Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Alliance NFT Protocol
Website:	thealliancenft.com
Platform:	Avalanche Network
Language:	Solidity
Date:	August 4th, 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 Alliance NFT Protocol to perform the Security audit of the Alliance NFT 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 August 4th, 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

- Alliance NFT is an NFT smart contract having functionalities like whitelisted users can mint NFT tokens by giving USDC token, stake NFT, unstake NFT, claim rewards in terms of USDC tokens, setting royalties, etc.
- The Alliance NFT Token contract inherits ERC721URIStorage, ReentrancyGuardUpgradeable, ERC721EnumerableUpgradeable, Address, OwnableUpgradeable, IERC20, ERC2981, SafeMathUpgradeable, MerkleProof, Ownable 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.

Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for Alliance NFT Protocol Smart Contracts
Platform	**Avalanche / Solidity**
File 1	AllianceNFT.sol
File 1 MD5 Hash	33BF8FE02E6D91FFE42B037CE11E0BF5
File 2	NodeManager.sol
File 2 MD5 Hash	E8FC4F6E8E743568832A86E81C55922B
File 3	RoyaltiesAddon.sol
File 3 MD5 Hash	4686079609560B3BB19C79E54949A499
File 4	Royalties.sol
File 4 MD5 Hash	D6FC083A9C1349C266EF8ED806BEEED6
Audit Date	August 4th,2022

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 AllianceNFT.sol • Name: The Alliance • Symbol: ALLIANCE • The owner can add, remove, update a tier. • The owner can set a USDC token address. • The owner can set blacklist and whitelist addresses. • Maximum NFT: 250 • Maximum NFT per user: 10	**YES, This is valid.**
File 2 NodeManager.sol • The owner can set a reward address, USDC token address, NFT address, etc. • The owner can set blacklist addresses.	**YES, This is valid.**
File 3 RoyaltiesAddon.sol • RoyaltiesAddon can internally set royalties addresses.	**YES, This is valid.**
File 4 Royalties.sol • Creator Royalties: 1% • Community Royalties: 3% • Collection Size: 10,000 • The owner can set royalties address, creator address, smaller collection size, etc.	**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 2 low and some very low level issues.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	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	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Moderated
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 4 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 Alliance NFT 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 Alliance NFT Protocol.

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

Some code parts are not well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given an Alliance NFT 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 not well commented. But the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://thealliancenft.com/ 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

AllianceNFT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__ERC721Enumerable_init	internal	access only Initializing	No Issue
3	__ERC721Enumerable_init_unchained	internal	access only Initializing	No Issue
4	supportsInterface	read	Passed	No Issue
5	tokenOfOwnerByIndex	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	tokenByIndex	read	Passed	No Issue
8	__ReentrancyGuard_init	internal	access only Initializing	No Issue
9	_beforeTokenTransfer	internal	Passed	No Issue
10	_addTokenToOwnerEnumeration	write	Passed	No Issue
11	_addTokenToAllTokensEnumeration	write	Passed	No Issue
12	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
13	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
14	__ReentrancyGuard_init_unchained	internal	access only Initializing	No Issue
15	nonReentrant	modifier	Passed	No Issue
16	_nonReentrantBefore	write	Passed	No Issue
17	_nonReentrantAfter	write	Passed	No Issue
18	__Ownable_init	internal	access only Initializing	No Issue
19	__Ownable_init_unchained	internal	access only Initializing	No Issue
20	onlyOwner	modifier	Passed	No Issue
21	owner	read	Passed	No Issue
22	_checkOwner	internal	Passed	No Issue
23	renounceOwnership	write	access only Owner	No Issue
24	transferOwnership	write	access only Owner	No Issue
25	_transferOwnership	internal	Passed	No Issue
26	supportsInterface	read	Passed	No Issue
27	royaltyInfo	read	Passed	No Issue
28	_feeDenominator	internal	Passed	No Issue
29	_setDefaultRoyalty	internal	Passed	No Issue
30	_deleteDefaultRoyalty	internal	Passed	No Issue
31	_setTokenRoyalty	internal	Passed	No Issue

32	_resetTokenRoyalty	internal	Passed	No Issue
33	initialize	write	Anyone can initialize contract	Refer audit findings
34	addTier	external	access only Owner	No Issue
35	updateTier	external	access only Owner	No Issue
36	removeTier	external	access only Owner	No Issue
37	setUsdcToken	write	access only Owner	No Issue
38	_baseURI	internal	Passed	No Issue
39	setURI	external	access only Owner	No Issue
40	tokenURI	read	Passed	No Issue
41	noBlacklist	modifier	Passed	No Issue
42	setBlacklist	write	access only Owner	No Issue
43	setWhitelist	write	access only Owner	No Issue
44	isAddressWhitelisted	read	Passed	No Issue
45	mintNode	write	Passed	No Issue
46	mint	write	Passed	No Issue
47	mintTo	write	access only Owner	No Issue
48	setMintReceiver	external	access only Owner	No Issue
49	removeFromArrayString	internal	Passed	No Issue
50	removeFromArray	internal	Unused internal function	Refer audit findings
51	nodeTiersCount	external	Passed	No Issue
52	supportsInterface	read	Passed	No Issue
53	setRoyalities	external	Function input parameters lack of check	Refer audit findings
54	setNftTier	external	Function input parameters lack of check	Refer audit findings
55	withdrawUSDC	external	access only Owner	No Issue

NodeManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__ReentrancyGuard_init_unchained	internal	access only Initializing	No Issue
3	nonReentrant	modifier	Passed	No Issue
4	_nonReentrantBefore	write	Passed	No Issue
5	_nonReentrantAfter	write	Passed	No Issue
6	__Ownable_init	internal	access only Initializing	No Issue
7	__Ownable_init_unchained	internal	access only Initializing	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	owner	read	Passed	No Issue

10	_checkOwner	internal	Passed	No Issue
11	renounceOwnership	write	access only Owner	No Issue
12	transferOwnership	write	access only Owner	No Issue
13	_transferOwnership	internal	Passed	No Issue
14	initialize	write	Anyone can initialize contract	Refer audit findings
15	setRewardAddress	write	access only Owner	No Issue
16	setUsdcToken	write	access only Owner	No Issue
17	setNFT	write	access only Owner	No Issue
18	noBlacklist	modifier	Passed	No Issue
19	setBlacklist	write	access only Owner	No Issue
20	stakeNode	external	Passed	No Issue
21	unstakeNode	external	Passed	No Issue
22	getNodesCountStaked	read	Passed	No Issue
23	getNodesStaked	read	Passed	No Issue
24	getDailyReward	read	Passed	No Issue
25	getAvailableReward	read	Passed	No Issue
26	getTierReward	internal	Passed	No Issue
27	claimAllRewards	write	Passed	No Issue
28	setTaxAccount	external	access only Owner	No Issue
29	getTax	external	access only Owner	No Issue
30	removeFromArray	internal	Passed	No Issue

RoyaltiesAddon.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	tokenURI	read	Passed	No Issue
3	_setTokenURI	internal	Passed	No Issue
4	_burn	internal	Passed	No Issue
5	_setRoyaltiesAddress	internal	Passed	No Issue
6	_beforeTokenTransfer	internal	Passed	No Issue

Royalties.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	_setOwner	write	Passed	No Issue
7	setTokenFeesAddress	external	access only Owner	No Issue

8	setCreatorAddress	external	access only Owner	No Issue
9	setCollectionSize	external	access only Owner	No Issue
10	setCreatorRoyalties	external	access only Owner	No Issue
11	setCommunityRoyalties	external	access only Owner	No Issue
12	getTotalRoyalties	read	Passed	No Issue
13	getRoyalties	read	Passed	No Issue
14	getTotalCollected	read	Passed	No Issue
15	getCreatorBalance	read	Passed	No Issue
16	getTokenTotalRoyalties	read	Passed	No Issue
17	getTokenBalance	read	Passed	No Issue
18	getTokensBalance	read	Passed	No Issue
19	getAddressClaims	read	Passed	No Issue
20	claimCommunity	write	Passed	No Issue
21	claimCommunityBatch	external	Infinite loops possibility	Refer audit findings
22	claimCreator	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.

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) Function input parameters lack of check: **AllianceNFT.sol**

Some functions require validation before execution.

Functions are:

- setNftTier() - tier variable not check - if tier exist or not
- setRoyalities()

Resolution: We suggest using validation like for numerical variables that should be greater than 0 and for address type check variables that are not address(0).

(2) Infinite loops possibility: **Royalties.sol**

```
/// @dev claim community from an array of tokenIDs
function claimCommunityBatch(uint256[] calldata tokenIDs) external {
    for (uint256 i=0; i<tokenIDs.length; i++) {
        claimCommunity(tokenIDs[i]);
    }
}
```

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

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

Very Low / Informational / Best practices:

(1) Unused variables: **NodeManager.sol**

```
uint256 public nodeCount;
```

```
string _baseTokenURI;
```

There are some variables defined but not used anywhere.

- nodeCount
- _baseTokenURI

Resolution: Remove unused variables from the code.

(2) Anyone can initialize contract:

AllianceNFT.sol

```
function initialize() initializer public {
    __Ownable_init();
    __ReentrancyGuard_init();// is on 100000 so to 10% it will be like 10000
    __ERC721_init("The Alliance", "ALLIANCE");
}
```

NodeManager.sol

```
function initialize() initializer public {
    __Ownable_init();
    __ReentrancyGuard_init();// is on 100000 so to 10% it will be like 10000

    lastClaimTax = block.timestamp;
}
```

initialize() function is public, so anyone can execute this function, And make the contract creator own itself.

Resolution: The owner has to make sure to initialize the contract after deploying.

(3) Unused internal function: **AllianceNFT.sol**

There are functions defined but not used in the functionality.

- removeFromArray()

Resolution: We suggest removing unused internal functions from code.

(4) SafeMathUpgradeable Library: **AllianceNFT.sol**

SafeMathUpgradeable Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will be not required to use, solidity automatically handles overflow/underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

(5) Spelling mistakes: **Royalties.sol**

There are some places where there are spelling mistakes:

Comments before function name getAddressClaims() -

/// @dev get address tot claims

- tot word should be total

Resolution: We suggest correcting spelling mistakes.

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:

- addTier: AllianceNFT owner can add new tier.
- updateTier: AllianceNFT owner can update tier.
- removeTier: AllianceNFT owner can remove tier.
- setUsdcToken: AllianceNFT owner can set USDC token.
- setURI: AllianceNFT owner can set the URI.
- setBlacklist: AllianceNFT owner can set blacklist address.
- setWhitelist: AllianceNFT owner can set whitelist address.
- mintTo: AllianceNFT owner can mint token from address.
- setMintReceiver: AllianceNFT owner can set the mint receiver address.
- setRoyalities: AllianceNFT owner can set royalties receiver address.
- setNftTier: AllianceNFT owner can set NFT Tier.
- withdrawUSDC: AllianceNFT owner can withdraw USDC address.
- setRewardAddress: NodeManager owner can set reward address.
- setUsdcToken: NodeManager owner can set USDC token address.
- setNFT: NodeManager owner can set NFT address.
- setBlacklist: NodeManager owner can set blacklist address.
- setTaxAccount: NodeManager owner can set tax account address.
- getTax: NodeManager owner can get tax.
- setTokenFeesAddress: Royalties owner can set royalties address.
- setCreatorAddress: Royalties owner can set the creator address to be another contract.
- setCollectionSize: Royalties owner can set only smaller collection size, can't increase the size.
- setCreatorRoyalties: Royalties owner can set creator royalties values.
- setCommunityRoyalties: Royalties owner can set community royalties value.
- claimCreator: Royalties owner can claim creator royalties.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We have observed 2 low severity issues and some informational issues in the smart contracts. **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.

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

The 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 is unlimited, the audit makes no statements or warranties on the 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.

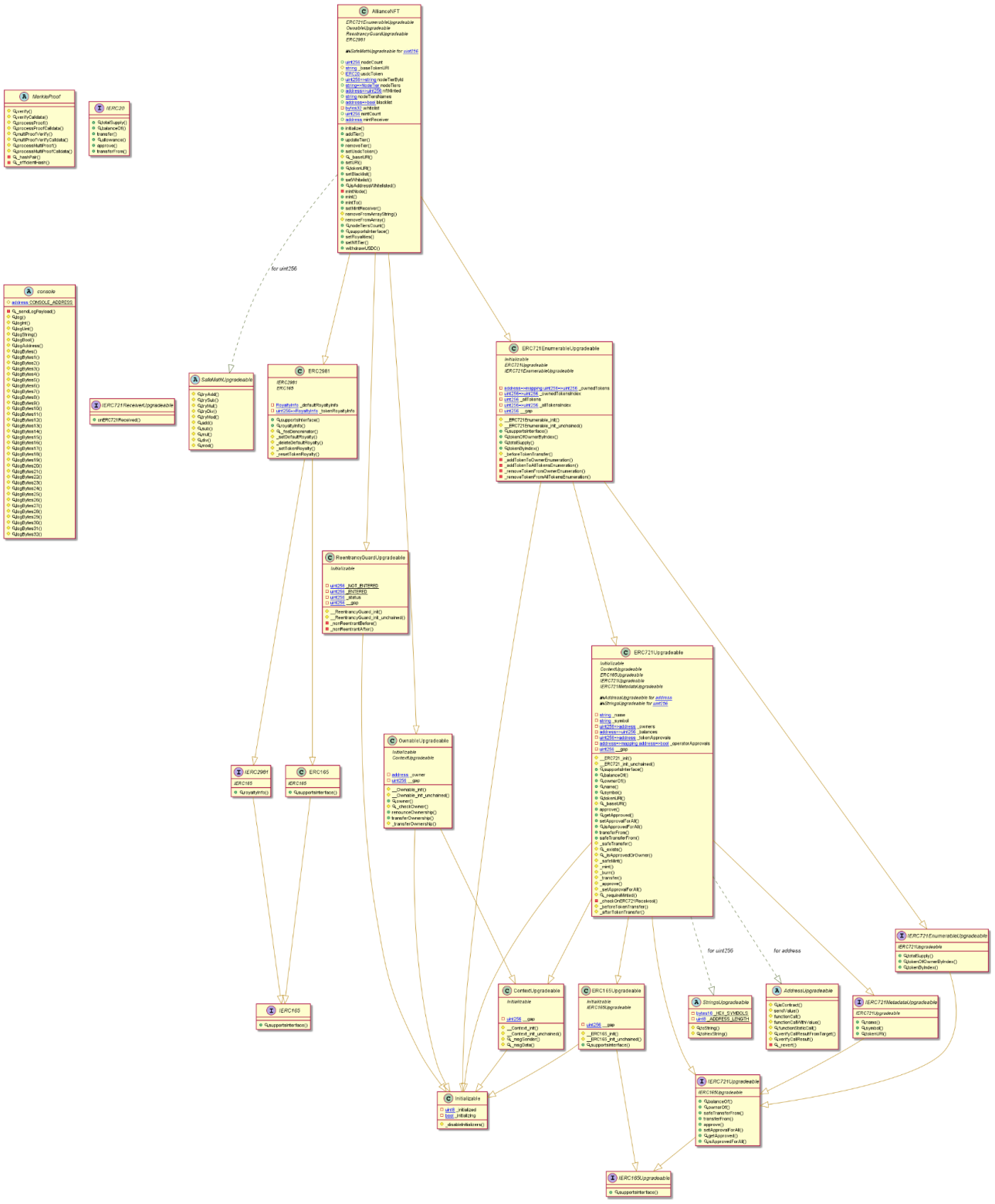
Technical Disclaimer

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.

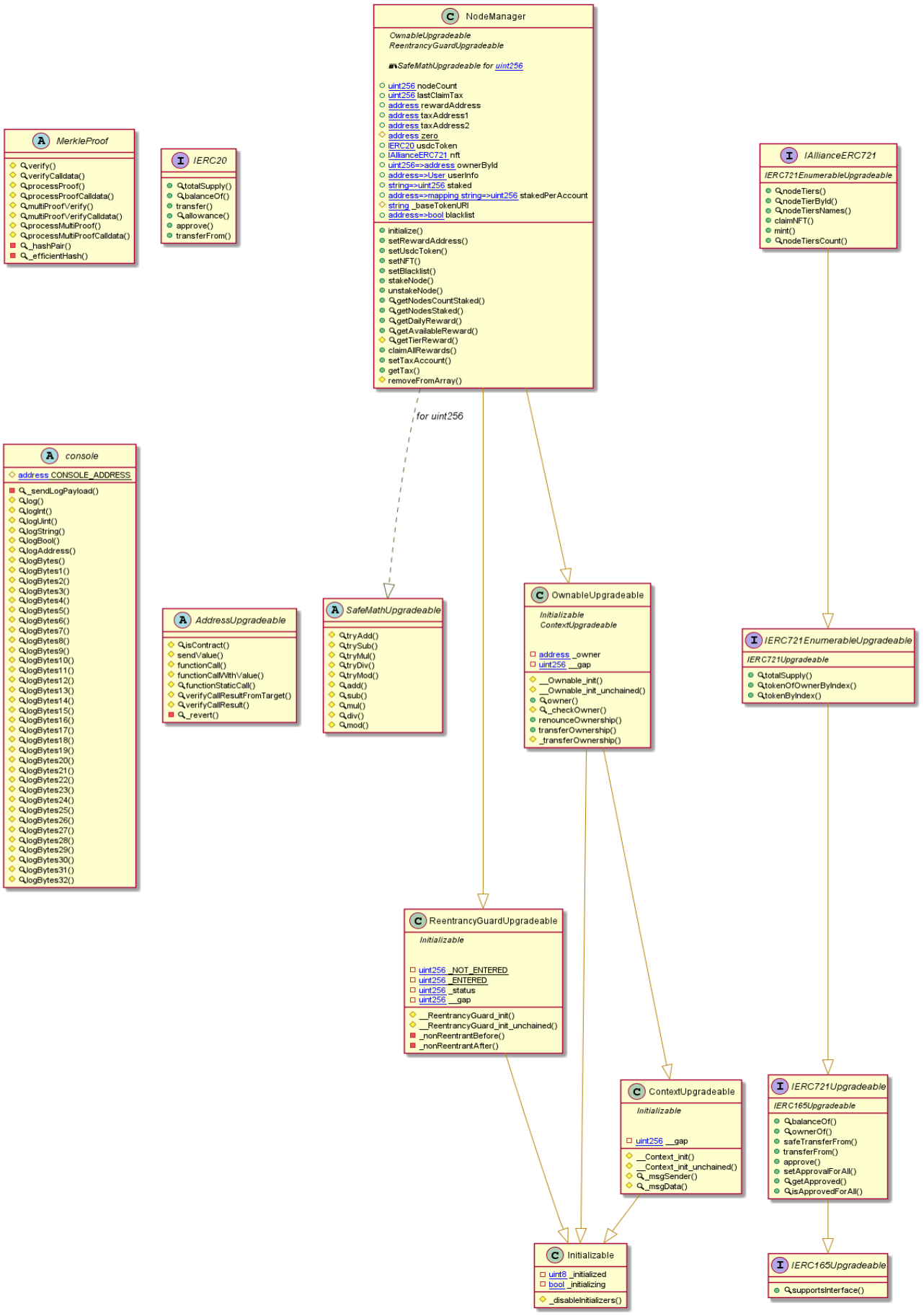
Appendix

Code Flow Diagram - Alliance NFT Protocol

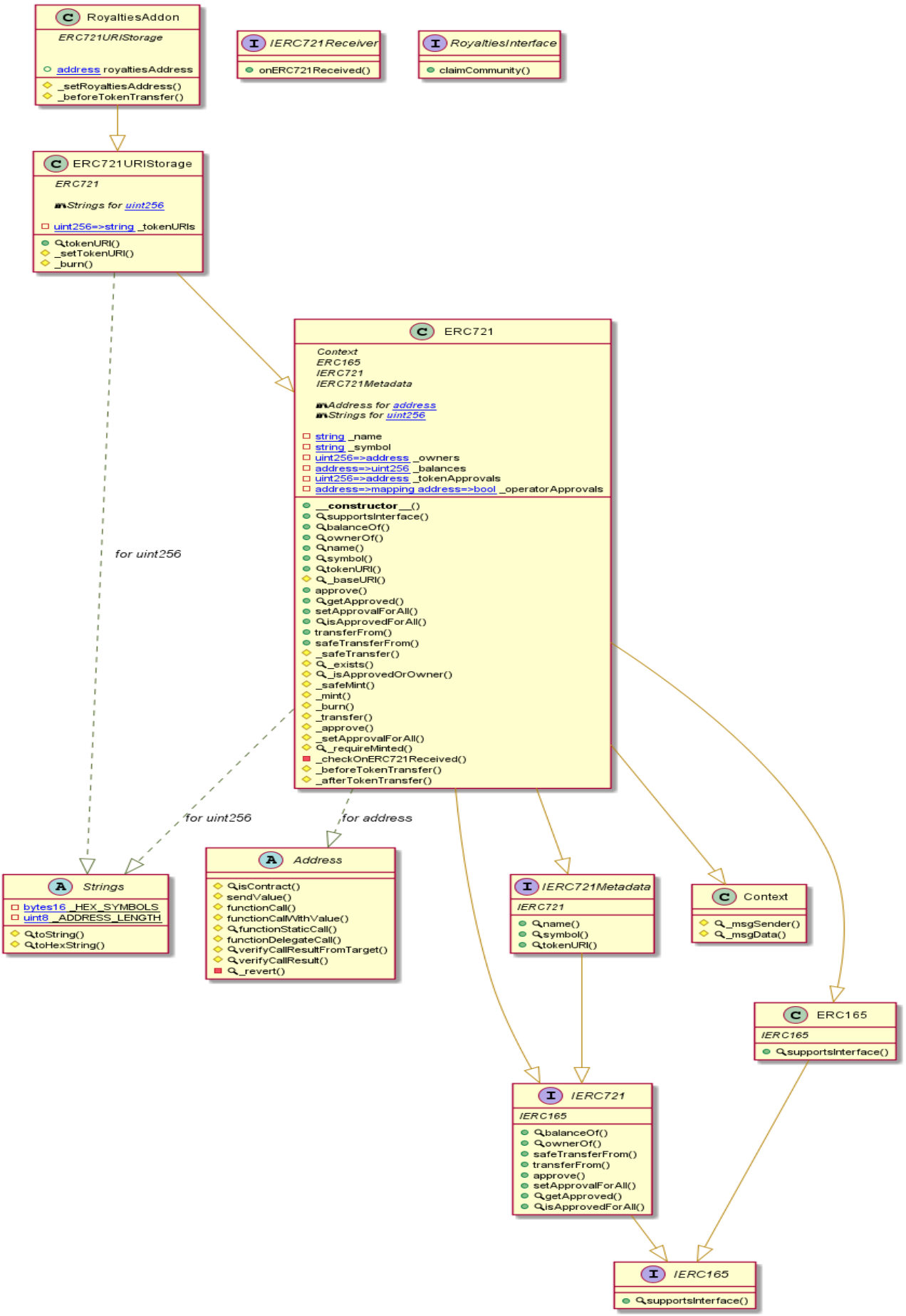
AllianceNFT Diagram



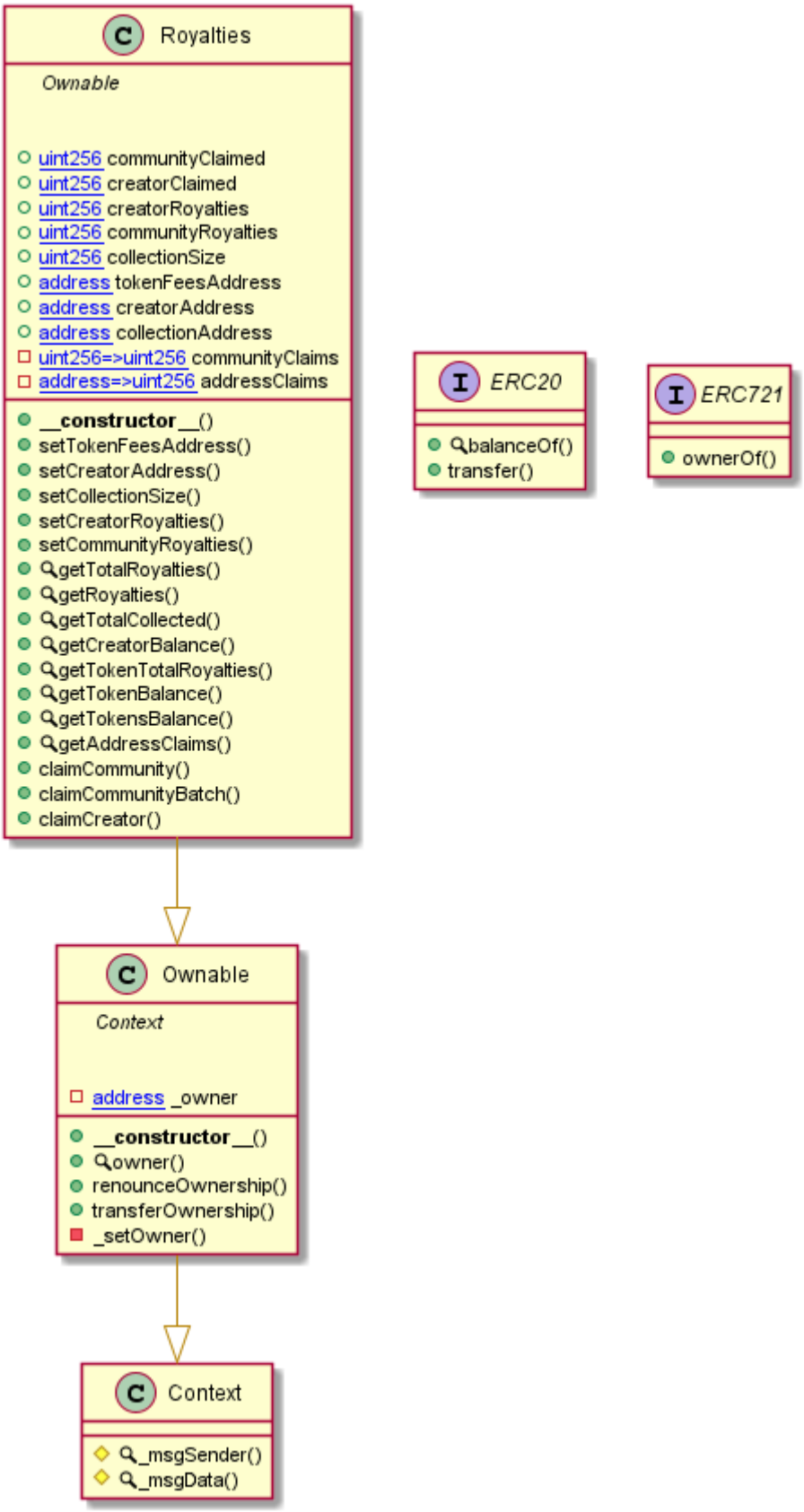
NodeManager Diagram



RoyaltiesAddon Diagram



Royalties Diagram



Slither Results Log

Slither log >> AllianceNFT.sol

```
INFO:Detectors:
AllianceNFT.mint(string,uint256,bytes32[]) (AllianceNFT.sol#3251-3268) compares to a boolean constant:
        -require(bool,string)(nodeTiers[_tier].canMint == true,NODE: Mint is disabled) (AllianceNFT.sol#3255)
AllianceNFT.noBlacklist(address) (AllianceNFT.sol#3209-3212) compares to a boolean constant:
        -require(bool,string)(blacklist[_user] == false,Node: You are blacklisted) (AllianceNFT.sol#3210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
INFO:Detectors:
console.slitherConstructorConstantVariables() (AllianceNFT.sol#246-1774) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (AllianceNFT.sol#247)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ReentrancyGuardUpgradeable.__gap (AllianceNFT.sol#2446) is never used in AllianceNFT (AllianceNFT.sol#3101-3331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
royaltyInfo(uint256,uint256) should be declared external:
        - ERC2981.royaltyInfo(uint256,uint256) (AllianceNFT.sol#2062-2072)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (AllianceNFT.sol#2475-2477)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (AllianceNFT.sol#2479-2482)
```

```
        - OwnableUpgradeable.renounceOwnership() (AllianceNFT.sol#2475-2477)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (AllianceNFT.sol#2479-2482)
name() should be declared external:
        - ERC721Upgradeable.name() (AllianceNFT.sol#2559-2561)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (AllianceNFT.sol#2566-2568)
tokenURI(uint256) should be declared external:
        - AllianceNFT.tokenURI(uint256) (AllianceNFT.sol#3199-3201)
        - ERC721Upgradeable.tokenURI(uint256) (AllianceNFT.sol#2573-2578)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (AllianceNFT.sol#2592-2602)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (AllianceNFT.sol#2616-2618)
transferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.transferFrom(address,address,uint256) (AllianceNFT.sol#2630-2639)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (AllianceNFT.sol#2644-2650)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenOfOwnerByIndex(address,uint256) (AllianceNFT.sol#2972-2975)
tokenByIndex(uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenByIndex(uint256) (AllianceNFT.sol#2987-2990)
initialize() should be declared external:
        - AllianceNFT.initialize() (AllianceNFT.sol#3123-3127)
setUsdcToken(address) should be declared external:
        - AllianceNFT.setUsdcToken(address) (AllianceNFT.sol#3187-3189)
setBlacklist(address,bool) should be declared external:
        - AllianceNFT.setBlacklist(address,bool) (AllianceNFT.sol#3214-3216)
setWhitelist(bytes32) should be declared external:
        - AllianceNFT.setWhitelist(bytes32) (AllianceNFT.sol#3224-3226)
mint(string,uint256,bytes32[]) should be declared external:
        - AllianceNFT.mint(string,uint256,bytes32[]) (AllianceNFT.sol#3251-3268)
mintTo(address[],string) should be declared external:
        - AllianceNFT.mintTo(address[],string) (AllianceNFT.sol#3270-3274)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AllianceNFT.sol analyzed (23 contracts with 75 detectors), 523 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Royalties.sol

```
INFO:Detectors:
Reentrancy in Royalties.claimCommunity(uint256) (Royalties.sol#184-196):
        External calls:
        - owner = ERC721(collectionAddress).ownerOf(tokenID) (Royalties.sol#187)
        - ERC20(tokenFeesAddress).transfer(owner,balance) (Royalties.sol#189)
        State variables written after the call(s):
        - addressClaims[owner] = addressClaims[owner] + balance (Royalties.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Royalties.claimCommunity(uint256) (Royalties.sol#184-196):
        External calls:
        - owner = ERC721(collectionAddress).ownerOf(tokenID) (Royalties.sol#187)
        - ERC20(tokenFeesAddress).transfer(owner,balance) (Royalties.sol#189)
        Event emitted after the call(s):
        - CommunityClaimed(owner,balance,tokenID) (Royalties.sol#193)
Reentrancy in Royalties.claimCreator() (Royalties.sol#206-213):
        External calls:
        - ERC20(tokenFeesAddress).transfer(creatorAddress,balance) (Royalties.sol#210)
        Event emitted after the call(s):
        - CreatorClaimed(balance) (Royalties.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (Royalties.sol#10-13) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (Royalties.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
```

```
INFO:Detectors:
Redundant expression "this (Royalties.sol#11)" inContext (Royalties.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Royalties.sol#34-36)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Royalties.sol#38-41)
getRoyalties() should be declared external:
        - Royalties.getRoyalties() (Royalties.sol#133-135)
getTokensBalance(uint256[]) should be declared external:
        - Royalties.getTokensBalance(uint256[]) (Royalties.sol#168-175)
getAddressClaims(address) should be declared external:
        - Royalties.getAddressClaims(address) (Royalties.sol#179-181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Royalties.sol analyzed (5 contracts with 75 detectors), 28 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> NodeManager.sol

```
INFO:Detectors:
Variable NodeManager.taxAddress1 (NodeManager.sol#2284) is too similar to NodeManager.taxAddress2 (NodeManager.sol#2285)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
console.slitherConstructorConstantVariables() (NodeManager.sol#228-1756) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (NodeManager.sol#229)
NodeManager.slitherConstructorConstantVariables() (NodeManager.sol#2268-2487) uses literals with too many digits:
        - zero = 0x0000000000000000000000000000000000000000 (NodeManager.sol#2286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ReentrancyGuardUpgradeable.__gap (NodeManager.sol#2220) is never used in NodeManager (NodeManager.sol#2268-2487)
NodeManager._baseTokenURI (NodeManager.sol#2296) is never used in NodeManager (NodeManager.sol#2268-2487)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
NodeManager._baseTokenURI (NodeManager.sol#2296) should be constant
NodeManager.nodeCount (NodeManager.sol#2281) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (NodeManager.sol#2249-2251)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (NodeManager.sol#2253-2256)
initialize() should be declared external:
        - NodeManager.initialize() (NodeManager.sol#2298-2303)
setRewardAddress(address) should be declared external:
        - NodeManager.setRewardAddress(address) (NodeManager.sol#2308-2310)
setUsdcToken(address) should be declared external:
        - NodeManager.setUsdcToken(address) (NodeManager.sol#2312-2314)
setNFT(address) should be declared external:
        - NodeManager.setNFT(address) (NodeManager.sol#2316-2318)
setBlacklist(address,bool) should be declared external:
        - NodeManager.setBlacklist(address,bool) (NodeManager.sol#2331-2333)
getNodesCountStaked(address) should be declared external:
        - NodeManager.getNodesCountStaked(address) (NodeManager.sol#2387-2389)
getNodesStaked(address) should be declared external:
        - NodeManager.getNodesStaked(address) (NodeManager.sol#2391-2393)
getDailyReward(address) should be declared external:
```

Slither log >> RoyaltiesAddon.sol

```
INFO:Detectors:
Pragma version^0.8.0 (RoyaltiesAddon.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (RoyaltiesAddon.sol#56-61):
        - (success) = recipient.call{value: amount}() (RoyaltiesAddon.sol#59)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (RoyaltiesAddon.sol#124-133):
        - (success,returndata) = target.call{value: value}(data) (RoyaltiesAddon.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (RoyaltiesAddon.sol#151-158):
        - (success,returndata) = target.staticcall(data) (RoyaltiesAddon.sol#156)
Low level call in Address.functionDelegateCall(address,bytes,string) (RoyaltiesAddon.sol#176-183):
        - (success,returndata) = target.delegatecall(data) (RoyaltiesAddon.sol#181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (RoyaltiesAddon.sol#428)" inContext (RoyaltiesAddon.sol#422-431)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (RoyaltiesAddon.sol#556-559)
name() should be declared external:
        - ERC721.name() (RoyaltiesAddon.sol#573-575)
symbol() should be declared external:
        - ERC721.symbol() (RoyaltiesAddon.sol#580-582)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (RoyaltiesAddon.sol#606-616)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (RoyaltiesAddon.sol#630-632)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (RoyaltiesAddon.sol#644-653)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (RoyaltiesAddon.sol#658-664)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:RoyaltiesAddon.sol analyzed (12 contracts with 75 detectors), 43 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

AllianceNFT.so

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AllianceNFT.mint(string,uint256,bytes32[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 3251:4:

Inline assembly:

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

more

Pos: 2889:20:

Gas & Economy

Gas costs:

Gas requirement of function AllianceNFT.isAddressWhitelisted 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: 3228:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 3271:8:

Email: audit@EtherAuthority.io

Miscellaneous

Constant/View/Pure functions:

AllianceNFT.removeFromArray(uint256[],uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3295:4:

Similar variable names:

AllianceNFT.isAddressWhitelisted(address,bytes32[]) : Variables have very similar names "whitelist" and "whitelisted". Note: Modifiers are currently not considered by this static analysis.
Pos: 3232:9:

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: 3296:8:

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: 3095:8:

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: 2069:32:

NodeManager.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in NodeManager.getTax(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2457:4:

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: 2302:23:

Gas & Economy

Gas costs:

Gas requirement of function NodeManager.initialize 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: 2298:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 2479:8:

Miscellaneous

Constant/View/Pure functions:

NodeManager.removeFromArray(uint256[],uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2476:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2477:8:

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: 2469:70:

RoyaltiesAddon.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 181:50:

Gas & Economy

Gas costs:

Gas requirement of function ERC721.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 573:4:

Gas costs:

Gas requirement of function ERC721.safeTransferFrom 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: 669:4:

Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.
more
Pos: 998:12:

Miscellaneous

Constant/View/Pure functions:

RoyaltiesInterface.claimCommunity(uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 1004:4:

Similar variable names:

ERC721URIStorage._setTokenURI(uint256,string) : Variables have very similar names "_tokenURI" and "_tokenURIs".
Pos: 986:30:

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: 985:8:

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: 998:12:

Royalties.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Royalties.claimCommunity(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 184:4:

Gas & Economy

Gas costs:

Gas requirement of function Royalties.getTotalRoyalties 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: 126:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 200:8:

Miscellaneous

Constant/View/Pure functions:

ERC721.ownerOf(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 57:4:

No return:

ERC721.ownerOf(uint256): Defines a return type but never explicitly returns a value.

Pos: 57:4:

Similar variable names:

Royalties.claimCommunity(uint256) : Variables have very similar names "communityClaimed" and "communityClaims". Note: Modifiers are currently not considered by this static analysis.

Pos: 192:35:

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: 207:8:

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: 209:8:

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: 157:12:

Solhint Linter

AllianceNFT.sol

```
AllianceNFT.sol:1778:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1786:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1793:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1802:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1809:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1840:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1851:18: Error: Parse error: missing ';' at '{'
AllianceNFT.sol:1862:18: Error: Parse error: missing ';' at '{'
```

NodeManager.sol

```
NodeManager.sol:1760:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1768:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1775:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1784:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1791:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1822:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1833:18: Error: Parse error: missing ';' at '{'
NodeManager.sol:1844:18: Error: Parse error: missing ';' at '{'
```

RoyaltiesAddon.sol

```
RoyaltiesAddon.sol:2:1: Error: Compiler version ^0.8.0 does not
satisfy the r semver requirement
RoyaltiesAddon.sol:232:13: Error: Avoid using inline assembly. It is
acceptable only in rare cases
RoyaltiesAddon.sol:538:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
RoyaltiesAddon.sol:903:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
RoyaltiesAddon.sol:931:24: Error: Code contains empty blocks
RoyaltiesAddon.sol:948:24: Error: Code contains empty blocks
```

Royalties.sol

```
Royalties.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
Royalties.sol:21:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Royalties.sol:85:5: Error: Explicitly mark visibility in function
```

```
(Set ignoreConstructors to true if using solidity >=0.7.0)
Royalties.sol:110:51: Error: Use double quotes for string
literalsRoyalties.sol:192:17: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
Royalties.sol:211:9: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
```

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