Cyberscope

Audit Report

Xocolatl HouseOfCoin

December 2022

Github	https://github.com/La-DAO/xocolatl-contracts
Commit	7d780e9a7573b88f042f8f45096a201442ea782e
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	HouseOfCoin
Testing Deploy	https://testnet.bscscan.com/address/0x9b11331c630492d69fe679abbd2407b6506695d1

Audit Updates

Initial Audit	26 Oct 2022 https://github.com/cyberscope-io/audits/blob/main/xocolatl/v1/houseOfCoin.pdf
Corrected Phase 2	19 Dec 2022

Source Files

Filename	SHA256
@openzeppelin/contracts/access/AccessControl.sol	5af1771388b4fe634e0a566716e32c6d00a5372875099127b274d4cf8a94e9d2
@openzeppelin/contracts/access/IAccessControl.sol	d03c1257f2094da6c86efa7aa09c1c07ebd33dd31046480c5097bc2542140e45
@openzeppelin/contracts/proxy/utils/Initializable.sol	36cf1b60e8da3e2bca15b187f775780310bb219c30dccd6258123c43fbf84ad8
@openzeppelin/contracts/token/ERC1155/IERC1155.sol	fd6a1801f1f2f8af0a3ece0b254da06ec24568aec02cfe94827061379aebc6f3
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Address.sol	1e0922f6c0bf6b1b8b4d480dcabb691b1359195a297bde6dc5172e79f3a1f826
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	4e45d53327d561848fbcf381262ec5c0ac91b2f1f06432210bf76db55279d945
@openzeppelin/contracts/utils/introspection/ERC165.sol	8806a632d7b656cadb8133ff8f2acae4405b3a64d8709d93b0fa6a216a8a6154
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
@openzeppelin/contracts/utils/Strings.sol	34127ad0054df5963b0fd694c1b313d17e9114a2f426b85526d6d976210298ab
contracts/abstract/OracleHouse.sol	bda23986b2c82b00d3600c6b5ffaaccd2a46b8c0c5508fc97432fc5d9671341c

contracts/HouseOfCoin.sol	2a0fb9d93299695bacaec3addf881f2dd735876cba17ce6381d4553567b0633e
contracts/interfaces/chainlink/IAggregatorV3.sol	299b7546616ad9fb756c778f0771f5d39aeca3f85fb2c4d794b19df0a8795bd3
contracts/interfaces/IAssetsAccountant.sol	9119e1160f73bf62a5ef77f66d6932615f52836ca70f66f3d5b82b59fe61b1e9
contracts/interfaces/IERC20Extension.sol	341c5d7640bd0c44aa86ec924574727c53604487e57352158fb9a11e3b671f8d
contracts/interfaces/IHouseOfReserve.sol	2cf3c1454c96809fe84a571802268e15539652ab80328dbc7cd99b1db5f7997e
contracts/interfaces/IOracle.sol	1f13347804c9d374a356eb2c5100a4f983c3873c164e5bd1d3890d79bc3786a4
contracts/interfaces/uma/IAddressWhitelist.sol	46235463375dd715f5f30b2dd2bca0423e0994a311f84204ab39e82ef5d0e95b
contracts/interfaces/uma/IdentifierWhitelistInterface.sol	9495496b5ab855df3397193c9ba6a31eaf4ee050bce789bb2215619130723d3d
contracts/interfaces/uma/IOptimisticOracleV2.sol	11203bc5f10d2e4a60dcdb0f3728aae9f315bea16d5dbfa75fe6d5f0038f8aad
contracts/interfaces/uma/IUMAFinder.sol	94e604d5efcb6f22ea5f73d3c38c849775ae8225b9c736551db3d3cbaaa3bc93
contracts/utils/redstone/PriceAware.sol	0c7096448999fe38e17ca708ea0ad6dbb8878991413bfecfd09f4a1d7c7070b5
contracts/utils/uma/UMAOracleHelper.sol	d78c692b5c37e42e1d57ae6b8c6e08bda2a5db8e02d77ee46efecdb60ec422b1
contracts/utils/uma/UMAOracleInterfaces.sol	81eab927f79ea99651be5db8f7c3ae1fadaeed577a6b8ca53cc2c1cc77f3b55b

Introduction

The HouseOfCoin is a collateral issuing mechanim. The contract is responsible for minting backed tokens is relation to the reserved tokens. The ratio between the backed and the reserved tokens is defined by two factors, the price and the collarationRatio. The price is defined by an oracle and the collarationRatio by the admin role.

The contract uses Oracles to receive off-chain data. Three oracles are configured Chainlink, Optimistic, and Redstone. The contract can use one Oracle at a time.

Roles

The admin role has authority:

- Configure the Oracles.
- Configure the liquidation parameters.

Users have the ability to

- Mint coins. Issue backed tokens proportionally to their reserved tokens.
- Payback coins. Burn backed their tokens.
- Liquidate. Get the reserves of a user by burning the caller’s backed tokens.
- Check the user health ratio.
- Check the cost of liquidation.
- Check the remaining minting power.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	LP	Liquidate Permissions	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

LP - Liquidate Permissions

Criticality	Minor / Informative
Location	contract.sol#L377
Status	Unresolved

Description

Any user has the authority to call the liquidateUser() method. There are two cases:

- If the targeted user health ratio is between marginCallThreshold and 'liquidationThreshold', then it will emit a 'MarginCall' event.
- If the health ratio is less than liquidationThreshold and the callers' role is not LIQUIDATOR, then the contract will abort the transaction, otherwise it will proceed.

This diversion may produce some miss concerns since any user can emit the event but specific users can proceed with the liquidation.

```
function liquidateUser(address userToLiquidate, address reserveAsset)
    external
{
...
}
```

Recommendation

The contract could allow only the LIQUIDATOR role to access the ‘liquidateUser()’ method.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/HouseOfCoin.sol#L118
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of your Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).

2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).

3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).

4. Use indentation to improve readability and structure.

5. Use spaces between operators and after commas.

6. Use comments to explain the purpose and behavior of your code.

7. Keep lines short (around 120 characters) to improve readability.

```
LiquidationParameters internal _liqParam
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

You can find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/HouseOfCoin.sol#L142
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
backedAssetDecimals = IERC20Extension(backedAsset).decimals()
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, you can help to ensure that the contract performs as intended and does not have any missing events that could cause issues with its arithmetic.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/HouseOfCoin.sol#L738
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in your contract. It's important to always initialize local variables with appropriate values before using them.

```
LiquidationParameters memory ltemp
```

Recommendation

By initializing local variables before using them, you can help ensure that your contract functions behave as expected and avoid potential issues.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/HouseOfCoin.sol#L820
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20Extension(backedAsset).transferFrom(
            msg.sender,
            address(this),
            costofLiquidation

       )
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
AccessControl	Implementation	Context, IAccessControl, ERC165		
	supportsInterface	Public		-
	hasRole	Public		-
	_checkRole	Internal		
	_checkRole	Internal		
	getRoleAdmin	Public		-
	grantRole	Public	✓	onlyRole
	revokeRole	Public	✓	onlyRole
	renounceRole	Public	✓	-
	_setupRole	Internal	✓	
	_setRoleAdmin	Internal	✓	
	_grantRole	Internal	✓	
	_revokeRole	Internal	✓	
IAccessControl	Interface			
	hasRole	External		-
	getRoleAdmin	External		-
	grantRole	External	✓	-
	revokeRole	External	✓	-
	renounceRole	External	✓	-
Initializable	Implementation			

	_disableInitializers	Internal	✓	
IERC1155	Interface	IERC165		
	balanceOf	External		-
	balanceOfBatch	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
	safeTransferFrom	External	✓	-
	safeBatchTransferFrom	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	

	verifyCallResult	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
ECDSA	Library			
	_throwError	Private		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toTypedDataHash	Internal		
ERC165	Implementation	IERC165		
	supportsInterface	Public		-
IERC165	Interface			
	supportsInterface	External		-
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		

OracleHouse	Implementation	PriceAware		
	_oracleHouse_init	Internal	✓	
	activeOracle	External		-
	_getLatestPrice	Internal		
	setActiveOracle	External	✓	-
	_setActiveOracle	Internal	✓	
	_oracle_redstone_init	Private	✓	
	_getLatestPriceRedstone	Internal		
	getRedstoneData	External		-
	isSignerAuthorized	Public		-
	setTickers	External	✓	-
	_setTickers	Internal	✓	
	authorizeSigner	External	✓	-
	_authorizeSigner	Internal	✓	
	_getLatestPriceUMA	Internal		
	setUMAOracleHelper	External	✓	-
	_setUMAOracleHelper	Internal	✓	
	_getLatestPriceChainlink	Internal		
	getChainlinkData	External		-
	setChainlinkAddrs	External	✓	-
	_setChainlinkAddrs	Internal	✓	
HouseOfCoinState	Implementation			
HouseOfCoin	Implementation	Initializable, AccessControl, OracleHouse, HouseOfCoinState		

	initialize	Public	✓	initializer
	activeOracle	External		-
	setActiveOracle	External		-
	setTickers	External		-
	getRedstoneData	External		-
	authorizeSigner	External	✓	onlyRole
	setUMAOracleHelper	External		-
	getChainlinkData	External		-
	setChainlinkAddrs	External		-
	getLatestPrice	Public		-
	_getLatestPrice	Internal		
	mintCoin	Public	✓	-
	paybackCoin	Public	✓	-
	liquidateUser	External	✓	-
	computeUserHealthRatio	Public		-
	computeCostOfLiquidation	Public		-
	getBackedTokenID	Public		-
	getLiqParams	Public		-
	setLiqParams	Public	✓	onlyRole
	checkRemainingMintingPower	Public		-
	_checkBalances	Internal		
	_checkRemainingMintingPower	Internal		
	_checkIfUserCanMintMore	Internal		
	_transformToBackAssetDecimalBase	Internal	✓	
	_computeUserHealthRatio	Internal		
	_computeCostOfLiquidation	Internal		
	_executeLiquidation	Internal	✓	
IAggregatorV3	Interface			

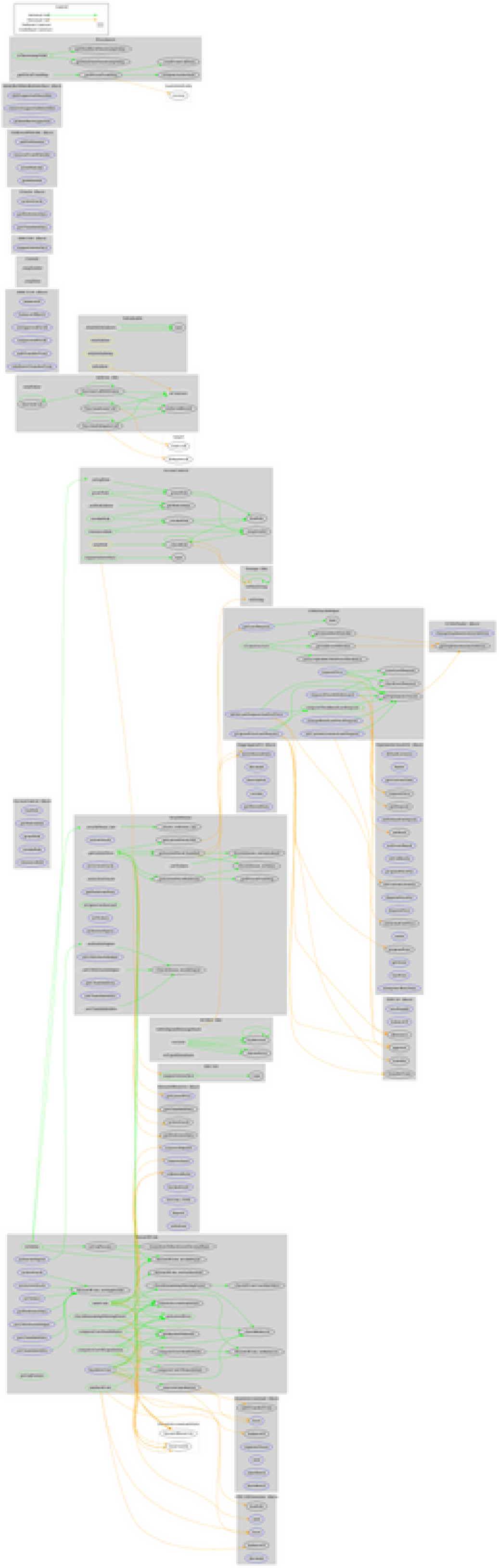
	decimals	External		-
	description	External		-
	version	External		-
	getRoundData	External		-
	latestRoundData	External		-
IAssetsAccountant	Interface	IERC1155		
	registerHouse	External	✓	-
	mint	External	✓	-
	mintBatch	External	✓	-
	burn	External	✓	-
	burnBatch	External	✓	-
IERC20Extension	Interface	IERC20, IAccessControl		
	decimals	External		-
	mint	External	✓	-
	burn	External	✓	-
IHouseOfReserve	Interface	IOracle		
	reserveAsset	External		-
	backedAsset	External		-
	reserveTokenID	External		-
	HOUSE_TYPE	External	✓	-
	collateralRatio	External		-
	getLatestPrice	External		-
	deposit	External	✓	-
	withdraw	External	✓	-

IOracle	Interface			
	activeOracle	External		-
	getRedstoneData	External		-
	getChainlinkData	External		-
IAddressWhitelist	Interface			
	addToWhitelist	External	✓	-
	removeFromWhitelist	External	✓	-
	isOnWhitelist	External		-
	getWhitelist	External		-
IdentifierWhitelistInterface	Interface			
	addSupportedIdentifier	External	✓	-
	removeSupportedIdentifier	External	✓	-
	isIdentifierSupported	External		-
IOptimisticOracleV2	Interface			
	defaultLiveness	External		-
	finder	External		-
	getCurrentTime	External		-
	requestPrice	External	✓	-
	setBond	External	✓	-
	setRefundOnDispute	External	✓	-
	setCustomLiveness	External	✓	-
	setEventBased	External	✓	-
	setCallbacks	External	✓	-
	proposePriceFor	External	✓	-
	proposePrice	External	✓	-

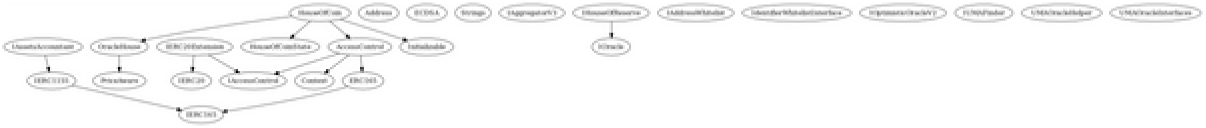
	disputePriceFor	External	✓	-
	disputePrice	External	✓	-
	settleAndGetPrice	External	✓	-
	settle	External	✓	-
	getRequest	External		-
	getState	External		-
	hasPrice	External		-
	stampAncillaryData	External		-
IUMAFinder	Interface			
	changeImplementationAddress	External	✓	-
	getImplementationAddress	External		-
PriceAware	Implementation			
	getMaxDataTimestampDelay	Public		-
	getMaxBlockTimestampDelay	Public		-
	isSignerAuthorized	Public		-
	isTimestampValid	Public		-
	_getPriceFromMsg	Internal		
	_getPricesFromMsg	Internal		
	_readFromCallData	Private		
UMAOracleHelper	Implementation			
		Public	✓	-
	getLastRequest	External		-
	requestPrice	External	✓	-
	requestPriceWithReward	External	✓	-
	setCustomLivenessLastRequest	External	✓	-
	changeBondLastPriceRequest	External	✓	-

	computeTotalBondLastRequest	Public		-
	proposePriceLastRequest	External	✓	-
	settleLastRequestAndGetPrice	External	✓	-
	setAcceptableUMAPriceObsolence	Public	✓	-
	_checkLastRequest	Internal		
	_resetLastRequest	Internal	✓	
	_getIdentifierWhitelist	Internal		
	_getAddressWhitelist	Internal		
	_getOptimisticOracle	Internal		
UMAOracleInterfaces	Library			

Contract Flow



Inheritance Graph



Summary

The HouseOfCoin is a collateral issuing mechanism. This audit investigates security issues and mentions business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io