EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Ansca Protocols
Website:	https://ansca.io
Platform:	Ethereum
Language:	Solidity
Date:	May 3rd, 2022

Table of contents

Introduction

EtherAuthority was contracted by the Ansca team to perform the Security audit of the Ansca 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 May 3rd, 2022.

The purpose of this audit was to address the following:

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

Project Background

- Ansca Protocol is Safe, Autonomous and Decentralized Peer-to-Peer DeFi Protocols.
- The Ansca contract inherits the IERC20, Strings, IERC721, ERC721Holder, IERC1155, ERC1155Holder, IERC165 standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

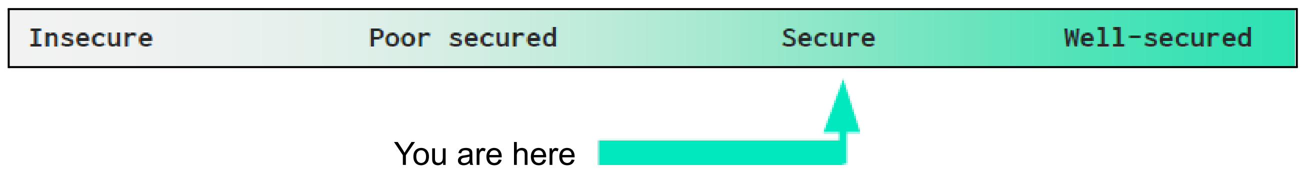
Name	**Code Review and Security Analysis Report for Ansca Protocol Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	ACE.sol
File 1 MD5 Hash	97C6880E8784EC9A0B5A5B025EF0B8BE
Updated File 1 MD5 Hash	716E770C4D095C16B7B6D376BAA9772F
File 2	ANE.sol
File 2 MD5 Hash	322DD98FE0E435864A98B9E001380B10
Updated File 2 MD5 Hash	8DE47B3F378EFC47C04DDB7BA10ED9DB
Audit Date	May 3rd, 2022
Revise Audit Date	May 31st, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 ACE.sol ● FEE: 0.75% ● Max Fee: 1% ● ACE has functions like: getAddressReputation, createEscrow, cancelEscrow, etc.	**YES, This is valid.**
File 2 ANE.sol ● FEE: 0.75% ● Max Fee: 1% ● ANE has functions like: createEscrow, cancelEscrow, etc.	**YES, This is valid.**

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, 1 medium and 1 low and some very low level issues.

All the issues have been fixed / acknowledged.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Passed
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 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 the Ansca 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 Ansca Protocol.

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

Code parts are well commented on smart contracts.

Documentation

We were given an Ansca Protocol smart contract code in the form of a File. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://ansca.io 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

ACE.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getEscrowById	external	Passed	No Issue
3	getAddressReputation	external	Passed	No Issue
4	escrowWentWell	external	Passed	No Issue
5	addressIsInvolved	external	Passed	No Issue
6	createEscrow	external	Passed	No Issue
7	getAllEscrowsForSender	external	Passed	No Issue
8	proposeAmountDiscount	external	Passed	No Issue
9	withdraw	external	Passed	No Issue
	changeHandler			
10	cancelEscrow	external	Passed	No Issue
11	proposeNewDeliveryTime	external	Passed	No Issue
12	deposit	external	Passed	No Issue
13	nativeDepositInternal	internal	Passed	No Issue
14	depositInternal	internal	Passed	No Issue
15	preDeposit	internal	Passed	No Issue
16	acceptNewDeliveryTime	external	Passed	No Issue
17	lock	external	Passed	No Issue
18	validateEscrow	external	Passed	No Issue
19	changeFeeCollectorAddress	external	Passed	No Issue
20	changeFee	external	access by owner	No Issue
21	pause	external	Passed	No Issue
22	isBuyerParty	internal	Passed	No Issue
23	isSellerParty	internal	Passed	No Issue
24	payParties	internal	Passed	No Issue
25	payParty	internal	Passed	No Issue
26	startEscrow	internal	Passed	No Issue
27	updateAction	internal	Passed	No Issue
28	emitACEEvent	internal	Passed	No Issue
29	emitCompleteEvent	internal	Passed	No Issue
30	isInOrGoingToBeOngoing	internal	Passed	No Issue
31	isInOrGoingToBeDispute	internal	Passed	No Issue
32	isInOrGoingToBeComplete	internal	Passed	No Issue

ANE.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onERC1155Received	write	Passed	No Issue

3	onERC1155BatchReceived	write	Passed	No Issue
4	escrowWentWell	external	Passed	No Issue
5	addressIsInvolved	external	Passed	No Issue
6	getAllEscrowsForSender	external	Passed	No Issue
7	getNftIdsByNftEscrowId	external	Passed	No Issue
8	getAddressesByNftEscrowId	external	Passed	No Issue
9	getNftQuantitiesByNftEscrowId	external	Passed	No Issue
10	getNftTypesByNftEscrowId	external	Passed	No Issue
11	getEscrowById	external	Passed	No Issue
12	createEscrow	external	Passed	No Issue
13	cancelEscrow	external	Passed	No Issue
14	buyNft	external	access by owner	No Issue
15	changeFeeCollectorAddress	external	access by owner	No Issue
16	changeFee	external	Passed	No Issue
17	pause	external	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

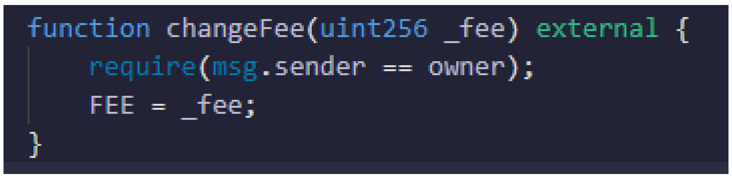
No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

(1) Function input parameters lack of check: **ACE.sol, ANE.sol**



```
function changeFee(uint256 _fee) external {
    require(msg.sender == owner);
    FEE = _fee;
}
```

Owner can set fees to 100%. Hence the buyer's or seller's receiving the amount gets 0.

Resolution: We suggest setting a minimum and maximum limit for fees.

Status: Fixed

Low

(1) Function input parameters lack of check: **ACE.sol**

Variable validation is not performed in below functions :

- createEscrow = _tokenAddress

Resolution: We advise to put validation : int type variables should not be empty and > 0 & address type variables should not be address(0).

Status: Fixed.

Very Low / Informational / Best practices:

(1) Critical operation lacks event log:

Missing event log for:

ACE.sol

- createEscrow
- proposeAmountDiscount
- withdraw
- cancelEscrow
- proposeNewDeliveryTime
- deposit
- acceptNewDeliveryTime
- validateEscrow

ANE.sol

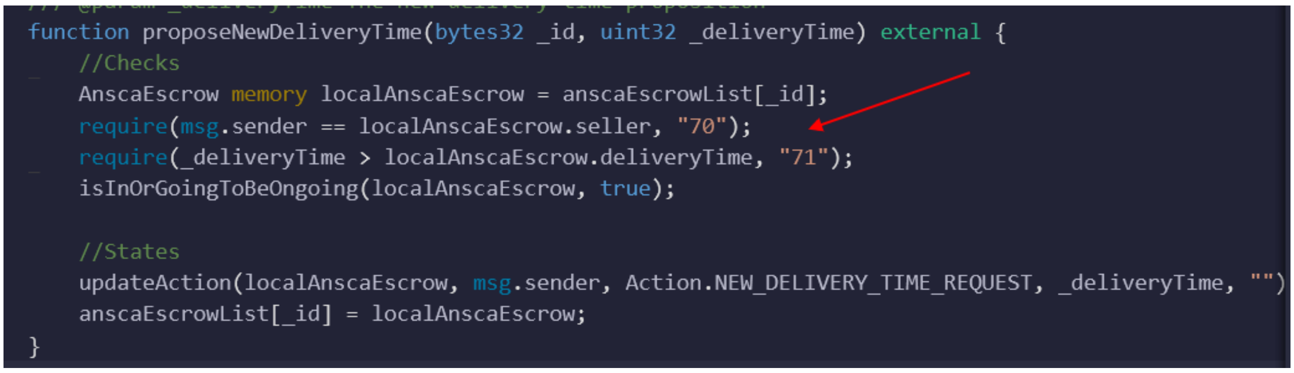
- buyNft
- cancelEscrow
- createEscrow

Resolution: Write an event log for listed events.

Status: Fixed

(2) Ambiguous Error Message:

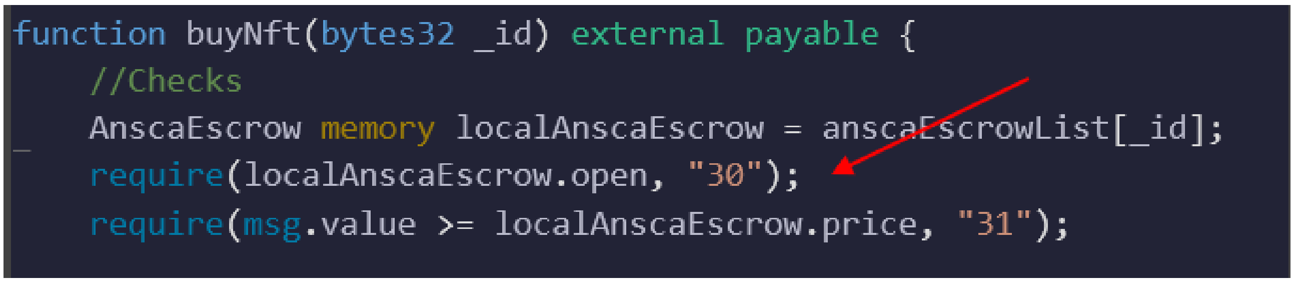
ACE.sol



```
function proposeNewDeliveryTime(bytes32 _id, uint32 _deliveryTime) external {
    //Checks
    AnscaEscrow memory localAnscaEscrow = anscaEscrowList[_id];
    require(msg.sender == localAnscaEscrow.seller, "70");
    require(_deliveryTime > localAnscaEscrow.deliveryTime, "71");
    isInOrGoingToBeOngoing(localAnscaEscrow, true);

    //States
    updateAction(localAnscaEscrow, msg.sender, Action.NEW_DELIVERY_TIME_REQUEST, _deliveryTime, "")
    anscaEscrowList[_id] = localAnscaEscrow;
}
```

ANE.sol



```
function buyNft(bytes32 _id) external payable {
    //Checks
    AnscaEscrow memory localAnscaEscrow = anscaEscrowList[_id];
    require(localAnscaEscrow.open, "30");
    require(msg.value >= localAnscaEscrow.price, "31");
```

In all the functions , The mentioned error messages do not explain exactly the error of the operation.

Resolution: As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system.

Status: Acknowledged.

Comments : To reduce the gas fee and size of the code, short error message codes have been used.

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:

- changeFee: ANE owner can update fees.
- changeFee: ACE owner can update fees.
- pause: ANE owner can set pause status.
- changeFeeCollectorAddress: ANE owner can change fee collector address.
- pause: ACE owner can set pause status.
- changeFeeCollectorAddress: ACE owner can change fee collector address.

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 files. And we have used all possible tests based on given objects as files. We had observed some issues in the smart contracts. All the issues have been fixed / acknowledged. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

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

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

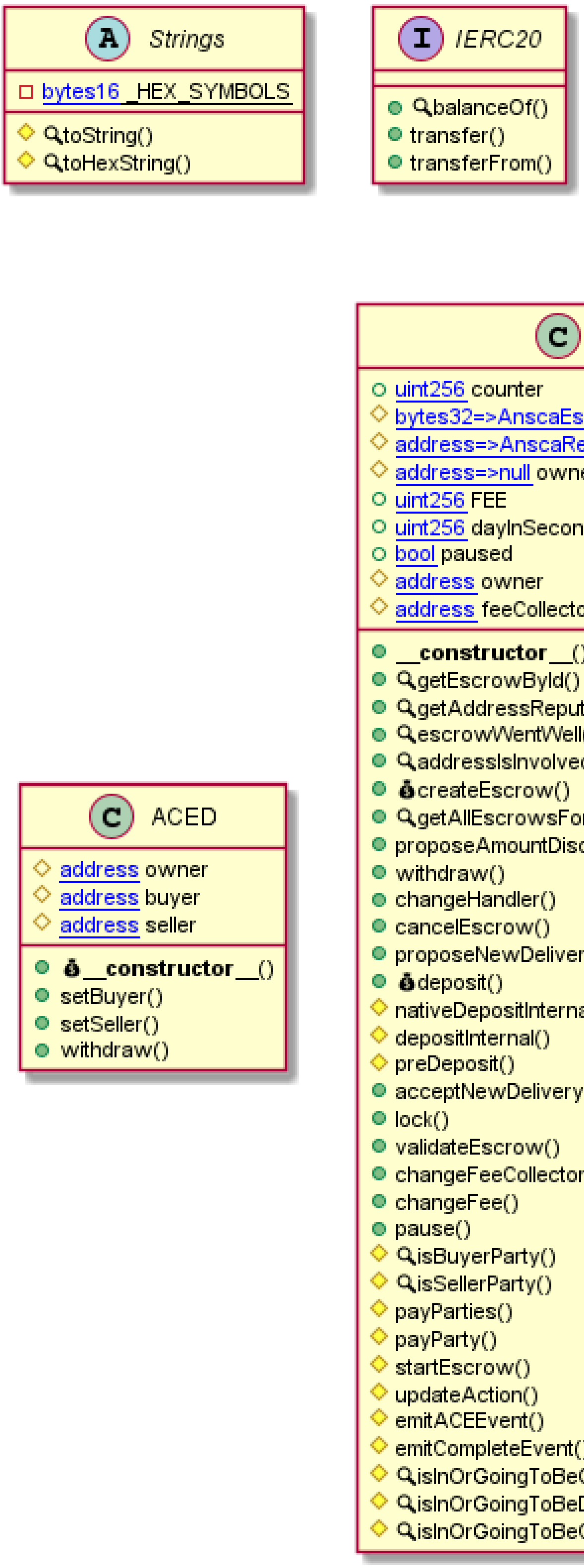
Technical Disclaimer

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

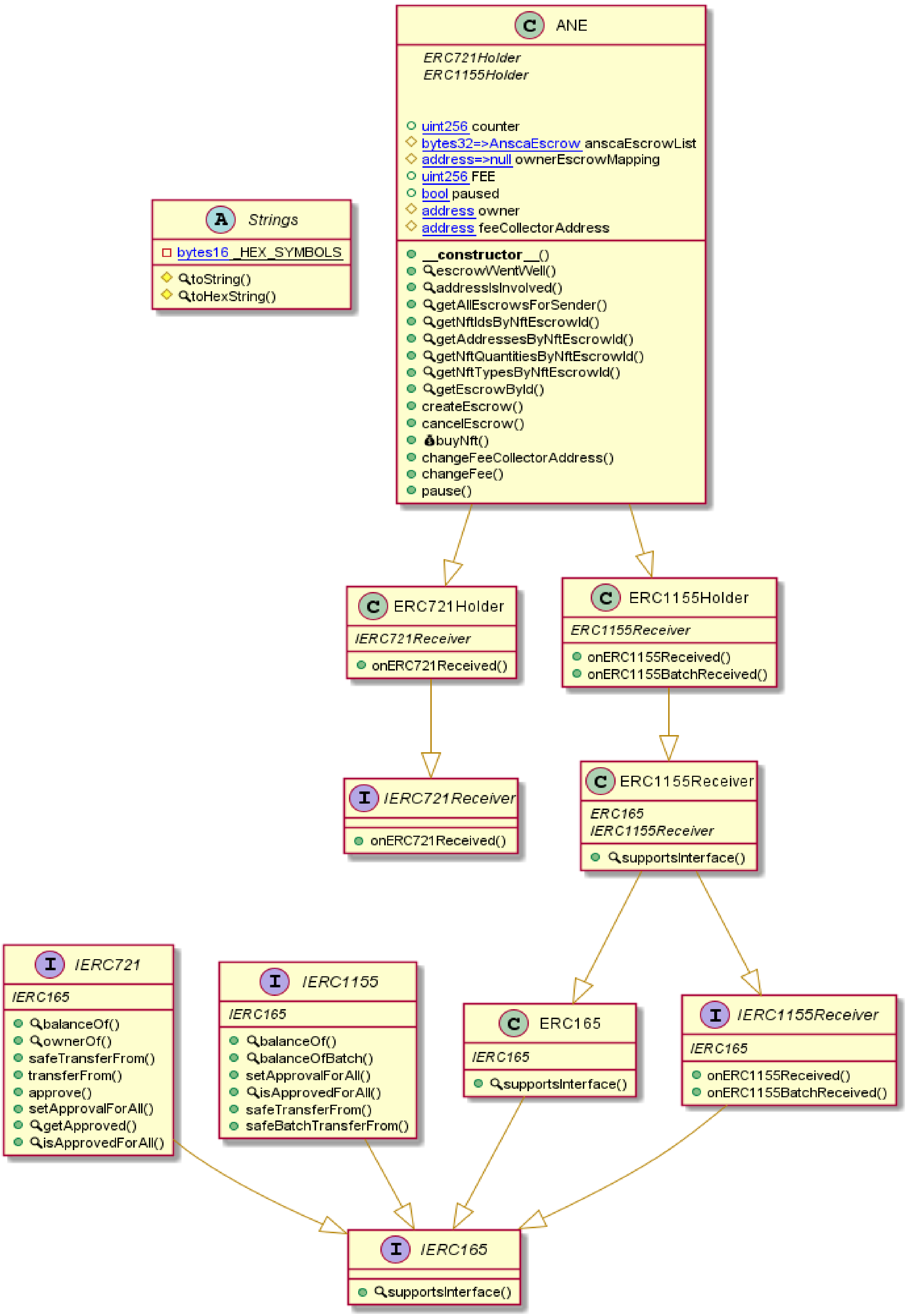
Appendix

Code Flow Diagram - Ansca Protocol

ACE Diagram

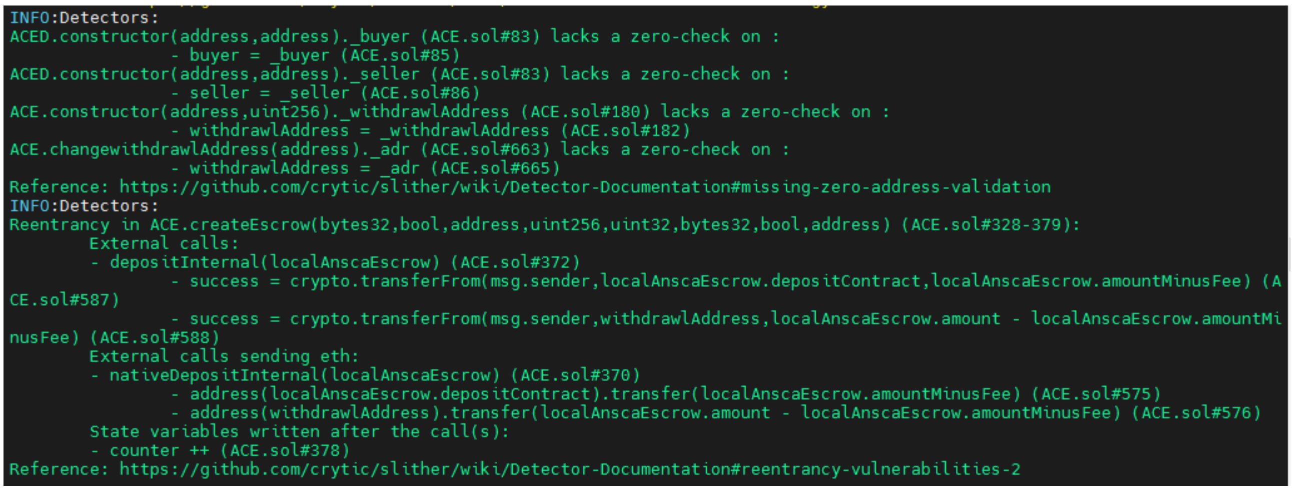


ANE Diagram

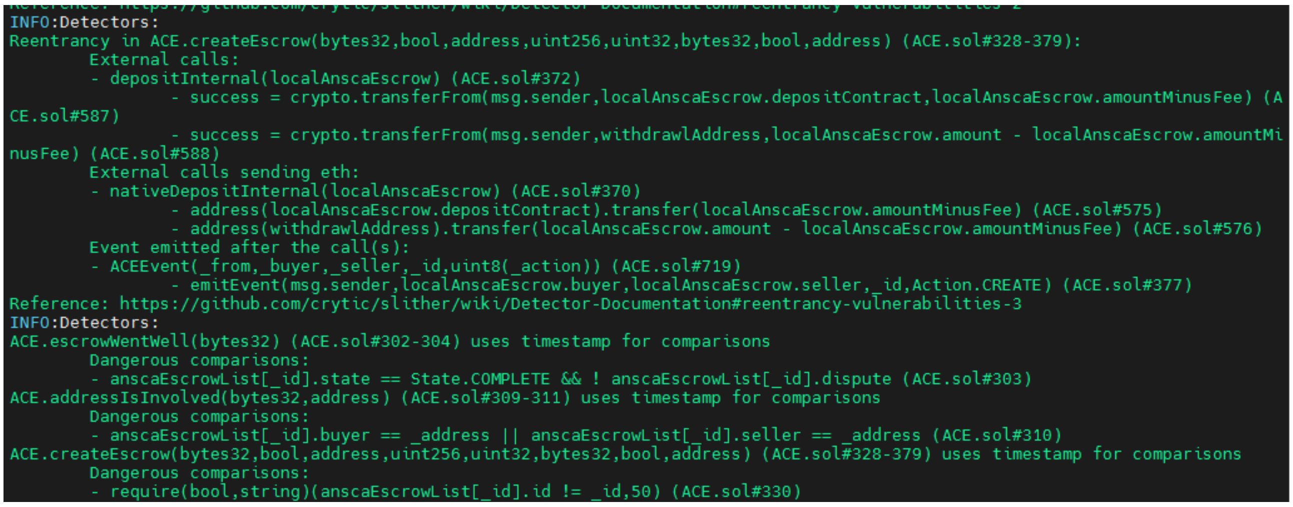

ANE
ERC721Holder
ERC1155Holder
uint256 counter
bytes32=>AnscaEscrow anscaEscrowList
address=>null ownerEscrowMapping
uint256 FEE
bool paused
address owner
address feeCollectorAddress
__constructor__()
escrowWentWell()
addressIsInvolved()
getAllEscrowsForSender()
getNftIdsByNftEscrowId()
getAddressesByNftEscrowId()
getNftQuantitiesByNftEscrowId()
getNftTypesByNftEscrowId()
getEscrowById()
createEscrow()
cancelEscrow()
buyNft()
changeFeeCollectorAddress()
changeFee()
pause()
Strings
bytes16 _HEX_SYMBOLS
toString()
toHexString()
ERC721Holder
IERC721Receiver
onERC721Received()
ERC1155Holder
ERC1155Receiver
onERC1155Received()
onERC1155BatchReceived()
IERC721Receiver
onERC721Received()
ERC1155Receiver
ERC165
IERC1155Receiver
supportsInterface()
IERC721
IERC165
balanceOf()
ownerOf()
safeTransferFrom()
transferFrom()
approve()
setApprovalForAll()
getApproved()
isApprovedForAll()
IERC1155
IERC165
balanceOf()
balanceOfBatch()
setApprovalForAll()
isApprovedForAll()
safeTransferFrom()
safeBatchTransferFrom()
ERC165
IERC165
supportsInterface()
IERC1155Receiver
IERC165
onERC1155Received()
onERC1155BatchReceived()
IERC165
supportsInterface()

Slither Results Log

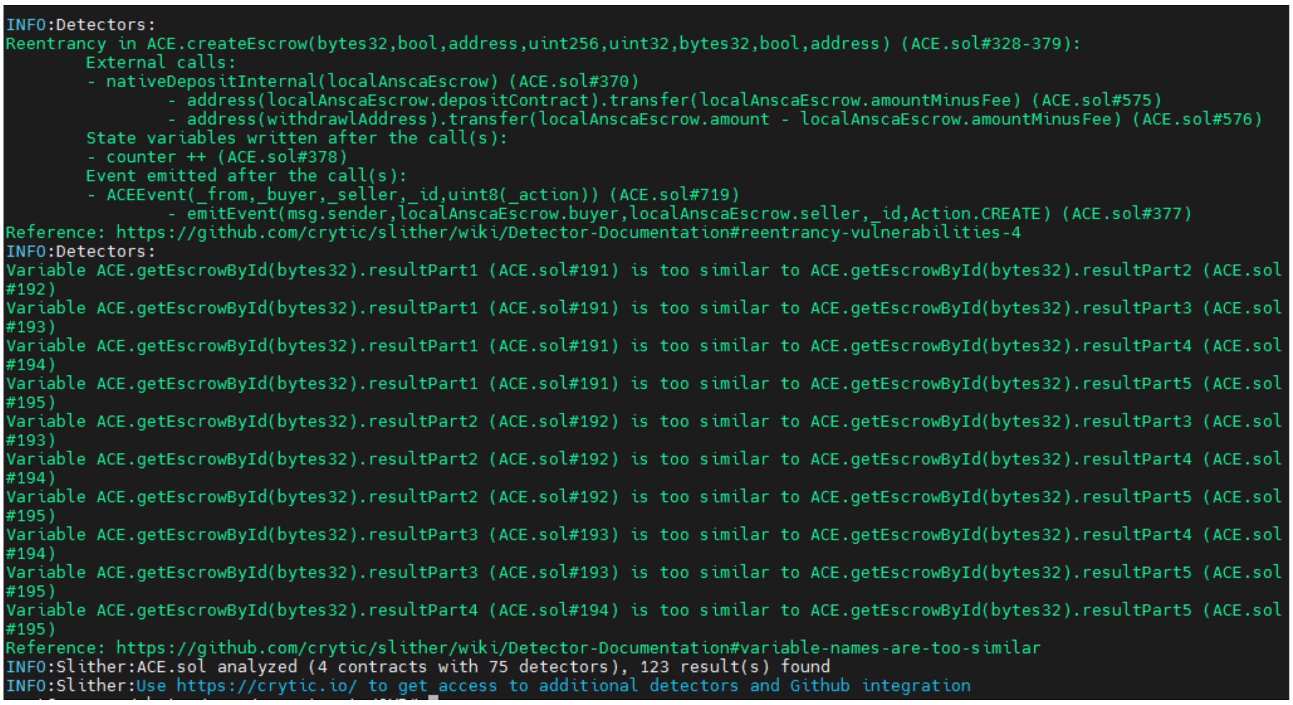
Slither log >> ACE.sol



```
INFO:Detectors:
ACED.constructor(address,address)._buyer (ACE.sol#83) lacks a zero-check on :
		- buyer = _buyer (ACE.sol#85)
ACED.constructor(address,address)._seller (ACE.sol#83) lacks a zero-check on :
		- seller = _seller (ACE.sol#86)
ACE.constructor(address,uint256)._withdrawlAddress (ACE.sol#180) lacks a zero-check on :
		- withdrawlAddress = _withdrawlAddress (ACE.sol#182)
ACE.changewithdrawlAddress(address)._adr (ACE.sol#663) lacks a zero-check on :
		- withdrawlAddress = _adr (ACE.sol#665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ACE.createEscrow(bytes32,bool,address,uint256,uint32,bytes32,bool,address) (ACE.sol#328-379):
	External calls:
	- depositInternal(localAnscaEscrow) (ACE.sol#372)
		- success = crypto.transferFrom(msg.sender,localAnscaEscrow.depositContract,localAnscaEscrow.amountMinusFee) (A
CE.sol#587)
		- success = crypto.transferFrom(msg.sender,withdrawlAddress,localAnscaEscrow.amount - localAnscaEscrow.amountMi
nusFee) (ACE.sol#588)
	External calls sending eth:
	- nativeDepositInternal(localAnscaEscrow) (ACE.sol#370)
		- address(localAnscaEscrow.depositContract).transfer(localAnscaEscrow.amountMinusFee) (ACE.sol#575)
		- address(withdrawlAddress).transfer(localAnscaEscrow.amount - localAnscaEscrow.amountMinusFee) (ACE.sol#576)
	State variables written after the call(s):
	- counter ++ (ACE.sol#378)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

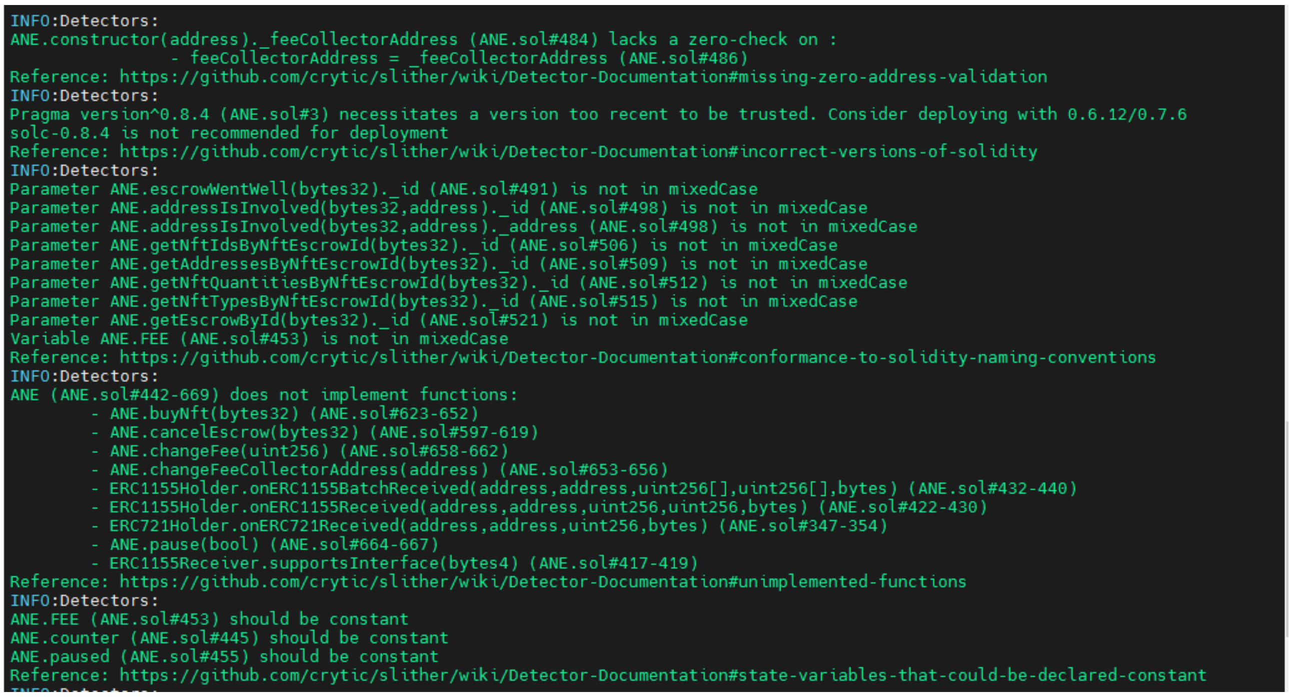


```
INFO:Detectors:
Reentrancy in ACE.createEscrow(bytes32,bool,address,uint256,uint32,bytes32,bool,address) (ACE.sol#328-379):
	External calls:
	- depositInternal(localAnscaEscrow) (ACE.sol#372)
		- success = crypto.transferFrom(msg.sender,localAnscaEscrow.depositContract,localAnscaEscrow.amountMinusFee) (A
CE.sol#587)
		- success = crypto.transferFrom(msg.sender,withdrawlAddress,localAnscaEscrow.amount - localAnscaEscrow.amountMi
nusFee) (ACE.sol#588)
	External calls sending eth:
	- nativeDepositInternal(localAnscaEscrow) (ACE.sol#370)
		- address(localAnscaEscrow.depositContract).transfer(localAnscaEscrow.amountMinusFee) (ACE.sol#575)
		- address(withdrawlAddress).transfer(localAnscaEscrow.amount - localAnscaEscrow.amountMinusFee) (ACE.sol#576)
	Event emitted after the call(s):
	- ACEEvent(_from,_buyer,_seller,_id,uint8(_action)) (ACE.sol#719)
		- emitEvent(msg.sender,localAnscaEscrow.buyer,localAnscaEscrow.seller,_id,Action.CREATE) (ACE.sol#377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ACE.escrowWentWell(bytes32) (ACE.sol#302-304) uses timestamp for comparisons
	Dangerous comparisons:
	- anscaEscrowList[_id].state == State.COMPLETE && ! anscaEscrowList[_id].dispute (ACE.sol#303)
ACE.addressIsInvolved(bytes32,address) (ACE.sol#309-311) uses timestamp for comparisons
	Dangerous comparisons:
	- anscaEscrowList[_id].buyer == _address || anscaEscrowList[_id].seller == _address (ACE.sol#310)
ACE.createEscrow(bytes32,bool,address,uint256,uint32,bytes32,bool,address) (ACE.sol#328-379) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(anscaEscrowList[_id].id != _id,50) (ACE.sol#330)
```

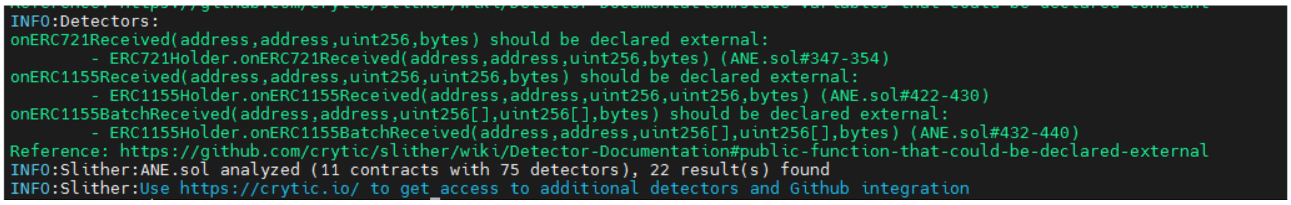


```
INFO:Detectors:
Reentrancy in ACE.createEscrow(bytes32,bool,address,uint256,uint32,bytes32,bool,address) (ACE.sol#328-379):
	External calls:
	- nativeDepositInternal(localAnscaEscrow) (ACE.sol#370)
		- address(localAnscaEscrow.depositContract).transfer(localAnscaEscrow.amountMinusFee) (ACE.sol#575)
		- address(withdrawlAddress).transfer(localAnscaEscrow.amount - localAnscaEscrow.amountMinusFee) (ACE.sol#576)
	State variables written after the call(s):
	- counter ++ (ACE.sol#378)
	Event emitted after the call(s):
	- ACEEvent(_from,_buyer,_seller,_id,uint8(_action)) (ACE.sol#719)
		- emitEvent(msg.sender,localAnscaEscrow.buyer,localAnscaEscrow.seller,_id,Action.CREATE) (ACE.sol#377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable ACE.getEscrowById(bytes32).resultPart1 (ACE.sol#191) is too similar to ACE.getEscrowById(bytes32).resultPart2 (ACE.sol
#192)
Variable ACE.getEscrowById(bytes32).resultPart1 (ACE.sol#191) is too similar to ACE.getEscrowById(bytes32).resultPart3 (ACE.sol
#193)
Variable ACE.getEscrowById(bytes32).resultPart1 (ACE.sol#191) is too similar to ACE.getEscrowById(bytes32).resultPart4 (ACE.sol
#194)
Variable ACE.getEscrowById(bytes32).resultPart1 (ACE.sol#191) is too similar to ACE.getEscrowById(bytes32).resultPart5 (ACE.sol
#195)
Variable ACE.getEscrowById(bytes32).resultPart2 (ACE.sol#192) is too similar to ACE.getEscrowById(bytes32).resultPart3 (ACE.sol
#193)
Variable ACE.getEscrowById(bytes32).resultPart2 (ACE.sol#192) is too similar to ACE.getEscrowById(bytes32).resultPart4 (ACE.sol
#194)
Variable ACE.getEscrowById(bytes32).resultPart2 (ACE.sol#192) is too similar to ACE.getEscrowById(bytes32).resultPart5 (ACE.sol
#195)
Variable ACE.getEscrowById(bytes32).resultPart3 (ACE.sol#193) is too similar to ACE.getEscrowById(bytes32).resultPart4 (ACE.sol
#194)
Variable ACE.getEscrowById(bytes32).resultPart3 (ACE.sol#193) is too similar to ACE.getEscrowById(bytes32).resultPart5 (ACE.sol
#195)
Variable ACE.getEscrowById(bytes32).resultPart4 (ACE.sol#194) is too similar to ACE.getEscrowById(bytes32).resultPart5 (ACE.sol
#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Slither:ACE.sol analyzed (4 contracts with 75 detectors), 123 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ANE.sol



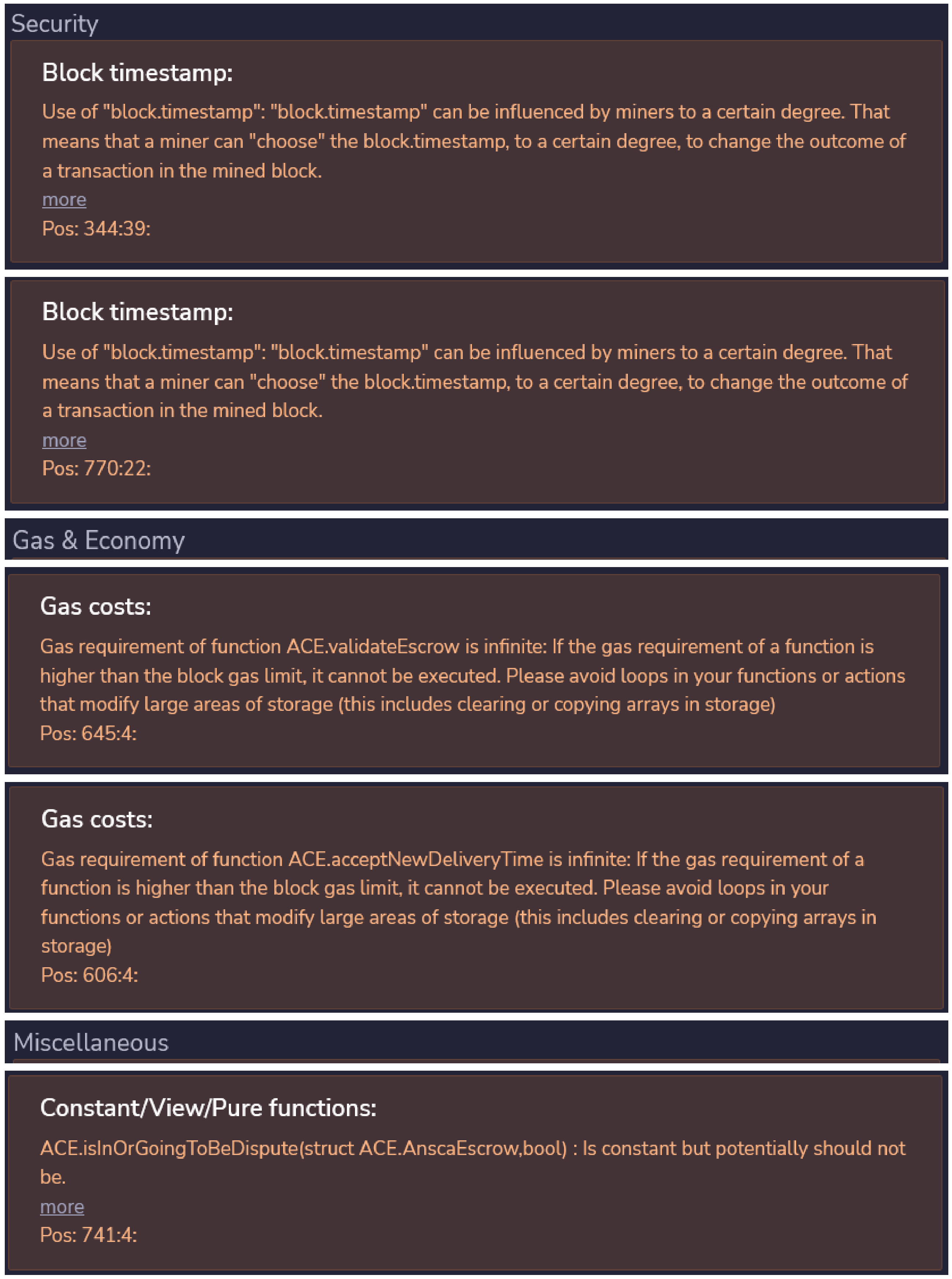
```
INFO:Detectors:
ANE.constructor(address)._feeCollectorAddress (ANE.sol#484) lacks a zero-check on :
                - feeCollectorAddress = _feeCollectorAddress (ANE.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Pragma version^0.8.4 (ANE.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter ANE.escrowWentWell(bytes32)._id (ANE.sol#491) is not in mixedCase
Parameter ANE.addressIsInvolved(bytes32,address)._id (ANE.sol#498) is not in mixedCase
Parameter ANE.addressIsInvolved(bytes32,address)._address (ANE.sol#498) is not in mixedCase
Parameter ANE.getNftIdsByNftEscrowId(bytes32)._id (ANE.sol#506) is not in mixedCase
Parameter ANE.getAddressesByNftEscrowId(bytes32)._id (ANE.sol#509) is not in mixedCase
Parameter ANE.getNftQuantitiesByNftEscrowId(bytes32)._id (ANE.sol#512) is not in mixedCase
Parameter ANE.getNftTypesByNftEscrowId(bytes32)._id (ANE.sol#515) is not in mixedCase
Parameter ANE.getEscrowById(bytes32)._id (ANE.sol#521) is not in mixedCase
Variable ANE.FEE (ANE.sol#453) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ANE (ANE.sol#442-669) does not implement functions:
        - ANE.buyNft(bytes32) (ANE.sol#623-652)
        - ANE.cancelEscrow(bytes32) (ANE.sol#597-619)
        - ANE.changeFee(uint256) (ANE.sol#658-662)
        - ANE.changeFeeCollectorAddress(address) (ANE.sol#653-656)
        - ERC1155Holder.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (ANE.sol#432-440)
        - ERC1155Holder.onERC1155Received(address,address,uint256,uint256,bytes) (ANE.sol#422-430)
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (ANE.sol#347-354)
        - ANE.pause(bool) (ANE.sol#664-667)
        - ERC1155Receiver.supportsInterface(bytes4) (ANE.sol#417-419)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ANE.FEE (ANE.sol#453) should be constant
ANE.counter (ANE.sol#445) should be constant
ANE.paused (ANE.sol#455) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```



```
INFO:Detectors:
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (ANE.sol#347-354)
onERC1155Received(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Holder.onERC1155Received(address,address,uint256,uint256,bytes) (ANE.sol#422-430)
onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Holder.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (ANE.sol#432-440)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ANE.sol analyzed (11 contracts with 75 detectors), 22 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ACE.sol



Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 344:39:

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: 770:22:

Gas & Economy

Gas costs:

Gas requirement of function ACE.validateEscrow 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: 645:4:

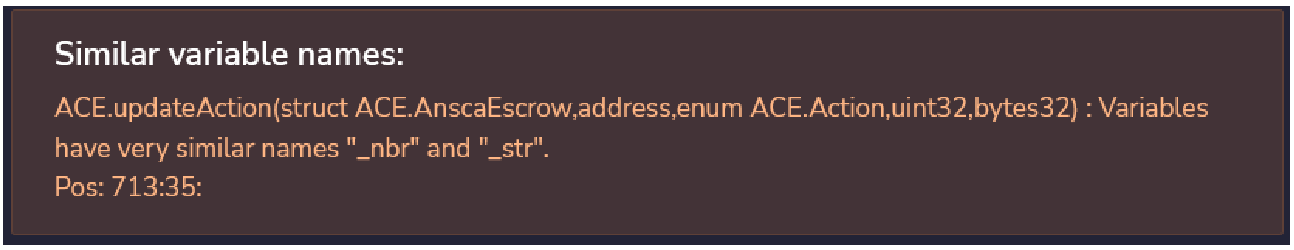
Gas costs:

Gas requirement of function ACE.acceptNewDeliveryTime 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: 606:4:

Miscellaneous

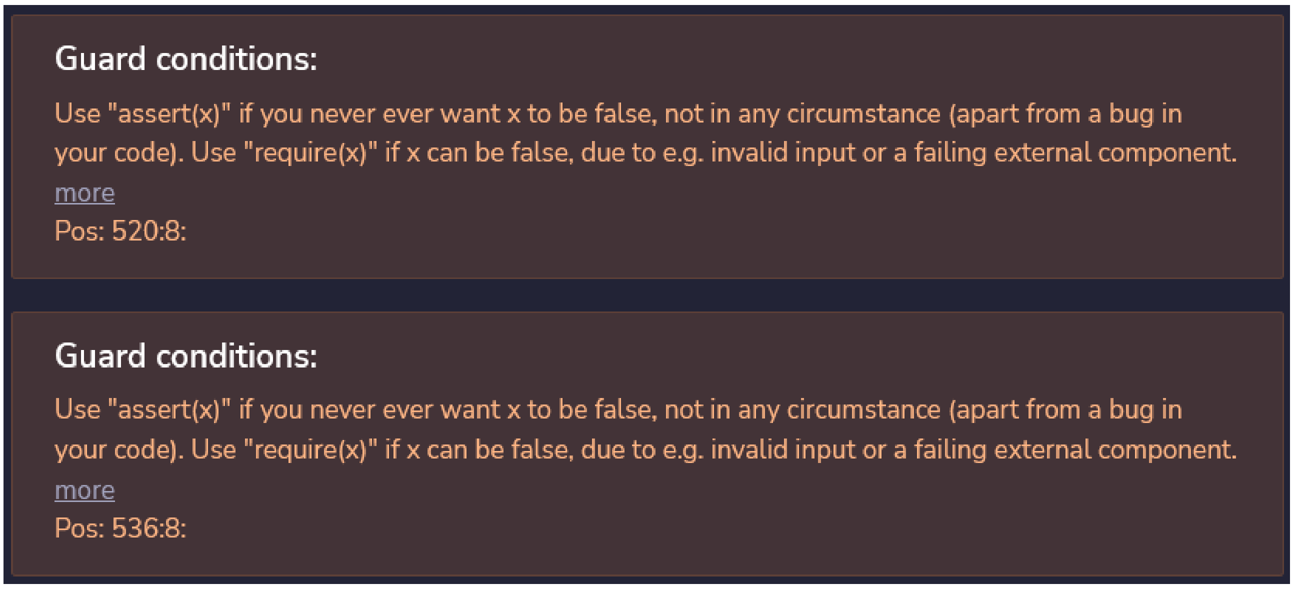
Constant/View/Pure functions:

ACE.isInOrGoingToBeDispute(struct ACE.AnscaEscrow,bool) : Is constant but potentially should not be.
more
Pos: 741:4:



Similar variable names:

ACE.updateAction(struct ACE.AnscaEscrow,address,enum ACE.Action,uint32,bytes32) : Variables have very similar names "_nbr" and "_str".
Pos: 713:35:

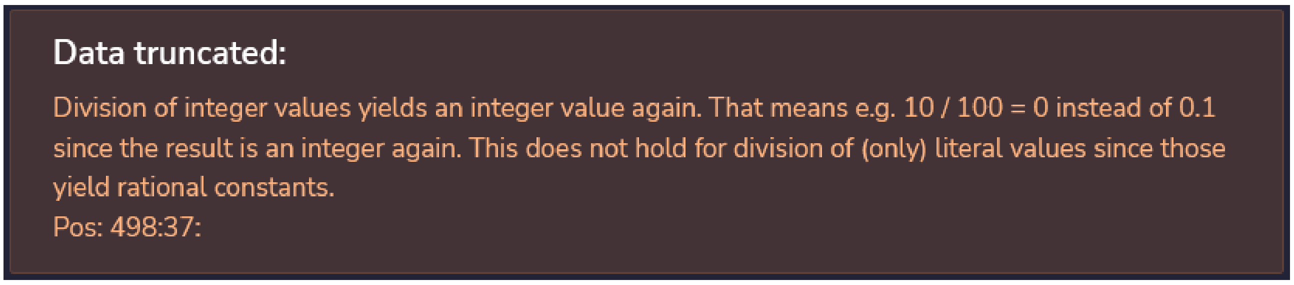


Guard conditions:

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: 520:8:

Guard conditions:

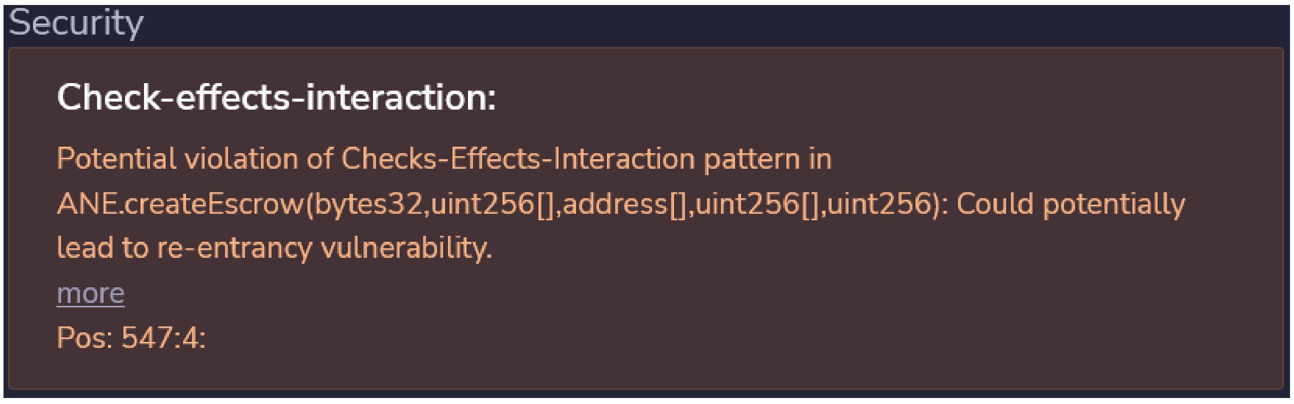
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 536:8:



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: 498:37:

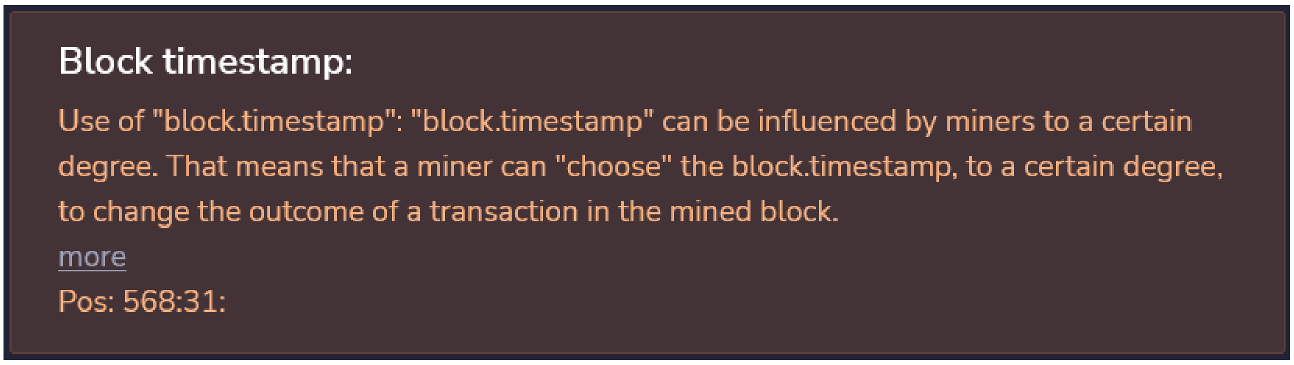
ANE.sol



Security

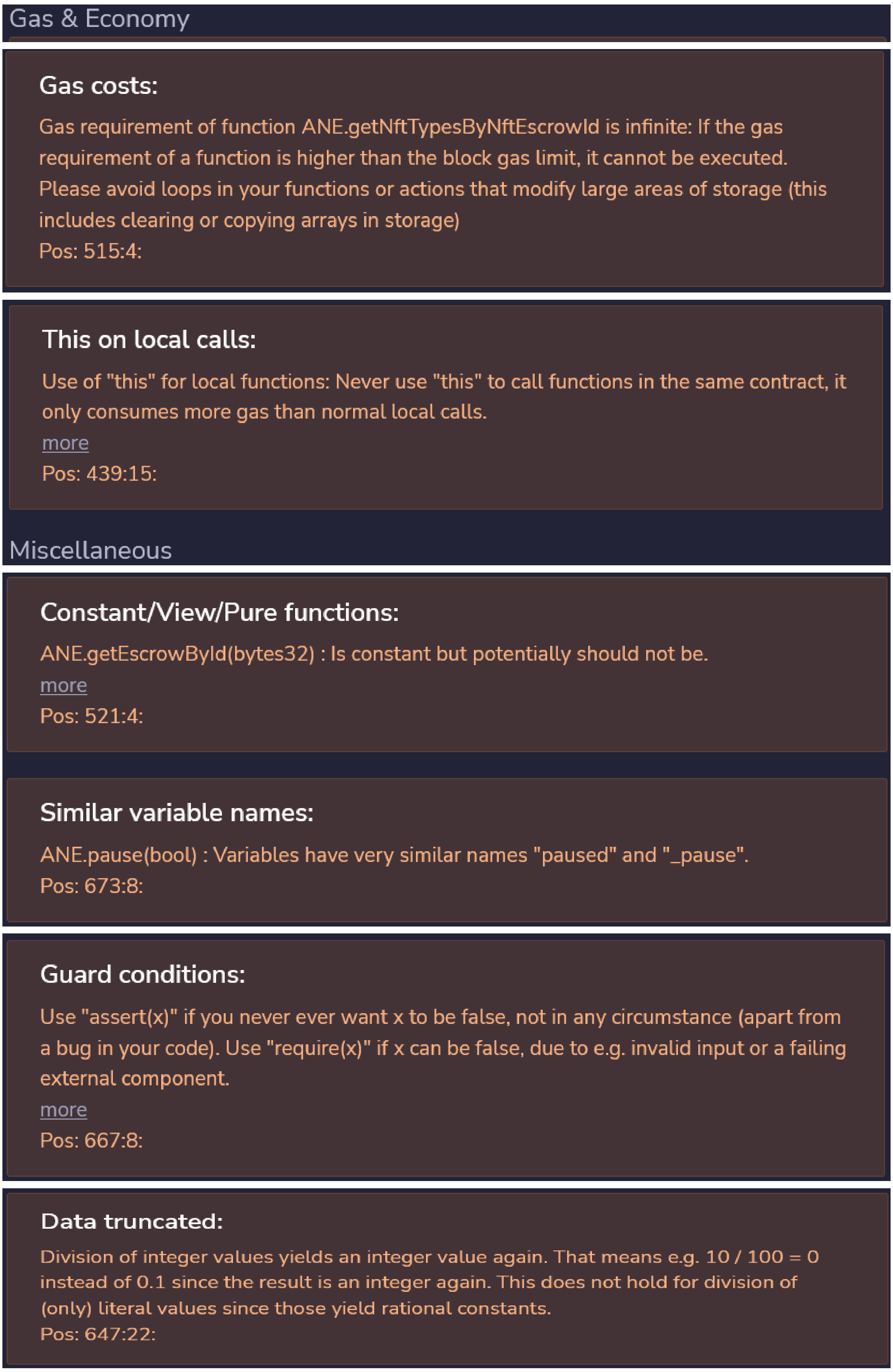
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ANE.createEscrow(bytes32,uint256[],address[],uint256[],uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 547:4:



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: 568:31:



Gas & Economy

Gas costs:

Gas requirement of function ANE.getNftTypesByNftEscrowId 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: 515:4:

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: 439:15:

Miscellaneous

Constant/View/Pure functions:

ANE.getEscrowById(bytes32) : Is constant but potentially should not be.
more
Pos: 521:4:

Similar variable names:

ANE.pause(bool) : Variables have very similar names "paused" and "_pause".
Pos: 673:8:

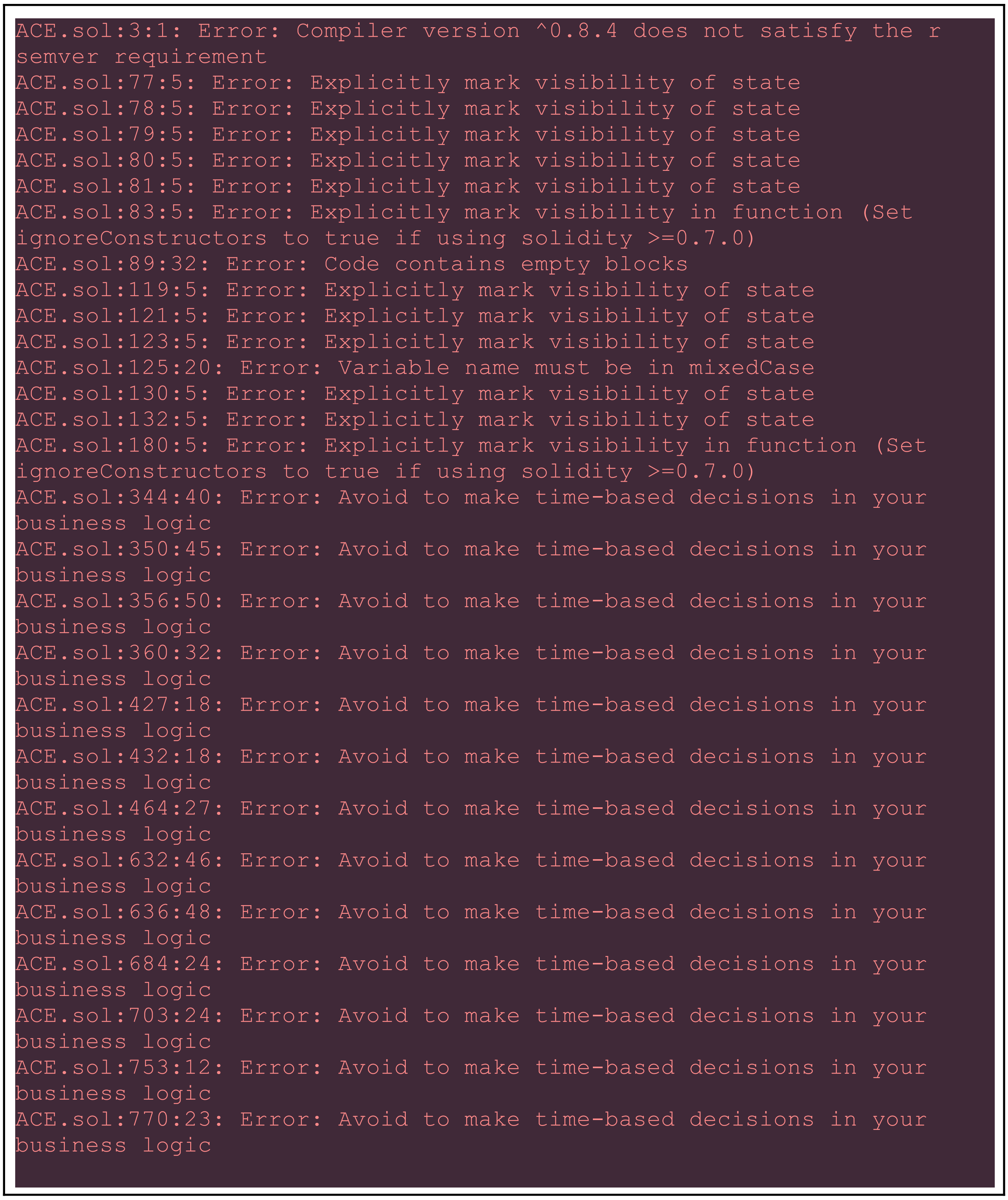
Guard conditions:

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: 667:8:

Data truncated:

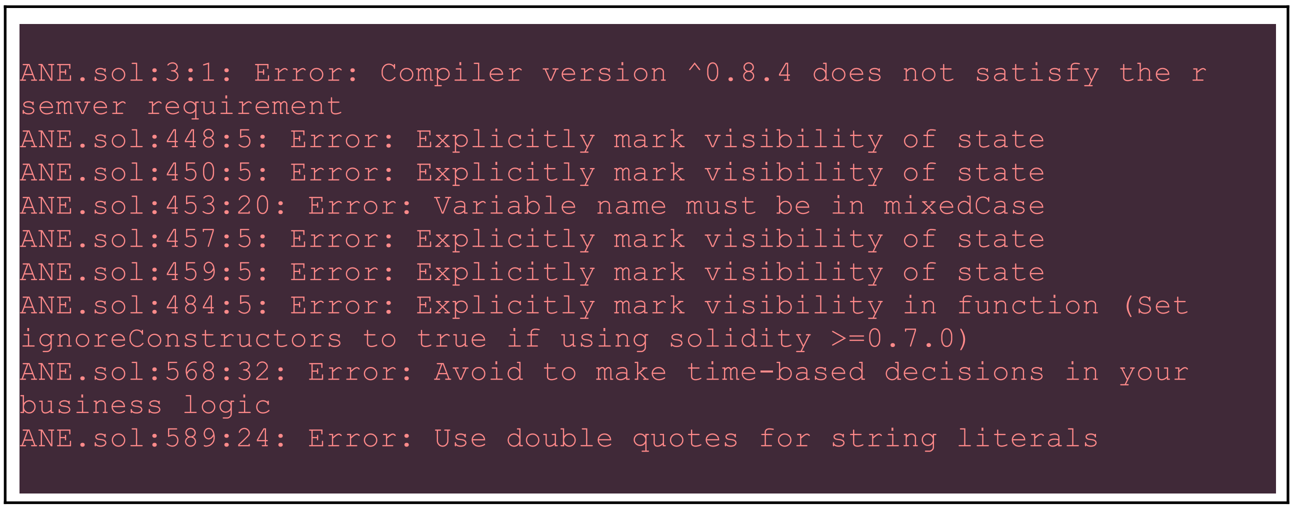
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 647:22:

ACE.sol



```
ACE.sol:3:1: Error: Compiler version ^0.8.4 does not satisfy the r
semver requirement
ACE.sol:77:5: Error: Explicitly mark visibility of state
ACE.sol:78:5: Error: Explicitly mark visibility of state
ACE.sol:79:5: Error: Explicitly mark visibility of state
ACE.sol:80:5: Error: Explicitly mark visibility of state
ACE.sol:81:5: Error: Explicitly mark visibility of state
ACE.sol:83:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
ACE.sol:89:32: Error: Code contains empty blocks
ACE.sol:119:5: Error: Explicitly mark visibility of state
ACE.sol:121:5: Error: Explicitly mark visibility of state
ACE.sol:123:5: Error: Explicitly mark visibility of state
ACE.sol:125:20: Error: Variable name must be in mixedCase
ACE.sol:130:5: Error: Explicitly mark visibility of state
ACE.sol:132:5: Error: Explicitly mark visibility of state
ACE.sol:180:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
ACE.sol:344:40: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:350:45: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:356:50: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:360:32: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:427:18: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:432:18: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:464:27: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:632:46: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:636:48: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:684:24: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:703:24: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:753:12: Error: Avoid to make time-based decisions in your
business logic
ACE.sol:770:23: Error: Avoid to make time-based decisions in your
business logic
```

ANE.sol



```
ANE.sol:3:1: Error: Compiler version ^0.8.4 does not satisfy the r
semver requirement
ANE.sol:448:5: Error: Explicitly mark visibility of state
ANE.sol:450:5: Error: Explicitly mark visibility of state
ANE.sol:453:20: Error: Variable name must be in mixedCase
ANE.sol:457:5: Error: Explicitly mark visibility of state
ANE.sol:459:5: Error: Explicitly mark visibility of state
ANE.sol:484:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
ANE.sol:568:32: Error: Avoid to make time-based decisions in your
business logic
ANE.sol:589:24: Error: Use double quotes for string literals
```

Overall Software analysis result:

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

EtherAuthority