Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Changeblock Protocol
Website: changeblock.com
Platform: Polygon Network
Language: Solidity
Date: July 5th, 2022

Table of contents

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 Changeblock Protocol to perform the Security audit of the Changeblock 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 July 5th, 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

Changeblock Protocol is a smart contract having functions like: mint, burn, withdraw, rebalance, approve, deploy, etc. The Changeblock Protocol contract inherits the ERC20, Ownable, IERC20, IERC721 standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

Name	**Code Review and Security Analysis Report for Changeblock Protocol Smart Contracts**
Platform	**Polygon / Solidity**
File 1	ChangeblockMarketplace.sol
File 1 MD5 Hash	BC7BCA9F16B05E5DBA3E33EB1A80C6E1
Updated File 1 MD5 Hash	51B24F82AFC5D3757D59FA81DCC65FF1
File 2	CBLKFixed.sol
File 2 MD5 Hash	2BCA918EE75295DDC1C80D4E6AC5CC33
File 3	CBLKUnfixed.sol
File 3 MD5 Hash	2B61141C4DE91B3EBF6FD3E0631664EA
File 4	CBT.sol

File 4 MD5 Hash	F91FEB29A263F80BD43C5B7B47326218
File 5	CBTFixedFactory.sol
File 5 MD5 Hash	62BC9E07FEFE48AB4267E73A01DA926D
File 6	CBTUnfixedFactory.sol
File 6 MD5 Hash	1B990FF508FF3498E877581EFFC5844F
File 7	CBTFactory.sol
File 7 MD5 Hash	8E1BA286D7032838DA4981C985B91FA1
Audit Date	July 5th,2022

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 ChangeblockMarketplace.sol • ChangeblockMarketplace can represent one or more ERC20 tokens listed for-sale. • ChangeblockMarketplace has functions like buyERC20, listERC20, etc. • Change Block Marketplace to list and purchase ERC20/ERC721 tokens.	**YES, This is valid.**
File 2 CBLKFixed.sol • Name: CBLK • Symbol: CBLK • CBLKFixed tokens represent a share of an underlying index of CBTs.	**YES, This is valid.**
File 3 CBLKUnfixed.sol • Name: CBLK • Symbol: CBLK • CBLKUnfixed tokens represents a share of an index of CBTs curated by an owner.	**YES, This is valid.**
File 4 CBT.sol • Name: CBT • Symbol: CBT	**YES, This is valid.**
File 5 CBTFixedFactory.sol • CBLKFixedFactory has functions like: approve, etc.	**YES, This is valid.**
File 6 CBTUnfixedFactory.sol • CBLKUnfixedFactory has functions like: approve, etc.	**YES, This is valid.**
File 7 CBTFactory.sol • CBTFactory has functions like: approve, deploy.	**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 1 critical, 0 high, 1 medium and 2 low and some very low level issues.
All the major issues have been fixed / acknowledged in the revised code.

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	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 7 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 Changeblock 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 Changeblock Protocol.

The Changeblock 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 a Changeblock 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. 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 www.changeblock.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

ChangeblockMarketplace.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	_transferOwnership	internal	Passed	No Issue
7	getListing	read	Passed	No Issue
8	onlyBuyer	modifier	Passed	No Issue
9	onlySeller	modifier	Passed	No Issue
10	buyERC20	write	Seller can bid/buy his own listing	Refer Audit Findings
11	buyERC721	write	Seller can bid/buy his own listing	Refer Audit Findings
12	listERC20	write	access only Seller	No Issue
13	listERC721	write	access only Seller	No Issue
14	delistERC20	write	Seller can delist listing after bidder bid on listing	Refer Audit Findings
15	delistERC721	write	Seller can delist listing after bidder bid on listing	Refer Audit Findings
16	updateERC20Price	external	Passed	No Issue
17	updateERC721Price	external	Passed	No Issue
18	bid	write	Seller can bid/buy his own listing	Refer Audit Findings
19	withdrawBid	write	Passed	No Issue
20	acceptBid	write	Passed	No Issue
21	setSellers	write	Infinite loops possibility	Refer Audit Findings
22	setBuyers	write	Infinite loops possibility	Refer Audit Findings
23	setFeeNumerator	external	Fee validation, Critical operation lacks event log	Refer Audit Findings
24	setFeeDenominator	external	Fee validation, Critical operation lacks event log	Refer Audit Findings
25	setBuyerWhitelisting	external	Critical operation lacks event log	Refer Audit Findings
26	_removeBid	internal	Passed	No Issue

CBLKFixed.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_spendAllowance	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue
19	_afterTokenTransfer	internal	Passed	No Issue
20	deposit	write	Passed	No Issue
21	withdraw	write	Passed	No Issue

CBLKUnfixed.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_spendAllowance	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue

19	_afterTokenTransfer	internal	Passed	No Issue
20	owner	read	Passed	No Issue
21	onlyOwner	modifier	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue
24	_transferOwnership	internal	Passed	No Issue
25	rebalance	external	access only Owner	No Issue
26	withdraw	write	Passed	No Issue
27	_unregisterToken	internal	Passed	No Issue

CBT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_spendAllowance	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue
19	_afterTokenTransfer	internal	Passed	No Issue
20	owner	read	Passed	No Issue
21	onlyOwner	modifier	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue
24	_transferOwnership	internal	Passed	No Issue
25	mint	write	Unlimited minting	Refer Audit Findings
26	burn	write	access only Owner	No Issue

CBTFixedFactory.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	_transferOwnership	internal	Passed	No Issue
7	approve	write	access only Owner	No Issue
8	deploy	write	Passed	No Issue

CBTUnfixedFactory.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	_transferOwnership	internal	Passed	No Issue
7	approve	write	access only Owner	No Issue
8	deploy	write	Passed	No Issue

CBTFactory.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	_transferOwnership	internal	Passed	No Issue
7	approve	write	access only Owner	No Issue
8	deploy	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

Critical Severity

(1) Transfer amount wrong:- **ChangeblockMarketplace.sol**

```
/// @notice Call to purchase a listed ERC721 token.
/// @dev Token price is included as a parameter to prevent price manipulation.
/// @param listingId The ID of the listing whose token the caller wishes to purchase.
/// @param price The price at which the caller wishes to purchase the token.
function buyERC721(uint256 listingId, uint256 price) public onlyBuyer {
    ERC721Listing memory listing = ERC721Listings[listingId];
    require(listing.price == price, 'Listed price not equal to input price');
    uint256 fee = (listing.price * FEE_NUMERATOR) / FEE_DENOMINATOR;
    IERC20(listing.currency).transferFrom(msg.sender, listing.vendor, listing.price);
    IERC20(listing.currency).transferFrom(msg.sender, TREASURY, fee);
    IERC721(listing.product).safeTransferFrom(address(this), msg.sender, listing.id);
    emit ERC721Sale(listingId, price, msg.sender);
}
```

In buyERC721(), function amount transfers to the vendor the listing's price and fees are also cut from the buyer account. So the buyer has to pay the listing price + fee.

Resolution: We suggest correcting the logic for price calculation, so that buyers just need to transfer the listing price and fees should be cut from that price only.

Status: Fixed

High Severity

No High severity vulnerabilities were found.

Medium

(1) Fee validation:- **ChangeblockMarketplace.sol**

```
function setFeeNumerator(uint256 feeNumerator) external onlyOwner {
    FEE_NUMERATOR = feeNumerator;
}

function setFeeDenominator(uint256 feeDenominator) external onlyOwner {
    FEE_DENOMINATOR = feeDenominator;
}
```

The owner can set the fee percentage to 100%. so the vendor cannot get any amount for his ERC20 and ERC721 Token.

Resolution: We suggest using some maximum limit for fees.

Status: Acknowledged

Low

(1) Seller can delist listing even after bidder bid on listing:**ChangeblockMarketplace.sol**

There are functions delistERC20() and delistERC721(), In these functions, sellers can remove listings. but if the bidder, Bid on listing, Then is seller delist listing, So bidder's token will collect in contract, After that no way to withdraw that token by bidder.

Resolution: This logic will be incorrect, if seller will delist listing in between listing on public and bidder's bid on Listing, So bidder's token will collect in contract. If this is a part of the plan then disregard this issue.

(2) Function input parameters lack of check:**ChangeblockMarketplace.sol**

```
function setFeeNumerator(uint256 feeNumerator) external onlyOwner {
    FEE_NUMERATOR = feeNumerator;
}

function setFeeDenominator(uint256 feeDenominator) external onlyOwner {
    FEE_DENOMINATOR = feeDenominator;
}
```

FEE_DENOMINATOR can be greater than FEE_NUMERATOR.

Resolution: We suggest validating for the FEE_DENOMINATOR and FEE_NUMERATOR before setting value for them.

Status: Acknowledged

(3) Seller can bid/buy his own listing:**ChangeblockMarketplace.sol**

Listing creators can bid/buy his own item. This is meaningless.

Resolution: We suggest not allowing the listing creator to bid/buy his own listing. If this is a part of the plan then disregard this issue.

Status: Acknowledged

Very Low / Informational / Best practices:

(1) Assign default value:**- ChangeblockMarketplace.sol**

```
bool buyerWhitelisting = false;
```

All the boolean variables have default as “false”. So, no need to explicitly assign the value. Although this does not raise any security or logical vulnerability, it is a good practice to avoid setting empty/default values explicitly.

Resolution: We suggest removing the default assignment.

Status: Fixed

(2) Unused event:**- ChangeblockMarketplace.sol**

```
event Removal(uint256 indexed listingId);
```

Removal event is defined but not used in code.

Resolution: We suggest removing unused events.

Status: Fixed

(3) Critical operation lacks event log:

Missing event log for:

- setFeeNumerator()
- setFeeDenominator()
- setBuyerWhitelisting()

Resolution: Please write an event log for listed events.

(4) Infinite loops possibility:**ChangeblockMarketplace.sol**

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.

- setSellers() - targets.length.
- setBuyers() - targets.length.

(5) Unlimited minting: **CBT.sol**

Token minting without any maximum limit is considered inappropriate for tokenomics. We recommend placing some limit on token minting to mitigate this issue.

Resolution: We suggest setting some limit for mint tokens.

(6) Owner can burn anyone's tokens: **CBT.sol**

Only the owner of the tokens is allowed to burn his tokens. But here the contract owner can burn anyone's tokens.

Resolution: We suggest confirming the burn functionality.

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:

- approve: CBTFactory owner can approve target address.
- approve: CBTUnfixedFactory owner can approve target address.
- approve: CBTKFixedFactory owner can approve target address.
- mint: CBT owner can mint an amount of CBT to a user's wallet.
- burn: CBT owners can burn an amount of CBT from a user's wallet.
- rebalance: CBLKUnfixed owner can add or remove CBTs to the CBLK's underlying tokens.
- setSellers: ChangeblockMarketplace owner can approve account(s) to allow them to create listings on the platform default is of course unapproved (false).
- setBuyers: ChangeblockMarketplace owner can set buyer whitelisting has been enabled.
- setFeeNumerator: ChangeblockMarketplace owner can set fee numerator.
- setFeeDenominator: ChangeblockMarketplace owner can set fee denominator.
- setBuyerWhitelisting: ChangeblockMarketplace owner can set buyer whitelisting address has been enabled.

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 some major issues in the smart contracts, but those issues have been resolved / acknowledged in the revised code. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

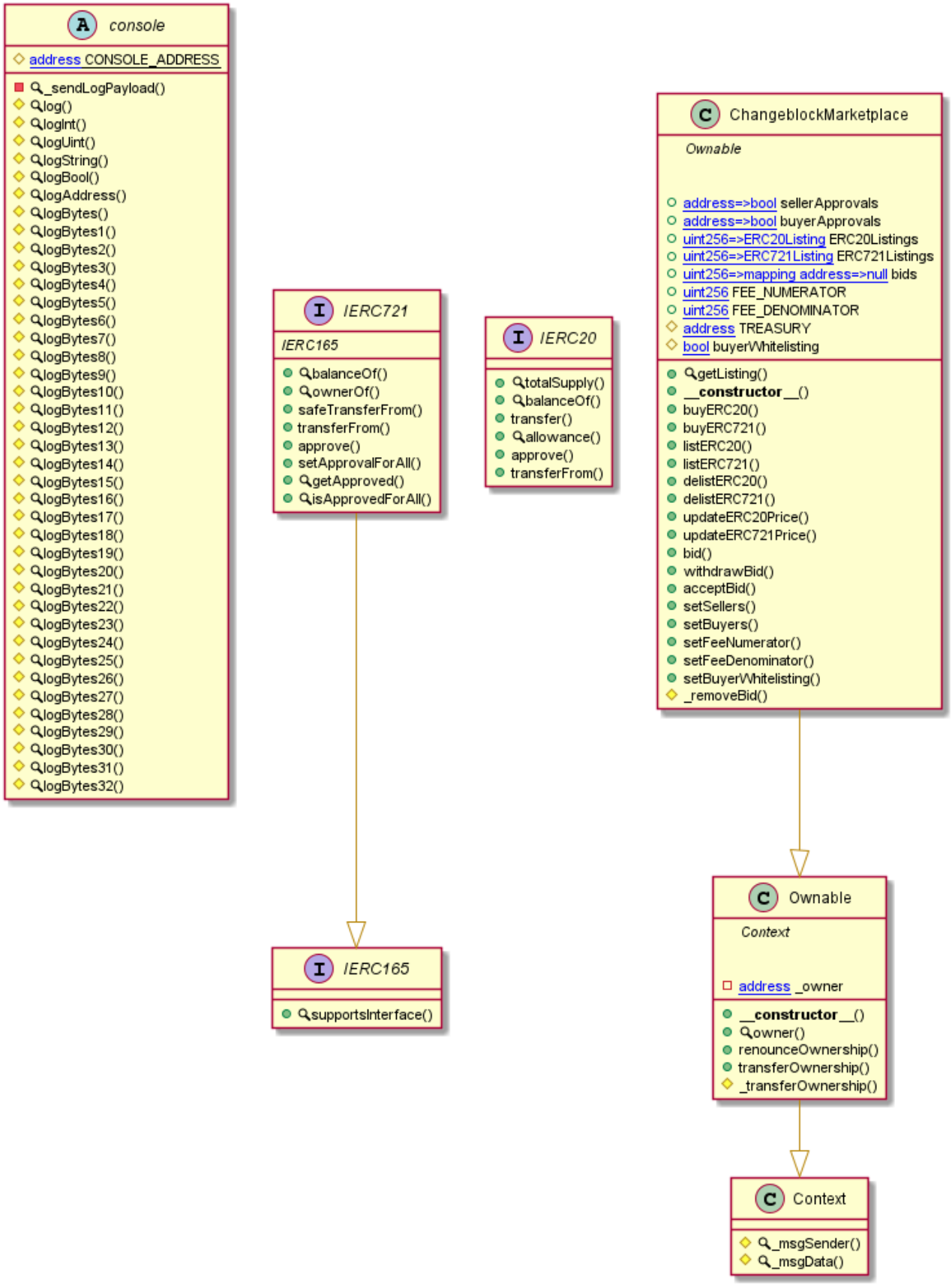
Technical Disclaimer

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

Appendix

Code Flow Diagram - Changeblock Protocol

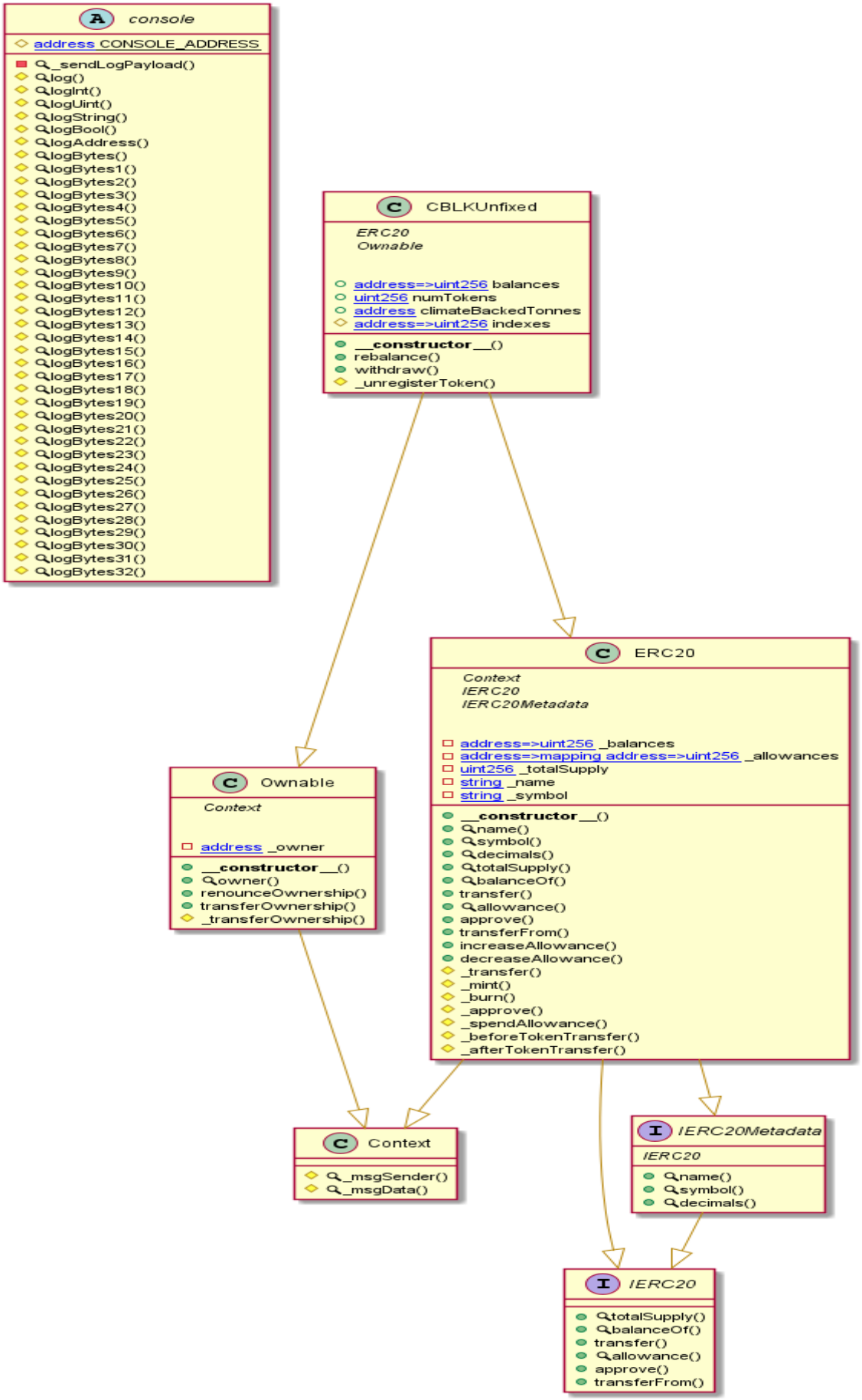
ChangeblockMarketplace Diagram



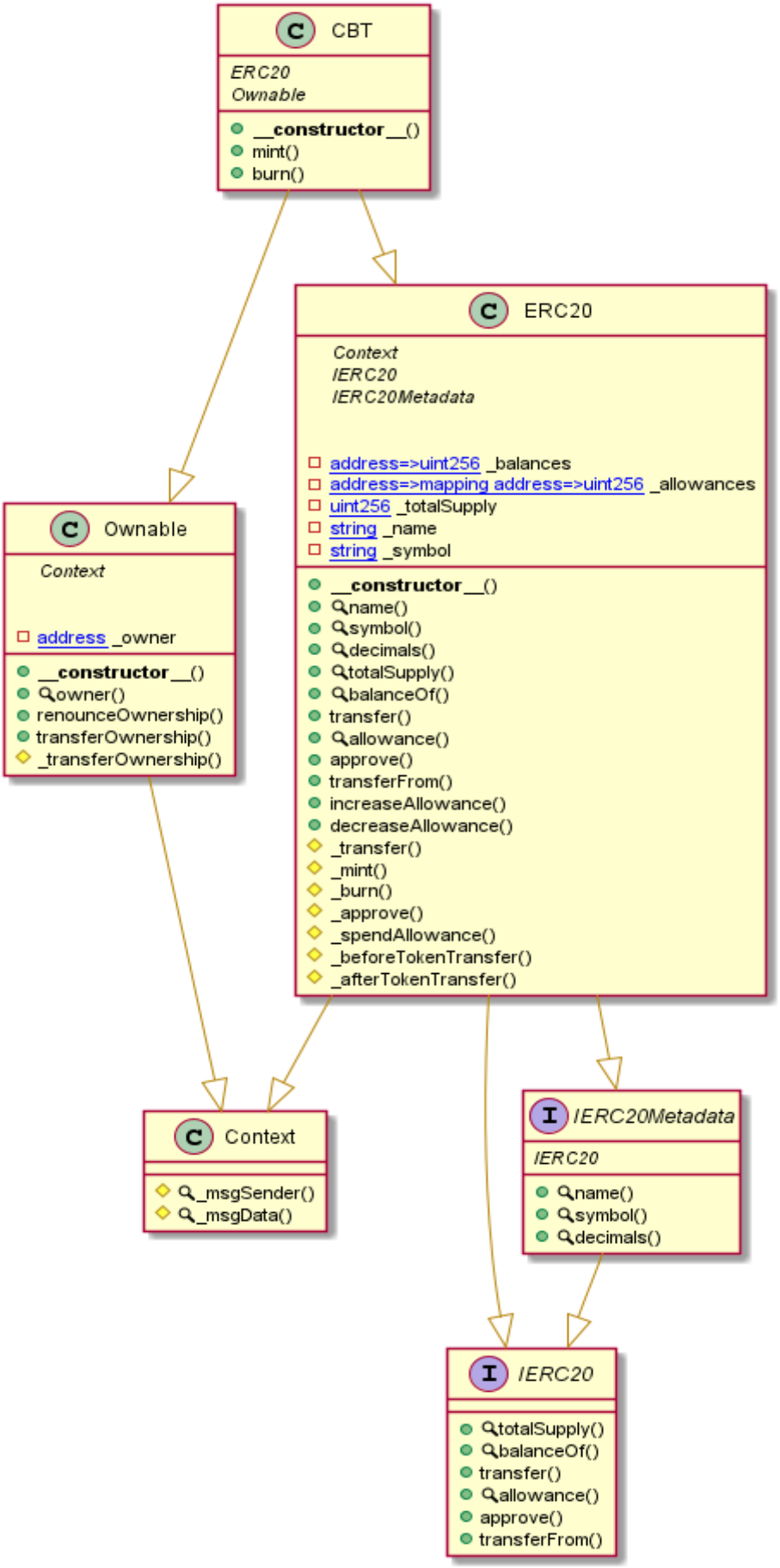
CBLKFixed Diagram

Email: audit@EtherAuthority.io

CBLKUnfixed Diagram

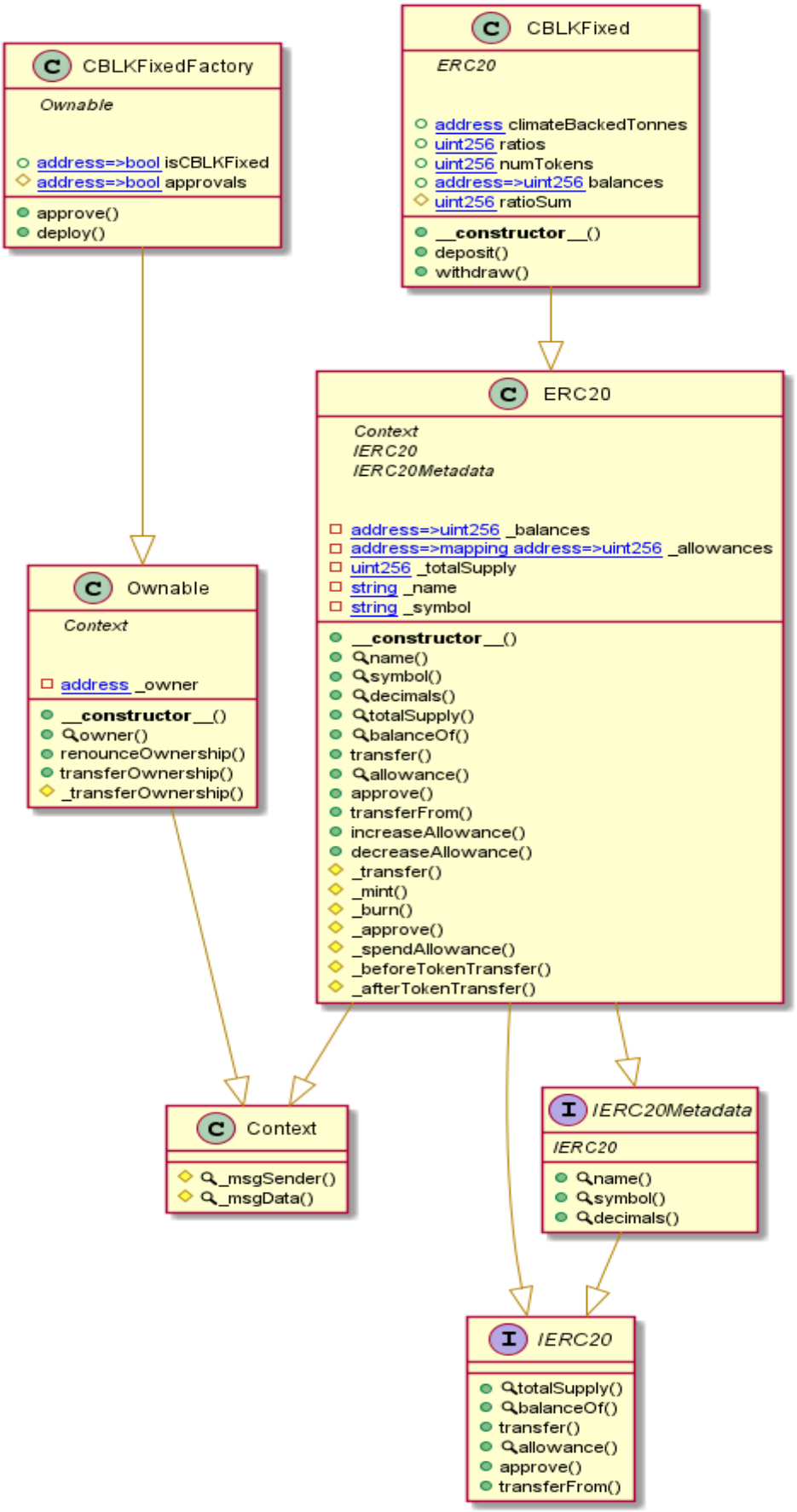


CBT Diagram

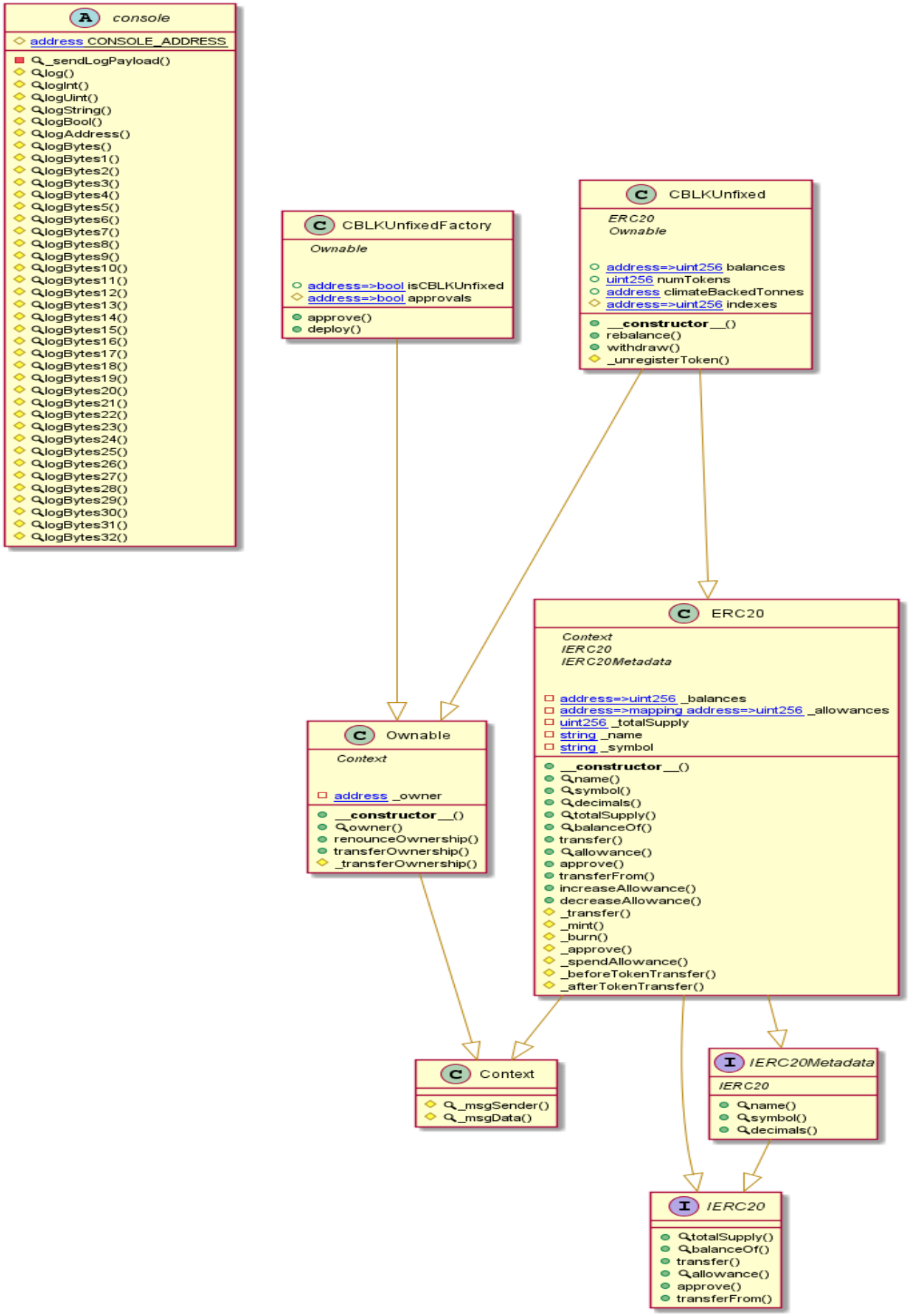


Email: audit@EtherAuthority.io

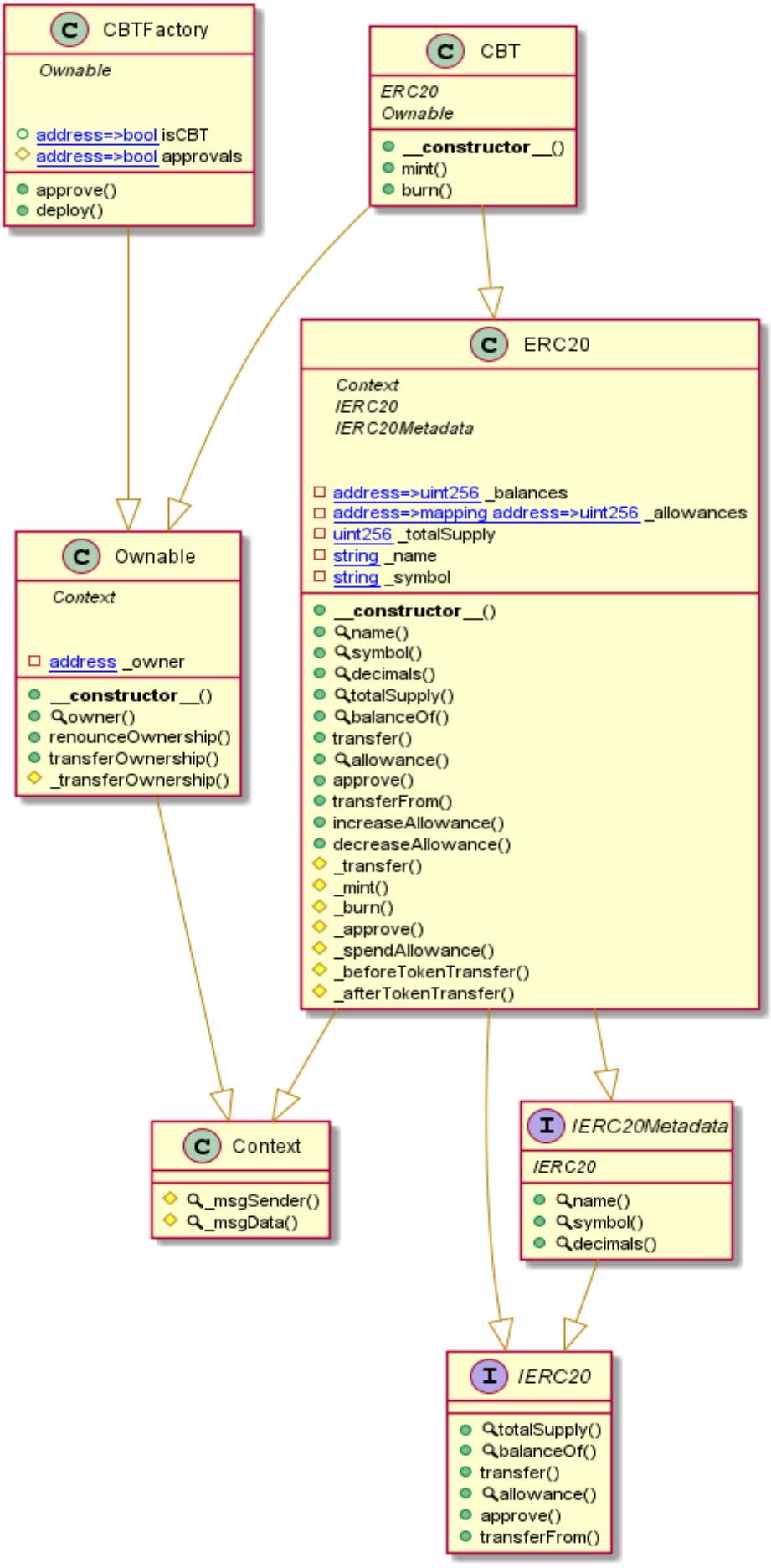
CBTFixedFactory Diagram



CBTUnfixedFactory Diagram



CBTFactory Diagram



Slither Results Log

Slither log >> ChangeblockMarketplace.sol

```
INFO:Detectors:
ChangeblockMarketplace.constructor(uint256,uint256,address).treasury (ChangeblockMarketplace.sol#1966) lacks a zero-check on :
                - TREASURY = treasury (ChangeblockMarketplace.sol#1970)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ChangeblockMarketplace.bid(uint256,uint256,uint256) (ChangeblockMarketplace.sol#2105-2119):
        External calls:
        - IERC20(ERC20Listings[listingId].currency).transferFrom(msg.sender,address(this),payment) (ChangeblockMarketplace.sol
#2110)
        State variables written after the call(s):
        - bids[listingId][msg.sender].push(Bid(quantity,payment)) (ChangeblockMarketplace.sol#2118)
Reentrancy in ChangeblockMarketplace.listERC20(uint256,uint256,address,address) (ChangeblockMarketplace.sol#2019-2036):
        External calls:
        - IERC20(product).transferFrom(msg.sender,address(this),amount) (ChangeblockMarketplace.sol#2025)
        State variables written after the call(s):
        - ERC20Listings[listingId] = ERC20Listing(amount + ERC20Listings[listingId].amount,price,msg.sender,product,currency)
(ChangeblockMarketplace.sol#2027-2033)
Reentrancy in ChangeblockMarketplace.listERC721(uint256,uint256,address,address) (ChangeblockMarketplace.sol#2042-2053):
        External calls:
        - IERC721(product).transferFrom(msg.sender,address(this),id) (ChangeblockMarketplace.sol#2048)
        State variables written after the call(s):
        - ERC721Listings[listingId] = ERC721Listing(id,price,msg.sender,product,currency) (ChangeblockMarketplace.sol#2050)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Pragma version^0.8.0 (ChangeblockMarketplace.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:
Contract console (ChangeblockMarketplace.sol#5-1533) is not in CapWords
Variable ChangeblockMarketplace.ERC20Listings (ChangeblockMarketplace.sol#1874) is not in mixedCase
Variable ChangeblockMarketplace.ERC721Listings (ChangeblockMarketplace.sol#1878) is not in mixedCase
Variable ChangeblockMarketplace.FEE_NUMERATOR (ChangeblockMarketplace.sol#1884) is not in mixedCase
Variable ChangeblockMarketplace.FEE_DENOMINATOR (ChangeblockMarketplace.sol#1885) is not in mixedCase
Variable ChangeblockMarketplace.TREASURY (ChangeblockMarketplace.sol#1887) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
console._sendLogPayload(bytes) (ChangeblockMarketplace.sol#8-15) uses assembly
        - INLINE ASM (ChangeblockMarketplace.sol#11-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
console.slitherConstructorConstantVariables() (ChangeblockMarketplace.sol#5-1533) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (ChangeblockMarketplace.sol#6)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ChangeblockMarketplace.sol#1798-1800)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ChangeblockMarketplace.sol#1806-1809)
getListing(uint256) should be declared external:
        - ChangeblockMarketplace.getListing(uint256) (ChangeblockMarketplace.sol#1834-1847)
buyERC20(uint256,uint256,uint256) should be declared external:
        - ChangeblockMarketplace.buyERC20(uint256,uint256,uint256) (ChangeblockMarketplace.sol#1980-1995)
buyERC721(uint256,uint256) should be declared external:
        - ChangeblockMarketplace.buyERC721(uint256,uint256) (ChangeblockMarketplace.sol#2001-2009)
listERC20(uint256,uint256,address,address) should be declared external:
        - ChangeblockMarketplace.listERC20(uint256,uint256,address,address) (ChangeblockMarketplace.sol#2019-2036)
listERC721(uint256,uint256,address,address) should be declared external:
        - ChangeblockMarketplace.listERC721(uint256,uint256,address,address) (ChangeblockMarketplace.sol#2042-2053)
delistERC20(uint256,uint256) should be declared external:
        - ChangeblockMarketplace.delistERC20(uint256,uint256) (ChangeblockMarketplace.sol#2059-2069)
delistERC721(uint256) should be declared external:
        - ChangeblockMarketplace.delistERC721(uint256) (ChangeblockMarketplace.sol#2074-2082)
bid(uint256,uint256,uint256) should be declared external:
        - ChangeblockMarketplace.bid(uint256,uint256,uint256) (ChangeblockMarketplace.sol#2105-2119)
withdrawBid(uint256,uint256) should be declared external:
        - ChangeblockMarketplace.withdrawBid(uint256,uint256) (ChangeblockMarketplace.sol#2125-2132)
acceptBid(uint256,address,uint256,uint256,uint256) should be declared external:
        - ChangeblockMarketplace.acceptBid(uint256,address,uint256,uint256,uint256) (ChangeblockMarketplace.sol#2141-2166)
setSellers(address[],bool[]) should be declared external:
        - ChangeblockMarketplace.setSellers(address[],bool[]) (ChangeblockMarketplace.sol#2173-2178)
setBuyers(address[],bool[]) should be declared external:
        - ChangeblockMarketplace.setBuyers(address[],bool[]) (ChangeblockMarketplace.sol#2182-2187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ChangeblockMarketplace.sol analyzed (7 contracts with 75 detectors), 435 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CBLKFixed.sol

```
INFO:Detectors:
CBLKFixed.constructor(string,string,address[],uint256[]).name (CBLKFixed.sol#500) shadows:
        - ERC20.name() (CBLKFixed.sol#137-139) (function)
        - IERC20Metadata.name() (CBLKFixed.sol#88) (function)
CBLKFixed.constructor(string,string,address[],uint256[]).symbol (CBLKFixed.sol#501) shadows:
        - ERC20.symbol() (CBLKFixed.sol#145-147) (function)
        - IERC20Metadata.symbol() (CBLKFixed.sol#93) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CBLKFixed.deposit(uint256[]) (CBLKFixed.sol#520-537) has external calls inside a loop: IERC20(token).transferFrom(msg.sender,a
ddress(this),amounts[i_scope_0]) (CBLKFixed.sol#532)
CBLKFixed.withdraw(uint256) (CBLKFixed.sol#542-560) has external calls inside a loop: IERC20(token).transfer(msg.sender,balanc
es[token]) (CBLKFixed.sol#547)
CBLKFixed.withdraw(uint256) (CBLKFixed.sol#542-560) has external calls inside a loop: IERC20(token_scope_1).transfer(msg.sende
r,withdrawal) (CBLKFixed.sol#554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
INFO:Detectors:
Reentrancy in CBLKFixed.deposit(uint256[]) (CBLKFixed.sol#520-537):
        External calls:
        - IERC20(token).transferFrom(msg.sender,address(this),amounts[i_scope_0]) (CBLKFixed.sol#532)
        State variables written after the call(s):
        - balances[token] += amounts[i_scope_0] (CBLKFixed.sol#533)
Reentrancy in CBLKFixed.withdraw(uint256) (CBLKFixed.sol#542-560):
        External calls:
        - IERC20(token_scope_1).transfer(msg.sender,withdrawal) (CBLKFixed.sol#554)
        State variables written after the call(s):
        - balances[token_scope_1] -= withdrawal (CBLKFixed.sol#555)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Context._msgData() (CBLKFixed.sol#105-107) 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 (CBLKFixed.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (CBLKFixed.sol#137-139)
symbol() should be declared external:
        - ERC20.symbol() (CBLKFixed.sol#145-147)
decimals() should be declared external:
        - ERC20.decimals() (CBLKFixed.sol#162-164)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (CBLKFixed.sol#176-178)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CBLKFixed.sol#188-192)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CBLKFixed.sol#211-215)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CBLKFixed.sol#233-242)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CBLKFixed.sol#256-260)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CBLKFixed.sol#276-285)
deposit(uint256[]) should be declared external:
        - CBLKFixed.deposit(uint256[]) (CBLKFixed.sol#520-537)
withdraw(uint256) should be declared external:
        - CBLKFixed.withdraw(uint256) (CBLKFixed.sol#542-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CBLKFixed.sol analyzed (5 contracts with 75 detectors), 25 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CBLKUnfixed.sol

```
INFO:Detectors:
CBLKUnfixed.constructor(string,string).name (CBLKUnfixed.sol#2085) shadows:
        - ERC20.name() (CBLKUnfixed.sol#1723-1725) (function)
        - IERC20Metadata.name() (CBLKUnfixed.sol#1617) (function)
CBLKUnfixed.constructor(string,string).symbol (CBLKUnfixed.sol#2085) shadows:
        - ERC20.symbol() (CBLKUnfixed.sol#1731-1733) (function)
        - IERC20Metadata.symbol() (CBLKUnfixed.sol#1622) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]) (CBLKUnfixed.sol#2094-2127) has external calls inside a loop: I
ERC20(inputTokens[i]).transferFrom(msg.sender,address(this),inputAmounts[i]) (CBLKUnfixed.sol#2106)
CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]) (CBLKUnfixed.sol#2094-2127) has external calls inside a loop: I
ERC20(outputTokens[i_scope_0]).transfer(msg.sender,outputAmounts[i_scope_0]) (CBLKUnfixed.sol#2112)
CBLKUnfixed.withdraw(uint256) (CBLKUnfixed.sol#2132-2149) has external calls inside a loop: IERC20(token).transfer(msg.sender,
withdrawal) (CBLKUnfixed.sol#2141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]) (CBLKUnfixed.sol#2094-2127):
        External calls:
        - IERC20(outputTokens[i_scope_0]).transfer(msg.sender,outputAmounts[i_scope_0]) (CBLKUnfixed.sol#2112)
        State variables written after the call(s):
        - balances[outputTokens[i_scope_0]] -= outputAmounts[i_scope_0] (CBLKUnfixed.sol#2113)
        - _unregisterToken(outputTokens[i_scope_0]) (CBLKUnfixed.sol#2115)
                - climateBackedTonnes[indexes[token]] = climateBackedTonnes[climateBackedTonnes.length - 1] (CBLKUnfixed.sol#2
153)
                - climateBackedTonnes.pop() (CBLKUnfixed.sol#2154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
console._sendLogPayload(bytes) (CBLKUnfixed.sol#11-18) uses assembly
        - INLINE ASM (CBLKUnfixed.sol#14-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Pragma version>=0.8.0 (CBLKUnfixed.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:
Contract console (CBLKUnfixed.sol#8-1536) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

INFO:Detectors:
console.slitherConstructorConstantVariables() (CBLKUnfixed.sol#8-1536) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (CBLKUnfixed.sol#9)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CBLKUnfixed.sol#1672-1674)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CBLKUnfixed.sol#1680-1683)
name() should be declared external:
        - ERC20.name() (CBLKUnfixed.sol#1723-1725)
symbol() should be declared external:
        - ERC20.symbol() (CBLKUnfixed.sol#1731-1733)
```

```
symbol() should be declared external:
        - ERC20.symbol() (CBLKUnfixed.sol#1731-1733)
decimals() should be declared external:
        - ERC20.decimals() (CBLKUnfixed.sol#1748-1750)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (CBLKUnfixed.sol#1762-1764)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CBLKUnfixed.sol#1774-1778)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CBLKUnfixed.sol#1797-1801)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CBLKUnfixed.sol#1819-1828)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CBLKUnfixed.sol#1842-1846)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CBLKUnfixed.sol#1862-1871)
withdraw(uint256) should be declared external:
        - CBLKUnfixed.withdraw(uint256) (CBLKUnfixed.sol#2132-2149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CBLKUnfixed.sol analyzed (7 contracts with 75 detectors), 410 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CBT.sol

```
INFO:Detectors:
CBT.constructor(string,string).name (CBT.sol#520) shadows:
        - ERC20.name() (CBT.sol#193-195) (function)
        - IERC20Metadata.name() (CBT.sol#87) (function)
CBT.constructor(string,string).symbol (CBT.sol#520) shadows:
        - ERC20.symbol() (CBT.sol#201-203) (function)
        - IERC20Metadata.symbol() (CBT.sol#92) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (CBT.sol#104-106) 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 (CBT.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:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CBT.sol#142-144)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CBT.sol#150-153)
name() should be declared external:
        - ERC20.name() (CBT.sol#193-195)
symbol() should be declared external:
        - ERC20.symbol() (CBT.sol#201-203)
decimals() should be declared external:
        - ERC20.decimals() (CBT.sol#218-220)
totalSupply() should be declared external:
        - ERC20.totalSupply() (CBT.sol#225-227)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (CBT.sol#232-234)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CBT.sol#244-248)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CBT.sol#267-271)
transferFrom(address,address,uint256) should be declared external:
```

```
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CBT.sol#289-298)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CBT.sol#312-316)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CBT.sol#332-341)
mint(address,uint256) should be declared external:
        - CBT.mint(address,uint256) (CBT.sol#525-527)
burn(address,uint256) should be declared external:
        - CBT.burn(address,uint256) (CBT.sol#532-534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CBT.sol analyzed (6 contracts with 75 detectors), 19 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CBTFixedFactory.sol

```
INFO:Detectors:
CBLKFixed.constructor(string,string,address[],uint256[]).name (CBTFixedFactory.sol#495) shadows:
        - ERC20.name() (CBTFixedFactory.sol#132-134) (function)
        - IERC20Metadata.name() (CBTFixedFactory.sol#83) (function)
CBLKFixed.constructor(string,string,address[],uint256[]).symbol (CBTFixedFactory.sol#496) shadows:
        - ERC20.symbol() (CBTFixedFactory.sol#140-142) (function)
        - IERC20Metadata.symbol() (CBTFixedFactory.sol#88) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CBLKFixed.deposit(uint256[]) (CBTFixedFactory.sol#515-532) has external calls inside a loop: IERC20(token).transferFrom(msg.se
nder,address(this),amounts[i_scope_0]) (CBTFixedFactory.sol#527)
CBLKFixed.withdraw(uint256) (CBTFixedFactory.sol#537-555) has external calls inside a loop: IERC20(token).transfer(msg.sender,
balances[token]) (CBTFixedFactory.sol#542)
CBLKFixed.withdraw(uint256) (CBTFixedFactory.sol#537-555) has external calls inside a loop: IERC20(token_scope_1).transfer(msg
.sender,withdrawal) (CBTFixedFactory.sol#549)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in CBLKFixed.deposit(uint256[]) (CBTFixedFactory.sol#515-532):
        External calls:
        - IERC20(token).transferFrom(msg.sender,address(this),amounts[i_scope_0]) (CBTFixedFactory.sol#527)
        State variables written after the call(s):
        - balances[token] += amounts[i_scope_0] (CBTFixedFactory.sol#528)
Reentrancy in CBLKFixed.withdraw(uint256) (CBTFixedFactory.sol#537-555):
```

```
        External calls:
        - IERC20(token_scope_1).transfer(msg.sender,withdrawal) (CBTFixedFactory.sol#549)
        State variables written after the call(s):
        - balances[token_scope_1] -= withdrawal (CBTFixedFactory.sol#550)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Context._msgData() (CBTFixedFactory.sol#100-102) 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 (CBTFixedFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (CBTFixedFactory.sol#132-134)
symbol() should be declared external:
        - ERC20.symbol() (CBTFixedFactory.sol#140-142)
decimals() should be declared external:
        - ERC20.decimals() (CBTFixedFactory.sol#157-159)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (CBTFixedFactory.sol#171-173)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CBTFixedFactory.sol#183-187)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CBTFixedFactory.sol#206-210)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CBTFixedFactory.sol#228-237)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CBTFixedFactory.sol#251-255)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CBTFixedFactory.sol#271-280)
deposit(uint256[]) should be declared external:
        - CBLKFixed.deposit(uint256[]) (CBTFixedFactory.sol#515-532)
withdraw(uint256) should be declared external:
        - CBLKFixed.withdraw(uint256) (CBTFixedFactory.sol#537-555)
renounceOwnership() should be declared external:
```

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CBTFixedFactory.sol#595-597)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CBTFixedFactory.sol#603-606)
approve(address,bool) should be declared external:
        - CBLKFixedFactory.approve(address,bool) (CBTFixedFactory.sol#634-637)
deploy(string,string,address[],uint256[]) should be declared external:
        - CBLKFixedFactory.deploy(string,string,address[],uint256[]) (CBTFixedFactory.sol#639-650)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CBTFixedFactory.sol analyzed (7 contracts with 75 detectors), 29 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CBTUnfixedFactory.sol

```
INFO:Detectors:
CBLKUnfixed.constructor(string,string).name (CBTUnfixedFactory.sol#2081) shadows:
        - ERC20.name() (CBTUnfixedFactory.sol#1719-1721) (function)
        - IERC20Metadata.name() (CBTUnfixedFactory.sol#1613) (function)
CBLKUnfixed.constructor(string,string).symbol (CBTUnfixedFactory.sol#2081) shadows:
        - ERC20.symbol() (CBTUnfixedFactory.sol#1727-1729) (function)
        - IERC20Metadata.symbol() (CBTUnfixedFactory.sol#1618) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]) (CBTUnfixedFactory.sol#2090-2123) has external calls inside a l
oop: IERC20(inputTokens[i]).transferFrom(msg.sender,address(this),inputAmounts[i]) (CBTUnfixedFactory.sol#2102)
CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]) (CBTUnfixedFactory.sol#2090-2123) has external calls inside a l
oop: IERC20(outputTokens[i_scope_0]).transfer(msg.sender,outputAmounts[i_scope_0]) (CBTUnfixedFactory.sol#2108)
CBLKUnfixed.withdraw(uint256) (CBTUnfixedFactory.sol#2128-2145) has external calls inside a loop: IERC20(token).transfer(msg.s
ender,withdrawal) (CBTUnfixedFactory.sol#2137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in CBLKUnfixedFactory.deploy(string,string) (CBTUnfixedFactory.sol#2177-2184):
        External calls:
        - cblk.transferOwnership(msg.sender) (CBTUnfixedFactory.sol#2180)
        State variables written after the call(s):
        - isCBLKUnfixed[address(cblk)] = true (CBTUnfixedFactory.sol#2181)
Reentrancy in CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]) (CBTUnfixedFactory.sol#2090-2123):
        External calls:
        - IERC20(outputTokens[i_scope_0]).transfer(msg.sender,outputAmounts[i_scope_0]) (CBTUnfixedFactory.sol#2108)
        State variables written after the call(s):
        - balances[outputTokens[i_scope_0]] -= outputAmounts[i_scope_0] (CBTUnfixedFactory.sol#2109)
        - _unregisterToken(outputTokens[i_scope_0]) (CBTUnfixedFactory.sol#2111)
```

```
INFO:Detectors:
Pragma version>=0.8.0 (CBTUnfixedFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.1
2/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Contract console (CBTUnfixedFactory.sol#4-1532) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
console.slitherConstructorConstantVariables() (CBTUnfixedFactory.sol#4-1532) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (CBTUnfixedFactory.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CBTUnfixedFactory.sol#1668-1670)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CBTUnfixedFactory.sol#1676-1679)
name() should be declared external:
        - ERC20.name() (CBTUnfixedFactory.sol#1719-1721)
symbol() should be declared external:
        - ERC20.symbol() (CBTUnfixedFactory.sol#1727-1729)
```

```
symbol() should be declared external:
        - ERC20.symbol() (CBTUnfixedFactory.sol#1727-1729)
decimals() should be declared external:
        - ERC20.decimals() (CBTUnfixedFactory.sol#1744-1746)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (CBTUnfixedFactory.sol#1758-1760)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CBTUnfixedFactory.sol#1770-1774)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CBTUnfixedFactory.sol#1793-1797)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CBTUnfixedFactory.sol#1815-1824)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CBTUnfixedFactory.sol#1838-1842)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CBTUnfixedFactory.sol#1858-1867)
withdraw(uint256) should be declared external:
        - CBLKUnfixed.withdraw(uint256) (CBTUnfixedFactory.sol#2128-2145)
approve(address,bool) should be declared external:
        - CBLKUnfixedFactory.approve(address,bool) (CBTUnfixedFactory.sol#2172-2175)
deploy(string,string) should be declared external:
        - CBLKUnfixedFactory.deploy(string,string) (CBTUnfixedFactory.sol#2177-2184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CBTUnfixedFactory.sol analyzed (8 contracts with 75 detectors), 414 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CBTFactory.sol

```
INFO:Detectors:
CBT.constructor(string,string).name (CBTFactory.sol#515) shadows:
        - ERC20.name() (CBTFactory.sol#188-190) (function)
        - IERC20Metadata.name() (CBTFactory.sol#82) (function)
CBT.constructor(string,string).symbol (CBTFactory.sol#515) shadows:
        - ERC20.symbol() (CBTFactory.sol#196-198) (function)
        - IERC20Metadata.symbol() (CBTFactory.sol#87) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in CBTFactory.deploy(string,string) (CBTFactory.sol#554-561):
        External calls:
        - cbt.transferOwnership(msg.sender) (CBTFactory.sol#557)
        State variables written after the call(s):
        - isCBT[address(cbt)] = true (CBTFactory.sol#558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CBTFactory.deploy(string,string) (CBTFactory.sol#554-561):
        External calls:
        - cbt.transferOwnership(msg.sender) (CBTFactory.sol#557)
        Event emitted after the call(s):
        - Deployment(address(cbt)) (CBTFactory.sol#559)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (CBTFactory.sol#99-101) 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 (CBTFactory.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:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CBTFactory.sol#137-139)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CBTFactory.sol#145-148)
name() should be declared external:
        - ERC20.name() (CBTFactory.sol#188-190)
symbol() should be declared external:
        - ERC20.symbol() (CBTFactory.sol#196-198)
decimals() should be declared external:
        - ERC20.decimals() (CBTFactory.sol#213-215)
totalSupply() should be declared external:
        - ERC20.totalSupply() (CBTFactory.sol#220-222)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (CBTFactory.sol#227-229)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (CBTFactory.sol#239-243)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (CBTFactory.sol#262-266)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (CBTFactory.sol#284-293)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (CBTFactory.sol#307-311)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (CBTFactory.sol#327-336)
mint(address,uint256) should be declared external:
        - CBT.mint(address,uint256) (CBTFactory.sol#520-522)
burn(address,uint256) should be declared external:
        - CBT.burn(address,uint256) (CBTFactory.sol#527-529)
approve(address,bool) should be declared external:
        - CBTFactory.approve(address,bool) (CBTFactory.sol#549-552)
deploy(string,string) should be declared external:
        - CBTFactory.deploy(string,string) (CBTFactory.sol#554-561)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CBTFactory.sol analyzed (7 contracts with 75 detectors), 23 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ChangeblockMarketplace.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ChangeblockMarketplace.buyERC20(uint256,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1980:4:

Gas & Economy

Gas costs:

Gas requirement of function ChangeblockMarketplace.getListing 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: 1834:4:

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: 2183:8:

Miscellaneous

Constant/View/Pure functions:

IERC20.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1748:4:

Email: audit@EtherAuthority.io

Similar variable names:

ChangeblockMarketplace.getListing(uint256) : Variables have very similar names "listing" and "listingId". Note: Modifiers are currently not considered by this static analysis.
Pos: 1845:8:

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: 2148:8:

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: 2158:22:

CBLKFixed.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CBLKFixed.deposit(uint256[]): Could potentially lead to re-entrancy vulnerability.
more
Pos: 520:4:

Gas & Economy

Gas costs:

Gas requirement of function CBLKFixed.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: 137:4:

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: 530:8:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.

more

Pos: 453:4:

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: 524:12:

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: 548:16:

Data truncated:

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

Pos: 552:37:

CBLKUnfixed.sol

Security

Check-effects-interaction:

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

more

Pos: 2094:4:

Gas & Economy

Gas costs:

Gas requirement of function CBLKUnfixed.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: 1723:4:

Gas costs:

Gas requirement of function CBLKUnfixed.withdraw 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: 2132:4:

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: 2136:8:

Miscellaneous

Email: audit@EtherAuthority.io

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2039:4:

Similar variable names:

CBLKUnfixed.withdraw(uint256) : Variables have very similar names "balance" and "balances". Note: Modifiers are currently not considered by this static analysis.
Pos: 2138:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1998:12:

Result not used:

A binary operation yields a value that is not used further. This is often caused by confusing assignment (=) and comparison (==).
Pos: 2103:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 2139:33:

CBT.sol

Gas & Economy

Gas costs:

Gas requirement of function CBT.burn 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: 532:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 509:4:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 425:49:

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: 468:12:

CBTFixedFactory.sol

Security

Check-effects-interaction:

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

Pos: 537:4:

Gas & Economy

Gas costs:

Gas requirement of function CBLKFixedFactory.deploy 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: 639:4:

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: 525:8:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 448:4:

Similar variable names:

CBLKFixedFactory.approve(address,bool) : Variables have very similar names "approval" and "approvals". Note: Modifiers are currently not considered by this static analysis.
Pos: 636:33:

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: 645:8:

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: 543:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 547:37:

CBTUnfixedFactory.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CBLKUnfixed.rebalance(address[],uint256[],address[],uint256[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2090:4:

Gas & Economy

Gas costs:

Gas requirement of function CBLKUnfixedFactory.deploy 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: 2177:4:

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: 2132:8:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2035:4:

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: 2178:8:

Result not used:

A binary operation yields a value that is not used further. This is often caused by confusing assignment (=) and comparison (==).
Pos: 2099:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 2135:33:

CBTFactory.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CBTFactory.deploy(string,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 554:4:

Gas & Economy

Gas costs:

Gas requirement of function CBTFactory.deploy 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: 554:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 504:4:

Similar variable names:

CBTFactory.approve(address,bool) : Variables have very similar names "approval" and "approvals". Note: Modifiers are currently not considered by this static analysis.
Pos: 551:33:

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: 555:8:

Solhint Linter

ChangeblockMarketplace.sol

```
ChangeblockMarketplace.sol:2:1: Error: Compiler version ^0.8.0 does
not satisfy the r semver requirement
ChangeblockMarketplace.sol:5:1: Error: Contract name must be in
CamelCase
ChangeblockMarketplace.sol:6:2: Error: Explicitly mark visibility of
state
ChangeblockMarketplace.sol:11:3: Error: Avoid using inline assembly.
It is acceptable only in rare cases
ChangeblockMarketplace.sol:13:8: Error: Variable "r" is unused
ChangeblockMarketplace.sol:1772:5: Error: Explicitly mark visibility
in function (Set ignoreConstructors to true if using solidity
>=0.7.0)
ChangeblockMarketplace.sol:1874:45: Error: Variable name must be in
mixedCase
ChangeblockMarketplace.sol:1878:46: Error: Variable name must be in
mixedCase
ChangeblockMarketplace.sol:1884:20: Error: Variable name must be in
mixedCase
ChangeblockMarketplace.sol:1885:20: Error: Variable name must be in
mixedCase
ChangeblockMarketplace.sol:1887:5: Error: Explicitly mark visibility
of state
ChangeblockMarketplace.sol:1887:13: Error: Variable name must be in
mixedCase
ChangeblockMarketplace.sol:1889:5: Error: Explicitly mark visibility
of state
ChangeblockMarketplace.sol:1946:49: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:1953:46: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:1963:5: Error: Explicitly mark visibility
in function (Set ignoreConstructors to true if using solidity
>=0.7.0)
ChangeblockMarketplace.sol:1986:41: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:1987:43: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:1993:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
ChangeblockMarketplace.sol:2003:41: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2063:13: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2065:43: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2067:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
ChangeblockMarketplace.sol:2078:13: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2086:64: Error: Use double quotes for
```

```
string literals
ChangeblockMarketplace.sol:2093:65: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2148:64: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2151:13: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2156:13: Error: Use double quotes for
string literals
ChangeblockMarketplace.sol:2163:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
```

CBLKFixed.sol

```
CBLKFixed.sol:280:18: Error: Parse error: missing ';' at '{'
CBLKFixed.sol:313:18: Error: Parse error: missing ';' at '{'
CBLKFixed.sol:362:18: Error: Parse error: missing ';' at '{'
CBLKFixed.sol:413:22: Error: Parse error: missing ';' at '{'
```

CBLKUnfixed.sol

```
CBLKUnfixed.sol:1866:18: Error: Parse error: missing ';' at '{'
CBLKUnfixed.sol:1899:18: Error: Parse error: missing ';' at '{'
CBLKUnfixed.sol:1948:18: Error: Parse error: missing ';' at '{'
CBLKUnfixed.sol:1999:22: Error: Parse error: missing ';' at '{'
```

CBT.sol

```
CBT.sol:336:18: Error: Parse error: missing ';' at '{'
CBT.sol:369:18: Error: Parse error: missing ';' at '{'
CBT.sol:418:18: Error: Parse error: missing ';' at '{'
CBT.sol:469:22: Error: Parse error: missing ';' at '{'
```

CBTFixedFactory.sol

```
CBTFixedFactory.sol:275:18: Error: Parse error: missing ';' at '{'
CBTFixedFactory.sol:308:18: Error: Parse error: missing ';' at '{'
CBTFixedFactory.sol:357:18: Error: Parse error: missing ';' at '{'
CBTFixedFactory.sol:408:22: Error: Parse error: missing ';' at '{'
```

CBTUnfixedFactory.sol

```
CBTUnfixedFactory.sol:1862:18: Error: Parse error: missing ';' at '{'
CBTUnfixedFactory.sol:1895:18: Error: Parse error: missing ';' at '{'
CBTUnfixedFactory.sol:1944:18: Error: Parse error: missing ';' at '{'
CBTUnfixedFactory.sol:1995:22: Error: Parse error: missing ';' at '{'
```

CBTFactory.sol

```
CBTFactory.sol:331:18: Error: Parse error: missing ';' at '{'
CBTFactory.sol:364:18: Error: Parse error: missing ';' at '{'
CBTFactory.sol:413:18: Error: Parse error: missing ';' at '{'
CBTFactory.sol:464:22: 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