Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Undoomed Protocol
Website: https://undoomed.space/
Platform: Astar Network
Language: Solidity
Date: April 30th, 2022

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by Undoomed to perform the Security audit of the Undoomed 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 April 30th, 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

The Undoomed Contracts have functions like _setInitializedVersion, getAdventures, adventure, redeemCoin, mint, addMinter, _exists, recycle, tokenByIndex, etc. The Undoomed Initializable 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 Undoomed Protocol Smart Contracts
Platform	**Astar / Solidity**
File 1	Adventure.sol
File 1 MD5 Hash	83D6C474603466B15CF525BF6B77BBE0
File 2	Building.sol
File 2 MD5 Hash	B34ED2C79065D047A296E170E477A044
File 3	CroesusToken.sol
File 3 MD5 Hash	B959509CB6B52D240F535E404B03C902
File 4	CrystalToken.sol
File 4 MD5 Hash	E550AFEF2D1808D9C7570BD0A6FF3CE6
File 5	ERC721.sol
File 5 MD5 Hash	BDEBC24FE78668F84429B655540F88C4
File 6	Hero.sol
File 6 MD5 Hash	89B3C762E7E970D6D4F77CC163F70085
File 7	Item.sol
File 7 MD5 Hash	82FB9B34BDD5D161C3373AED81157616
File 8	heroCoupon.sol
File 8 MD5 Hash	1C198530A9E0786534A093E413B320E7
Audit Date	April 30th,2022

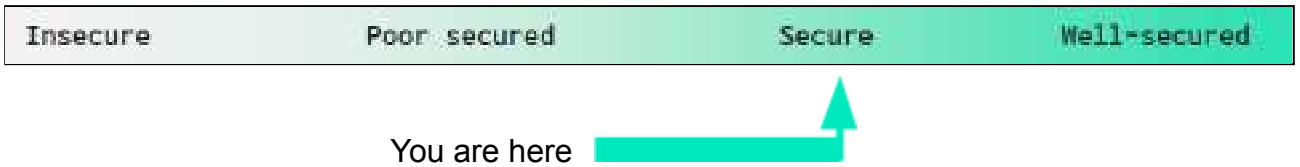
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Adventure.sol • USDT Decimals:6 • Maximum Points: 1 Billion • Adventure has functions like: getSummonersTotalPoints, getAdventures, etc.	**YES, This is valid.**
File 2 Building.sol • DAY: 1 days • Half Award Duration:15 • Building has functions like: getPledgeInfo, updateWallet, etc.	**YES, This is valid.**
File 3 CroesusToken.sol • Name: CROESUS • Symbol: COSS • Decimals: 18 • Maximum Supply: 21 Million	**YES, This is valid.**
File 4 CrystalToken.sol • Name: CRYSTAL-UNDOOMED • Symbol: CRYSTAL • Decimals: 18	**YES, This is valid.**
File 5 ERC721.sol • ERC721 has functions like: balanceOf, ownerOf, approve, getApproved, etc.	**YES, This is valid.**
File 6 Hero.sol • Name: Undoomed-Hero • Symbol: UDH • USDT Decimals: 6	**YES, This is valid.**

File 7 Item.sol • Name: Undoomed-Item • Symbol: UDI • USDT Decimals: 6	**YES, This is valid.**
File 8 HeroCoupon.sol • Name: HeroCoupon • Symbol: HeroCoupon • Maximum Supply: 20,000	**YES, This is valid.**

`Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. Also, these contracts do contain owner control, which does not make them fully decentralized.



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

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

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 8 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 Undoomed 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 Undoomed Protocol.

The Undoomed 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 **not** well commented on smart contracts.

Documentation

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

As mentioned above, code parts are **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://undoomed.space/ 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

Adventure.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	_setInitializedVersion	write	Passed	No Issue
7	initialize	write	Passed	No Issue
8	updateWallet	external	access only Owner	No Issue
9	getSummonersTotalPoints	external	Passed	No Issue
10	getAdventures	external	Passed	No Issue
11	getSummonersLastAdventure	external	Passed	No Issue
12	adventure	external	Infinite loops possibility	Refer Audit Findings
13	_armyLevelById	internal	Passed	No Issue
14	updateAdventureResult	external	Passed	No Issue
15	_openItem	write	Passed	No Issue
16	_updateRankingList	write	Passed	No Issue
17	getDailyRankingList	external	Passed	No Issue
18	getWeeklyRankingList	external	Passed	No Issue
19	publishDailyRanking	external	Passed	No Issue
20	publishWeeklyRanking	external	Passed	No Issue
21	recieveDailyRankingUsdtAward	external	Passed	No Issue
22	recieveWeeklyRankingUsdtAward	external	Passed	No Issue
23	_arryaPush	write	Passed	No Issue
24	queryAwardCroesus	read	Passed	No Issue
25	getAwardCroesusHistory	external	Passed	No Issue
26	_calcPageInfo	write	Passed	No Issue
27	recieveCroesus	external	Passed	No Issue
28	addCoresusMinter	external	access only Owner	No Issue
29	updateItemAddress	external	access only Owner	No Issue
30	removeCoresusMinter	external	access only Owner	No Issue
31	addRankingPublisher	external	access only Owner	No Issue
32	removeRankingPublisher	external	access only Owner	No Issue
33	mintCroesus	external	Passed	No Issue
34	_isCoresusMinter	read	Passed	No Issue
35	_isRankingPublisher	read	Passed	No Issue

36	addAllowedUpdateAdventureAddresses	external	access only Owner	No Issue
37	removeAllowedUpdateAdventureAddresses	external	access only Owner	No Issue
38	_isCallFromAllowedUpdateAdventureAddresses	read	Passed	No Issue
39	_isApprovedOrOwnerOfSummoner	read	Passed	No Issue
40	_arrayContains	write	Passed	No Issue
41	_addressesContains	write	Passed	No Issue
42	_addressArrayDelete	internal	Passed	No Issue
43	_getTimestampDay	write	Passed	No Issue
44	_getTimestampWeek	write	Passed	No Issue
45	onlyOwner	modifier	Passed	No Issue

Building.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	_setInitializedVersion	write	Passed	No Issue
7	initialize	write	Passed	No Issue
8	getPledgeInfo	external	Passed	No Issue
9	updateBuildingConfigAddress	external	access only Owner	No Issue
10	updateWallet	external	access only Owner	No Issue
11	init	external	Passed	No Issue
12	create	external	Passed	No Issue
13	_consumeCroesus	write	Passed	No Issue
14	getArmyBuildingProduceRateAndStorageLimit	external	Passed	No Issue
15	_calcArmyBuildingProduceRateAndStorageLimit	read	Passed	No Issue
16	redeemCoin	write	Passed	No Issue
17	multiRedeemCoin	external	Passed	No Issue
18	multiReceiveCrystal	external	Passed	No Issue
19	_prepareRecieveCrystal	write	Passed	No Issue
20	changePledgeCoin	write	Passed	No Issue
21	multiChangePledgeCoin	external	Passed	No Issue
22	_calculateCurrentAward	internal	Passed	No Issue
23	getCurrentAwards	read	Passed	No Issue
24	getCurrentAward	read	Passed	No Issue
25	_calculateAwardBaseRate	internal	Passed	No Issue

26	_getPledgeRate	internal	Passed	No Issue
27	pledgeHero	write	Passed	No Issue
28	multiPledgeHero	external	Passed	No Issue
29	redeemHero	write	Passed	No Issue
30	multiRedeemHero	external	Passed	No Issue
31	getOwnedBuildings	read	Passed	No Issue
32	getOwnedBuildingCount	external	Passed	No Issue
33	_isApprovedOrOwnerOfSummoner	internal	Passed	No Issue
34	_arrayDelete	internal	Passed	No Issue
35	_arrayContains	write	Passed	No Issue
36	onlyOwner	modifier	Passed	No Issue

CroesusToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_mint	internal	Passed	No Issue
7	_transfer	internal	Passed	No Issue
8	delegate	external	Passed	No Issue
9	delegateBySig	external	Passed	No Issue
10	getCurrentVotes	external	Passed	No Issue
11	getPriorVotes	external	Passed	No Issue
12	_delegate	internal	Passed	No Issue
13	_moveDelegates	internal	Passed	No Issue
14	_writeCheckpoint	internal	Passed	No Issue
15	safe32	internal	Passed	No Issue
16	getChainId	internal	Passed	No Issue
17	mint	write	access only Minter	No Issue
18	addMinter	write	access only Owner	No Issue
19	delMinter	write	access only Owner	No Issue
20	getMinterLength	read	Passed	No Issue
21	isMinter	read	Passed	No Issue
22	getMinter	read	access only Owner	No Issue
23	onlyMinter	modifier	Passed	No Issue

CrystalToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_mint	internal	Passed	No Issue
7	_transfer	internal	Passed	No Issue
8	delegate	external	Passed	No Issue
9	delegateBySig	external	Passed	No Issue
10	getCurrentVotes	external	Passed	No Issue
11	getPriorVotes	external	Passed	No Issue
12	_delegate	internal	Passed	No Issue
13	_moveDelegates	internal	Passed	No Issue
14	_writeCheckpoint	internal	Passed	No Issue
15	safe32	internal	Passed	No Issue
16	getChainId	internal	Passed	No Issue
17	mint	write	access only Minter	No Issue
18	addMinter	write	access only Owner	No Issue
19	delMinter	write	access only Owner	No Issue
20	getMinterLength	read	Passed	No Issue
21	isMinter	read	Passed	No Issue
22	getMinter	read	access only Owner	No Issue
23	onlyMinter	modifier	Passed	No Issue

ERC721.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	balanceOf	read	Passed	No Issue
3	ownerOf	read	Passed	No Issue
4	_baseURI	internal	Passed	No Issue
5	approve	write	Passed	No Issue
6	getApproved	read	Passed	No Issue
7	setApprovalForAll	write	Passed	No Issue
8	_isContract	internal	Passed	No Issue
9	isApprovedForAll	read	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	safeTransferFrom	write	Passed	No Issue
12	_safeTransfer	internal	Passed	No Issue
13	_exists	internal	Passed	No Issue
14	_isApprovedOrOwner	internal	Passed	No Issue

15	_safeMint	internal	Passed	No Issue
16	_safeMint	internal	Passed	No Issue
17	_mint	internal	Passed	No Issue
18	_burn	internal	Passed	No Issue
19	_transfer	internal	Passed	No Issue
20	_approve	internal	Passed	No Issue
21	_checkOnERC721Received	write	Passed	No Issue
22	_beforeTokenTransfer	internal	Passed	No Issue

Hero.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	_setInitializedVersion	write	Passed	No Issue
7	getOwnerdTokens	read	Passed	No Issue
8	getOwnedTokensByAddress	read	Passed	No Issue
9	tokenOfOwnerByIndex	read	Passed	No Issue
10	totalSupply	read	Passed	No Issue
11	tokenByIndex	read	Passed	No Issue
12	_beforeTokenTransfer	internal	Passed	No Issue
13	_addTokenToOwnerEnumeration	write	Passed	No Issue
14	_addTokenToAllTokensEnumeration	write	Passed	No Issue
15	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
16	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
17	supportsInterface	external	Passed	No Issue
18	initialize	write	Passed	No Issue
19	mergeSummoners	external	Passed	No Issue
20	setSummonerName	external	Passed	No Issue
21	recycle	external	Passed	No Issue
22	isWorking	read	Passed	No Issue
23	level_up	external	Passed	No Issue
24	_consumeCroesus	write	Passed	No Issue
25	summoners	external	Passed	No Issue
26	abilityScores	external	Passed	No Issue
27	summon	external	Passed	No Issue
28	summonByCoupon	external	Passed	No Issue

29	_summonRoll	write	Passed	No Issue
30	_summon	write	Passed	No Issue
31	_roll	write	Passed	No Issue
32	_getInitAbility	write	Passed	No Issue
33	_randomBetween	write	Passed	No Issue
34	setPledged	external	Passed	No Issue
35	setAdventuring	external	Passed	No Issue
36	addAllowedPledgeFromAddress	external	access only Owner	No Issue
37	removeAllowedPledgeFromAddress	external	access only Owner	No Issue
38	updateWallet	external	access only Owner	No Issue
39	_isCallFromAllowedPledgeFromAddresses	write	Passed	No Issue
40	addAllowedUpdateAdventuringAddresses	external	access only Owner	No Issue
41	removeAllowedUpdateAdventuringAddresses	external	access only Owner	No Issue
42	_isCallFromAllowedUpdateAdventuringAddresses	read	Passed	No Issue
43	_random	write	Passed	No Issue
44	_beforeTokenTransfer	internal	Passed	No Issue
45	setItemAddress	external	access only Owner	No Issue
46	_addressArrayDelete	internal	Passed	No Issue
47	_addressesContains	write	Passed	No Issue
48	tokenURI	external	Passed	No Issue
49	toBytes	internal	Passed	No Issue
50	onlyOwner	modifier	Passed	No Issue

Item.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	_setInitializedVersion	write	Passed	No Issue
7	getOwnerdTokens	read	Passed	No Issue
8	getOwnedTokensByAddress	read	Passed	No Issue
9	tokenOfOwnerByIndex	read	Passed	No Issue
10	totalSupply	read	Passed	No Issue
11	tokenByIndex	read	Passed	No Issue
12	_beforeTokenTransfer	internal	Passed	No Issue

13	_addTokenToOwnerEnumeration	write	Passed	No Issue
14	_addTokenToAllTokensEnumeration	write	Passed	No Issue
15	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
16	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
17	supportsInterface	external	Passed	No Issue
18	initialize	write	Passed	No Issue
19	mergeItems	external	Passed	No Issue
20	_consumeCroesus	write	Passed	No Issue
21	updateCroesus	external	access only Owner	No Issue
22	updateWallet	external	access only Owner	No Issue
23	open	external	Passed	No Issue
24	_open	write	Passed	No Issue
25	_mintItem	write	Passed	No Issue
26	openItemRandom	write	Passed	No Issue
27	recycle	external	Passed	No Issue
28	items	external	Passed	No Issue
29	summonersWithItems	read	Passed	No Issue
30	getSummonersWeardItems	read	Passed	No Issue
31	getWearingItems	read	Passed	No Issue
32	isUsing	read	Passed	No Issue
33	_removeItem	write	Passed	No Issue
34	_wearItem	write	Passed	No Issue
35	wearItems	external	Infinite loops possibility	Refer Audit Findings
36	removeAllExpiredItems	external	Passed	No Issue
37	clearWearingItems	external	Passed	No Issue
38	_clearWearingItems	write	Passed	No Issue
39	addAllowedOpenItemAddress	external	access only Owner	No Issue
40	removeAllowedOpenItemAddress	external	access only Owner	No Issue
41	_isCallFromAllowedOpenItemAddresses	read	Passed	No Issue
42	_arrayFirstZeroIndex	write	Passed	No Issue
43	_arrayContains	write	Passed	No Issue
44	_roll	write	Passed	No Issue
45	_beforeTokenTransfer	internal	Passed	No Issue
46	_isApprovedOrOwnerOfSummoner	internal	Passed	No Issue
47	_isApprovedOrOwnerOfItem	internal	Passed	No Issue
48	_random	write	Passed	No Issue
49	_addressesContains	write	Passed	No Issue

50	_addressArrayDelete	internal	Passed	No Issue
51	tokenURI	read	Passed	No Issue
52	toBytes	internal	Passed	No Issue
53	onlyOwner	modifier	Passed	No Issue

HeroCoupon.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	mint	write	access only Minter	No Issue
7	transfer	write	access only Owner	No Issue
8	transferFrom	write	access only Hero Contract	No Issue
9	setHeroAddress	write	access only Owner	No Issue
10	addMinter	write	access only Owner	No Issue
11	delMinter	write	access only Owner	No Issue
12	getMinterLength	read	Passed	No Issue
13	isMinter	read	Passed	No Issue
14	getMinter	read	access only Owner	No Issue
15	onlyMinter	modifier	Passed	No Issue
16	onlyHeroContract	modifier	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.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

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

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

Adventure.sol

- adventure() - _armys.length

Item.sol

- wearItems() - _excluded.length.

Very Low / Informational / Best practices:

(1) Unused struct: **Building.sol**

```
struct BuildingPledgeSummoners {
    address owner;
    uint256 buildingId;
    uint256[] summoners;
}
```

```
struct EconomyRecord {
    uint256 buildingId;
    uint256 pledgeTimestamp;
    uint256 lastRecieveAwardTimestamp;
    uint256 startTimestamp;
    uint256 endTimestamp;
    uint256 award;
    uint256 currentTimestamp;
    uint256 pledgeCoinRuleChangeTimestamp;
    uint256 pledgeWeeks;
    uint256 pledgeCoins;
}
```

structs are defined but not used in code:

1. EconomyRecord
2. BuildingPledgeSummoners

Resolution: We suggest removing unused struct.

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:

- updateCroesus: Item owner can update croesus address.
- updateWallet: Item owner can update address.
- addAllowedOpenItemAddress: Item owner can add allowed open item address.
- removeAllowedOpenItemAddress: Item owner can remove allowed open item address.
- updateWallet: Adventure owner can update address.
- addCoresusMinter: Adventure owner can add coresus minter address.
- updateItemAddress: Adventure owner can update item address.
- removeCoresusMinter: Adventure owner can remove coresus minter address.
- addRankingPublisher: Adventure owner can add ranking publisher address.
- removeRankingPublisher: Adventure owner can remove ranking publisher address.
- addAllowedUpdateAdventureAddresses: Adventure owner can add allowed update adventure addresses.
- removeAllowedUpdateAdventureAddresses: Adventure owners can remove allowed update adventure addresses.
- updateBuildingConfigAddress: Building Owner can update config address.
- updateWallet: Building Owner can update wallet address.
- addMinter: CroesusToken owner can add minter address.
- delMinter: CroesusToken owner can remove minter address.
- getMinter: CroesusToken owner can get minter address.
- addMinter: CrystalToken owner can add minter address.
- delMinter: CrystalToken owner can remove minter address.
- getMinter: CrystalToken owner can get minter address.
- addAllowedPledgeFromAddress: Hero owner can add allowed pledge from address.
- removeAllowedPledgeFromAddress: Hero owner can remove allowed pledge from address.
- updateWallet: Hero owner can update wallet address.

- addAllowedUpdateAdventuringAddresses: Hero owner can add allowed update adventure addresses.
- removeAllowedUpdateAdventuringAddresses: Hero owner can remove allowed update adventure addresses.
- setItemAddress: Hero owner can set item address.
- getMinter: HeroCoupon owner can get minter address.
- delMinter: HeroCoupon owner can remove minter address.
- addMinter: HeroCoupon owner can add minter address.
- setHeroAddress: HeroCoupon owner can set hero address.
- transfer: HeroCoupon owner can transfer amount.

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, but they were resolved in the revised smart contract code. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

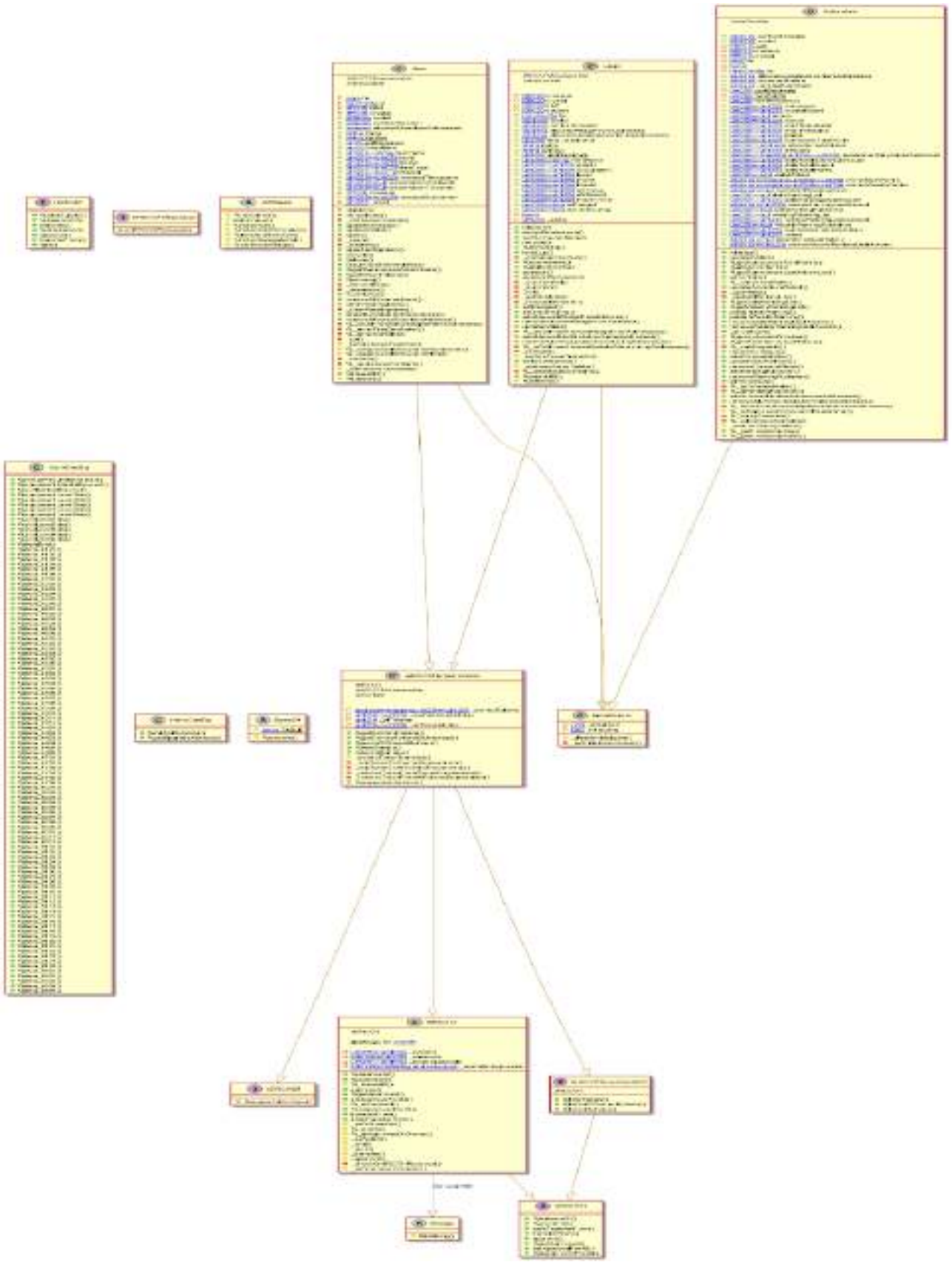
Technical Disclaimer

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

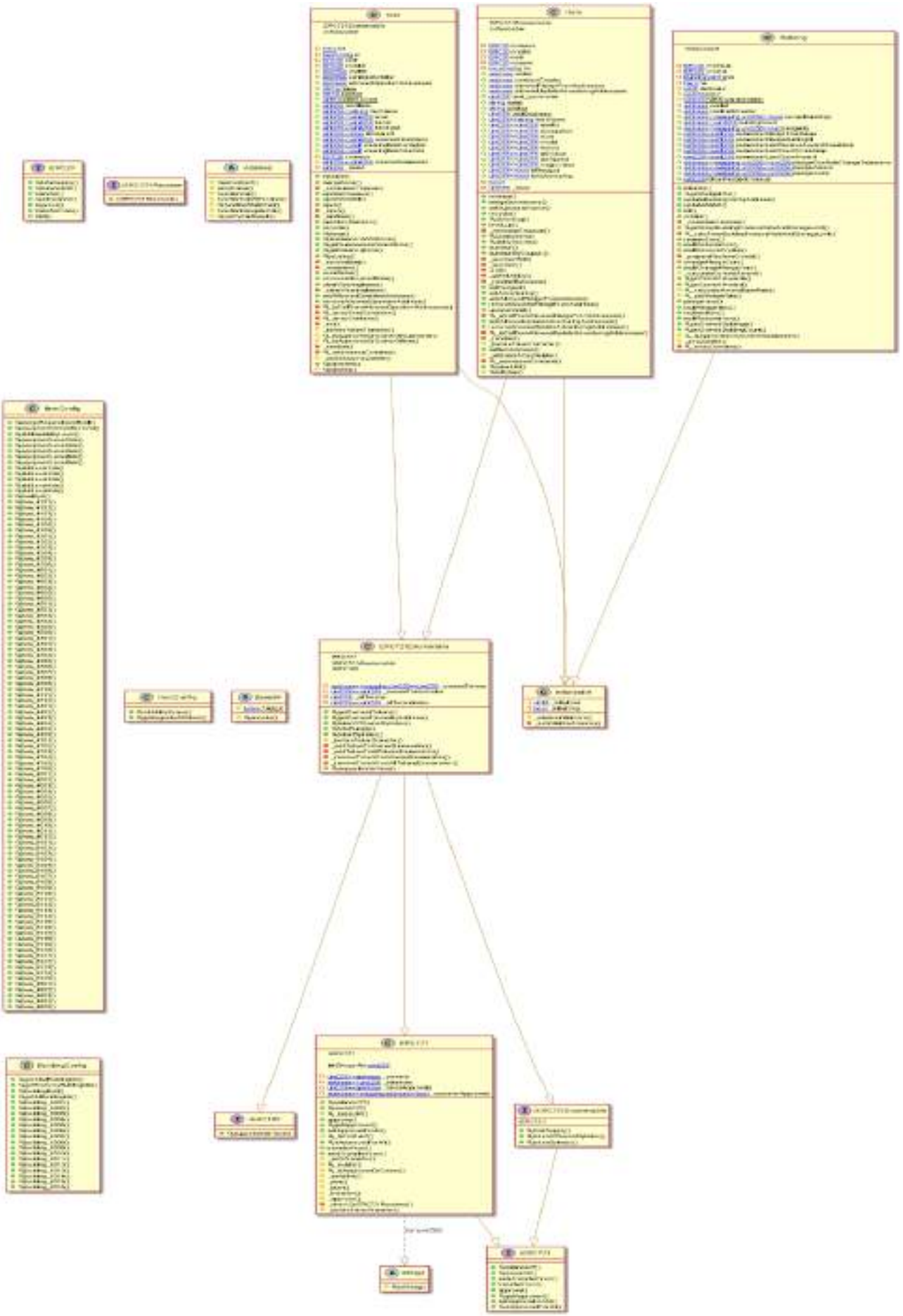
Appendix

Code Flow Diagram - Undoomed Protocol

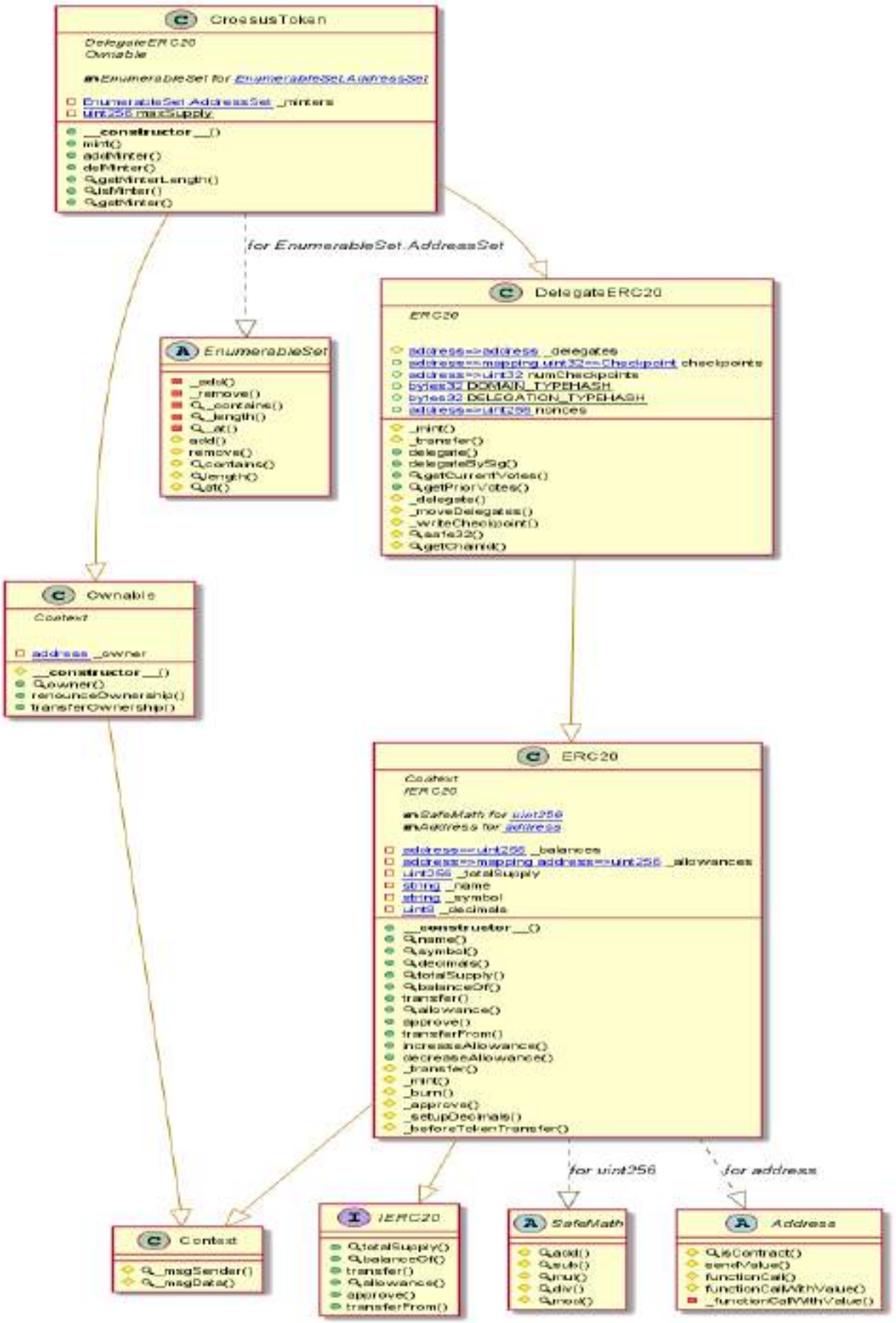
Adventure Diagram



Building Diagram

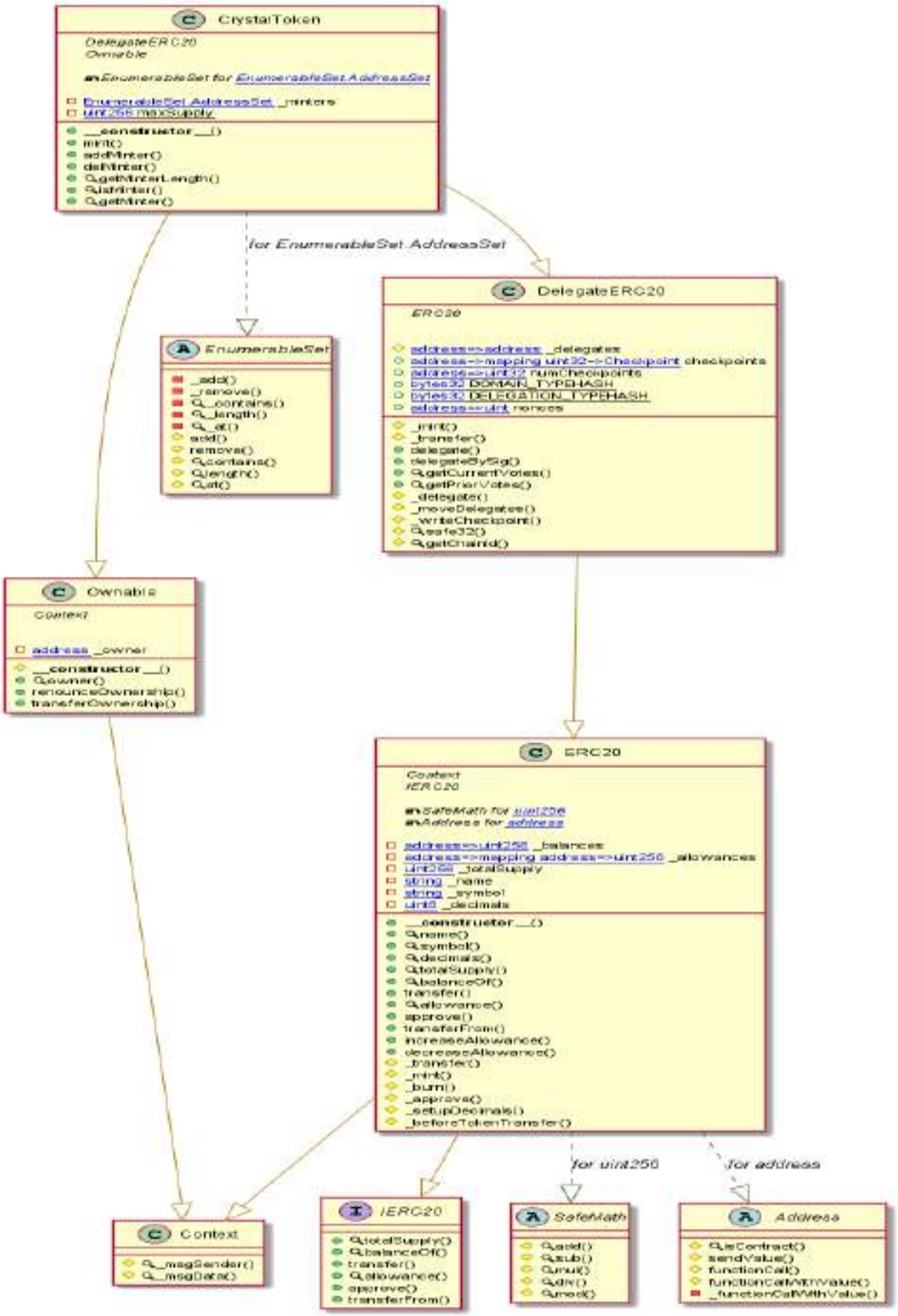


CroesusToken Diagram

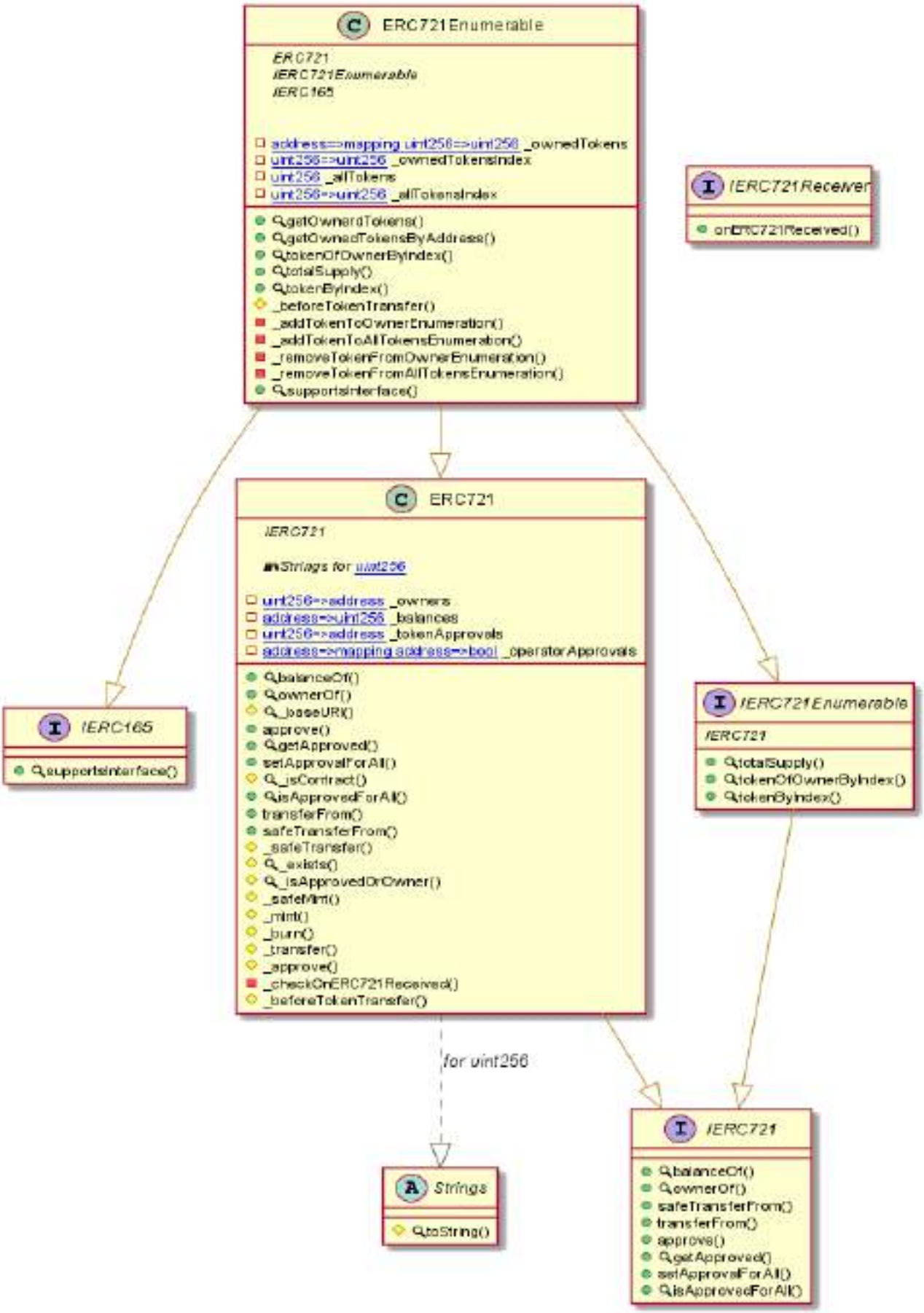


Email: audit@EtherAuthority.io

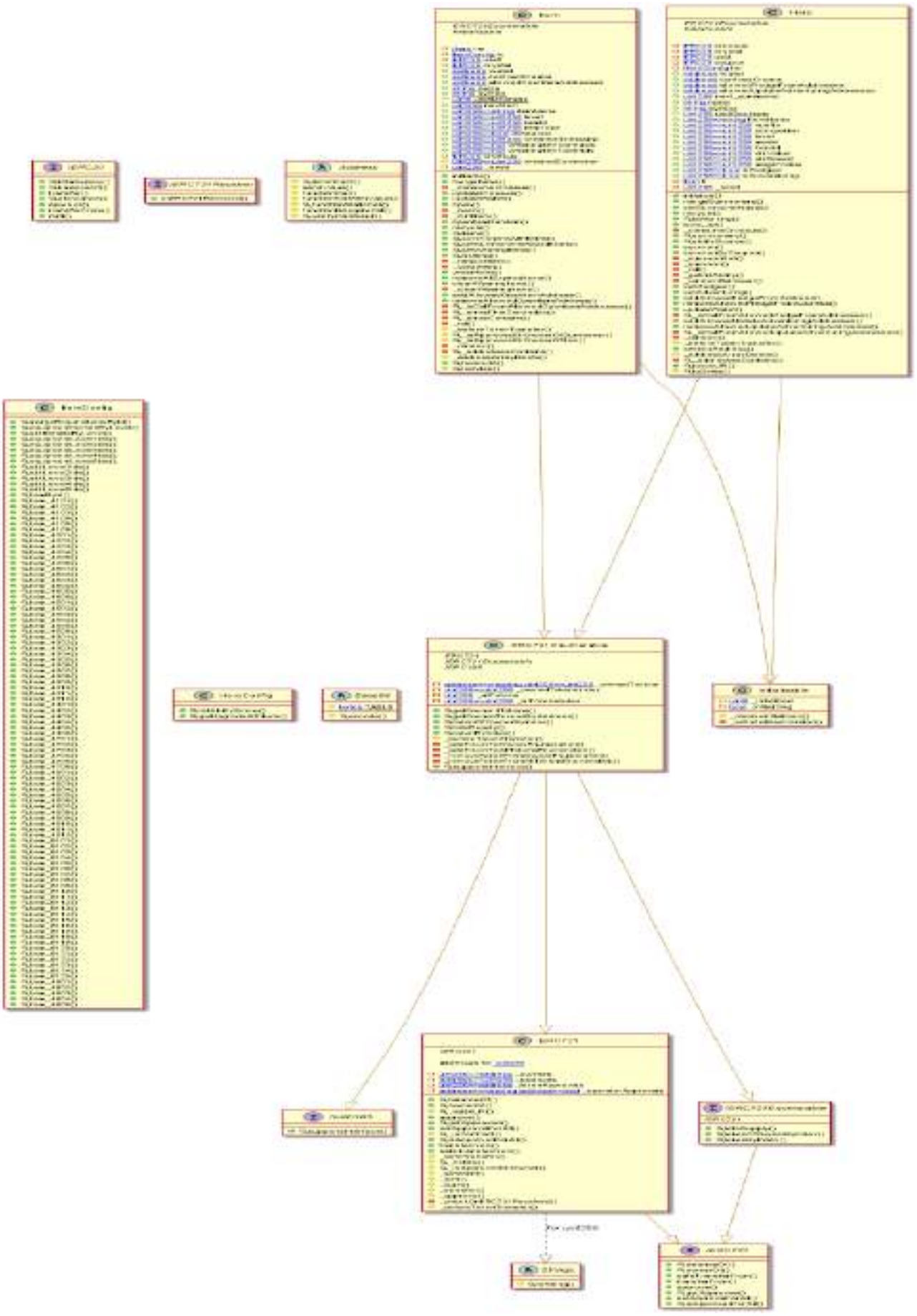
CrystalToken Diagram



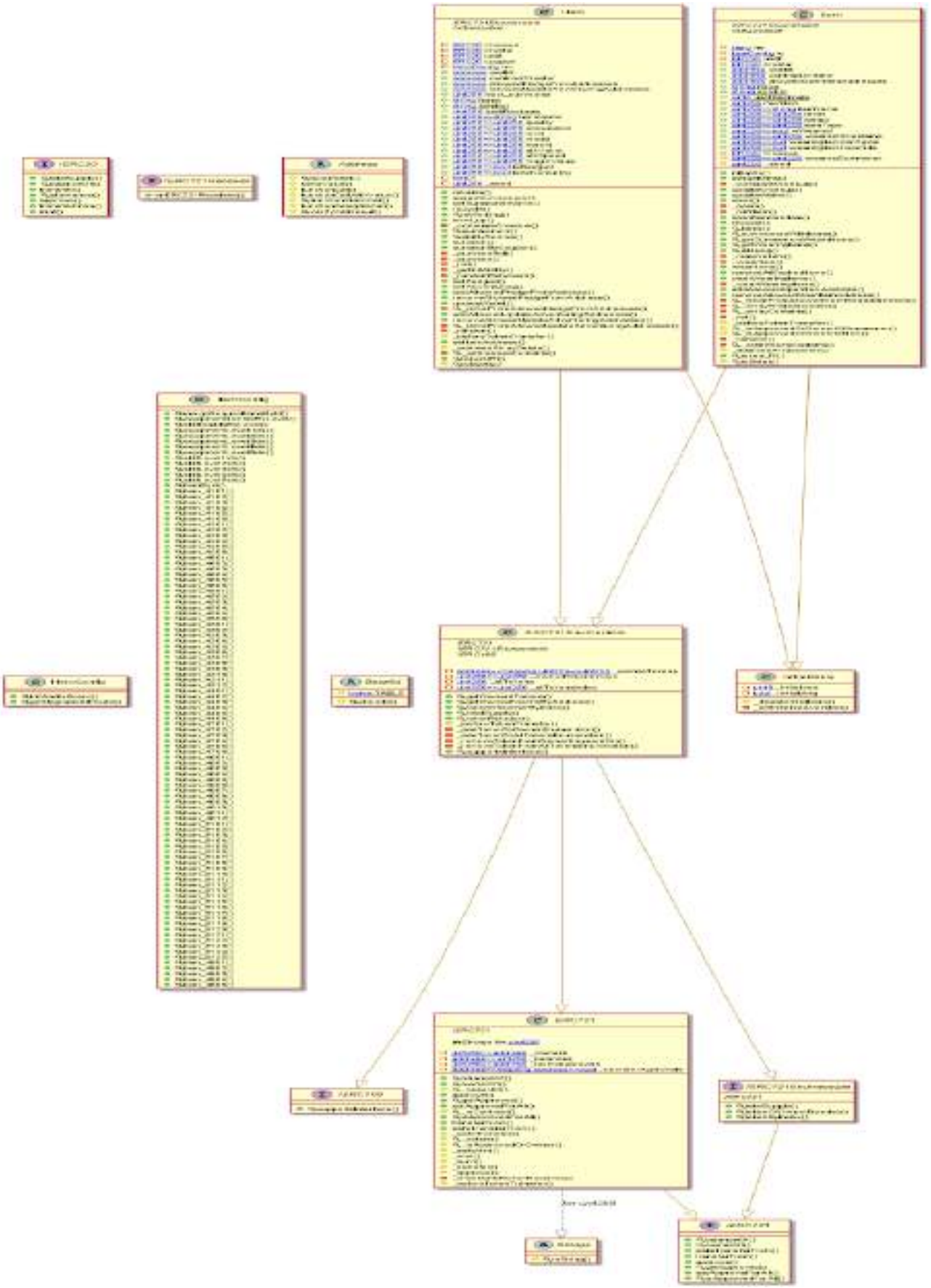
ERC721 Diagram



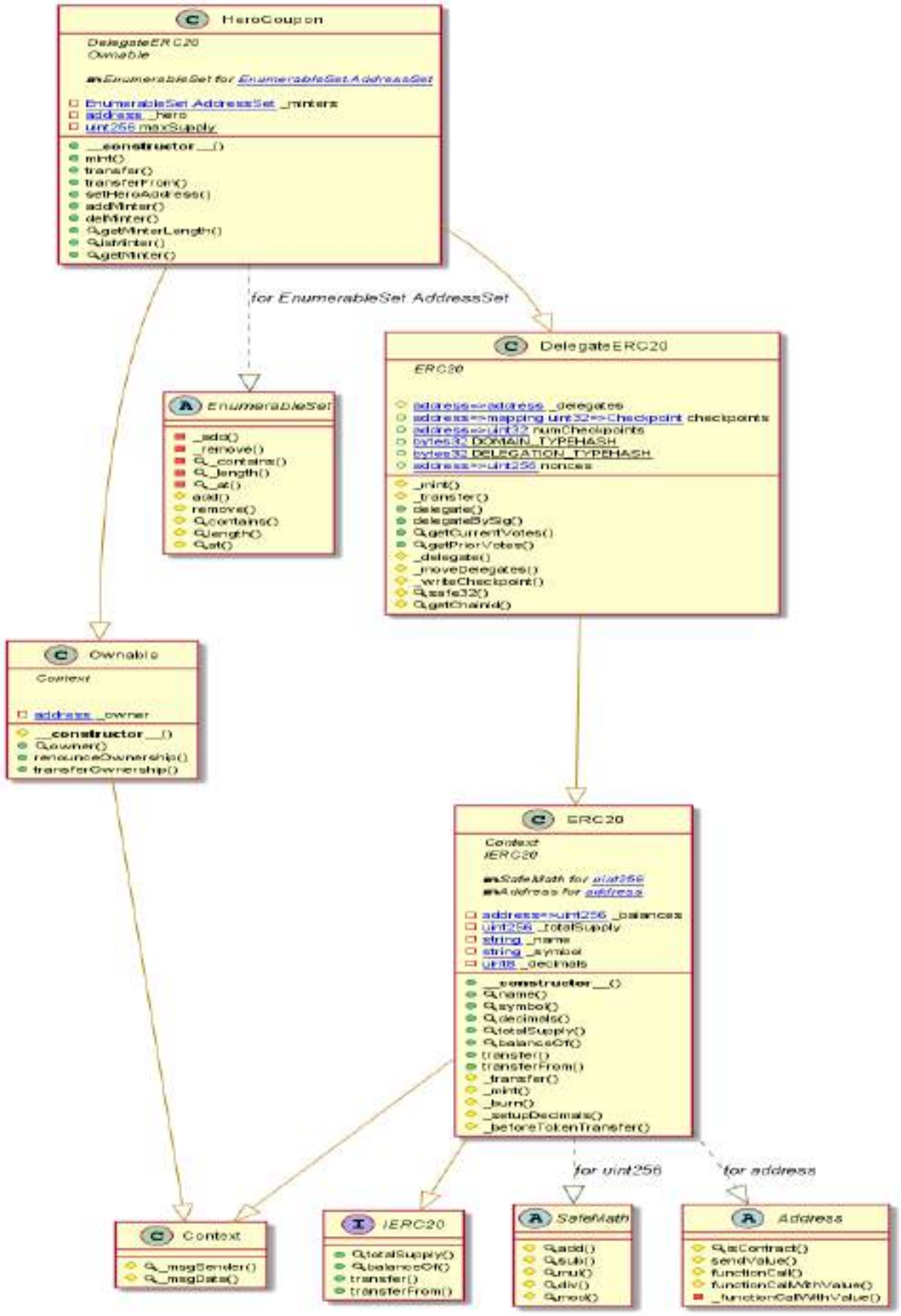
Hero Diagram



Item Diagram

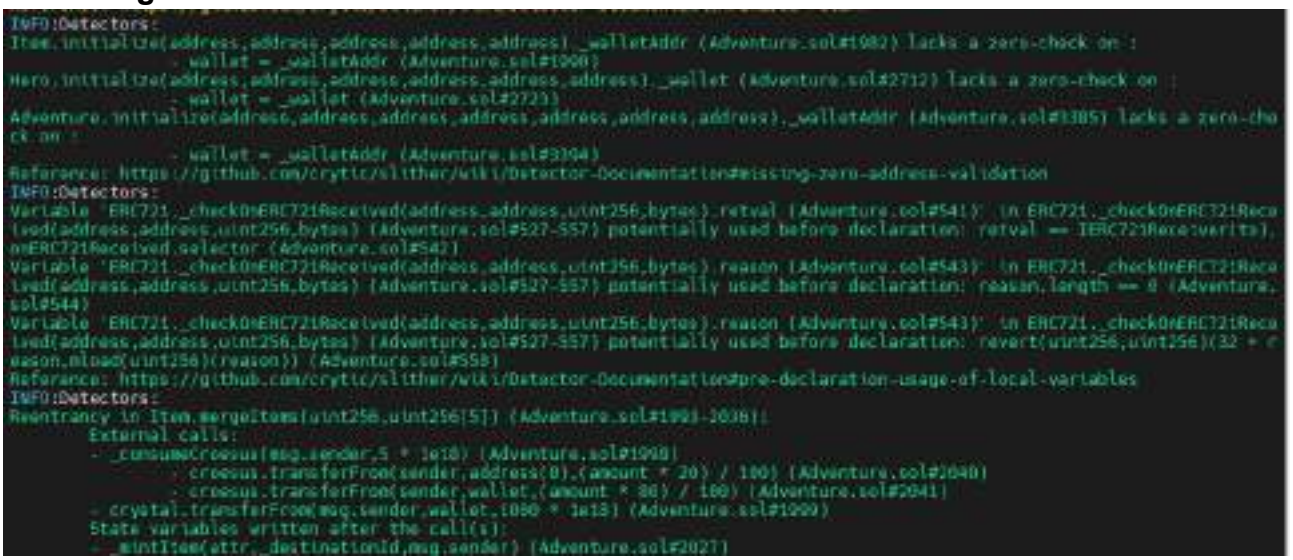


HeroCoupon Diagram

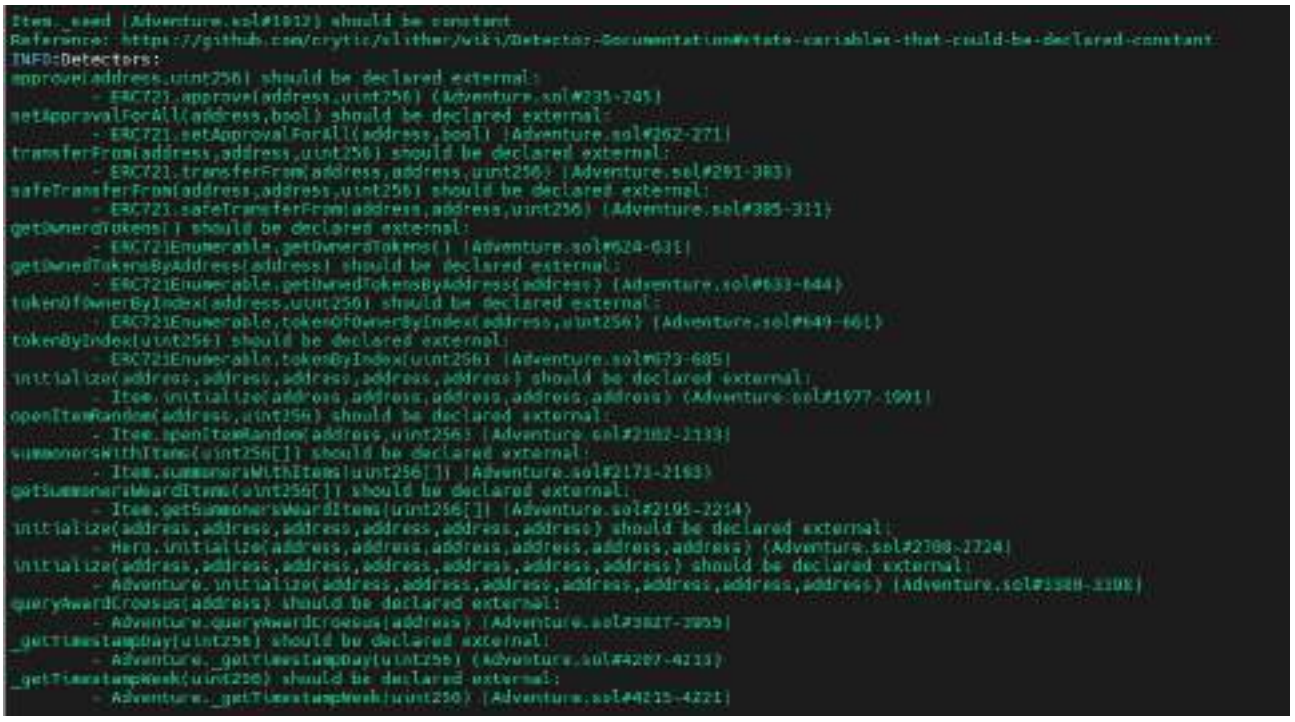


Slither Results Log

Slither log >> Adventure.sol



```
INFO:Detectors:
Item.initialize(address,address,address,address,address)._walletAddr (Adventure.sol#1982) lacks a zero-check on :
                - wallet = _walletAddr (Adventure.sol#1990)
Hero.initialize(address,address,address,address,address,address)._wallet (Adventure.sol#2712) lacks a zero-check on :
                - wallet = _wallet (Adventure.sol#2723)
Adventure.initialize(address,address,address,address,address,address,address)._walletAddr (Adventure.sol#3385) lacks a zero-che
ck on :
                - wallet = _walletAddr (Adventure.sol#3394)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (Adventure.sol#541)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (Adventure.sol#527-557) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (Adventure.sol#547)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Adventure.sol#543)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (Adventure.sol#527-557) potentially used before declaration: reason.length == 0 (Adventure.sol#544)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Adventure.sol#543)' in ERC721._checkOnERC721Rece
ived(address,address,uint256,bytes) (Adventure.sol#527-557) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (Adventure.sol#550)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in Item.mergeItems(uint256,uint256[5]) (Adventure.sol#1993-2036):
        External calls:
        - _consumeCroesus(msg.sender,5 * 1e18) (Adventure.sol#1998)
                - croesus.transferFrom(sender,address(0),(amount * 20) / 100) (Adventure.sol#2040)
                - croesus.transferFrom(sender,wallet,(amount * 80) / 100) (Adventure.sol#2041)
        - crystal.transferFrom(msg.sender,wallet,1000 * 1e18) (Adventure.sol#1999)
        State variables written after the call(s):
        - _mintItem(attr,_destinationId,msg.sender) (Adventure.sol#2027)
```

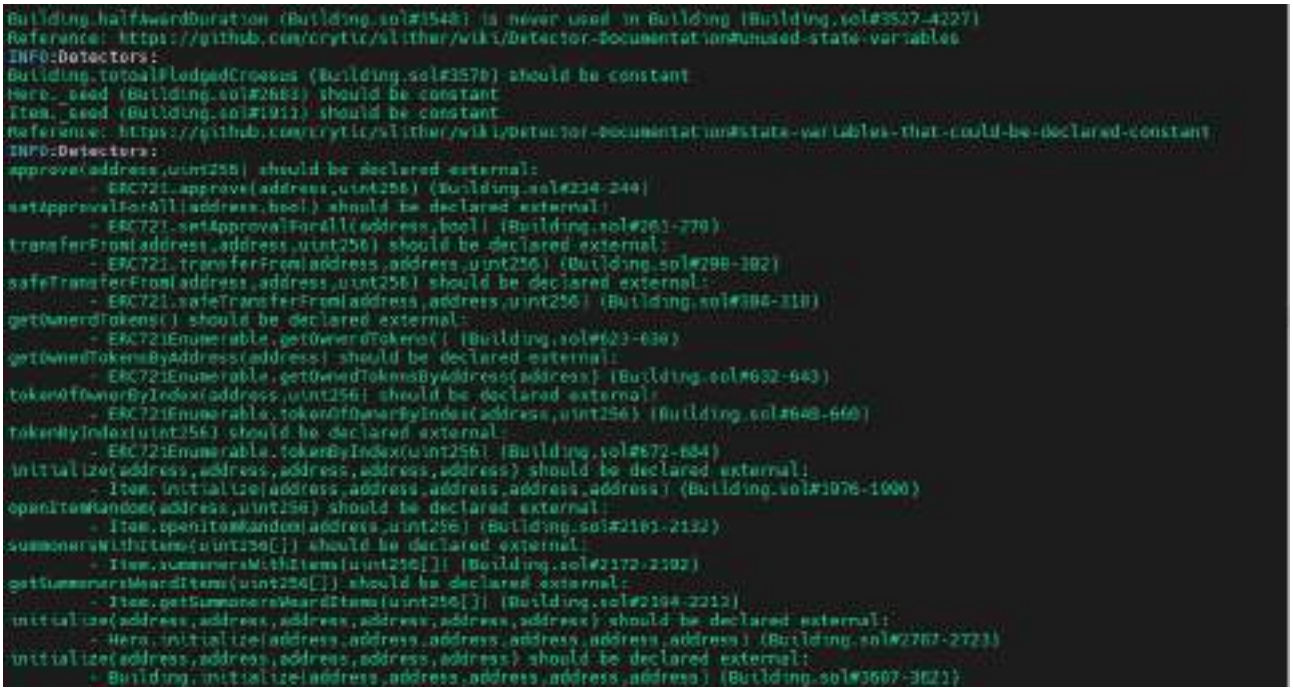


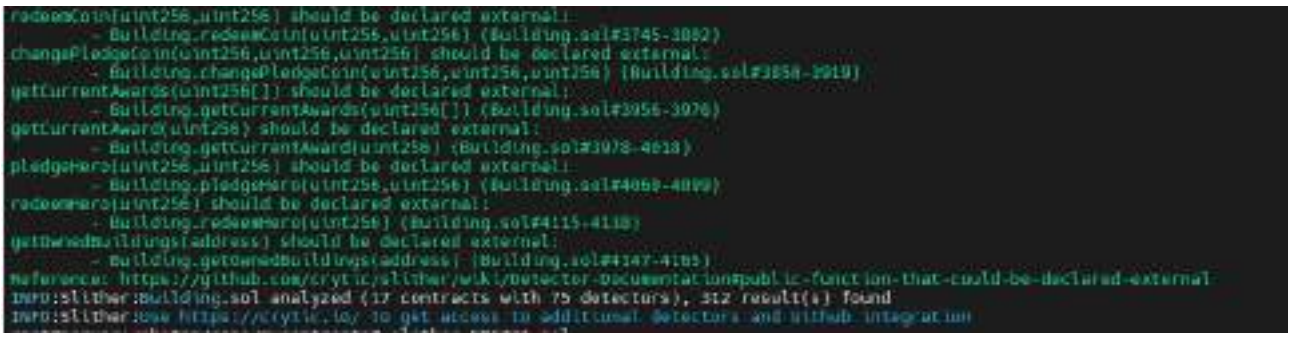
```
Item._seed (Adventure.sol#1813) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (Adventure.sol#235-245)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (Adventure.sol#262-271)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (Adventure.sol#291-303)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (Adventure.sol#305-311)
getOwnerTokens() should be declared external:
        - ERC721Enumerable.getOwnerTokens() (Adventure.sol#620-631)
getOwnedTokensByAddress(address) should be declared external:
        - ERC721Enumerable.getOwnedTokensByAddress(address) (Adventure.sol#633-644)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (Adventure.sol#649-661)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (Adventure.sol#673-685)
initialize(address,address,address,address,address,address) should be declared external:
        - Item.initialize(address,address,address,address,address) (Adventure.sol#1977-1991)
openItemRandom(address,uint256) should be declared external:
        - Item.openItemRandom(address,uint256) (Adventure.sol#2102-2133)
summonersWithItems(uint256[]) should be declared external:
        - Item.summonersWithItems(uint256[]) (Adventure.sol#2171-2183)
getSummonersWeardItems(uint256[]) should be declared external:
        - Item.getSummonersWeardItems(uint256[]) (Adventure.sol#2195-2214)
initialize(address,address,address,address,address,address,address) should be declared external:
        - Hero.initialize(address,address,address,address,address,address,address) (Adventure.sol#2708-2724)
initialize(address,address,address,address,address,address,address,address) should be declared external:
        - Adventure.initialize(address,address,address,address,address,address,address,address) (Adventure.sol#3380-3398)
queryAwardCroesus(address) should be declared external:
        - Adventure.queryAwardCroesus(address) (Adventure.sol#3827-3859)
_getTimestampDay(uint256) should be declared external:
        - Adventure._getTimestampDay(uint256) (Adventure.sol#4207-4213)
_getTimestampWeek(uint256) should be declared external:
        - Adventure._getTimestampWeek(uint256) (Adventure.sol#4215-4221)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Adventure.sol analyzed (16 contracts with 75 detectors), 303 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

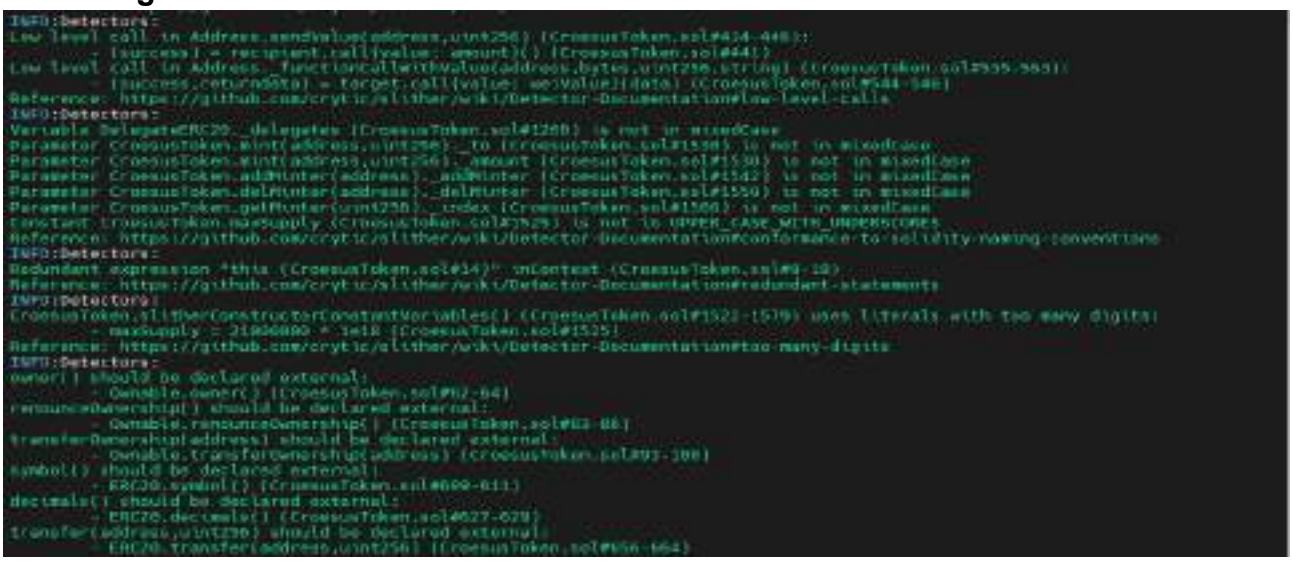
Slither log >> Building.sol

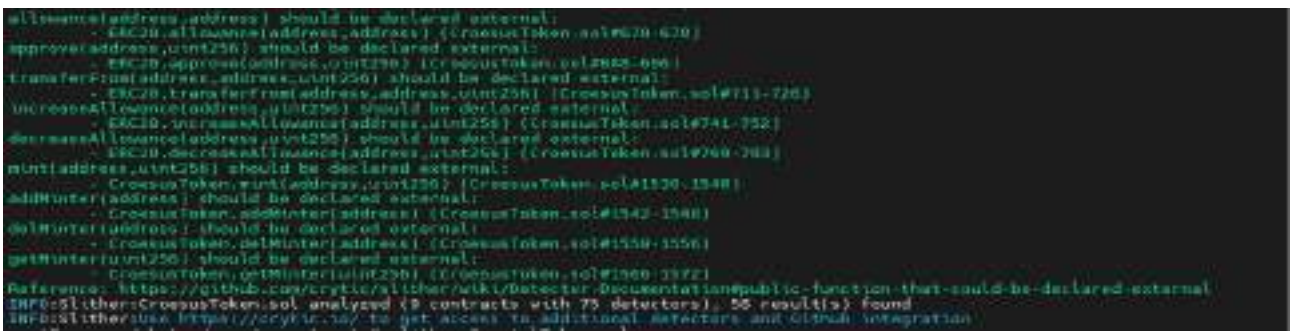
```
INFO:Detectors:
Item.initialize(address,address,address,address,address)._walletAddr (Building.sol#1881) lacks a zero-check on :
                - wallet = _walletAddr (Building.sol#1889)
Hero.initialize(address,address,address,address,address,address)._wallet (Building.sol#2711) lacks a zero-check on :
                - wallet = _wallet (Building.sol#2722)
Building.initialize(address,address,address,address,address)._walletAddr (Building.sol#3612) lacks a zero-check on :
                - wallet = _walletAddr (Building.sol#3619)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (Building.sol#540)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (Building.sol#526-556) potentially used before declaration: retval == IERC721Receiver(to).onERC721Received.selector (Building.sol#541)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Building.sol#543)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (Building.sol#526-556) potentially used before declaration: reason.length == 0 (Building.sol#543)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Building.sol#542)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (Building.sol#526-556) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (Building.sol#549)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```



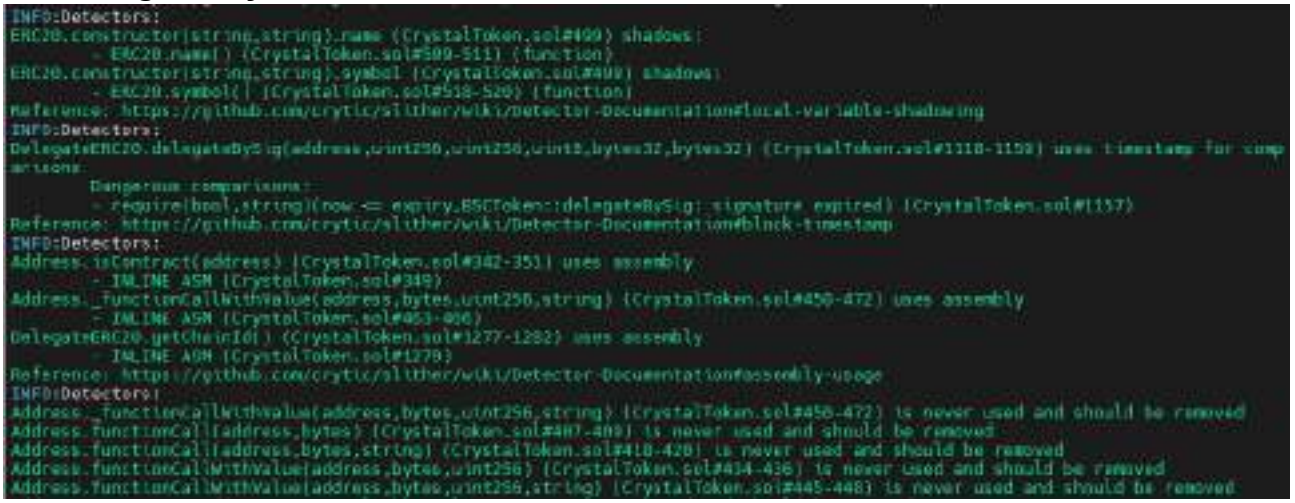


Slither log >> CroesusToken.sol

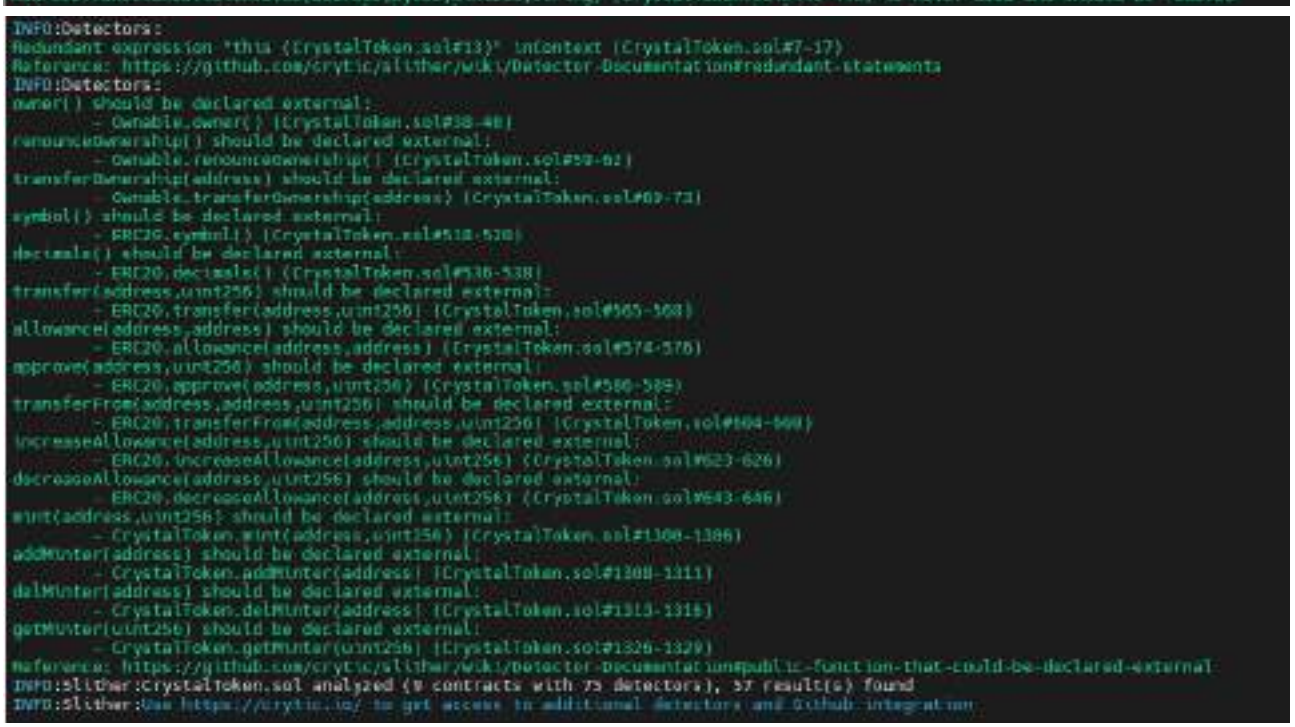




Slither log >> CrystalToken.sol



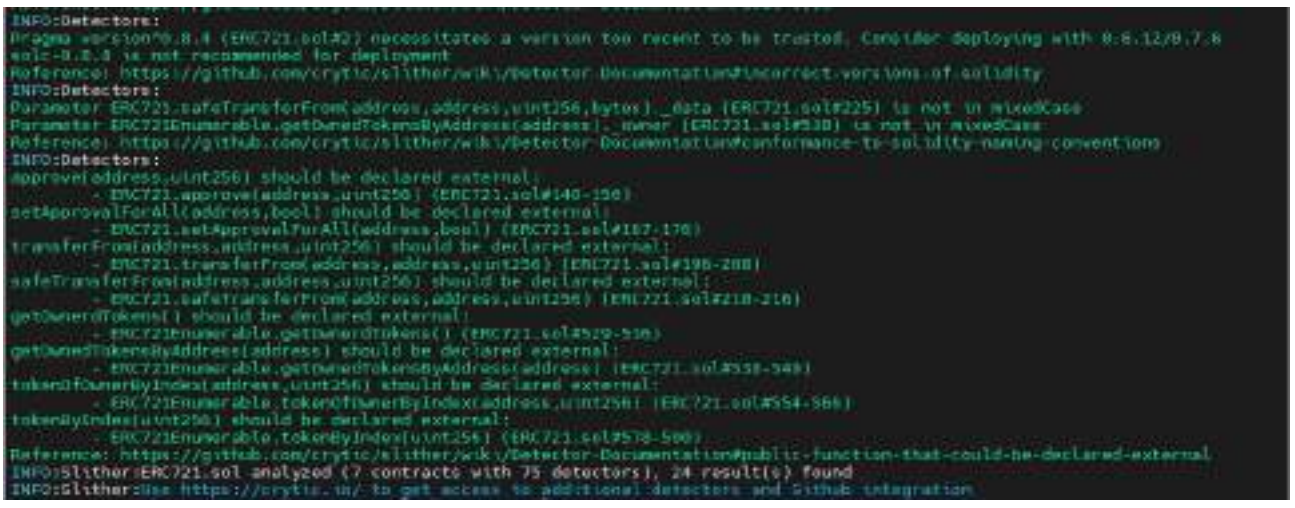
```
INFO:Detectors:
ERC20.constructor(string,string).name (CrystalToken.sol#409) shadows:
        - ERC20.name() (CrystalToken.sol#509-511) (function)
ERC20.constructor(string,string).symbol (CrystalToken.sol#409) shadows:
        - ERC20.symbol() (CrystalToken.sol#518-520) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
DelegateERC20.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (CrystalToken.sol#1118-1159) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,BSCToken::delegateBySig: signature expired) (CrystalToken.sol#1157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (CrystalToken.sol#342-351) uses assembly
        - INLINE ASM (CrystalToken.sol#349)
Address._functionCallWithValue(address,bytes,uint256,string) (CrystalToken.sol#450-472) uses assembly
        - INLINE ASM (CrystalToken.sol#463-466)
DelegateERC20.getChainId() (CrystalToken.sol#1277-1282) uses assembly
        - INLINE ASM (CrystalToken.sol#1279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (CrystalToken.sol#450-472) is never used and should be removed
Address.functionCall(address,bytes) (CrystalToken.sol#407-409) is never used and should be removed
Address.functionCall(address,bytes,string) (CrystalToken.sol#418-420) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (CrystalToken.sol#434-436) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (CrystalToken.sol#445-448) is never used and should be removed
```



```
INFO:Detectors:
Redundant expression "this (CrystalToken.sol#13)" inContext (CrystalToken.sol#7-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (CrystalToken.sol#38-40)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CrystalToken.sol#59-62)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CrystalToken.sol#69-73)
symbol() should be declared external:
        - ERC20.symbol() (CrystalToken.sol#518-520)
decimals() should be declared external:
        - ERC20.decimals() (CrystalToken.sol#536-538)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CrystalToken.sol#565-568)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (CrystalToken.sol#574-576)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CrystalToken.sol#586-589)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CrystalToken.sol#604-608)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CrystalToken.sol#623-626)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CrystalToken.sol#643-646)
mint(address,uint256) should be declared external:
        - CrystalToken.mint(address,uint256) (CrystalToken.sol#1300-1306)
addMinter(address) should be declared external:
        - CrystalToken.addMinter(address) (CrystalToken.sol#1308-1311)
delMinter(address) should be declared external:
        - CrystalToken.delMinter(address) (CrystalToken.sol#1313-1316)
getMinter(uint256) should be declared external:
        - CrystalToken.getMinter(uint256) (CrystalToken.sol#1326-1329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CrystalToken.sol analyzed (9 contracts with 75 detectors), 57 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

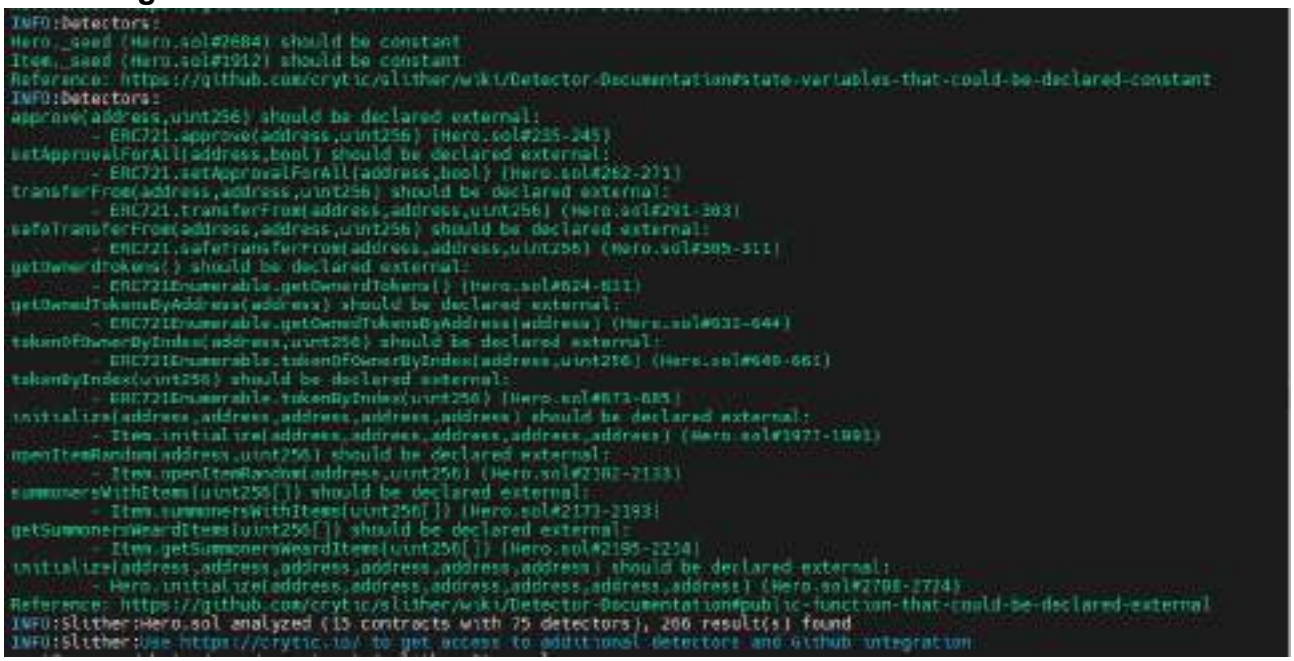
Slither log >> ERC721.sol

```
INFO:Detectors:
ERC721._isContract(address) (ERC721.sol#178-184) uses assembly
        - INLINE ASM (ERC721.sol#180-182)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (ERC721.sol#432-462) uses assembly
        - INLINE ASM (ERC721.sol#454-456)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

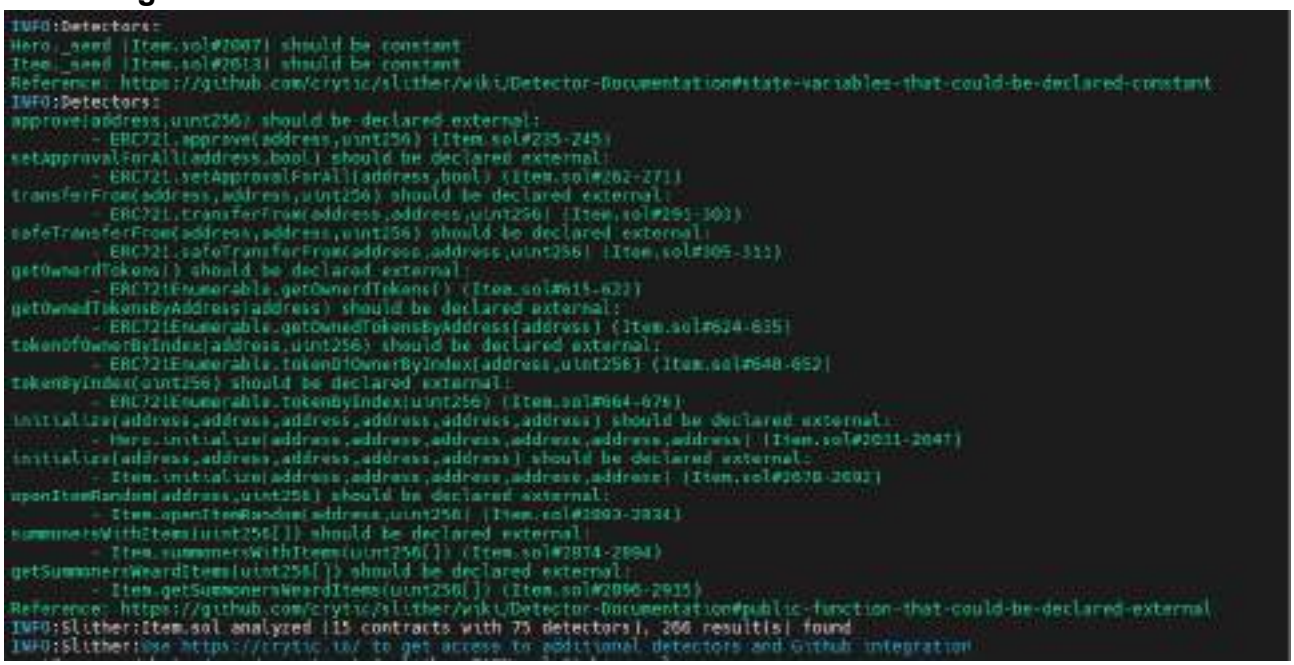


```
INFO:Detectors:
Pragma version^0.8.4 (ERC721.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (ERC721.sol#225) is not in mixedCase
Parameter ERC721Enumerable.getOwnedTokensByAddress(address)._owner (ERC721.sol#538) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (ERC721.sol#140-150)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (ERC721.sol#162-170)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (ERC721.sol#190-208)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (ERC721.sol#210-216)
getOwnedTokens() should be declared external:
        - ERC721Enumerable.getOwnedTokens() (ERC721.sol#529-536)
getOwnedTokensByAddress(address) should be declared external:
        - ERC721Enumerable.getOwnedTokensByAddress(address) (ERC721.sol#558-560)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (ERC721.sol#554-556)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (ERC721.sol#578-580)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ERC721.sol analyzed (7 contracts with 75 detectors), 24 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

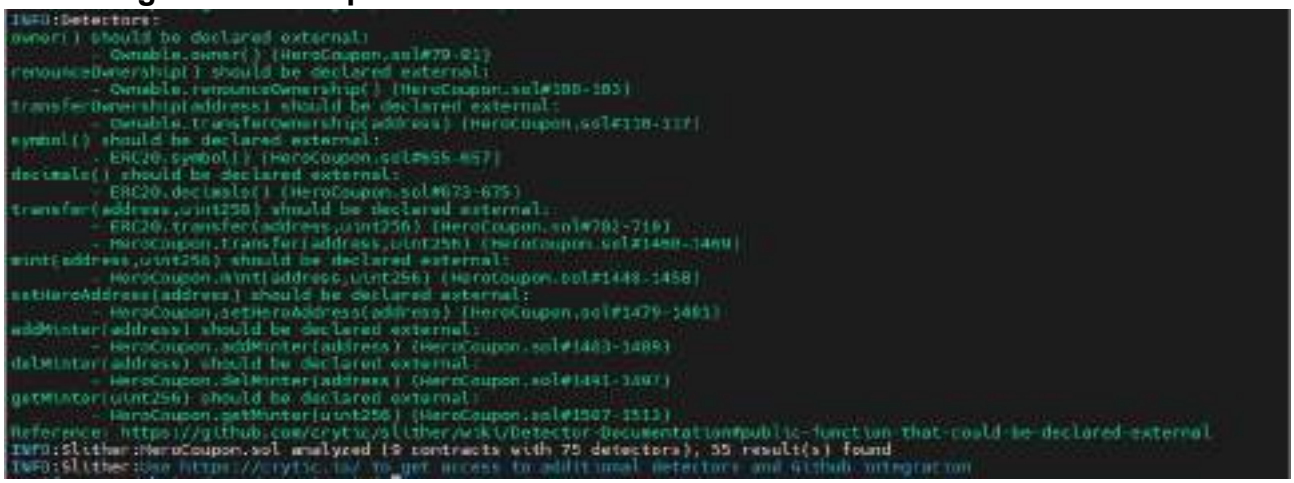
Slither log >> Hero.sol



Slither log >> Item.sol



Slither log >> HeroCoupon.sol



Solidity Static Analysis

Adventure.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 625:42:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Adventure.recieveDailyRankingUsdtAward(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3890:2:

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: 3618:40:

Gas & Economy

Gas costs:

Gas requirement of function Adventure.recieveWeeklyRankingUsdtAward 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: 3897:39:

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: 4196:10:

Miscellaneous

Constant/View/Pure functions:

Adventure.getAwardCroesusHistory(address,uint256,uint256,bool) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3976:11:

Similar variable names:

Adventure.publishWeeklyRanking(uint256) : Variables have very similar names "amount" and "amount2". Note: Modifiers are currently not considered by this static analysis.
Pos: 3861:19:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 4101:6:

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: 792:8:

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: 4231:14:

Building.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 4129:32:

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: 3919:0:

Gas & Economy

Gas costs:

Gas requirement of function Building.getOwnedBuildings 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: 4152:11:

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: 4215:6:

Miscellaneous

Similar variable names:

Building.redeemHero(uint256) : Variables have very similar names "bc" and "he". Note: Modifiers are currently not considered by this static analysis.

Pos: 4121:23:

Guard conditions:

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

more

Pos: 4121:1:

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: 4062:6:

CroesusToken.sol

Security

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: 1500:8:

Gas & Economy

Gas costs:

Gas requirement of function CroesusToken.getMinter 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: 1566:4:

Miscellaneous

Constant/View/Pure functions:

CroesusToken.getMinter(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1566:4:

Similar variable names:

DelegateERC20._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "numCheckpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.

Pos: 1483:12:

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: 1576:8:

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: 1411:36:

CrystalToken.sol

Security

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: 1279:8:

Gas & Economy

Gas costs:

Gas requirement of function CrystalToken.getMinter 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: 1326:4:

Miscellaneous

Constant/View/Pure functions:

CrystalToken.getMinter(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1326:4:

Similar variable names:

DelegateERC20._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "numCheckpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.

Pos: 1266:40:

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: 1333:8:

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: 1207:36:

ERC721.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

more

Pos: 530:42:

Gas & Economy

Gas costs:

Gas requirement of function ERC721.approve 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: 140:4:

Miscellaneous

Constant/View/Pure functions:

ERC721._isContract(address) : Is constant but potentially should not be.

more

Pos: 178:4:

Similar variable names:

ERC721.balanceOf(address) : Variables have very similar names "_owners" and "owner".

Pos: 118:25:

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: 259:8:

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: 373:8:

No return:

IERC165.supportsInterface(bytes4): Defines a return type but never explicitly returns a value.

Pos: 11:4:

Hero.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Hero.mergeSummoners(uint256[5],string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 2726:32:

Block hash:

Use of "blockhash": "blockhash(uint blockNumber)" is used to access the last 256 block hashes. A miner computes the block hash by "summing up" the information in the current block mined. By "summing up" the information cleverly, a miner can try to influence the outcome of a transaction in the current block. This is especially easy if there are only a small number of equally likely outcomes.

Pos: 3089:26:

Gas & Economy

Gas costs:

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

Pos: 3143:25:

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: 3136:0:

Miscellaneous

Constant/View/Pure functions:

Base64.encode(bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 3206:26:

Similar variable names:

Hero._getInitAbility(uint256,uint256) : Variables have very similar names "hc" and "it". Note: Modifiers are currently not considered by this static analysis.
Pos: 3003:11:

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: 3196:4:

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: 3213:38:

Item.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Item.mergeItems(uint256,uint256[5]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2694:78:

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: 2942:9:

Gas & Economy

Gas costs:

Gas requirement of function Item.wearItems 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: 2990:7:

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: 3191:0:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 3065:27:

Similar variable names:

Item._beforeTokenTransfer(address,address,uint256) : Variables have very similar names "he" and "to". Note: Modifiers are currently not considered by this static analysis.

Pos: 3144:16:

Guard conditions:

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

more

Pos: 2838:31:

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: 2745:2:

Data truncated:

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

Pos: 2746:3:

Adventure.sol

Security

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: 1417:8:

Gas & Economy

Gas costs:

Gas requirement of function HeroCoupon.getMinter 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: 1507:4:

Miscellaneous

Constant/View/Pure functions:

HeroCoupon.getMinter(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1507:4:

Similar variable names:

DelegateERC20._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "numCheckpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.
Pos: 1400:40:

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: 1517:8:

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: 1328:36:

Solhint Linter

Adventure.sol

```
Adventure.sol:2:1: Error: Compiler version ^0.8.4 does not satisfy
the r semver requirement
Adventure.sol:275:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Adventure.sol:549:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Adventure.sol:577:24: Error: Code contains empty blocks
Adventure.sol:625:43: Error: Avoid to use tx.origin
Adventure.sol:627:69: Error: Avoid to use tx.origin
Adventure.sol:871:28: Error: Avoid using low level calls.
Adventure.sol:1021:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Adventure.sol:1510:5: Error: Function name must be in mixedCase
Adventure.sol:1514:5: Error: Function name must be in mixedCase
```

Building.sol

```
Building.sol:1517:5: Error: Function name must be in mixedCase
Building.sol:1878:5: Error: Function name must be in mixedCase
Building.sol:1884:1: Error: Contract has 19 states declarations but
allowed no more than 15
Building.sol:1894:28: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:1895:28: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:1896:5: Error: Explicitly mark visibility of state
Building.sol:1896:20: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:1910:5: Error: Explicitly mark visibility of state
Building.sol:1930:5: Error: Event name must be in CamelCase
Building.sol:2237:38: Error: Avoid to make time-based decisions in
your business logic
Building.sol:2654:1: Error: Contract has 23 states declarations but
allowed no more than 15
Building.sol:2666:20: Error: Variable name must be in mixedCase
Building.sol:2667:28: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:2668:28: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:2669:5: Error: Explicitly mark visibility of state
Building.sol:2669:22: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:2685:5: Error: Event name must be in CamelCase
Building.sol:2803:9: Error: Variable name must be in
mixedCaseBuilding.sol:3216:9: Error: Avoid to use inline assembly. It
is acceptable only in rare cases
Building.sol:3222:26: Error: Code contains empty blocks
```

```
Building.sol:3374:5: Error: Function name must be in mixedCase
Building.sol:3527:1: Error: Contract has 19 states declarations but
allowed no more than 15
Building.sol:3547:5: Error: Explicitly mark visibility of state
Building.sol:3548:5: Error: Explicitly mark visibility of state
Building.sol:3548:22: Error: Constant name must be in capitalized
SNAKE_CASE
Building.sol:3737:26: Error: Avoid to make time-based decisions in
your business logic
Building.sol:3840:55: Error: Avoid to use tx.origin
Building.sol:3846:53: Error: Avoid to make time-based decisions in
your business logic
Building.sol:3939:59: Error: Avoid to use tx.origin
Building.sol:3946:57: Error: Avoid to make time-based decisions in
your business logic
Building.sol:3989:32: Error: Avoid to make time-based decisions in
your business logic
Building.sol:4055:38: Error: Avoid to use tx.origin
Building.sol:4092:46: Error: Avoid to make time-based decisions in
your business logic
Building.sol:4126:26: Error: Avoid to use tx.origin
Building.sol:4129:56: Error: Avoid to use tx.origin
```

CroesusToken.sol

```
CroesusToken.sol:6:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
CroesusToken.sol:941:24: Error: Code contains empty blocks
CroesusToken.sol:1361:17: Error: Avoid to make time-based decisions
in your business logic
CroesusToken.sol:1500:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
CroesusToken.sol:1525:30: Error: Constant name must be in capitalized
SNAKE_CASE
CroesusToken.sol:1527:51: Error: Code contains empty blocks
```

CrystalToken.sol

```
CrystalToken.sol:6:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
CrystalToken.sol:786:96: Error: Code contains empty blocks
CrystalToken.sol:1157:17: Error: Avoid to make time-based decisions
in your business logic
CrystalToken.sol:1279:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
CrystalToken.sol:1295:30: Error: Constant name must be in capitalized
SNAKE_CASE
CrystalToken.sol:1296:62: Error: Code contains empty blocks
```

ERC721.sol

```
ERC721.sol:2:1: Error: Compiler version ^0.8.4 does not satisfy the r
semver requirement
ERC721.sol:180:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ERC721.sol:454:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ERC721.sol:482:24: Error: Code contains empty blocks
ERC721.sol:530:43: Error: Avoid to use tx.origin
ERC721.sol:532:69: Error: Avoid to use tx.origin
```

Hero.sol

```
Hero.sol:275:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Hero.sol:549:21: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Hero.sol:577:24: Error: Code contains empty blocks
Hero.sol:625:43: Error: Avoid to use tx.origin
Hero.sol:627:69: Error: Avoid to use tx.origin
Hero.sol:871:28: Error: Avoid using low level calls.
Hero.sol:1021:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Hero.sol:1510:5: Error: Function name must be in mixedCase
```

Item.sol

```
Item.sol:2:1: Error: Compiler version ^0.8.4 does not satisfy the r
semver requirement
Item.sol:275:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Item.sol:549:21: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Item.sol:577:24: Error: Code contains empty blocks
Item.sol:616:43: Error: Avoid to use tx.origin
Item.sol:618:69: Error: Avoid to use tx.origin
Item.sol:862:28: Error: Avoid using low level calls.
Item.sol:990:51: Error: Avoid using low level calls.
Item.sol:1012:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Item.sol:1603:5: Error: Function name must be in
mixedCaseItem.sol:1615:5: Error: Function name must be in mixedCase
Item.sol:1619:5: Error: Function name must be in mixedCase
```

HeroCoupon.sol

```
HeroCoupon.sol:6:1: Error: Compiler version ^0.6.12 does not satisfy
```

```
the r semver requirement
HeroCoupon.sol:858:24: Error: Code contains empty blocks
HeroCoupon.sol:1278:17: Error: Avoid to make time-based decisions in
your business logic
HeroCoupon.sol:1417:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
HeroCoupon.sol:1443:30: Error: Constant name must be in capitalized
SNAKE_CASE
HeroCoupon.sol:1445:60: Error: Code contains empty blocks
```

Software analysis result:

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

Email: audit@EtherAuthority.io