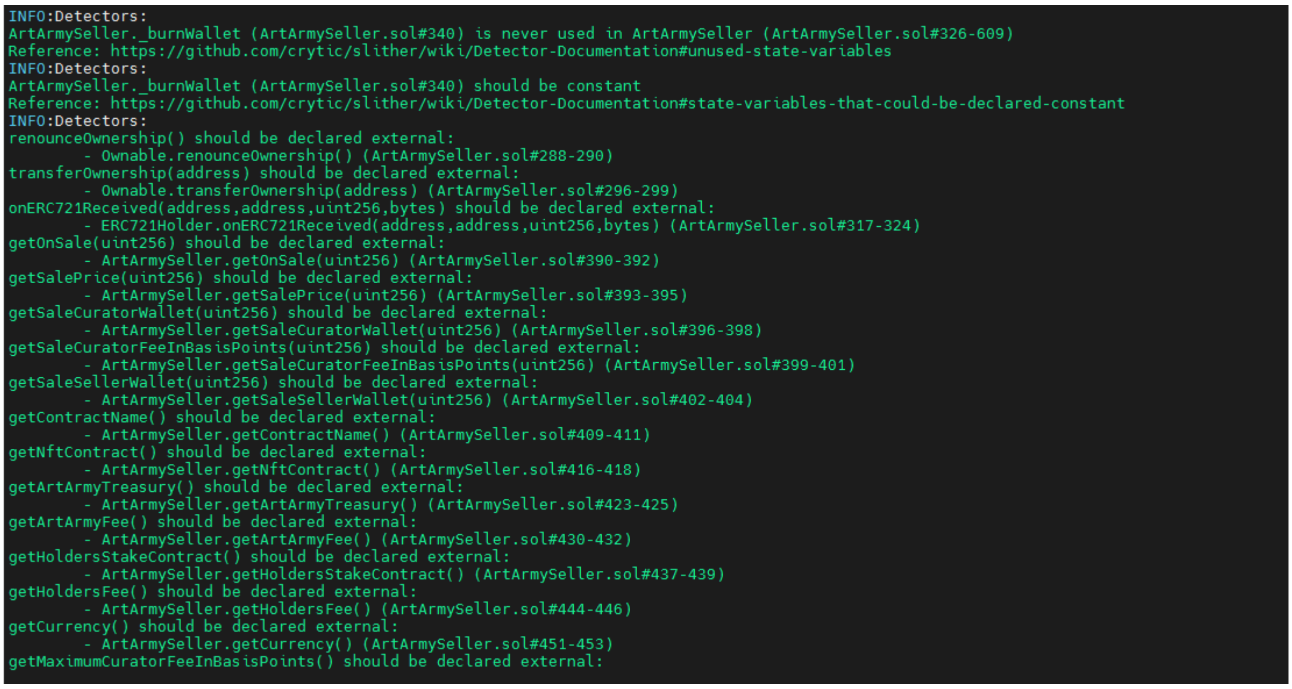


```
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (ArtArmyArtwork.sol#1831-1833)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (ArtArmyArtwork.sol#1844-1846)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (ArtArmyArtwork.sol#1862-1866)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (ArtArmyArtwork.sol#1949-1951)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerable.getRoleMemberCount(bytes32) (ArtArmyArtwork.sol#1957-1959)
getAuctionContract() should be declared external:
        - ArtArmyArtwork.getAuctionContract() (ArtArmyArtwork.sol#2036-2038)
getSellerContract() should be declared external:
        - ArtArmyArtwork.getSellerContract() (ArtArmyArtwork.sol#2043-2045)
getMaximumRoyaltyInBasisPoints() should be declared external:
        - ArtArmyArtwork.getMaximumRoyaltyInBasisPoints() (ArtArmyArtwork.sol#2073-2075)
mint(address,string,address,uint256) should be declared external:
        - ArtArmyArtwork.mint(address,string,address,uint256) (ArtArmyArtwork.sol#2097-2108)
pause() should be declared external:
        - ArtArmyArtwork.pause() (ArtArmyArtwork.sol#2119-2122)
unpause() should be declared external:
        - ArtArmyArtwork.unpause() (ArtArmyArtwork.sol#2133-2136)
getArtistWallet(uint256) should be declared external:
        - ArtArmyArtwork.getArtistWallet(uint256) (ArtArmyArtwork.sol#2176-2178)
getArtistRoyalty(uint256) should be declared external:
        - ArtArmyArtwork.getArtistRoyalty(uint256) (ArtArmyArtwork.sol#2180-2182)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ArtArmyArtwork.sol analyzed (24 contracts with 75 detectors), 108 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

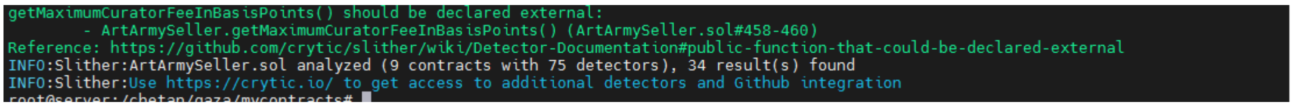
Slither log >> ArtArmySeller.sol



```
INFO:Detectors:
ArtArmySeller.constructor(string,address,uint256,uint256,IERC20,uint256,IArtArmyStakeHolders).artArmyTreasury (ArtArmySeller.sol#350) lac
ks a zero-check on :
                - _artArmyTreasury = artArmyTreasury (ArtArmySeller.sol#363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ArtArmySeller.makePurchase(uint256) (ArtArmySeller.sol#549-558):
        External calls:
        - endSale(tokenId,address(_msgSender())) (ArtArmySeller.sol#554)
                - (sent,data) = _to.call{value: amount}() (ArtArmySeller.sol#604)
                - _stakeHoldersContract.distributeEarns{value: holdersBenefit}() (ArtArmySeller.sol#578)
        Event emitted after the call(s):
        - purchaseDone(tokenId) (ArtArmySeller.sol#556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ArtArmySeller._tokenExists(uint256) (ArtArmySeller.sol#498-505) is never used and should be removed
Context._msgData() (ArtArmySeller.sol#249-251) 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 (ArtArmySeller.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 ArtArmySeller.sendViaCall(address,uint256) (ArtArmySeller.sol#603-606):
        - (sent,data) = _to.call{value: amount}() (ArtArmySeller.sol#604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Event ArtArmySellersaleLaunched(uint256) (ArtArmySeller.sol#344) is not in CapWords
Event ArtArmySellerpurchaseDone(uint256) (ArtArmySeller.sol#345) is not in CapWords
Parameter ArtArmySeller.sendViaCall(address,uint256)._to (ArtArmySeller.sol#603) is not in mixedCase
Variable ArtArmySeller._burnWallet (ArtArmySeller.sol#340) is not in mixedCase
Variable ArtArmySeller._stakeHoldersContract (ArtArmySeller.sol#342) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ArtArmySeller.slitherConstructorVariables() (ArtArmySeller.sol#326-609) uses literals with too many digits:
        - _burnWallet = 0x0000000000000000000000000000000000000000 (ArtArmySeller.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
```

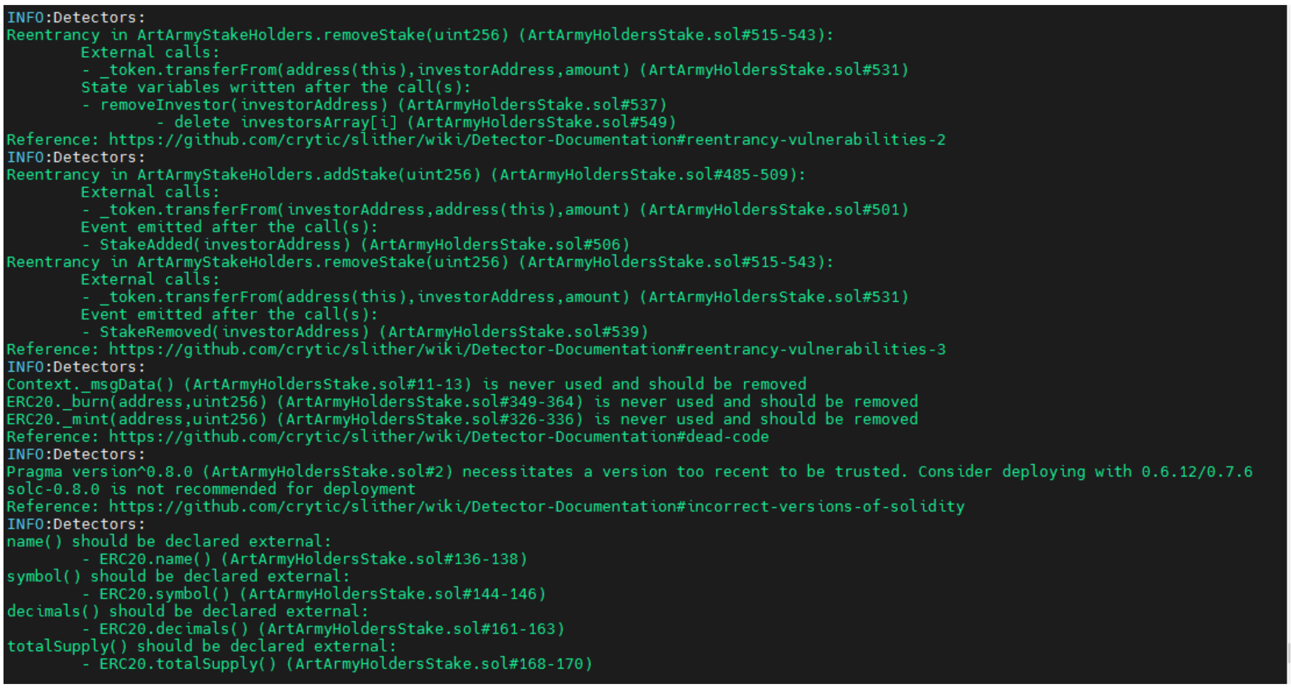


```
INFO:Detectors:
ArtArmySeller._burnWallet (ArtArmySeller.sol#340) is never used in ArtArmySeller (ArtArmySeller.sol#326-609)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
ArtArmySeller._burnWallet (ArtArmySeller.sol#340) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ArtArmySeller.sol#288-290)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ArtArmySeller.sol#296-299)
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (ArtArmySeller.sol#317-324)
getOnSale(uint256) should be declared external:
        - ArtArmySeller.getOnSale(uint256) (ArtArmySeller.sol#390-392)
getSalePrice(uint256) should be declared external:
        - ArtArmySeller.getSalePrice(uint256) (ArtArmySeller.sol#393-395)
getSaleCuratorWallet(uint256) should be declared external:
        - ArtArmySeller.getSaleCuratorWallet(uint256) (ArtArmySeller.sol#396-398)
getSaleCuratorFeeInBasisPoints(uint256) should be declared external:
        - ArtArmySeller.getSaleCuratorFeeInBasisPoints(uint256) (ArtArmySeller.sol#399-401)
getSaleSellerWallet(uint256) should be declared external:
        - ArtArmySeller.getSaleSellerWallet(uint256) (ArtArmySeller.sol#402-404)
getContractName() should be declared external:
        - ArtArmySeller.getContractName() (ArtArmySeller.sol#409-411)
getNftContract() should be declared external:
        - ArtArmySeller.getNftContract() (ArtArmySeller.sol#416-418)
getArtArmyTreasury() should be declared external:
        - ArtArmySeller.getArtArmyTreasury() (ArtArmySeller.sol#423-425)
getArtArmyFee() should be declared external:
        - ArtArmySeller.getArtArmyFee() (ArtArmySeller.sol#430-432)
getHoldersStakeContract() should be declared external:
        - ArtArmySeller.getHoldersStakeContract() (ArtArmySeller.sol#437-439)
getHoldersFee() should be declared external:
        - ArtArmySeller.getHoldersFee() (ArtArmySeller.sol#444-446)
getCurrency() should be declared external:
        - ArtArmySeller.getCurrency() (ArtArmySeller.sol#451-453)
getMaximumCuratorFeeInBasisPoints() should be declared external:
```

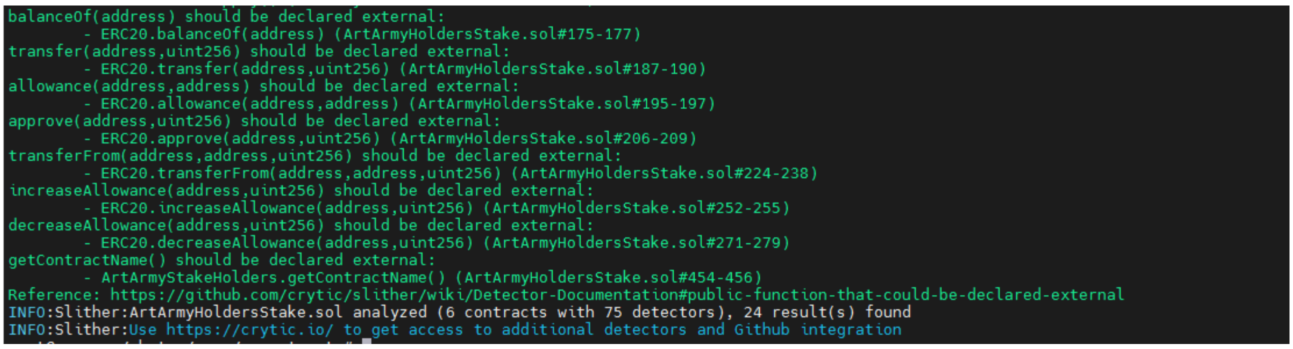


```
getMaximumCuratorFeeInBasisPoints() should be declared external:
        - ArtArmySeller.getMaximumCuratorFeeInBasisPoints() (ArtArmySeller.sol#458-460)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ArtArmySeller.sol analyzed (9 contracts with 75 detectors), 34 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

Slither log >> ArtArmyHoldersStake.sol



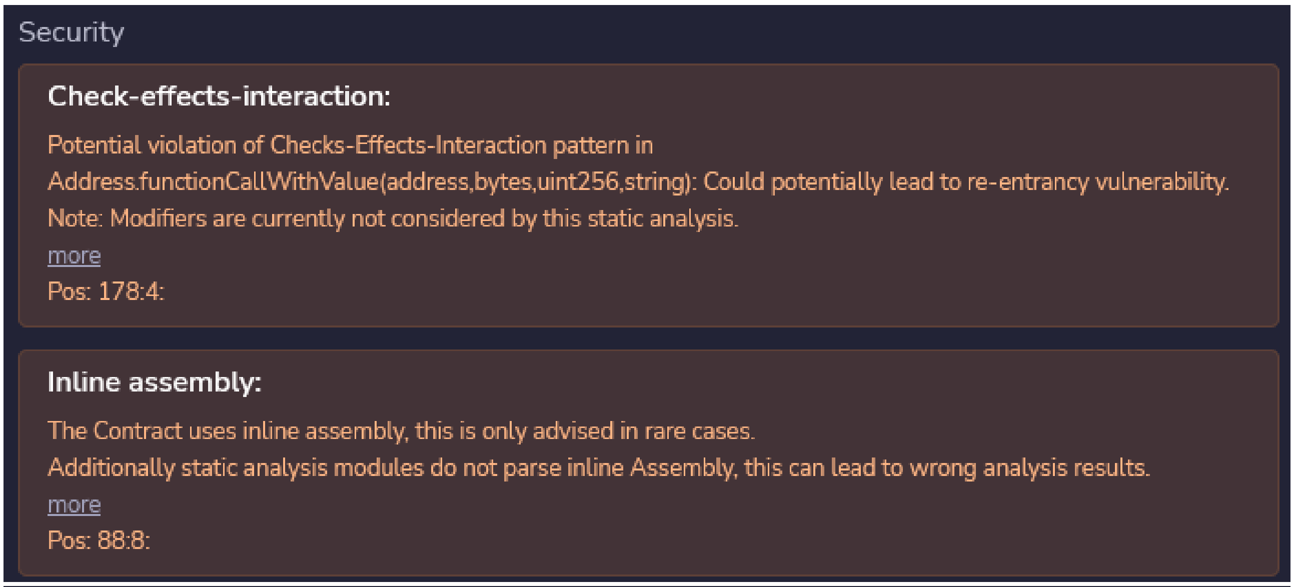
```
INFO:Detectors:
Reentrancy in ArtArmyStakeHolders.removeStake(uint256) (ArtArmyHoldersStake.sol#515-543):
        External calls:
        - _token.transferFrom(address(this),investorAddress,amount) (ArtArmyHoldersStake.sol#531)
        State variables written after the call(s):
        - removeInvestor(investorAddress) (ArtArmyHoldersStake.sol#537)
                - delete investorsArray[i] (ArtArmyHoldersStake.sol#549)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ArtArmyStakeHolders.addStake(uint256) (ArtArmyHoldersStake.sol#485-509):
        External calls:
        - _token.transferFrom(investorAddress,address(this),amount) (ArtArmyHoldersStake.sol#501)
        Event emitted after the call(s):
        - StakeAdded(investorAddress) (ArtArmyHoldersStake.sol#506)
Reentrancy in ArtArmyStakeHolders.removeStake(uint256) (ArtArmyHoldersStake.sol#515-543):
        External calls:
        - _token.transferFrom(address(this),investorAddress,amount) (ArtArmyHoldersStake.sol#531)
        Event emitted after the call(s):
        - StakeRemoved(investorAddress) (ArtArmyHoldersStake.sol#539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (ArtArmyHoldersStake.sol#11-13) is never used and should be removed
ERC20._burn(address,uint256) (ArtArmyHoldersStake.sol#349-364) is never used and should be removed
ERC20._mint(address,uint256) (ArtArmyHoldersStake.sol#326-336) 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 (ArtArmyHoldersStake.sol#2) 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:
name() should be declared external:
        - ERC20.name() (ArtArmyHoldersStake.sol#136-138)
symbol() should be declared external:
        - ERC20.symbol() (ArtArmyHoldersStake.sol#144-146)
decimals() should be declared external:
        - ERC20.decimals() (ArtArmyHoldersStake.sol#161-163)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ArtArmyHoldersStake.sol#168-170)
```



```
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ArtArmyHoldersStake.sol#175-177)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ArtArmyHoldersStake.sol#187-190)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ArtArmyHoldersStake.sol#195-197)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ArtArmyHoldersStake.sol#206-209)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ArtArmyHoldersStake.sol#224-238)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ArtArmyHoldersStake.sol#252-255)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ArtArmyHoldersStake.sol#271-279)
getContractName() should be declared external:
        - ArtArmyStakeHolders.getContractName() (ArtArmyHoldersStake.sol#454-456)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ArtArmyHoldersStake.sol analyzed (6 contracts with 75 detectors), 24 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ArtArmyArtwork.sol



Security

Check-effects-interaction:

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

more

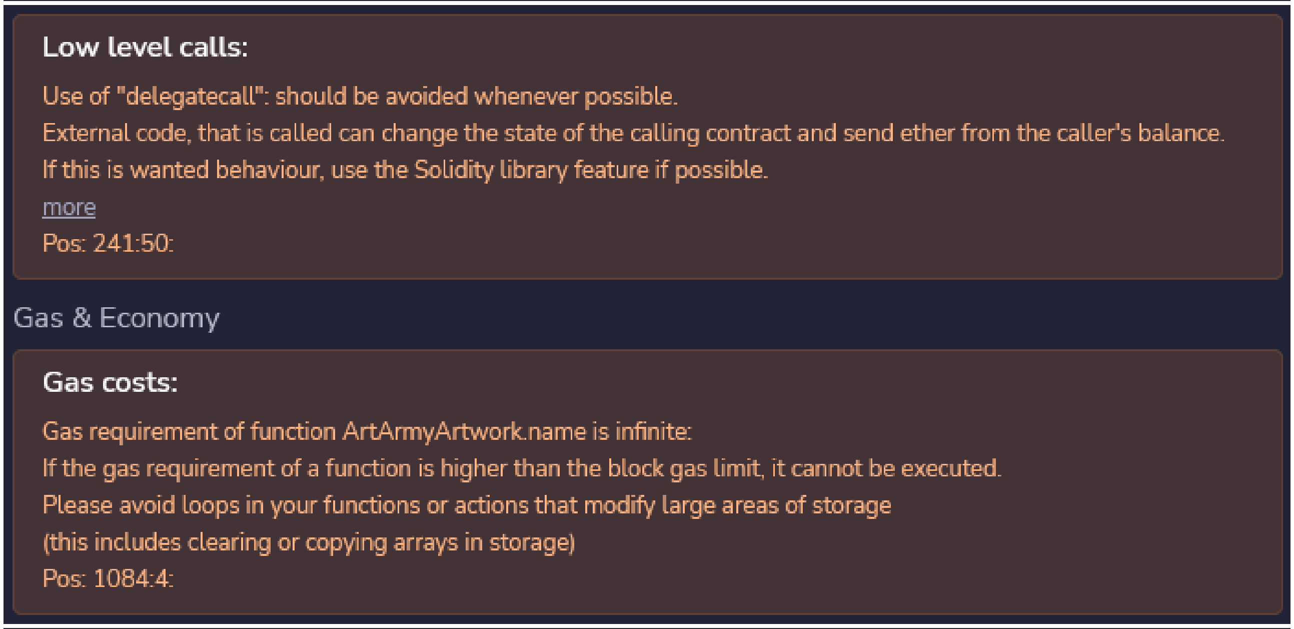
Pos: 178:4:

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: 88:8:



Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

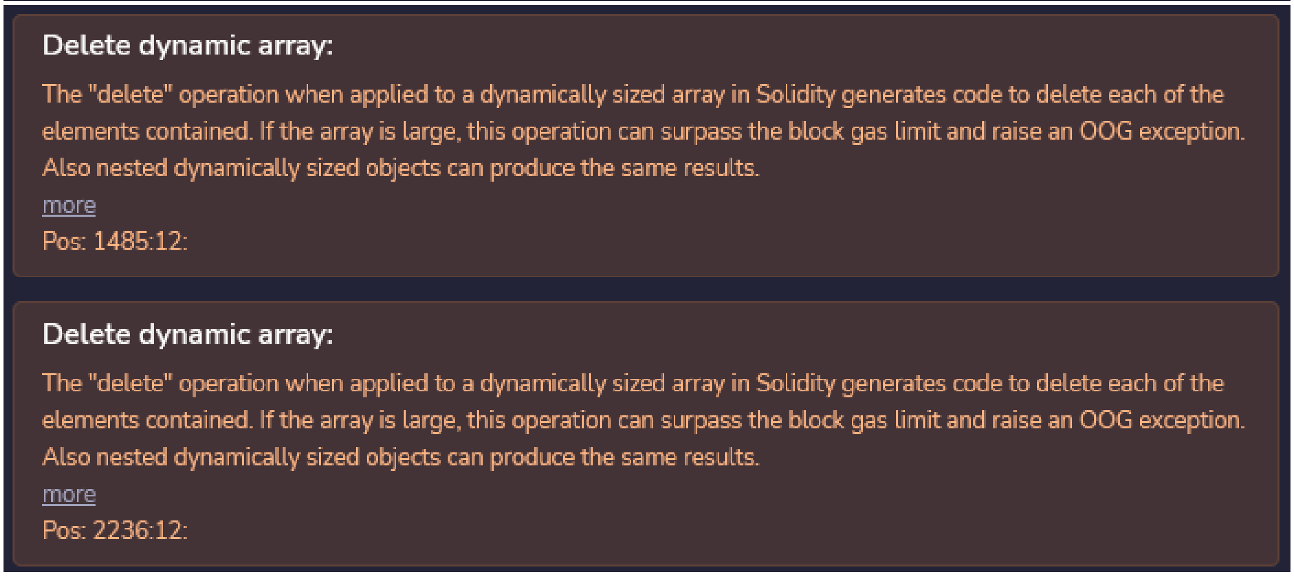
Pos: 241:50:

Gas & Economy

Gas costs:

Gas requirement of function ArtArmyArtwork.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: 1084:4:



Delete dynamic array:

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

more

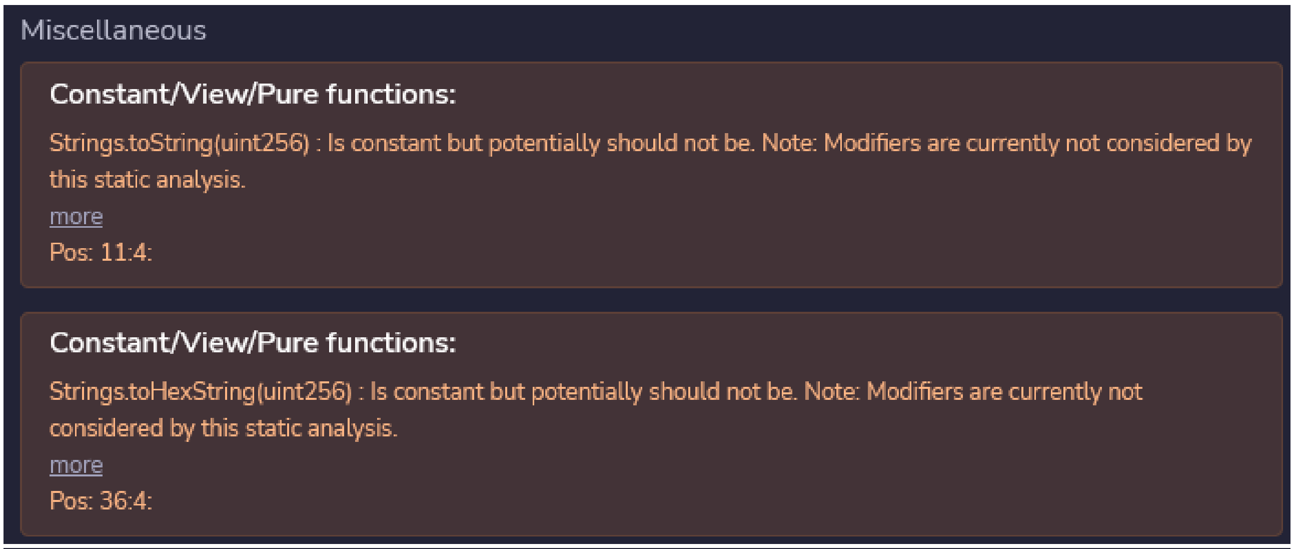
Pos: 1485:12:

Delete dynamic array:

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

more

Pos: 2236:12:

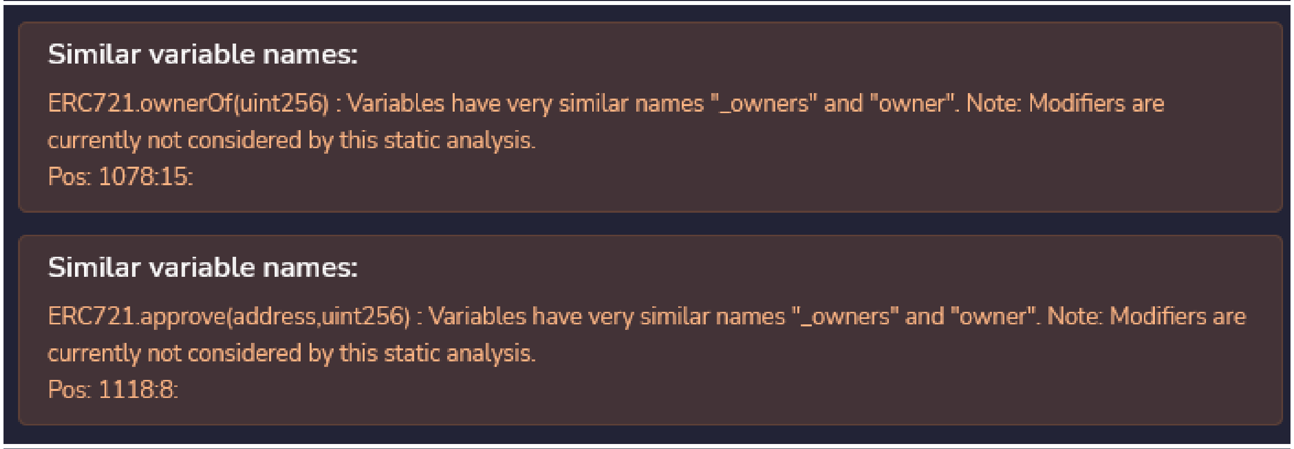

Miscellaneous

Constant/View/Pure functions:

Strings.toString(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 11:4:

Constant/View/Pure functions:

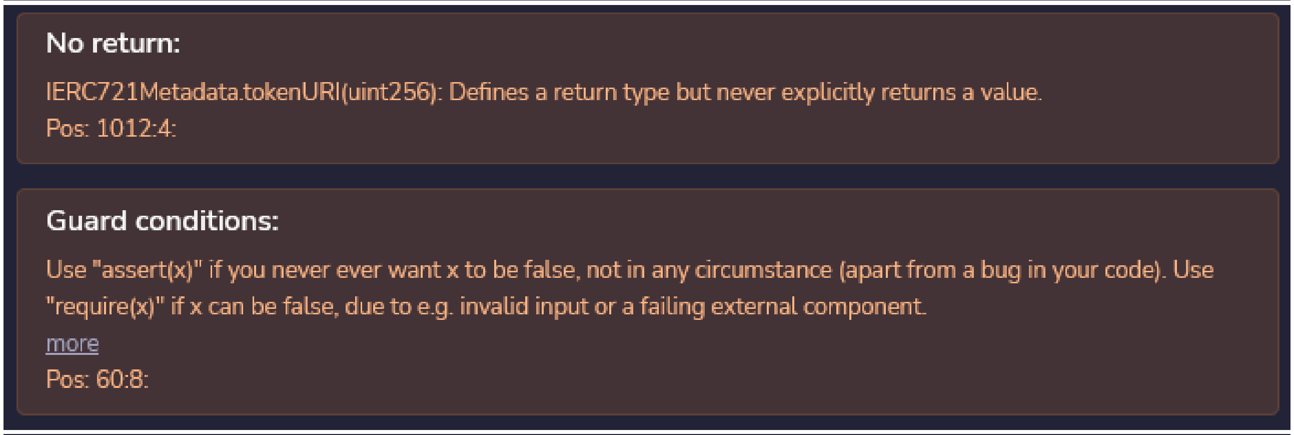
Strings.toHexString(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 36:4:


Similar variable names:

ERC721.ownerOf(uint256) : Variables have very similar names "_owners" and "owner". Note: Modifiers are currently not considered by this static analysis.
Pos: 1078:15:

Similar variable names:

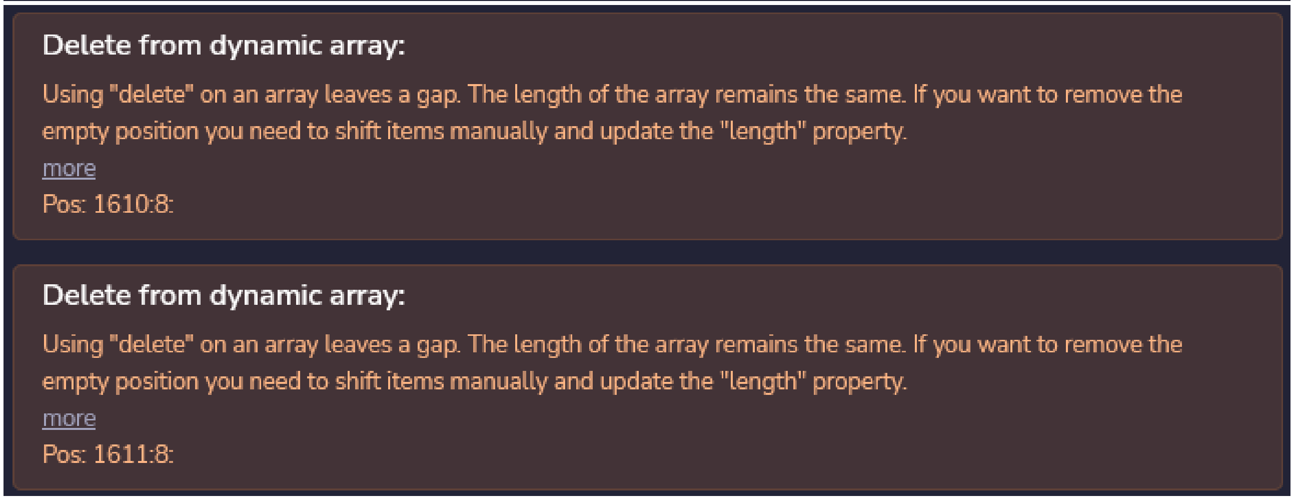
ERC721.approve(address,uint256) : Variables have very similar names "_owners" and "owner". Note: Modifiers are currently not considered by this static analysis.
Pos: 1118:8:


No return:

IERC721Metadata.tokenURI(uint256): Defines a return type but never explicitly returns a value.
Pos: 1012:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 60:8:

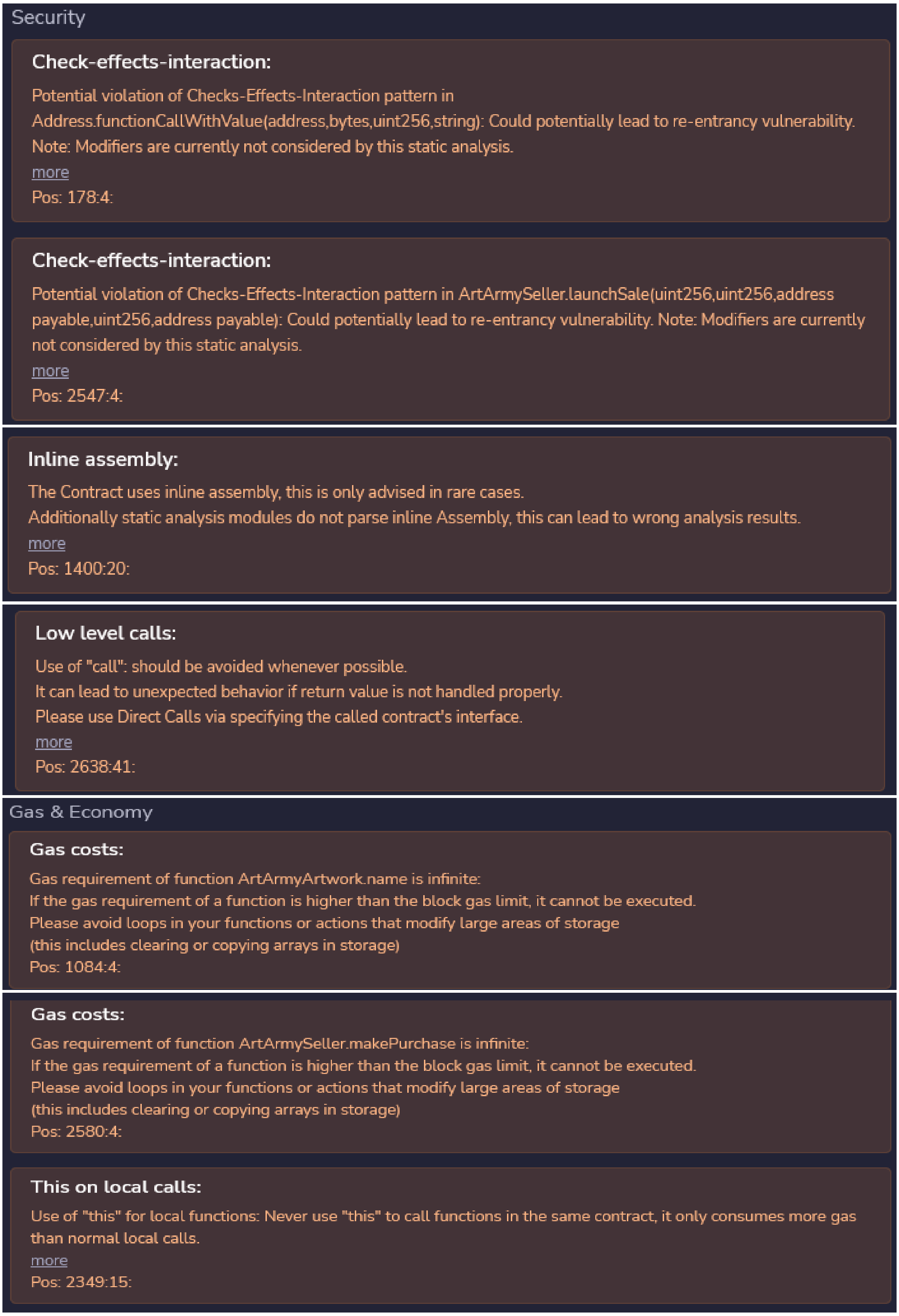

Delete from dynamic array:

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: 1610:8:

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: 1611:8:

ArtArmySeller.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 178:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ArtArmySeller.launchSale(uint256,uint256,address payable,uint256,address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2547:4:

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: 1400:20:

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: 2638:41:

Gas & Economy

Gas costs:

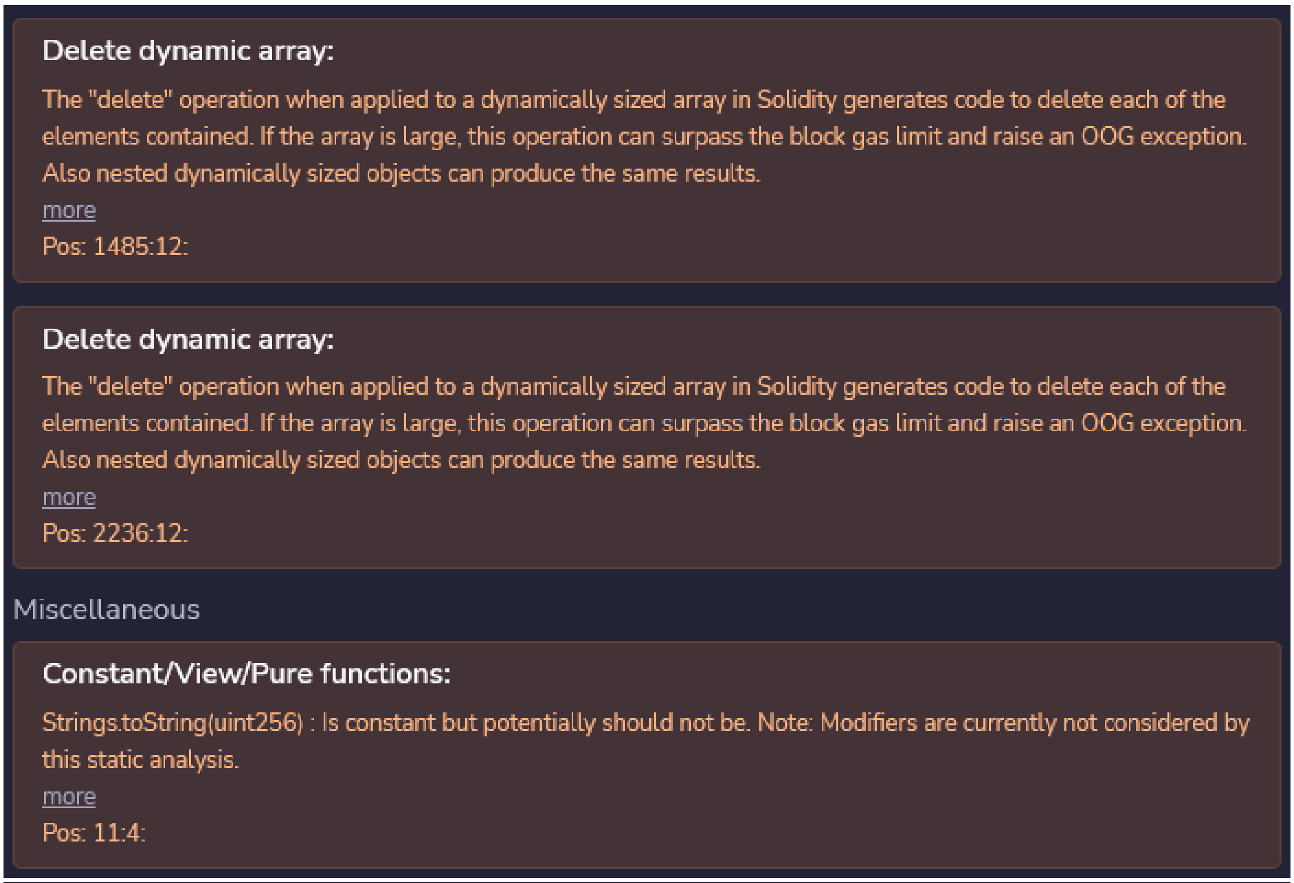
Gas requirement of function ArtArmyArtwork.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1084:4:

Gas costs:

Gas requirement of function ArtArmySeller.makePurchase 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: 2580:4:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 2349:15:



Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.
more
Pos: 1485:12:

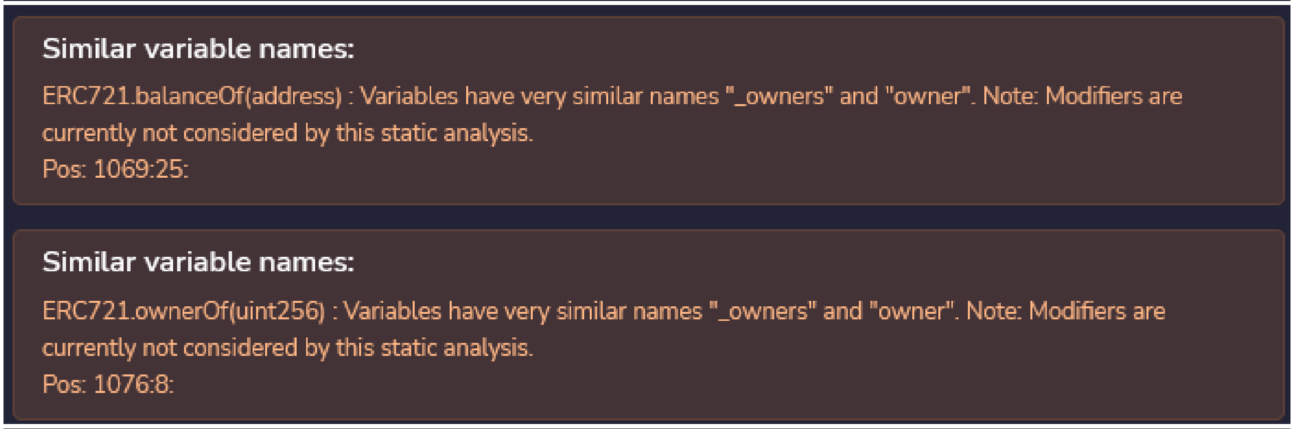
Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.
more
Pos: 2236:12:

Miscellaneous

Constant/View/Pure functions:

Strings.toString(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 11:4:

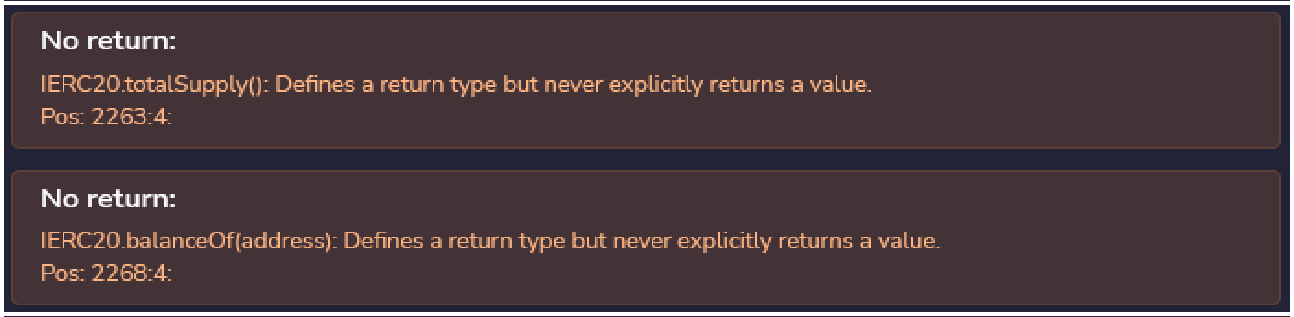


Similar variable names:

ERC721.balanceOf(address) : Variables have very similar names "_owners" and "owner". Note: Modifiers are currently not considered by this static analysis.
Pos: 1069:25:

Similar variable names:

ERC721.ownerOf(uint256) : Variables have very similar names "_owners" and "owner". Note: Modifiers are currently not considered by this static analysis.
Pos: 1076:8:

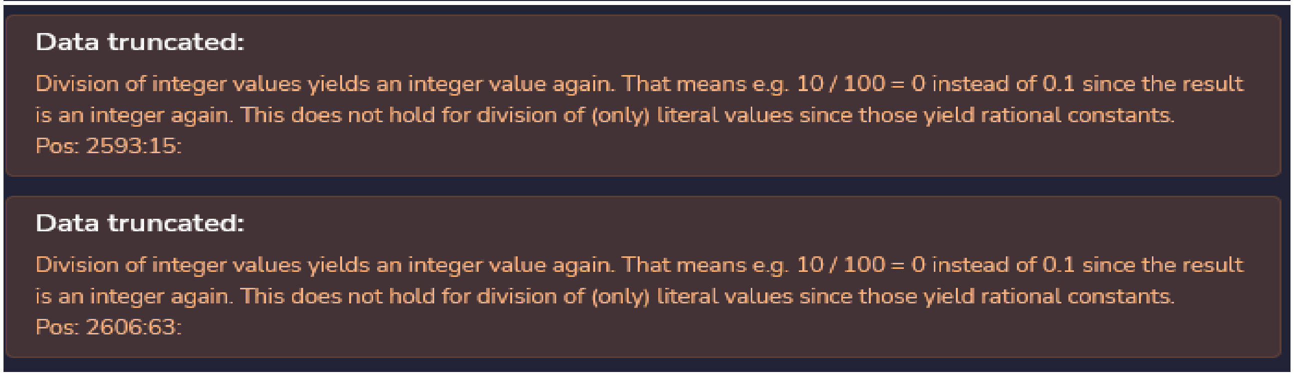


No return:

IERC20.totalSupply(): Defines a return type but never explicitly returns a value.
Pos: 2263:4:

No return:

IERC20.balanceOf(address): Defines a return type but never explicitly returns a value.
Pos: 2268:4:



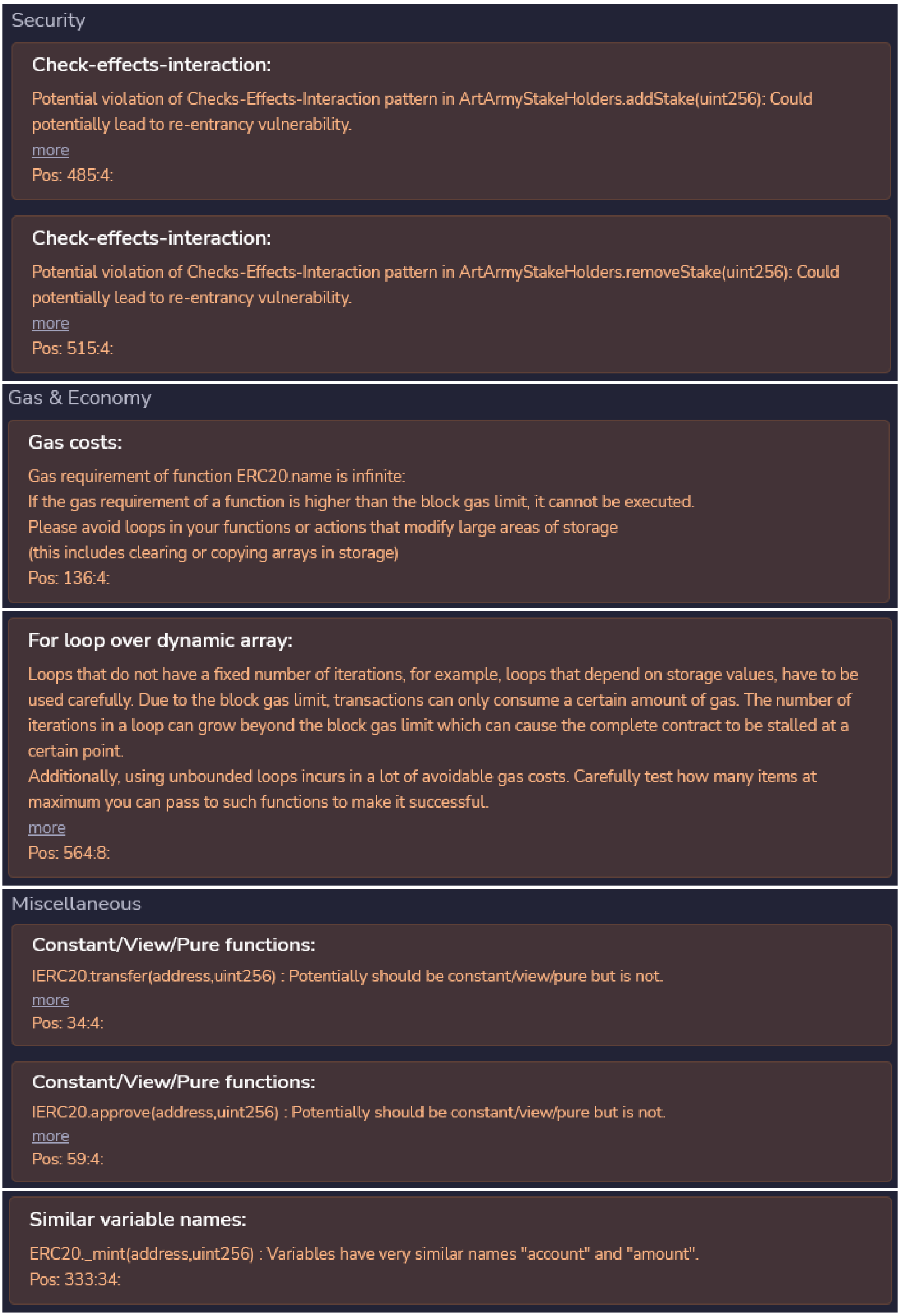
Data truncated:

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: 2593:15:

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: 2606:63:

ArtArmyHoldersStake.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ArtArmyStakeHolders.addStake(uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 485:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ArtArmyStakeHolders.removeStake(uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 515:4:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.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: 136:4:

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: 564:8:

Miscellaneous

Constant/View/Pure functions:

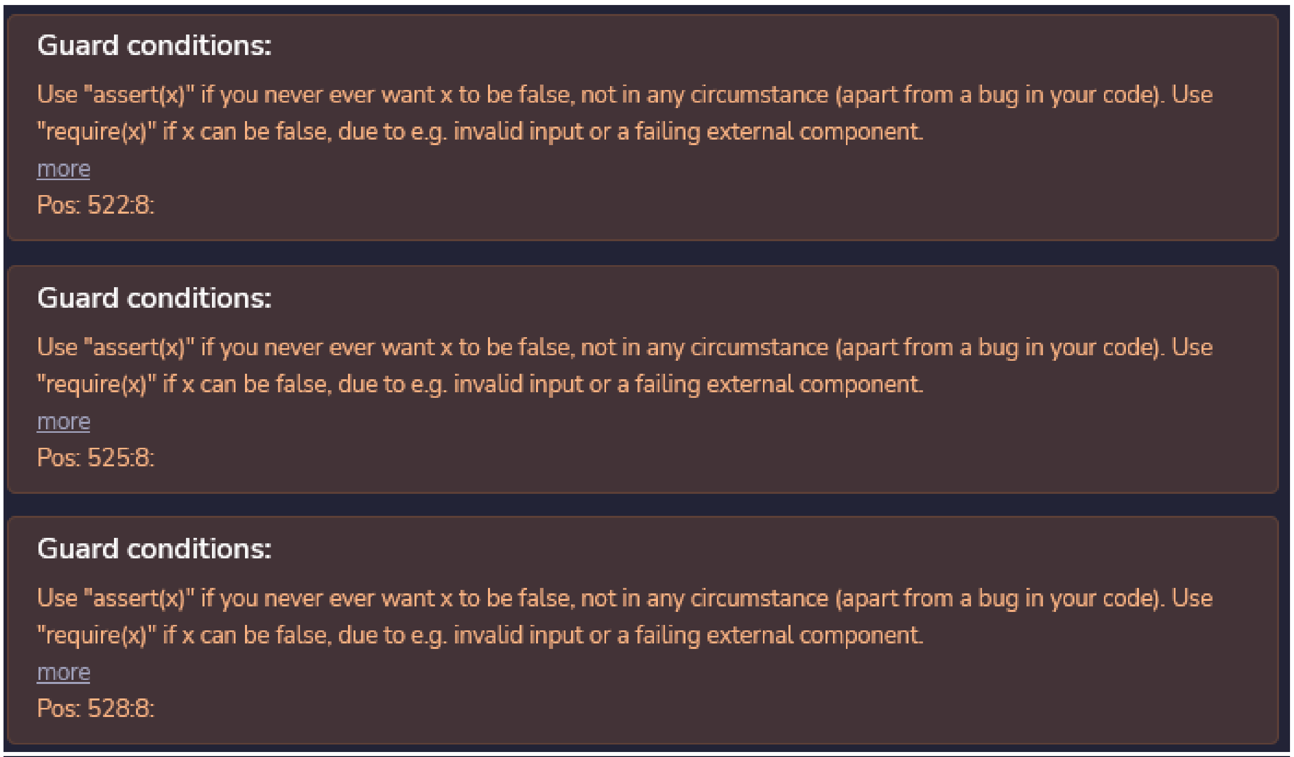
IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 34:4:

Constant/View/Pure functions:

IERC20.approve(address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 59:4:

Similar variable names:

ERC20._mint(address,uint256) : Variables have very similar names "account" and "amount".
Pos: 333:34:



Guard conditions:

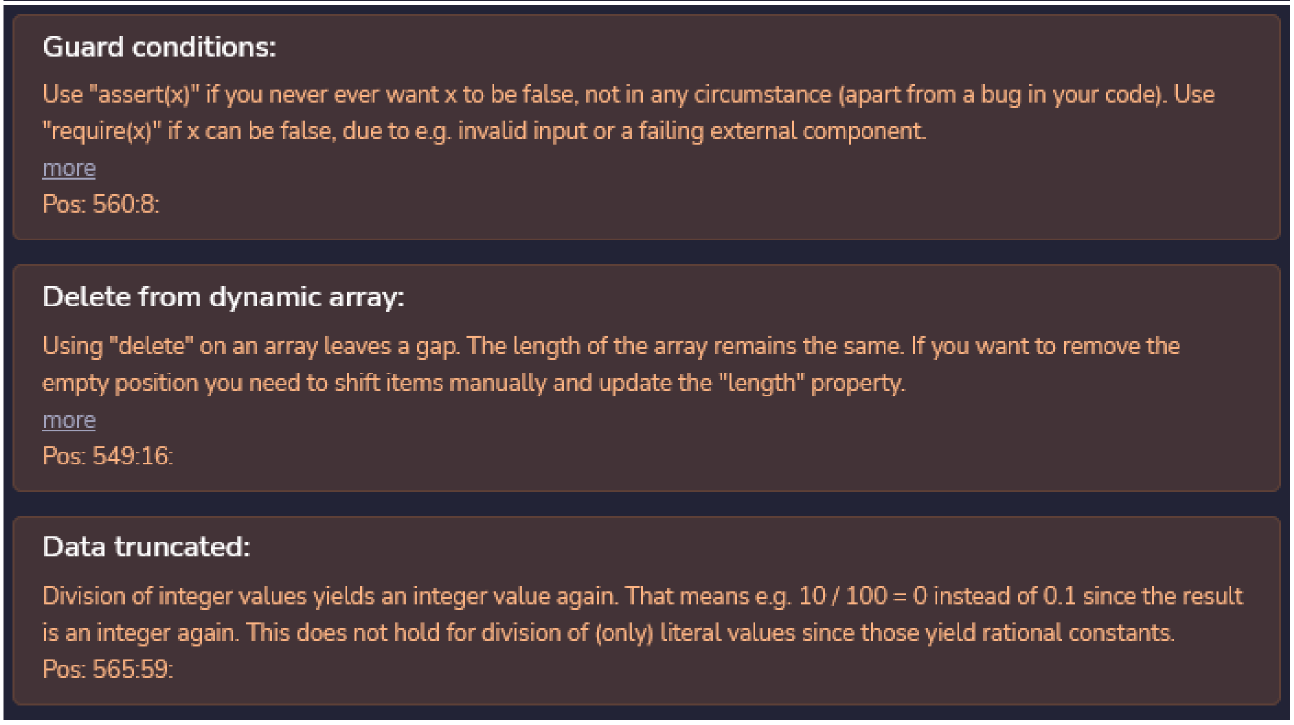
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 522:8:

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: 525:8:

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: 528:8:



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: 560:8:

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: 549:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 565:59:

Solhint Linter

ArtArmyArtwork.sol

```
ArtArmyArtwork.sol:286:18: Error: Parse error: missing ';' at '{'
ArtArmyArtwork.sol:294:18: Error: Parse error: missing ';' at '{'
```

ArtArmySeller.sol

```
ArtArmySeller.sol:286:18: Error: Parse error: missing ';' at '{'
ArtArmySeller.sol:294:18: Error: Parse error: missing ';' at '{'
```

ArtArmyHoldersStake.sol

```
ArtArmyHoldersStake.sol:233:18: Error: Parse error: missing ';' at
'{'
ArtArmyHoldersStake.sol:274:18: Error: Parse error: missing ';' at
'{'
ArtArmyHoldersStake.sol:307:18: Error: Parse error: missing ';' at
'{'
ArtArmyHoldersStake.sol:356:18: Error: Parse error: missing ';' at
'{'
```

Software analysis result:

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

EtherAuthority

Email: audit@EtherAuthority.io