QuillAudits

Audit Report
October, 2023

For

ASTRA
DAO

Table of Content

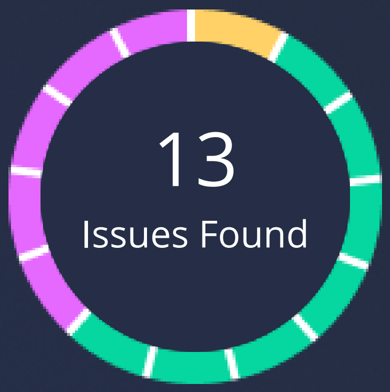
Table of Content

Table of Content

Executive Summary

Project Name	Astra DAO
Project URL	*https://astradao.org/*
Overview	Astra DAO is a decentralized and non-custodial automated crypto asset allocator. Astra DAO provides convenient and practical access to crypto-oriented investment strategies. ASTRA token is responsible for governance of the whole ecosystem which can be earned through various investment products/indices, participation units marketplace, user staking, harvesting investment strategies profits.
Audit Scope	*https://github.com/astradao/astra-private/tree/foundry-migrate/astra-smartcontracts/src*
Contracts in Scope	poolv2.sol chefv2.sol itoken-staking.sol governance.sol governance-layer2.sol timelock.sol
Commit Hash	1420b3f57d38e4318e7852b5fdfa2090dca0ee7f
Language	Solidity
Blockchain	ARBITRUM
Method	Manual Analysis, Functional Testing, Automated Tests
Review 1	4th September 2023 - 11th October 2023
Updated Code Received	12th October 2023
Review 2	13th October 2023 - 18th October 2023
Fixed In	f6a8f3781a0efa60e8fe2a91fcc8230c0da1c722

Number of Security Issues per Severity



🟥 High	🟨 Medium
🟩 Low	🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	1	2	2
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	5	3

Checked Vulnerabilities

- ✓ Access Management
- ✓ Arbitrary write to storage
- ✓ Centralization of control
- ✓ Ether theft
- ✓ Improper or missing events
- ✓ Logical issues and flaws
- ✓ Arithmetic Correctness
- ✓ Race conditions/front running
- ✓ SWC Registry
- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC's conformance
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Multiple Sends
- ✓ Using suicide
- ✓ Using delegatecall
- ✓ Upgradeable safety
- ✓ Using throw

Checked Vulnerabilities

- ✓ Using inline assembly
- ✓ Style guide violation
- ✓ Unsafe type inference
- ✓ Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Solhint, Mythril, Slither, Solidity Statistic Analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

A. Contract - poolv2.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

A.1 Missing require statement

Description

In contract poolV2 the oracle provides the details such as name, symbol, etc., As for the new upgraded deployment the team is going to remove the oracle related functionality from the code. The function **addPublicPool()** takes a tokens array which should check the length of the token after the code is removed.

Remediation

Consider adding require check e.g require(_tokens.length > 0, “ERR”).

Status

Resolved

Informational Issues

A.2 Use of incorrect error message string identifier

Description

E15 is used for **require(_tokens.length > 0, "E15");** as an error message on **L184** and on **L617**. But the **E15** belongs to the "Zero address" according to comments.

E02 is used for **require(_tokens.length == _weights.length, "E02");** as an error message but the **E02** belongs to the "Invalid Pool Index" according to comments. **E06** should be used here.

A.2 Use of incorrect error message string identifier

Remediation

Use another error if the used error string is not intentional.

Status

Resolved

B. Contract - itoken-staking.sol

NOTE about emergency withdrawal:

We expect that the development team is aware that even after using emergency withdrawal functionality in an emergency the users can still try to withdraw or claim tokens (in case not all token are removed). As discussed with the development team, the team is going to add the withdrawn tokens back again once the issue is patched so that the previously stored contract state can be used.

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

B.1 Misleading comment

Description

In contract itoken-staking there is emergency only functionality for withdrawing astra tokens and it's comment says: // Withdraw without caring about rewards. EMERGENCY ONLY. But the astra token is itself a reward token so the comment is inappropriate.

B.1 Misleading comment

Remediation

Please remove the comment or change it to appropriate meaning.

Status

Resolved

B.2 Event parameter related issue

Description

In contract itoken-staking there is emergency only functionality where an event is emitted for withdrawing astra tokens and pid is passed as **type(uint).max**. As in reality, **emergencyAstraWithdraw()** function does not take any pid.

Remediation

Check that the pid value **type(uint).max** getting passed to the **emergencyAstraWithdraw()** function is intentional and won't create any problem with the service listening for the events. Additionally new event can be created for this which won't take a pid.

Status

Resolved

Informational Issues

No issues were found.

C. Contract - Chefv2.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

C.1 Wrong event parameter passed

Description

In **emergencyNFTWithdraw()** function the **EmergencyWithdraw()** event is getting emitted, where the **pid** is getting passed as **_tokenId**.

Remediation

Another event can be added for nft withdrawal which won't take pid.

Status

Resolved

Informational Issues

C.2 emergencyNFTWithdraw() allows to transfer 1 NFT at a time

Description

emergencyNFTWithdraw() allows to transfer **1** NFT lp at a time which would be slow in the time of emergency.

Remediation

Arrays can be taken to take multiple nft ids which will get transferred. Still there would be limitations for how many ids can be passed in one call because of gas limitations.

Status

Resolved

D. Contract - governance.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Informational Issues

D.1 Storage collision because of struct modification

Description

In the new implementation there's a new **string chain** variable added in the **Proposal** struct which is getting used in **proposals** mapping. It can create a collision as it is getting added between the previous variable sequence. **chain** is getting added to the place of the target address array where it would be storing the length of the array. It should be added after all previous variables in the struct.

Remediation

Add new variables after the previous variables in the struct.

Status

Acknowledged

Astra DAO team's comment

We will not be upgarding the contracts anymore, will be deploying new set of contarcts on arbitrum.

Auditor's comment

The issue was added as medium severity but later moved to informational as the Astra Dao team is not planning to upgrade and because it has no impact on new deployment.

E. Contract - governance-layer2.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

E.1 Check that there's no need to send value for crosschain call

Description

Axelar's gateway contract calls **execute()** from **AxelarExecutable** which is getting inherited by the **GovernorBeta** (i.e governance-layer2). The **execute()** calls overridden _**execute()** which makes a call on the target contract with the value from the decoded data.

It needs to be noted that **execute()** is not payable so it can't receive the native blockchain token value in case the call in _**execute()** requires it.

If axelar doesn't supports the call with native value then the **receive()** function can be added in the **governance-layer2** to ensure that the contract can receive the value which can be then used by **call{}()** in _**execute()**.

Remediation

Consider verifying the business requirement and testing the call on target contract which requires blockchain native value.

Status

Resolved

Astra DAO team's comment

Resolved by adding receive() function in governance-layer2. The team will send native token balance to governance-layer2 contract which will be used while calling on target contract.

Informational Issues

No issues were found.

F. Contract - timelock.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

F.1 Check that theres no need to attach tokens for crosschain call

Description

Currently in **executeTransaction()** it is calling **gateway.callContract()** which comes without the ability to attach a token. to attach tokens axelar recommends **callContractWithToken()** function, check the use of **callContractWithToken** is not needed for intentional logic.

Remediation

Check the use of **callContractWithToken()** is not needed.

Status

Acknowledged

Astra DAO team's comment

There is no need to send any tokens with cross-chain call.

F.2 Non-EVM chain compatibility is missing

Description

setL2GovernanceContract() functions stores **_L2GovernanceContracts** mapping value for chain. which maps string chain to governance contract address. In case the governance-layer2 contract is deployed on non evm chain where the address length is different than 20 bytes, the function won't be able to store the addreses which are not 20 bytes.

F.2 Non-EVM chain compatibility is missing

Remediation

If the protocol is only going to be working on evm chains then it can be kept as it is, if not then please consider changing **_L2GovernanceContracts** mapping to **string => string** instead of **string => address**.

Status

Acknowledged

Astra DAO team's comment

We will only be extending to EVM compatible chains.

Informational Issues

F.3 Unused events

Description

NewChainAdded() is declared but never emitted.

Remediation

Remove or use the unused event.

Status

Resolved

F.4 Axelar specific assumptions

Description

1. It is assumed that Axelar takes care of replay protection, once the call is executed on the destination chain then that can't be executed again for that same request.

2. If the gas value sent is not enough for cross chain transaction and the transaction is pending on the axelar network then the gas value can be increased on axelar so that it will execute. According to astra dao development team they will be calculating the gas required for the transaction using the axelar api so that enough value can be sent for the transaction.

Status

Acknowledged

G. Common issues

High Severity Issues

No issues were found.

Medium Severity Issues

G.1 Centralization risk

Description

Functions like **emergencyNFTWithdraw()**, **emergencyWithdraw()** and **emergencyWithdraw()**, **emergencyAstraWithdraw()** from **chefv2** and **itoken-staking** respectively create centralization risk where malicious owner can withdraw tokens deposited by users and rewards earned on them.

A multisig wallet should be used to handle these type of actions so that risk of centralization can be mitigated to some extent.

Remediation

Use multisig wallet for handling the emergency withdrawal functionalities.

Status

Acknowledged

Astra DAO team's comment

The ownership will be transferred to governance after deployment.

Low Severity Issues

No issues were found.

Informational Issues

No issues were found.

Functional Tests

Some of the tests performed are mentioned below:

- poolv2.sol

✓ Should be able to rebalance with base stable coin

- itoken-staking.sol

✓ Should be able to withdraw itokens with emergencyWithdraw

✓ Should be able to withdraw astra tokens with emergencyAstraWithdraw

- chefv2.sol

✓ Should be able to withdraw deposited NFT lp with emergencyNFTWithdraw

- governance-layer2.sol

✓ Only owner should be able to set the manager address

✓ Only owner should be able to set the manager chain

✓ _execute() should execute call with data as calldata when the signature string length is 0

✓ _execute() should execute call by creating calldata when the signature string length is nonzero

- timelock.sol

✓ Should be able to set governance-layer2 contract address for a chain

✓ Should be able to send payload to the gateway contract using callContract

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Astra DAO. We performed our audit according to the procedure described above.

Some issues of Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Astra DAO smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Astra DAO smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services.. It is the responsibility of the Astra DAO to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

850+
Audits Completed

$30B
Secured

$30B
Lines of Code Audited

Follow Our Journey

Audit Report
October, 2023

For

ASTRA
DAO

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com