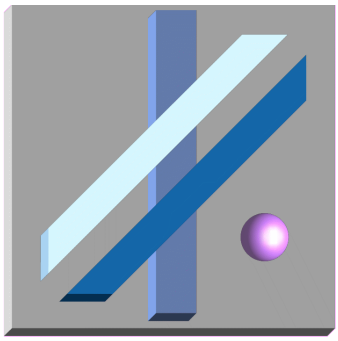

&& Servo Farms

Ajna Rewards and Governance Security Review

Reviewers

Kurt Barry (Fixed Point Solutions, LLC)

Brian McMichael (Servo Farms, LLC)

Report prepared by: Kurt Barry

January 12, 2024

Contents

1 About The Review Team

Fixed Point Solutions, LLC is a provider of Web3 security reviews and other bespoke technical consulting services. Servo Farms, LLC is a provider of Web3 security reviews and other bespoke technical consulting services.

2 Introduction

Ajna is a decentralized lending protocol with governance-minimized design. This security review covered a number of auxiliary contracts found in the `ajna-finance/contracts` and `ajna-finance/ecosystem-coordination` repositories:

- `PositionManager` (in `contracts`): allows bundling lending positions into ERC721 tokens.
- `RewardsManager` (in `contracts`): allows positions wrapped via PositionManager to be used to collect incentive rewards granted to lenders.
- `BurnWrapper` (in `ecosystem-coordination`): burns AJNA tokens, used for one-way transfers to L2s.
- `GrantFund` (in `ecosystem-coordination`): used to vote on how to disburse Ajna treasury funds.

The locked commits for the two repositories were:

- a2557591bb44ef02cfc0aee14c38e31ec66ad523 for `ajna-finance/contracts`;
- c6574925712a25a1612e7e690c8817a1ac6c67be for `ajna-finance/ecosystem-coordination`.

Disclaimer: No assessment can guarantee the absolute safety or security of a software-based system. Further, a system can become unsafe or insecure over time as it and/or its environment evolves. This assessment aimed to discover as many issues and make as many suggestions for improvement as possible within the specified timeframe. Undiscovered issues, even serious ones, may remain. Issues may also exist in components and dependencies not included in the assessment scope.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 28 days in total, Ajna Finance engaged with Fixed Point Solutions, LLC and Servo Farms, LLC to review Ajna Finance. In this period of time a total of 16 issues were found.

Summary

Project Name	Ajna Finance
Repository	ajna-finance/contracts
Commit	a2557591bb44ef02cfc0aee...
Repository	ajna-finance/ecosystem-coordination
Commit	c6574925712a25a1612e7e6...
Type of Project	Lending, DeFi
Audit Timeline	Aug 7th 2023 - Aug 31st 2023
Methods	Manual Review

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	2
Gas Optimizations	6
Informational	8
Total Issues	16

5 Findings

5.1 Low Risk

5.1.1 `getDelegateReward` Does Not Account for Whether The Voter Has Cast Screening Votes

Context: GrantFund.sol#L1095

Description: Voters are only eligible to claim delegate rewards after the funding stage if they also voted during the screening stage. However, the `view` function `getDelegateReward`, which is intended to return the amount of delegate rewards available to a voter in a distribution, does not account for this rule. As a consequence, UIs or users relying on this function may suffer confusion and wasted gas because the function will erroneously report non-zero rewards in some cases.

Recommendation: Return zero from `getDelegateReward` if `voter.screeningVotesCast == 0`.

Ajna: Fixed in commit 545ec4290f41b4b943eec99be1de78169aefc14e.

Review Team: Fix verified.

5.1.2 UpdateExchangeRate Event Can Be Emitted With Indexes That Weren't Updated

Context: RewardsManager.sol#L664

Description: If `curBurnEpoch != 0`, `totalBurnedInEpoch != 0`, and `block.timestamp > curBurnTime + UPDATE_PERIOD`, no exchange rate updates are actually done but the full list of indexes is still emitted in the `UpdateExchangeRates` event. This could be misleading for consumers of the event.

Recommendation: Only emit the event if updates to exchange rates actually occurred.

Ajna: Fixed in commit f404a7e2e40f55749238c55bbce7368674106cc3.

Review Team: Fix verified.

5.2 Gas Optimization

5.2.1 Unnecessary Check and Storage Usage For Surplus Update Status

Context: GrantFund.sol#L64

Description: It's unnecessary to check `_isSurplusFundsUpdated` on the indicated line of `startNewDistributionPeriod()`, as this is the only function that calls `_updateTreasury` which in turn is the only function that modifies `_isSurplusFundsUpdated`--because `startNewDistribution()` increments the distribution id, the surplus will be updated exactly once per distribution. Further since `_isSurplusFundsUpdated` isn't relied on anywhere else, it could be eliminated entirely.

Recommendation: Remove the `_isSurplusFundsUpdated` mapping to reduce gas costs and code complexity.

Ajna: Acknowledged.

5.2.2 Absolute Value Function Does Not Support All Input Values and Can Be Optimized

Context: Maths.sol#L28

Description: This absolute value computation will revert when its input is equal to `-2**255` (`-x` specifically is checked for overflow in Solidity 0.8.x), even though `2**255` is representable as a `uint256`. This edge case likely isn't a concern in practice, but strictly speaking the function does not support its full range of possible inputs. Further, given that the sign of `x` is being checked and the casting is from `int256` to `uint256`, the use of the `SafeCast` library is unnecessary.

An example of an implementation that supports the full range of possible input values and is a bit cheaper gas-wise is the following:

```
function abs(int256 x) internal pure returns (uint256 z) {
    unchecked {
        z = x >= 0 ? uint256(x) : uint256(-x);
    }
}
```

Recommendation: Consider adopting an absolute value function implementation that supports all possible inputs and does not perform unnecessary checks.

Ajna: Acknowledged, prefer not to make unnecessary changes.

5.2.3 Redundant Check in `_updateBucketExchangeRates`

Context: RewardsManager.sol#L614

Description: The behavior of `_getEpochInfo()` is to return zero for all return values if the epoch argument is zero, so this condition could be just `if (totalBurnedInEpoch == 0)` instead of `if (curBurnEpoch == 0 || totalBurnedInEpoch == 0)`.

Recommendation: Simplify the conditional to `if (totalBurnedInEpoch == 0)` to reduce gas costs and code verbosity.

Ajna: Fixed in commit f404a7e2e40f55749238c55bbce7368674106cc3.

Review Team: Fix verified.

5.2.4 RewardsManager: Reliance on nested helpers obfuscates code duplication

Context: RewardsManager.sol

Description: Heavy reliance on helper functions throughout codebase can obfuscate code path and hide multiple calls to external contracts.

Recommendation: Use of helper functions can produce code duplication that add to cost of use are difficult to uncover. For example, in `RewardsManager.stake()` the `currentBurnEpoch` is called in the `stake` function and again in the `_updateBucketExchangeRates` helper. These external calls can be removed, and cost of interaction reduced, by re-using the data.

See example fix at https://github.com/ajna-finance/contracts/pull/940

Ajna: Changes were merged at https://github.com/ajna-finance/ajna-core/pull/940

Review Team: Understood.

5.2.5 BurnWrapper: Can remove parameter from constructor

Context: BurnWrapper.sol#L57

Description: BurnWrapper requires parameter input exactly matching a constant variable. This procedure can be simplified and optimized.

Recommendation: See ecosystem-coordination#127 for a proposed solution and gas effects.

Ajna: Acknowledged, but the contract has already been deployed so will not take any action.

Review Team: Understood.

5.2.6 Convert `_nextId` from `uint176` to `uint256`

Context: PositionManager.sol#L52

Description: `uint176` for `_nextId` increases gas costs. See PR#932 for detailed gas calculations.

Recommendation: Convert to `uint256`

Ajna: Changes integrated at https://github.com/ajna-finance/ajna-core/commit/d343cc9531b1236443e293a566a4e07da180c13

Review Team: Understood.

5.3 Informational

5.3.1 `AlreadyVoted` Error Is Unused

Context: IGrantFundErrors.sol#L18

Description: The `AlreadyVoted` error is not used in the GrantFund contract; it appears further that it is not needed in the code as it exists, as voters are limited in their voting activities by their voting weight as derived from snapshotted voting balances.

Recommendation: Remove the unused error.

Ajna: Fixed in commit b65ee3a172fc4013533cb1ddc882df73e116d1b0.

Review Team: Fix verified.

5.3.2 GrantFund: Potential Tally Incompatibility

Context: GrantFund.sol

Description: The public function signatures of the GrantFund do not appear to be fully compatible with Tally's OpenZeppelin Governor or Compound Governor Bravo.

From Tally:

To be compatible with Tally, your Governor will need these function signatures:

```
function votingDelay() public view virtual returns (uint256);
function votingPeriod() public view virtual returns (uint256);
function quorum(uint256 blockNumber) public view virtual returns (uint256);
function proposalThreshold() public view virtual returns (uint256);
function state(uint256 proposalId) public view virtual override returns (
    ProposalState
);

function getVotes(
    address account,
    uint256 blockNumber
) public view virtual returns (uint256);

function propose(
```

```
    address[] memory targets,
    uint256[] memory values,
    bytes[] memory calldatas,
    string memory description
) public virtual returns (uint256 proposalId);

function execute(
    address[] memory targets,
    uint256[] memory values,
    bytes[] memory calldatas,
    bytes32 descriptionHash
) public payable virtual returns (uint256 proposalId);

function castVote(
    uint256 proposalId,
    uint8 support
) public virtual returns (uint256 balance);

function castVoteWithReason(
    uint256 proposalId,
    uint8 support,
    string calldata reason
) public virtual returns (uint256 balance);

function castVoteBySig(
    uint256 proposalId,
    uint8 support,
    uint8 v,
    bytes32 r,
    bytes32 s
) public virtual returns (uint256 balance);
```

Recommendation: The team has expressed an interest in supporting ecosystem governance UI's, Tally was specifically mentioned. Review the public interfaces of the GrantFund and determine which functions should be added or altered to conform to the Tally UI. Not sure I have a recommendation for a fix because Ajna uses a multi-phase voting system.

Ajna: Noted that there are some incompatibilities. We attempted to strike a balance between having a consistent interface, and not including additional methods and complexity that increased gas costs and risk unnecessarily.

5.3.3 GrantFund Script: Do not suggest transfer.

Context: GrantFund.s.sol

Description: Output from GrantFund Deployment script could cause confusion.

The GrantFund deployment script says `"Please transfer %s AJNA (%s WAD) into the treasury"`, this is a somewhat incorrect statement as sending funds directly to the Grant fund using AJNA's `transfer` function will cause the funds to be locked. Funds must be added to the treasury using the `fundTreasury()` function for them to be allocated to the treasury.

Recommendation: Update the script to explicitly state funds must be transferred to the treasury using the `fundTreasury` function in the grant fund.

BONUS POINTS: Add a permissionless function to the Grant Fund that tops up the treasury balance if `AJNA.balanceOf(GrantFund) > treasury`, this will allow the treasury to be increased by any Ajna tokens accidentally or intentionally sent directly to the Grant fund contract, otherwise they are effectively burned.

Ajna: Fixed in commit d31162fa1d0eed44fcacbaccd46a30ad21742231.

Review Team: Fix Verified.

5.3.4 General: Beware deleting structs with internal mappings does not delete underlying mapping data.

Context: PositionManager.sol, RewardsManager.sol

Description: Note that there are several instances in these contracts where `mapping`s contain `struct`s that also include `mapping`s. Note that solidity will not clear up the sub-mapping data via the `delete` keyword.

See: https://docs.soliditylang.org/en/latest/types.html

For example, in `PositionManager`, deletion of `positionTokens[tokenId_]` will not clear up the `positions` mapping in the underlying `TokenInfo` struct, this should not be an issue as new positionTokens will generate a new `tokenId`, so in this case consider whether the deletion is necessary at all.

In `RewardsManager`, the `snapshot` mapping is cleared up in `_unstake`, and the invariant that it's all cleaned up is covered in the tests, but this should probably be tested again after a re-stake when the `stakes` mapping is re-initialized with a tokenId.

Recommendation: Evaluate all scenarios where a mapping contains a mapping and consider whether deletion of the key is sufficient protection from a re-initialized parent mapping location.

Ajna: Acknowledged.

5.3.5 BurnWrapper: Is ERC20Burnable required?

Context: BurnWrapper.sol#L9

Description: BurnWrapper inherits from ERC20Burnable, which enables the burn-wrapped token itself to be burned. It is unclear if this is necessary in this context or if there is a wider purpose for burning the wrapped token. Preventing the BurnWrapper ERC20 token from being burned directly might allow users to quickly inspect how many Ajna have been burned for L2 use by way of the `totalSupply` function, however, if these tokens are burned themselves the `totalSupply` will decrease and provide a less accurate high-level view of the number of tokens that have been moved off mainnet via this mechanism.

Recommendation: Consider whether burning of this token is a requirement and, if not, consider removing the inheritance of ERC20Burnable interface from the contract.

Ajna: Acknowledged.

5.3.6 Missing Documentation of External Calls In PositionManager Functions

Context: [1]PositionManager.sol#L163-L165 [2]PositionManager.sol#L269-L271

Description: [1] The list of external calls for `memorializePositions` is incomplete; it omits the call to `IPool.lpAllowance` on line 207. [2] The list of external calls for `moveLiquidity` is incomplete; it omits the call to `IPool.updateInterest` on line 307.

Recommendation: Add the omitted external calls to the function comments.

Ajna: Fixed.

Review Team: Fix verified.

5.3.7 PositionManager: Missing tests of ERC721 Pools

Context: PositionManager.t.sol

Description: The test harness builds a setup that tests against ERC20 non-subset pools only.

Recommendation: Include additional tests that touch pools generated by the ERC721 pool factory and ERC721 subset pools.

Ajna: Fixed in commit 362cc74acac5196be1015b94c95d98c7e99dc944.

Review Team: Fix verified.

5.3.8 Increase usability by removing `poolSubsetHash_` requirement for most pools.

Context: PositionManager.sol#L252 PositionManager.sol#L555

Description: `poolSubsetHash` is known for most standard pools and is really only required for ERC721 subset pools. Further, ERC721 subset pools do not emit, return, or store this hash upon pool creation, requiring offline calculation to reconstitute the hash. Requiring most use-cases to track down this information when it is already available on the respective factories is an additional development burden on integrators.

Recommendation: See a proposed solution in PR#937, which solves this problem for all pools by allowing users to provide only the `pool` address as a parameter for the affected functions, the existing interface remains unchanged for subset ERC721 pools. There are additional optimizations and benefits described in the PR.

Ajna: Acknowledged but not plans to make this change unless an issue is discovered.

Review Team: Understood.