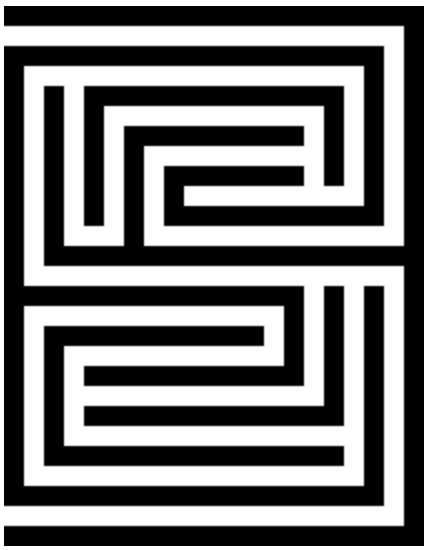

Aurox Public Report

PROJECT: Aurox
January 2022

Prepared For:

Giorgi Khazaradze | Aurox

giorgi@getaurox.com

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Executive Summary **3**
- Scope of Engagement 3
- Timeline 3
- Engagement Goals 3
- Contract Specification 3
- Overall Assessment 3
- Timeliness of Content 4

General Recommendations **6**

Best Practices & Solidity Development Guidelines **6**
- Administrative functions should emit an event 6
- Commented out code should be removed throughout 6
- Code does not follow the Solidity style guide 6

Specific Recommendations **7**
- Highly permissive owner account and centralization of power 7
- _stakeEndTime should be renamed or rewritten 7
- Adding liquidity and pulling in the same epoch results in improper rewards calculation 7

Toolset Warnings **8**
- Overview 8
- Compilation Warnings 8
- Test Coverage 8
- Static Analysis Coverage 8

Directory Structure **9**

Aurox Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in January of 2022 to perform a comprehensive security review of the Aurox smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of five days by a member of Bramah Systems, LLC. executive staff.

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

Timeline

Review Commencement: January 10th, 2022

Report Delivery: January 22nd, 2022

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Aurox 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, along with a verbose specification document which provided justification for infrastructure decisions and structural fundamentals.

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within

the codebase of the Aurox Protocol. During the course of our engagement, Bramah Systems found few instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT, as our report details.

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 Aurox 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 Aurox protocol or any other relevant product, service or asset of Aurox or otherwise. This report is not and should not be relied upon by Aurox 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 Aurox 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

Administrative functions should emit an event

Administrative functions (like those within **StakingMaster.sol**) should emit an event in the event the function is called to manipulate some element of the protocol (e.g. changing the provider address).

Resolution: Added event emitting to administrative functions.

Commented out code should be removed throughout

Throughout the protocol and test suite, there exists commented out code. As the code was presented within a version control system, the commented out code serves no purpose and can be otherwise removed (as previous versions featuring such snippets may be viewed).

Resolution: The main contracts appear to not have any commented code, the remaining commented code is likely left over from the test suites that were written for the first version. Due to time constraints the commented code will remain.

Code does not follow the Solidity style guide

There are numerous instances wherein the code does not follow the Solidity style guide (particularly around error message length). Where appropriate, a linter should be introduced in order to remove offending style guide violations.

Resolution: Whilst this is a valid point, due to time constraints the style guide violations will remain.

Specific Recommendations

Unique to the Aurox Protocol

Highly permissive owner account and centralization of power

The deploying account possesses a number of highly actions (namely, initiating protocol defaults and modifying payout parameters). 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 Aurox team provides the following, which Bramah feels sufficiently addresses the concern raised:

> *To resolve this it is optimal to create a Multi-Signature wallet and once all contracts are deployed to transfer the owner of those contracts to the Multi-Signature wallet.*
>
> *The Aurox Token, Staking Master and Provider are all Ownable inherited contracts, this is an accepted standard that allows the ownership of those contracts to be transferred at any time. Once deployment is complete it is recommended to transfer the ownership to the Multi-Signature wallet. This is achieved through a function call on each of the contracts.*

_stakeEndTime should be renamed or rewritten

The **_stakeEndTime** variable should be renamed or rewritten, as the end time must be exceeded in order for a claim to be valid, not simply reached. This could be resolved by changing the less than sign into less than or equal to, or by renaming the variable.

Resolution: Whilst that is a valid point, for the sake of maintaining similarities with the new versions of the contract, it is much simpler to leave this as is.

Adding liquidity and pulling in the same epoch results in improper rewards calculation

A user cannot properly add liquidity and pull within the same epoch. While this is vocalized to users within a code-comment, it should be made apparent within design documentation or prohibited from occurring.

Resolution: Design documentation updated to include this change. The user should not be

rewarded for adding liquidity and pulling it out within the same epoch. The code design is correct, documentation will be adjusted accordingly.

Toolset Warnings

Unique to the Aurox LLC 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

No warnings were present at time of compilation.

Test Coverage

The contract repository possesses extensive unit test coverage throughout. This testing provides a variety of unit tests which encompass the various operational stages of the 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 standard), or a concern stemming from the discovered risk was elevated to a larger issue and is referenced

above.

Directory Structure

At time of review, the directory structure of the Aurox smart contracts repository appeared as it does below. Our review, at request of Aurox, covers the Solidity code (*.sol) as of commit-hash **c0bd15483f110e11bf690228e52c2ade16af626c** of the Aurox repository.

```
.
├── Aurox LLC Audit.docx
├── README.md
├── constants
│   ├── New.json
│   ├── StakeStates.json
│   └── old_StakeStates.json
├── contracts
│   ├── Migrations.sol
│   ├── Provider
│   │   ├── EpochHelpers.sol
│   │   ├── IProvider.sol
│   │   ├── Provider.sol
│   │   ├── RewardHelpers.sol
│   │   └── artifacts
│   │       ├── IProvider.json
│   │       ├── IProvider_metadata.json
│   │       ├── Provider.json
│   │       └── Provider_metadata.json
│   ├── StakingMaster
│   │   ├── IStakingMaster.sol
│   │   ├── StakingMaster.sol
│   │   └── artifacts
│   │       ├── AuroxToken.json
│   │       ├── AuroxToken_metadata.json
```

```
│   │       ├── IStakingMaster.json
│   │       ├── IStakingMaster_metadata.json
│   │       ├── StakingMaster.json
│   │       └── StakingMaster_metadata.json
│   ├── TestHelpers
│   │   ├── ERC20.sol
│   │   ├── ERC20Mintable.sol
│   │   └── artifacts
│   │       ├── ERC20.json
│   │       └── ERC20_metadata.json
│   ├── Token
│   │   ├── AuroxToken.sol
│   │   ├── IAuroxToken.sol
│   │   ├── TokenVesting.sol
│   │   └── artifacts
│   │       ├── AuroxToken.json
│   │       ├── AuroxToken_metadata.json
│   │       ├── TokenVesting.json
│   │       └── TokenVesting_metadata.json
│   ├── __mocks__
│   │   └── Token.sol
│   └── artifacts
│       ├── AuroxToken.json
│       ├── AuroxToken_metadata.json
│       ├── AuroxVesting.json
│       ├── AuroxVesting_metadata.json
│       ├── VestingVault.json
│       └── VestingVault_metadata.json
├── coverage.json
├── hardhat.config.ts
├── old_migrations
```

```
│   ├── 1_initial_migration.js
│   ├── 2_Aurox_migration.js
│   ├── 3_MainContract_migrations.js
│   ├── backup
│   │   └── 2_Aurox_migration.js
│   └── helpers
│       ├── returnEthAmount.js
│       └── returnEthPrice.js
├── old_tests
│   └── javascript
│       ├── Integration
│       │   ├── all.js
│       │   └── claimingRewardsAddingLiquidity.js
│       ├── Provider
│       │   ├── addLiquidity.js
│       │   ├── addLiquidityFinal.js
│       │   ├── addLiquidityMoreComplex.js
│       │   ├── multipleEpochsMultipleUsers.js
│       │   ├── multipleEpochsSingleUser.js
│       │   ├── removeLiquidity.js
│       │   ├── removeLiquidityCheckAllFields.js
│       │   ├── returnAPR.js
│       │   ├── returnEpochToTimestamp.js
│       │   ├── returnIfInFirstDayOfEpoch.js
│       │   ├── returnRewardAmount.js
│       │   ├── returnTotalRewardsForEpoch.js
│       │   └── singleEpochsMultipleUsers.js
│       ├── StakingMaster
│       │   ├── addToStake.js
│       │   ├── closeStake.js
│       │   ├── currentStakeValue.js
```

```
│       │   ├── removeUsersStake.js
│       │   ├── returnInterestPercentage.js
│       │   ├── returnUsersTotalStakeValue.js
│       │   ├── returnValidUsersProviderStake.js
│       │   ├── simpleAndCompoundInterest.js
│       │   └── stakeState.js
│       └── helpers
│           ├── convertWeiToString.js
│           ├── fastForward.js
│           ├── removeAccuracy.js
│           └── returnStakeAddress.js
├── package-lock.json
├── package.json
├── secrets.json
├── tests
│   ├── Provider
│   │   ├── bugFixing.test.ts
│   │   ├── claimRewardsToStaking.test.ts
│   │   ├── rewards.test.ts
│   │   └── singleEpochsMultipleUsers.test.ts
│   ├── StakingMaster
│   │   ├── __mocks__
│   │   │   └── stakes.mock.ts
│   │   ├── addToStake.test.ts
│   │   ├── claimRewards.test.ts
│   │   ├── closeStake.test.ts
│   │   ├── createStake.test.ts
│   │   ├── createStaking.test.ts
│   │   ├── recreateStake.test.ts
│   │   └── returnStakesClaimableRewards.test.ts
│   └── helpers
```

```
│   ├── api
│   │   ├── ProviderAPI
│   │   │   ├── createProviderAPI.ts
│   │   │   ├── helpers
│   │   │   │   └── skipEpoch.ts
│   │   │   └── returnTotalRewardsForPastEpochs.ts
│   │   ├── StakingAPI
│   │   │   ├── createStaking.ts
│   │   │   ├── createStakingAPI.ts
│   │   │   ├── helpers.ts
│   │   │   └── returnFormattedStakingState.ts
│   │   ├── TokenAPI
│   │   │   └── createTokenAPI.ts
│   │   └── setup.ts
│   ├── assertions.ts
│   ├── common.ts
│   ├── constants.ts
│   ├── createAPI.ts
│   ├── deployLiquidityProvider.ts
│   ├── deployStakingMaster.ts
│   ├── deployToken.ts
│   └── modifiers.ts
├── truffle-config.js
├── tsconfig.json
├── types
│   └── typechain
│       ├── AuroxToken.d.ts
│       ├── ERC20.d.ts
│       ├── ERC20Mintable.d.ts
│       ├── EpochHelpers.d.ts
│       ├── IAuroxToken.d.ts
```

```
│   ├── IERC20.d.ts
│   ├── IERC20Metadata.d.ts
│   ├── IProvider.d.ts
│   ├── IStakingMaster.d.ts
│   ├── Migrations.d.ts
│   ├── Ownable.d.ts
│   ├── Provider.d.ts
│   ├── RewardHelpers.d.ts
│   ├── StakingMaster.d.ts
│   ├── Token.d.ts
│   ├── TokenVesting.d.ts
│   ├── common.d.ts
│   ├── factories
│   │   ├── AuroxToken__factory.js
│   │   ├── AuroxToken__factory.ts
│   │   ├── ERC20Mintable__factory.js
│   │   ├── ERC20Mintable__factory.ts
│   │   ├── ERC20__factory.js
│   │   ├── ERC20__factory.ts
│   │   ├── EpochHelpers__factory.js
│   │   ├── EpochHelpers__factory.ts
│   │   ├── IAuroxToken__factory.js
│   │   ├── IAuroxToken__factory.ts
│   │   ├── IERC20Metadata__factory.js
│   │   ├── IERC20Metadata__factory.ts
│   │   ├── IERC20__factory.js
│   │   ├── IERC20__factory.ts
│   │   ├── IProvider__factory.js
│   │   ├── IProvider__factory.ts
│   │   ├── IStakingMaster__factory.js
│   │   ├── IStakingMaster__factory.ts
```

```
│   │   ├── Migrations__factory.js
│   │   ├── Migrations__factory.ts
│   │   ├── Ownable__factory.js
│   │   ├── Ownable__factory.ts
│   │   ├── Provider__factory.js
│   │   ├── Provider__factory.ts
│   │   ├── RewardHelpers__factory.js
│   │   ├── RewardHelpers__factory.ts
│   │   ├── StakingMaster__factory.js
│   │   ├── StakingMaster__factory.ts
│   │   ├── TokenVesting__factory.js
│   │   ├── TokenVesting__factory.ts
│   │   ├── Token__factory.js
│   │   └── Token__factory.ts
│   ├── hardhat.d.ts
│   ├── index.js
│   └── index.ts
└── yarn.lock

34 directories, 163 files
```