Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project Name: Catecoin NFT
Website: play.catecoin.club
Platform: Binance Smart Chain
Language: Solidity
Date: October 29th, 2021

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 the Catecoin team to perform the Security audit of the Catecoin 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 October 29th, 2021.

The purpose of this audit was to address the following:

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

Project Background

Catecoin NFT system is based on ROCNFT token having functionality like mint, mintBatch, etc from ERC1155 and burn, burnBatch, etc from ERC1155Burnable.

Audit scope

Name	**Code Review and Security Analysis Report for Catecoin NFT Smart Contracts**
Platform	**BSC / Solidity**
File 1	ROCBlindBox1.sol
File 1 MD5 Hash	88A087FEECFA46852BEFBD43CAA0501A
File 1 Updates MD5 Hash	8AF8BBD7E7A81B34847F4A241583DC0C
File 2	ROCNFT.sol
File 2 MD5 Hash	F0A8168F470E0E8ECD312DB062E9248F
File 2 Updates MD5 Hash	C045E69C3F9B733214FC73633CE35728
Audit Date	October 29th, 2021
Revised Audit Date	November 2nd, 2021

Claimed Smart Contracts Features

Claimed Feature Detail	Our Observation
File 1: ROCBlindBox1.sol • Limited Buy Count: 2	**YES, This is valid.**
File 2: ROCNFT.sol • Name: Rise of Cats • Token URL: https://nft.riseofcats.com/cat/	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These Protocol contracts do contain owner control, which does not make it fully decentralized.

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

We found 0 critical, 1 high, 0 medium and 3 low and some very low level issues. Major issues have been resolved / acknowledged by auditee, so it's good to go for the production.

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	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 2 smart contracts files. Smart contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Catecoin 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 Catecoin NFT Protocol.

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

Code parts are **not** well commented on smart contracts.

Documentation

We were given a Catecoin NFT Protocol smart contracts code in the form of a BSCScan web link.The hash of that code is mentioned above in the table.

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

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

Use of Dependencies

As per our observation, the libraries are used in this smart contract 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

ROCBlindBox1.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	paused	read	Passed	No Issue
4	whenNotPaused	modifier	Passed	No Issue
5	whenPaused	modifier	Passed	No Issue
6	_pause	internal	access when Not Paused	No Issue
7	_unpause	internal	access when Paused	No Issue
8	owner	read	Passed	No Issue
9	onlyOwner	modifier	Passed	No Issue
10	renounceOwnership	write	Passed	No Issue
11	transferOwnership	write	access only Owner	No Issue
12	_setOwner	write	Passed	No Issue
13	boxIdIsValid	modifier	Passed	No Issue
14	whenNotSoldOut	modifier	Passed	No Issue
15	setupBlindBoxInfo	write	Passed	No Issue
16	pause	write	access only Owner	No Issue
17	unpause	write	access only Owner	No Issue
18	updatePrice	write	access only Owner	No Issue
19	getLeftLimitBuyCount	read	Passed	No Issue
20	getSoldCount	read	Passed	No Issue
21	getTotalCount	read	Passed	No Issue
22	getPrice	read	Passed	No Issue
23	payByToken	external	Passed	No Issue

ROCNFT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	uri	read	Passed	No Issue
3	uint2str	internal	Passed	No Issue
4	setURI	write	access only Owner/Admin Role	No Issue

5	pause	write	access only Owner	No Issue
6	unpause	write	access only Owner	No Issue
7	mint	write	Function input parameters lack of check	Refer Audit Findings
8	mintBatch	write	access only Minter Role	No Issue
9	mintForBatch	write	Removed	No Issue
10	_beforeTokenTransfer	internal	Passed	No Issue
11	supportsInterface	read	Passed	No Issue
12	supportsInterface	read	Passed	No Issue
13	uri	read	Passed	No Issue
14	balanceOf	read	Passed	No Issue
15	balanceOfBatch	read	Passed	No Issue
16	setApprovalForAll	write	Passed	No Issue
17	isApprovedForAll	read	Passed	No Issue
18	safeTransferFrom	write	Passed	No Issue
19	safeBatchTransferFrom	write	Passed	No Issue
20	_safeTransferFrom	internal	Passed	No Issue
21	_safeBatchTransferFrom	internal	Passed	No Issue
22	_setURI	internal	Passed	No Issue
23	_mint	internal	Passed	No Issue
24	_mintBatch	internal	Passed	No Issue
25	_burn	internal	Passed	No Issue
26	_burnBatch	internal	Passed	No Issue
27	_beforeTokenTransfer	internal	Passed	No Issue
28	_doSafeTransferAcceptanceCheck	write	Passed	No Issue
29	_doSafeBatchTransferAcceptanceCheck	write	Passed	No Issue
30	_asSingletonArray	write	Passed	No Issue
31	owner	read	Passed	No Issue
32	onlyOwner	modifier	Passed	No Issue
33	renounceOwnership	write	access only Owner	No Issue
34	transferOwnership	write	access only Owner	No Issue
35	_setOwner	write	Passed	No Issue
36	paused	read	Passed	No Issue
37	whenNotPaused	modifier	Passed	No Issue
38	whenPaused	modifier	Passed	No Issue
39	_pause	internal	access when Not Paused	No Issue
40	_unpause	internal	access when Paused	No Issue
41	onlyRole	modifier	Passed	No Issue
42	hasRole	read	Passed	No Issue
43	supportsInterface	read	Passed	No Issue
44	_checkRole	internal	Passed	No Issue
45	getRoleAdmin	read	Passed	No Issue

46	grantRole	write	access only Owner/Admin Role	No Issue
47	revokeRole	write	access only Owner/Admin Role	No Issue
48	renounceRole	write	Passed	No Issue
49	_setupRole	internal	Passed	No Issue
50	_setRoleAdmin	internal	Passed	No Issue
51	_grantRole	write	Passed	No Issue
52	_revokeRole	write	Passed	No Issue
53	burn	write	Passed	No Issue
54	burnBatch	write	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens 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

ROCBlindBox1.sol

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

No Information severity vulnerabilities were found.

ROCNFT.sol

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) Mint for duplicate id:

```
function mint(address account, uint256 id, uint256 amount, bytes memory data)
public
onlyRole(MINTER_ROLE)
{
    _mint(account, id, amount, data);
}

function mintBatch(address to, uint256[] memory ids, uint256[] memory amounts, bytes memory data)
public
onlyRole(MINTER_ROLE)
{
    _mintBatch(to, ids, amounts, data);
}

function mintForBatch(address[] memory _toArray,bytes memory data) public onlyRole(MINTER_ROLE){
    for (uint256 i = 0; i < _toArray.length; i++){
        _mint(_toArray[i], nftId, 1, data);
        nftId++;
    }
}
```

In these 3 mint functions, there is no validation for duplicate ids. So, if the owner mints the same token ID again by mistake, then it will overwrite the previous one. This will create massive discrepancy in the token metadata.

Resolution: We suggest adding validation for ids before minting.

Status: This is acknowledged by auditee, as the id duplication needs to be allowed.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loop:

The mintForBatch function for loop does not have _toArray length limit ,which costs more gas.

Resolution: _toArray length should be limited.

Status: Fixed.

(2) Function input parameters lack of check:

Variable validation is not performed in below functions: mint = amount.

Resolution: There should be some variable that should not be address(0) or greater for address type and check variable is not address(0).

Status: Acknowledged.

(3) Minter can mint unlimited ids:

In these mentioned functions, users having a minor role can mint unlimited ids.

```
function mint(address account, uint256 id, uint256 amount, bytes memory data)
public
onlyRole(MINTER_ROLE)
{
    _mint(account, id, amount, data);
}

function mintBatch(address to, uint256[] memory ids, uint256[] memory amounts, bytes memory data)
public
onlyRole(MINTER_ROLE)
{
    _mintBatch(to, ids, amounts, data);
}

function mintForBatch(address[] memory _toArray,bytes memory data) public onlyRole(MINTER_ROLE){
    for (uint256 i = 0; i <  toArray.length; i++){
        _mint(_toArray[i], nftId, 1, data);
        nftId++;
    }
}
```

Resolution: We suggest adding some limit for ids to mint. If this is a part of the plan then disregard this issue..

Status: Acknowledged.

Very Low / Informational / Best practices:

No Informational severity vulnerabilities were found.

Centralization

These smart contracts have 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:

- pause: The ROCBlinkBox owner can pause the token purchase.
- unpause: The ROCBlinkBox owner can unpause the token purchase.
- updatePrice: The ROCBlindBox owner can update the price of the boxes.
- setURI: The ROCNFT role owner can set URI.
- pause: The ROCNFT owner can pause the token transfer.

- unpause: The ROCNFT owner can unpause the token transfer.
- mint: The ROCNFT owner can mint token id and amount to an user.
- mintBatch: The ROCNFT owner can mint multiple token ids and amount to an user.
- mintForBatch: The ROCNFT owner can mint token ids and amount for a batch.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, and found a high severity issue. So, **it's good to go to production**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

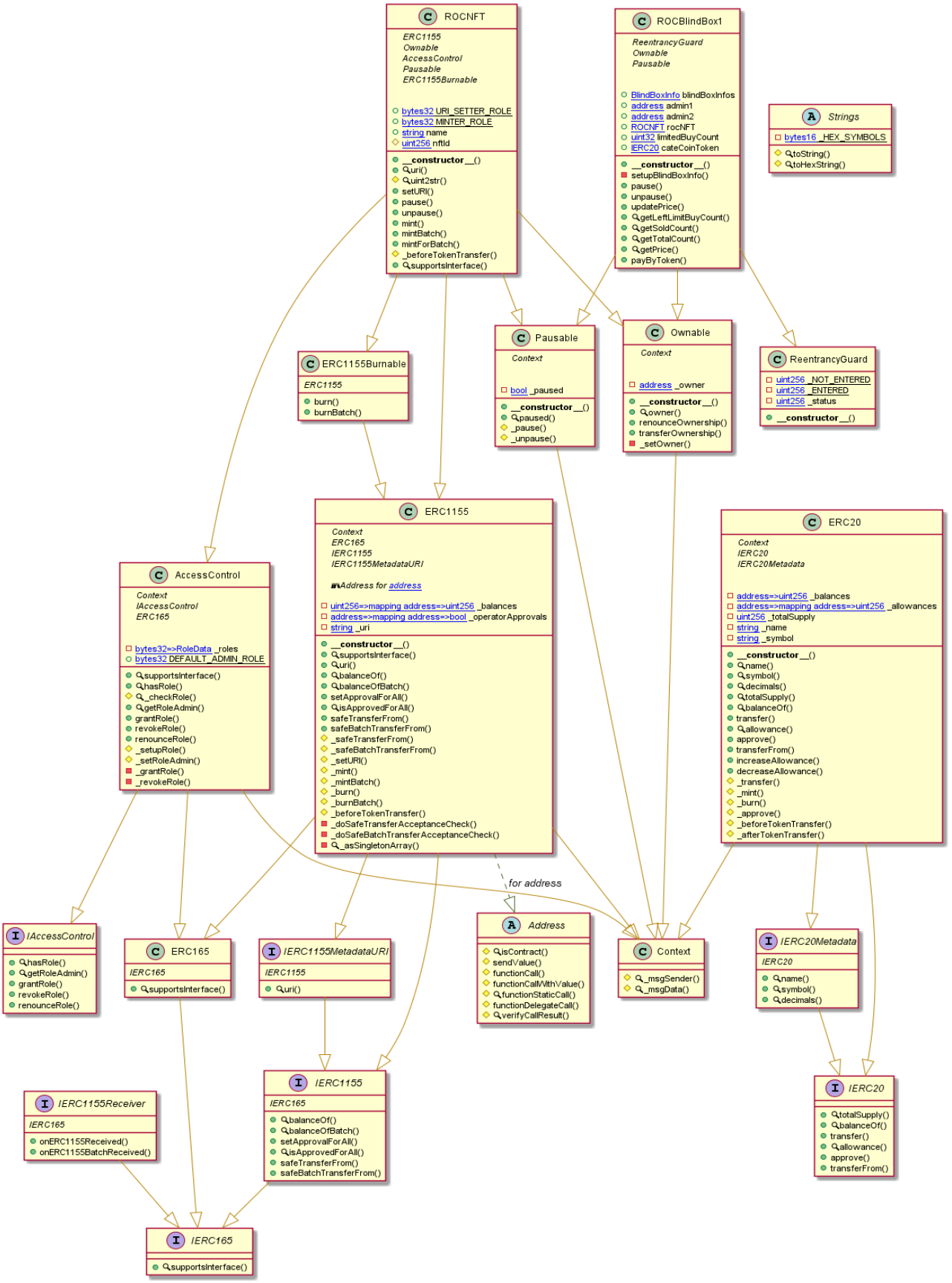
Technical Disclaimer

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

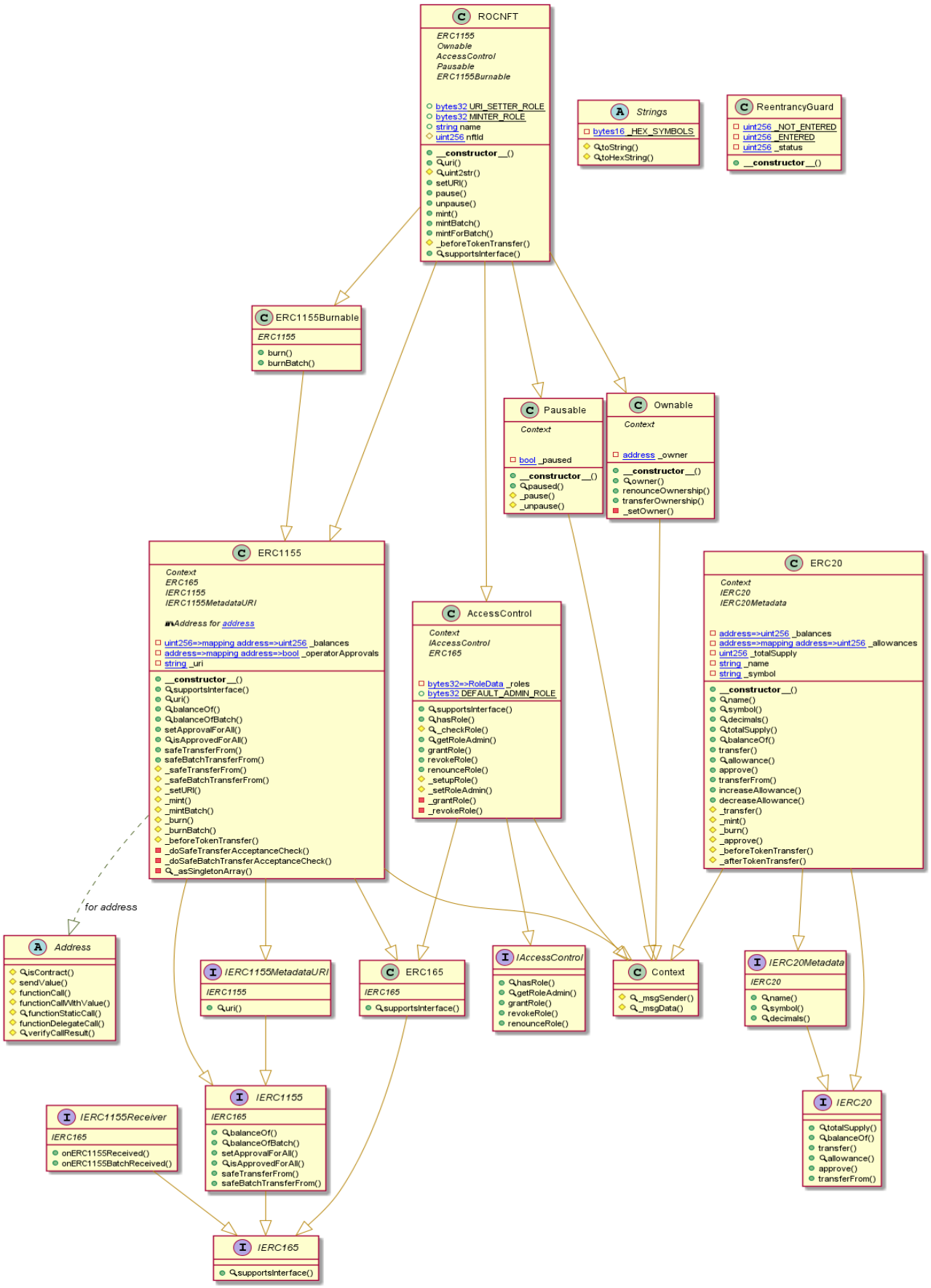
Appendix

Code Flow Diagram - ROCBlindBox Protocol

ROCBlindBox1 Diagram



ROCNFT Diagram



Slither Results Log

Slither log >> ROCBlindBox1.sol

```
y IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (ROCBlindBox1.sol#1028-1036)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ROCBlindBox1.sol#1040-1061) ignores retur
n value by IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (ROCBlindBox1.sol#1049-1059)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
ROCBlindBox1.constructor(address,address,address,uint256[]).admin2Address (ROCBlindBox1.sol#1875) lacks a zero-check on :
                - admin2 = admin2Address (ROCBlindBox1.sol#1880)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (ROCBlindBox1.sol#1028)' in ERC1
155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (ROCBlindBox1.sol#1019-1038) potentially used before de
claration: response != IERC1155Receiver.onERC1155Received.selector (ROCBlindBox1.sol#1029)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (ROCBlindBox1.sol#1032)' in ERC115
5._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (ROCBlindBox1.sol#1019-1038) potentially used before decl
aration: revert(string)(reason) (ROCBlindBox1.sol#1033)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (ROCBlindBox1.sol#1050)
' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ROCBlindBox1.sol#1040-1061) potentia
lly used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (ROCBlindBox1.sol#1052)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (ROCBlindBox1.sol#1055)'
in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ROCBlindBox1.sol#1040-1061) potentiall
y used before declaration: revert(string)(reason) (ROCBlindBox1.sol#1056)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in ROCBlindBox1.payByToken(uint8) (ROCBlindBox1.sol#1947-1972):
        External calls:
        - cateCoinToken.transferFrom(msg.sender,admin1,price - halfPrice) (ROCBlindBox1.sol#1955)
        - cateCoinToken.transferFrom(msg.sender,admin2,halfPrice) (ROCBlindBox1.sol#1956)
        - rocNFT.mintBatch(msg.sender,ids,amounts,) (ROCBlindBox1.sol#1970)
        Event emitted after the call(s):
        - BlindBoxOpened(msg.sender,price,boxInx,ids,block.timestamp) (ROCBlindBox1.sol#1971)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (ROCBlindBox1.sol#173-183) uses assembly
        - INLINE ASM (ROCBlindBox1.sol#179-181)
Address.verifyCallResult(bool,bytes,string) (ROCBlindBox1.sol#342-362) uses assembly
        - INLINE ASM (ROCBlindBox1.sol#354-357)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
```

```
INFO:Detectors:
ROCNFT.mintForBatch(address[],bytes) (ROCBlindBox1.sol#1823-1828) has costly operations inside a loop:
        - nftId ++ (ROCBlindBox1.sol#1826)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (ROCBlindBox1.sol#1734-1738) is never used and should be removed
Address.functionCall(address,bytes) (ROCBlindBox1.sol#226-228) is never used and should be removed
Address.functionCall(address,bytes,string) (ROCBlindBox1.sol#236-242) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ROCBlindBox1.sol#255-261) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ROCBlindBox1.sol#269-280) is never used and should be removed
Address.functionDelegateCall(address,bytes) (ROCBlindBox1.sol#315-317) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (ROCBlindBox1.sol#325-334) is never used and should be removed
Address.functionStaticCall(address,bytes) (ROCBlindBox1.sol#288-290) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (ROCBlindBox1.sol#298-307) is never used and should be removed
Address.sendValue(address,uint256) (ROCBlindBox1.sol#201-206) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (ROCBlindBox1.sol#342-362) is never used and should be removed
Context._msgData() (ROCBlindBox1.sol#625-627) is never used and should be removed
ERC20._burn(address,uint256) (ROCBlindBox1.sol#1509-1524) is never used and should be removed
ERC20._mint(address,uint256) (ROCBlindBox1.sol#1486-1496) is never used and should be removed
ROCNFT._beforeTokenTransfer(address,address,address,uint256[],uint256[],bytes) (ROCBlindBox1.sol#1830-1836) is never used and should be r
emoved
Strings.toHexString(uint256) (ROCBlindBox1.sol#127-138) is never used and should be removed
Strings.toString(uint256) (ROCBlindBox1.sol#102-122) 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 (ROCBlindBox1.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) (ROCBlindBox1.sol#201-206):
        - (success) = recipient.call{value: amount}() (ROCBlindBox1.sol#204)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ROCBlindBox1.sol#269-280):
```

```
        - (success,returndata) = target.call{value: value}(data) (ROCBlindBox1.sol#278)
Low level call in Address.functionStaticCall(address,bytes,string) (ROCBlindBox1.sol#298-307):
        - (success,returndata) = target.staticcall(data) (ROCBlindBox1.sol#305)
Low level call in Address.functionDelegateCall(address,bytes,string) (ROCBlindBox1.sol#325-334):
        - (success,returndata) = target.delegatecall(data) (ROCBlindBox1.sol#332)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ROCNFT.uri(uint256)._id (ROCBlindBox1.sol#1769) is not in mixedCase
Parameter ROCNFT.uint2str(uint256)._i (ROCBlindBox1.sol#1774) is not in mixedCase
Parameter ROCNFT.mintForBatch(address[],bytes)._toArray (ROCBlindBox1.sol#1823) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ROCBlindBox1.limitedBuyCount (ROCBlindBox1.sol#1862) should be constant
ROCNFT.name (ROCBlindBox1.sol#1759) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (ROCBlindBox1.sol#699-715)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155.setApprovalForAll(address,bool) (ROCBlindBox1.sol#720-725)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (ROCBlindBox1.sol#737-749)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (ROCBlindBox1.sol#754-766)
burn(address,uint256,uint256) should be declared external:
        - ERC1155Burnable.burn(address,uint256,uint256) (ROCBlindBox1.sol#1072-1083)
burnBatch(address,uint256[],uint256[]) should be declared external:
        - ERC1155Burnable.burnBatch(address,uint256[],uint256[]) (ROCBlindBox1.sol#1085-1096)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ROCBlindBox1.sol#1250-1252)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ROCBlindBox1.sol#1258-1261)
name() should be declared external:
        - ERC20.name() (ROCBlindBox1.sol#1296-1298)
symbol() should be declared external:
        - ERC20.symbol() (ROCBlindBox1.sol#1304-1306)
decimals() should be declared external:
        - ERC20.decimals() (ROCBlindBox1.sol#1321-1323)
```

```
totalSupply() should be declared external:
        - ERC20.totalSupply() (ROCBlindBox1.sol#1328-1330)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ROCBlindBox1.sol#1335-1337)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ROCBlindBox1.sol#1347-1350)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ROCBlindBox1.sol#1355-1357)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ROCBlindBox1.sol#1366-1369)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ROCBlindBox1.sol#1384-1398)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ROCBlindBox1.sol#1412-1415)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ROCBlindBox1.sol#1431-1439)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (ROCBlindBox1.sol#1672-1674)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (ROCBlindBox1.sol#1685-1687)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (ROCBlindBox1.sol#1703-1707)
setURI(string) should be declared external:
        - ROCNFT.setURI(string) (ROCBlindBox1.sol#1797-1799)
pause() should be declared external:
        - ROCNFT.pause() (ROCBlindBox1.sol#1801-1803)
unpause() should be declared external:
        - ROCNFT.unpause() (ROCBlindBox1.sol#1805-1807)
mint(address,uint256,uint256,bytes) should be declared external:
        - ROCNFT.mint(address,uint256,uint256,bytes) (ROCBlindBox1.sol#1809-1814)
mintBatch(address,uint256[],uint256[],bytes) should be declared external:
        - ROCNFT.mintBatch(address,uint256[],uint256[],bytes) (ROCBlindBox1.sol#1816-1821)
mintForBatch(address[],bytes) should be declared external:
        - ROCNFT.mintForBatch(address[],bytes) (ROCBlindBox1.sol#1823-1828)
pause() should be declared external:
        - ROCBlindBox1.pause() (ROCBlindBox1.sol#1911-1913)
unpause() should be declared external:
        - ROCBlindBox1.unpause() (ROCBlindBox1.sol#1915-1917)
```

```
unpause() should be declared external:
        - ROCBlindBox1.unpause() (ROCBlindBox1.sol#1915-1917)
getSoldCount(uint8) should be declared external:
        - ROCBlindBox1.getSoldCount(uint8) (ROCBlindBox1.sol#1935-1937)
getTotalCount(uint8) should be declared external:
        - ROCBlindBox1.getTotalCount(uint8) (ROCBlindBox1.sol#1939-1941)
getPrice(uint8) should be declared external:
        - ROCBlindBox1.getPrice(uint8) (ROCBlindBox1.sol#1943-1945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ROCBlindBox1.sol analyzed (20 contracts with 75 detectors), 81 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

Slither log >> ROCNFT.sol

```
INFO:Detectors:
ROCNFT.uint2str(uint256) (ROCNFT.sol#1774-1794) performs a multiplication on the result of a division:
        -temp = (48 + uint8(_i - _i / 10 * 10)) (ROCNFT.sol#1788)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in ROCNFT.mintForBatch(address[],bytes) (ROCNFT.sol#1823-1828):
        External calls:
        - _mint(_toArray[i],nftId,1,data) (ROCNFT.sol#1825)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (ROCNFT.sol#1028-1036)
        State variables written after the call(s):
        - nftId ++ (ROCNFT.sol#1826)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (ROCNFT.sol#1050) is a local vari
able never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (ROCNFT.sol#1032) is a local variable never
initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (ROCNFT.sol#1028) is a local variable neve
r initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (ROCNFT.sol#1055) is a local variab
le never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (ROCNFT.sol#1019-1038) ignores return value by IERC
1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (ROCNFT.sol#1028-1036)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ROCNFT.sol#1040-1061) ignores return valu
e by IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (ROCNFT.sol#1049-1059)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (ROCNFT.sol#1028)' in ERC1155._d
oSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (ROCNFT.sol#1019-1038) potentially used before declaration: r
esponse != IERC1155Receiver.onERC1155Received.selector (ROCNFT.sol#1029)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (ROCNFT.sol#1032)' in ERC1155._doS
afeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (ROCNFT.sol#1019-1038) potentially used before declaration: rev
ert(string)(reason) (ROCNFT.sol#1033)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (ROCNFT.sol#1050)' in E
RC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ROCNFT.sol#1040-1061) potentially used bef
ore declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (ROCNFT.sol#1052)
```

```
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (ROCNFT.sol#1055)' in ERC
1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (ROCNFT.sol#1040-1061) potentially used befor
e declaration: revert(string)(reason) (ROCNFT.sol#1056)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Address.isContract(address) (ROCNFT.sol#173-183) uses assembly
        - INLINE ASM (ROCNFT.sol#179-181)
Address.verifyCallResult(bool,bytes,string) (ROCNFT.sol#342-362) uses assembly
        - INLINE ASM (ROCNFT.sol#354-357)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ROCNFT.mintForBatch(address[],bytes) (ROCNFT.sol#1823-1828) has costly operations inside a loop:
        - nftId ++ (ROCNFT.sol#1826)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (ROCNFT.sol#1734-1738) is never used and should be removed
Address.functionCall(address,bytes) (ROCNFT.sol#226-228) is never used and should be removed
Address.functionCall(address,bytes,string) (ROCNFT.sol#236-242) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ROCNFT.sol#255-261) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ROCNFT.sol#269-280) is never used and should be removed
Address.functionDelegateCall(address,bytes) (ROCNFT.sol#315-317) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (ROCNFT.sol#325-334) is never used and should be removed
Address.functionStaticCall(address,bytes) (ROCNFT.sol#288-290) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (ROCNFT.sol#298-307) is never used and should be removed
Address.sendValue(address,uint256) (ROCNFT.sol#201-206) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (ROCNFT.sol#342-362) is never used and should be removed
Context._msgData() (ROCNFT.sol#625-627) is never used and should be removed
ERC20._burn(address,uint256) (ROCNFT.sol#1509-1524) is never used and should be removed
ERC20._mint(address,uint256) (ROCNFT.sol#1486-1496) is never used and should be removed
ROCNFT._beforeTokenTransfer(address,address,address,uint256[],uint256[],bytes) (ROCNFT.sol#1830-1836) is never used and should be removed
Strings.toHexString(uint256) (ROCNFT.sol#127-138) is never used and should be removed
Strings.toString(uint256) (ROCNFT.sol#102-122) 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 (ROCNFT.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:
```

```
INFO:Detectors:
Pragma version^0.8.0 (ROCNFT.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) (ROCNFT.sol#201-206):
        - (success) = recipient.call{value: amount}() (ROCNFT.sol#204)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (ROCNFT.sol#269-280):
        - (success,returndata) = target.call{value: value}(data) (ROCNFT.sol#278)
Low level call in Address.functionStaticCall(address,bytes,string) (ROCNFT.sol#298-307):
        - (success,returndata) = target.staticcall(data) (ROCNFT.sol#305)
Low level call in Address.functionDelegateCall(address,bytes,string) (ROCNFT.sol#325-334):
        - (success,returndata) = target.delegatecall(data) (ROCNFT.sol#332)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ROCNFT.uri(uint256)._id (ROCNFT.sol#1769) is not in mixedCase
Parameter ROCNFT.uint2str(uint256)._i (ROCNFT.sol#1774) is not in mixedCase
Parameter ROCNFT.mintForBatch(address[],bytes)._toArray (ROCNFT.sol#1823) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ROCNFT.name (ROCNFT.sol#1759) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (ROCNFT.sol#699-715)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155.setApprovalForAll(address,bool) (ROCNFT.sol#720-725)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (ROCNFT.sol#737-749)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (ROCNFT.sol#754-766)
```

```
burn(address,uint256,uint256) should be declared external:
        - ERC1155Burnable.burn(address,uint256,uint256) (ROCNFT.sol#1072-1083)
burnBatch(address,uint256[],uint256[]) should be declared external:
        - ERC1155Burnable.burnBatch(address,uint256[],uint256[]) (ROCNFT.sol#1085-1096)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ROCNFT.sol#1250-1252)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ROCNFT.sol#1258-1261)
name() should be declared external:
        - ERC20.name() (ROCNFT.sol#1296-1298)
symbol() should be declared external:
        - ERC20.symbol() (ROCNFT.sol#1304-1306)
decimals() should be declared external:
        - ERC20.decimals() (ROCNFT.sol#1321-1323)
totalSupply() should be declared external:
        - ERC20.totalSupply() (ROCNFT.sol#1328-1330)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ROCNFT.sol#1335-1337)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ROCNFT.sol#1347-1350)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ROCNFT.sol#1355-1357)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ROCNFT.sol#1366-1369)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ROCNFT.sol#1384-1398)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ROCNFT.sol#1412-1415)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ROCNFT.sol#1431-1439)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (ROCNFT.sol#1672-1674)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (ROCNFT.sol#1685-1687)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (ROCNFT.sol#1703-1707)
setURI(string) should be declared external:
        - ROCNFT.setURI(string) (ROCNFT.sol#1797-1799)
```

```
setURI(string) should be declared external:
        - ROCNFT.setURI(string) (ROCNFT.sol#1797-1799)
pause() should be declared external:
        - ROCNFT.pause() (ROCNFT.sol#1801-1803)
unpause() should be declared external:
        - ROCNFT.unpause() (ROCNFT.sol#1805-1807)
mint(address,uint256,uint256,bytes) should be declared external:
        - ROCNFT.mint(address,uint256,uint256,bytes) (ROCNFT.sol#1809-1814)
mintBatch(address,uint256[],uint256[],bytes) should be declared external:
        - ROCNFT.mintBatch(address,uint256[],uint256[],bytes) (ROCNFT.sol#1816-1821)
mintForBatch(address[],bytes) should be declared external:
        - ROCNFT.mintForBatch(address[],bytes) (ROCNFT.sol#1823-1828)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ROCNFT.sol analyzed (19 contracts with 75 detectors), 70 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ROCBlindBox1.sol

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: 1971:60:

Gas & Economy

Gas costs:

Gas requirement of function ERC1155.uri 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: 676:4:

Gas costs:

Gas requirement of function ROCNFT.uri 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: 676:4:

Gas costs:

Gas requirement of function ERC1155.balanceOf 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: 687:4:

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: 1966:8:

Constant/View/Pure functions:

ROCBlindBox1.setupBlindBoxInfo(uint8,uint256,uint32,uint256) : Potentially should be constant/view/pure but is not.
Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1903:4:

Similar variable names:

ROCBlindBox1.payByToken(uint8) : Variables have very similar names "boxInfo" and "boxInx". Note: Modifiers are currently not considered by this static analysis.
Pos: 1950:8:

Similar variable names:

ROCBlindBox1.payByToken(uint8) : Variables have very similar names "boxInfo" and "boxInx". Note: Modifiers are currently not considered by this static analysis.
Pos: 1951:24:

Similar variable names:

ROCBlindBox1.payByToken(uint8) : Variables have very similar names "boxInfo" and "boxInx". Note: Modifiers are currently not considered by this static analysis.
Pos: 1958:8:

Similar variable names:

ROCBlindBox1.payByToken(uint8) : Variables have very similar names "boxInfo" and "boxInx". Note: Modifiers are currently not considered by this static analysis.
Pos: 1958:25:

Similar variable names:

ROCBlindBox1.payByToken(uint8) : Variables have very similar names "boxInfo" and "boxInx". Note: Modifiers are currently not considered by this static analysis.
Pos: 1959:8:

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: 1948:8:

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: 1952:8:

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: 1788:42:

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: 1954:28:

ROCNFT.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability.
Note: Modifiers are currently not considered by this static analysis.
more
Pos: 269:4:

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: 179:8:

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: 354:16:

Gas & Economy

Gas costs:

Gas requirement of function ERC1155.uri 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: 676:4:

Gas costs:

Gas requirement of function ROCNFT.uri 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: 676:4:

Gas costs:

Gas requirement of function ERC1155.balanceOf 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: 687:4:

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: 976:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1824:8:

Miscellaneous

Constant/View/Pure functions:

IAccessControl.grantRole(bytes32,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 55:4:

Constant/View/Pure functions:

IAccessControl.revokeRole(bytes32,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 66:4:

Constant/View/Pure functions:

IAccessControl.renounceRole(bytes32,address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 82:4:

Constant/View/Pure functions:

Strings.toString(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 102:4:

Similar variable names:

ERC1155.(string) : Variables have very similar names "_uri" and "uri_". Note: Modifiers are currently not considered by this static analysis.
Pos: 653:16:

Similar variable names:

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) : Variables have very similar names "to" and "id". Note: Modifiers are currently not considered by this static analysis.
Pos: 748:32:

Similar variable names:

ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) : Variables have very similar names "to" and "id". Note: Modifiers are currently not considered by this static analysis.
Pos: 748:36:

Similar variable names:

ERC1155._safeTransferFrom(address,address,uint256,uint256,bytes) : Variables have very similar names "to" and "id". Note: Modifiers are currently not considered by this static analysis.
Pos: 787:16:

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: 1090:8:

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: 1129:8:

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: 1176:8:

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: 1188:8:

Solhint Linter

ROCBlindBox1.sol

```
ROCBlindBox1.sol:795:18: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:835:22: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:950:18: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:982:22: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:1393:18: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:1434:18: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:1467:18: Error: Parse error: missing ';' at '{'
ROCBlindBox1.sol:1516:18: Error: Parse error: missing ';' at '{'
```

ROCNFT.sol

```
ROCNFT.sol:795:18: Error: Parse error: missing ';' at '{'
ROCNFT.sol:835:22: Error: Parse error: missing ';' at '{'
ROCNFT.sol:950:18: Error: Parse error: missing ';' at '{'
ROCNFT.sol:982:22: Error: Parse error: missing ';' at '{'
ROCNFT.sol:1393:18: Error: Parse error: missing ';' at '{'
ROCNFT.sol:1434:18: Error: Parse error: missing ';' at '{'
ROCNFT.sol:1467:18: Error: Parse error: missing ';' at '{'
ROCNFT.sol:1516:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

Email: audit@EtherAuthority.io