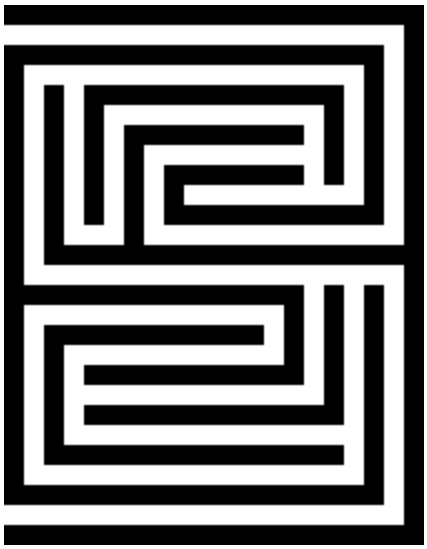

Coin Audit Public Report

PROJECT: Coin Audit

December 2020

Prepared For:

Coin Team | Coin XYZ, Inc.

https://coindefi.org/

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Coin Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in December of 2020 to perform a comprehensive security review of the Coin smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of four days by both members of the Bramah Systems, LLC. executive staff.

Bramah Systems completed the assessment using manual, static and dynamic analysis techniques.

Timeline

Review Commencement: December 22nd, 2020

Report Delivery: December 26th, 2020

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Coin protocol, with a specific focus on trading actions. In specific, the engagement sought to answer the following questions:

- Is it possible for an attacker to steal or freeze tokens?
- Does the Solidity code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?
- Are the arithmetic calculations trustworthy?

Contract Specification

Contract specification was provided in the form of code comments and functional unit tests. The contract is heavily influenced by the Synthetix rewards pool, a contract that has undergone intense scrutiny and was considered the de facto standard for reward distribution for quite some time, as noted by prior reviewers of the distribution mechanism.

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the Coin Protocol. During the course of our engagement, Bramah Systems found relatively few instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT, as our report details.

These aside, the team otherwise used **thoroughly** reviewed and vetted components and provided details as to the token structure, economics, and intent, which helped Bramah highlight any potential concerns with their approach.

Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the Coin Protocol, with the understanding that distributed ledger technologies ("DLT") remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within "Scope of Engagement" and contained within "Directory Structure". The report does NOT cover, review, or opine upon security considerations unique to the Solidity compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.

The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the Coin protocol or any other relevant product, service or asset of Coin or otherwise. This report is not and should not be relied upon by Coin or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the Coin Protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this report is provided to such individuals.

General Recommendations

Best Practices & Solidity Development Guidelines

Function scope should be marked external to save gas

In **public** functions, Solidity copies array arguments to **memory**, whereas **external** functions can read directly from **calldata**. In terms of gas, memory allocation is quite expensive, whereas reading from **calldata** is cheap.

For **external** functions, the compiler doesn't allow **internal** calls (which are executed via jumps in the code, and have array arguments passed by pointers to memory), instead allowing arguments to be read directly from **calldata**, saving a copying step (and the relevant gas associated with this process).

The function **initialize(address,address,address)** should be declared external

Location: contracts/core/BonusRewards.sol#88

The function **allLocked(address)** should be declared external

Location: contracts/general/LockedTokenWrapper.sol#88-90

The function **allTimes(address)** should be declared external

Location: contracts/general/LockedTokenWrapper.sol#96-98

The function **lockedTotalSupply()** should be declared external

Location: contracts/general/LockedTokenWrapper.sol#103-105

Resolution: The team has resolved these findings through augmenting the function scope accordingly.

Solidity version should be updated

The bulk of the protocol uses **pragma version^0.6.6**. As this pragma is out of date and misses many compiler optimizations and potential security considerations of later Solidity versions, it should be updated where possible.

In particular, two compiler bugs were found that potentially impact the contracts, both of medium overall severity.:

Bug Name	Description
DynamicArrayCleanup When assigning a dynamically-sized array with types of size at most 16 bytes in storage causing the assigned array to shrink, some parts of deleted slots were not zeroed out.	Consider a dynamically-sized array in storage whose base-type is small enough such that multiple values can be packed into a single slot, such as `uint128[]`. Let us define its length to be `l`. When this array gets assigned from another array with a smaller length, say `m`, the slots between elements `m` and `l` have to be cleaned by zeroing them out. However, this cleaning was not performed properly. Specifically, after the slot corresponding to `m`, only the first packed value was cleaned up. If this array gets resized to a length larger than `m`, the indices corresponding to the unclean parts of the slot contained the original value, instead of 0. The resizing here is performed by assigning to the array `length`, by a `push()` or via inline assembly. You are not affected if you are only using `.push()` or if you assign a value (even zero) to the new elements after increasing the length of the array. - First Introduced: - Fixed in Version: 0.7.3 - Published: - Severity<: medium

Bug Name	Description
EmptyByteArrayCopy Copying an empty byte array (or string) from memory or calldata to storage can result in data corruption if the target array's length is increased subsequently without storing new data.	The routine that copies byte arrays from memory or calldata to storage stores unrelated data from after the source array in the storage slot if the source array is empty. If the storage array's length is subsequently increased either by using ``.push()`` or by assigning to its ``.length`` attribute (only before 0.6.0), the newly created byte array elements will not be zero-initialized, but contain the unrelated data. You are not affected if you do not assign to ``.length`` and do not use ``.push()`` on byte arrays, or only use ``.push()`` or manually initialize the new elements. - First Introduced:

- Fixed in Version: 0.7.4
- Published:
- Severity<: medium

Thankfully, the protocol does possess mitigations for the first compiler bug (setting values to 0), and is not impacted by the second (as relevant arrays within the protocol are of type **uint256**.

Resolution: The team acknowledges the risk posed by earlier Solidity versions and will continue to utilize the **pragma version^0.6.6.**

Specific Recommendations

Unique to the Coin Protocol

Sensitive variable changes should emit an event

The changeReservePercent function within **StakingRewards.sol**, which takes in a **uint256** (**_reservePercent**) should emit an event for line 229 (listed below).

- reservePercent = _reservePercent

Location: contracts/core/StakingRewards.sol#225-230

The setRewardDistribution function within **BonusRewards.sol**, which takes in an **address** (**_rewardDistribution**), should emit an event for line 267 (listed below):

- **rewardDistribution = _rewardDistribution**

Location: contracts/core/BonusRewards.sol#262-268

Resolution: The team has introduced multiple event "emitters" that create an event upon changing of sensitive variables.

Setter function should check for zero

Setter functions should check that the value they are setting is not 0 (the default value of an uninitialized variable or in the case of addresses, the oft chosen "burn address"

The **changeStakingRewards** function in **Reserve.sol** lacks a zero-check on :

- **stakingRewards = _stakingRewards**

Location: contracts/core/Reserve.sol#46

The **changeBonusRewards** function in **Reserve.sol** lacks a zero-check on :

- bonusRewards = _bonusRewards

Location: contracts/core/Reserve.sol#56

Resolution: As these values may intentionally be set to zero, the team has noted this finding but opted to keep such logic in (as this is intentional behaviour of the function).

Highly permissive owner account and centralization of power

The deploying account possesses a number of highly actions (namely, changing various distribution and reward preferences). This deploying account should (where possible) minimize usage of the associated key (e.g. performing transactions, using as a regular user account) and perform other operational security best practices. Potentially, this could involve transferring ownership to a MultiSignature governance.

Resolution: The team has provided the following: “The centralization of power is understood/accepted and "owner" will be transferred to a DAO as soon as possible”

Design principles rely upon a “closed” system

By design, many principles within the protocol rely upon having a closed system design, wherein various functionality exists within a “wrapper” in lieu of the native functionality supported by the ERC20 token.

While this is an intentional design choice and used to facilitate proper execution of the contracts, users should be aware that these functions may perform differently than their ERC20 counterparts (e.g. how **totalSupply** is calculated for **LockedTokenWrapper.sol** differs from the standard **totalSupply** that an ERC20 token could return).

Resolution: This is by design and necessary for the successful implementation of the protocol.

Toolset Warnings

Unique to the Coin Protocol

Overview

In addition to our manual review, our process involves utilizing static analysis and formal methods in order to perform additional verification of the presence of security vulnerabilities (or lack thereof). An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.

The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable.

Compilation Warnings

Our review, at request of Coin, covers the Solidity code (*.sol) as of sha256sum **338fc13c99840f95448f1df63266d44787af71a692ee29f9ef58f390758e8cff** of the CoinStaking.7z archive. This codebase had a compilation error as follows:

Data location must be "calldata" for parameter in external function, but "memory" was given.

Location: **BonusRewards.sol**

Following initial delivery of the audit report, the Coin team provided an updated archive with their fixes (which included resolving this compilation error). That archive possessed the sha256sum of **48a4445b605f3332e2f05761ddd405679943db23cabc63d8006b405c7ed10198**.

Test Coverage

The contract repository features basic unit tests provided in the form of a TypeScript file that validates various functional stages of the smart contract.

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- Securify
- MAIAN
- Mythril
- Oyente
- Slither

In each case, the team had either mitigated relevant concerns raised by each of these tools or provided adequate justification for the risk (such as adhering to the ERC-20 token standard).

Surya Coverage Report

Contract	**Type**	**Bases**		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
StakingRewards	Implementation	LockedTokenWrapper, Ownable, IRewardDistributionRecipient		
L		Public ❗	🛑	NO ❗
L	setRewardDistribution	External ❗	🛑	onlyOwner
L	lastTimeRewardApplicable	Public ❗		NO ❗
L	rewardPerToken	Public ❗		NO ❗
L	earned	Public ❗		NO ❗
L	stake	Public ❗	🛑	updateReward updateLock

└	lock	Public ❗	🛑	updateReward updateLock
└	exit	External ❗	🛑	NO ❗
└	getReward	Public ❗	🛑	updateReward updateLock
└	notifyRewardAmount	External ❗	🛑	updateReward
└	getReserveReward	Internal 🔒		
└	changeReservePercent	External ❗	🛑	onlyOwner

BonusRewards	Implementation	Ownable, IRewardDistributionRecipientBonus		
└	initialize	Public ❗	🛑	NO ❗
└	updateReward	Public ❗	🛑	NO ❗
└	getReward	External ❗	🛑	onlyStakingRewards

L	lastTimeReward Applicable	Public ❗		NO ❗
L	rewardPerToken	Public ❗		NO ❗
L	viewRewards	External ❗		NO ❗
L	currentRewards	External ❗		NO ❗
L	_earned	Internal 🔒		
L	_deleteRewards	Internal 🔒	🛑	
L	setRewardDistrib ution	External ❗	🛑	onlyOwner
L	notifyRewardAm ount	External ❗	🛑	onlyReward Distribution

Ownable	Implementation			
L	initialize	Internal 🔒	🛑	
L	owner	Public ❗		NO ❗
L	isOwner	Public ❗		NO ❗
L	renounceOwners hip	Public ❗	🛑	onlyOwner

L	transferOwnership	Public ❗	🛑	onlyOwner
L	acceptOwnership	Public ❗	🛑	NO ❗
L	_transferOwnership	Internal 🔒	🛑	

SafeERC20	Library			
L	safeTransfer	Internal 🔒	🛑	
L	safeTransferFrom	Internal 🔒	🛑	
L	safeApprove	Internal 🔒	🛑	
L	safeIncreaseAllowance	Internal 🔒	🛑	
L	safeDecreaseAllowance	Internal 🔒	🛑	
L	callOptionalReturn	Private 🔐	🛑	

Address	Library			
L	isContract	Internal 🔒		

L	toPayable	Internal 🔒		
L	sendValue	Internal 🔒	🛑	

SafeMath	Library			
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	sub	Internal 🔒		
L	add	Internal 🔒		
L	mod	Internal 🔒		

IERC20	Interface			
L	totalSupply	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	transfer	External ❗	🛑	NO ❗
L	allowance	External ❗		NO ❗

L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗

Math	Library			
L	max	Internal 🔒		
L	min	Internal 🔒		
L	average	Internal 🔒		

IStakingRewards	Interface			
L	lockedTotalSupply	External ❗		NO ❗
L	allLocked	External ❗		NO ❗
L	allTimes	External ❗		NO ❗
IRewardDistributionRecipientBonus	Interface			

L	notifyRewardAmount	External ❗	🛑	NO ❗
L	setRewardDistribution	External ❗	🛑	NO ❗

LockedTokenWrapper	Implementation			
L	totalSupply	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	balanceLocked	Public ❗		NO ❗
L	available	Public ❗		NO ❗
L	canUnlock	Public ❗		NO ❗
L	stake	Public ❗	🛑	NO ❗
L	withdraw	Public ❗	🛑	NO ❗
L	lock	Public ❗	🛑	NO ❗
L	allLocked	Public ❗		NO ❗
L	allTimes	Public ❗		NO ❗

L	lockedTotalSupply	Public ❗		NO ❗
L	_unlockable	Internal 🔒	🛑	
L	_deleteLock	Internal 🔒	🛑	

IBonusRewards	Interface	IRewardDistributionRecipient		
L	initialize	External ❗	🛑	NO ❗
L	updateReward	External ❗	🛑	NO ❗
L	viewRewards	External ❗		NO ❗
L	currentRewards	External ❗		NO ❗
L	getReward	External ❗	🛑	NO ❗

IRewardDistributionRecipient	Interface			
L	notifyRewardAmount	External ❗	🛑	NO ❗

L	setRewardDistribution	External ❗	🛑	NO ❗

ERC20	Implementation	Context, IERC20		
L		Public ❗	🛑	NO ❗
L	name	Public ❗		NO ❗
L	symbol	Public ❗		NO ❗
L	decimals	Public ❗		NO ❗
L	totalSupply	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	allowance	Public ❗		NO ❗
L	approve	Public ❗	🛑	NO ❗
L	transferFrom	Public ❗	🛑	NO ❗
L	increaseAllowance	Public ❗	🛑	NO ❗

L	decreaseAllowance	Public ❗	🛑	NO ❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
L	_setupDecimals	Internal 🔒	🛑	
L	_beforeTokenTransfer	Internal 🔒	🛑	

Context	Implementation			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		

IERC20	Interface			
L	totalSupply	External ❗		NO ❗

└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗

SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		

ERC20Mock	Implementation	ERC20		
└	mintToSelf	Public ❗	🛑	NO ❗
└	mint	Public ❗	🛑	NO ❗

Reserve	Implementation	Ownable		
└		Public ❗	🛑	NO ❗
└	approve	External ❗	🛑	NO ❗
└	changeStakingRewards	External ❗	🛑	onlyOwner
└	changeBonusRewards	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Directory Structure

At time of review, the directory structure of the Coin smart contracts repository appeared as it does below. Our review, at request of Coin, covers the Solidity code (*.sol) as of sha256sum **338fc13c99840f95448f1df63266d44787af71a692ee29f9ef58f390758e8cff** of the CoinStaking.7z archive.

```
├── core
│   ├── BonusRewards.sol
│   ├── Reserve.sol
│   └── StakingRewards.sol
├── general
│   ├── LockedTokenWrapper.sol
│   ├── Ownable.sol
│   └── SafeERC20.sol
├── interfaces
│   ├── IBonusRewards.sol
│   ├── IERC20.sol
│   ├── IRewardDistributionRecipient.sol
│   ├── IRewardDistributionRecipientBonus.sol
│   └── IStakingRewards.sol
├── libraries
│   ├── Address.sol
│   ├── Math.sol
│   └── SafeMath.sol
└── mocks
    └── ERC20Mock.sol

5 directories, 15 files
```