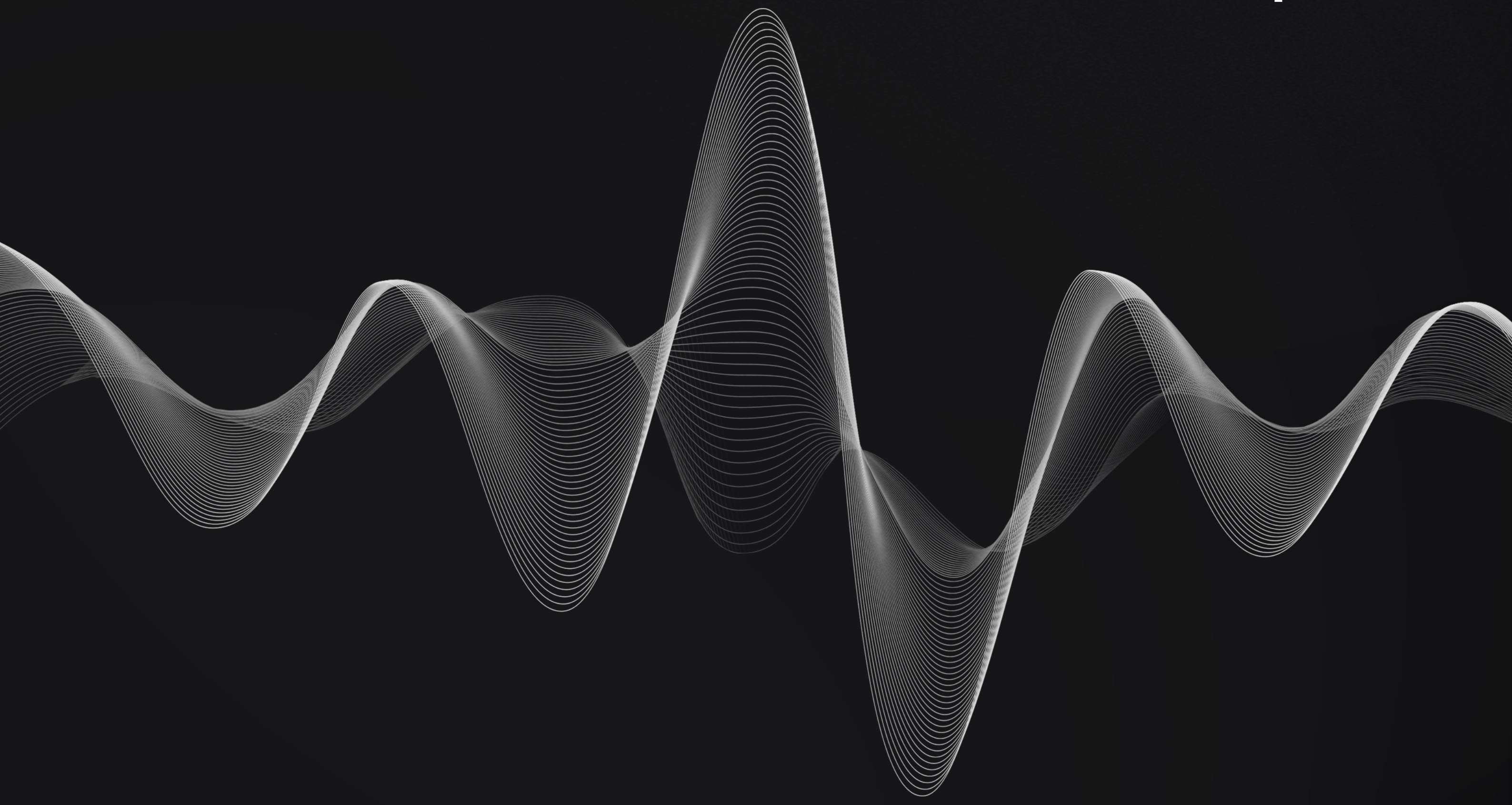
RESONANCE

EclipseFi

Equinox Protocol Audit Report



Blockchain, Emerging Technology, and Web2
CYBERSECURITY PRODUCT & SERVICE ADVISORY

Document Control

PUBLIC **FINAL**(v2.1)

Audit_Report_ECLP-EQN_FINAL_21

Sep 6, 2024	○	v0.1	Michal Bazyli: Initial draft
Sep 6, 2024	○	v0.2	Michal Bazyli: Added findings
Sep 6, 2024	●	v1.0	Charles Dray: Approved
Sep 9, 2024	○	v1.1	Michal Bazyli: Updated findings
Sep 16, 2024	○	v1.2	Michal Bazyli: Reviewed findings
Sep 20, 2024	●	v2.0	Charles Dray: Finalized
Oct 2, 2024	○	v2.1	Charles Dray: Published

Points of Contact	Ayush Shaw	EclipseFi	ayushs@eclipsefi.io
	Charles Dray	Resonance	charles@resonance.security
Testing Team	Michał Bazyli	Resonance	michal@resonance.security
	Michal Bajor	Resonance	michal.bajor@resonance.security
	João Simões	Resonance	joao@resonance.security

Copyright and Disclaimer

Contents

Executive Summary

EclipseFi contracted the services of Resonance to conduct a comprehensive security audit of their smart contracts ***between 19th August, 2023*** and ***6th of September, 2023***. The primary objective of the assessment was to identify any potential security vulnerabilities and ensure the correct functioning of smart contract operations.

During the engagement, Resonance allocated ***3*** engineers to perform the security review. The engineers, including an accomplished professional with extensive proficiency in blockchain and smart-contract security, encompassing specialized skills in advanced penetration testing, and in-depth knowledge of multiple blockchain protocols, devoted ***13 days*** to the project. The project's test targets, overview, and coverage details are available throughout the next sections of the report.

The ultimate goal of the audit was to provide **EclipseFi** with a detailed summary of the findings, including any identified vulnerabilities, and recommendations to mitigate any discovered risks. The results of the audit are presented in detail further below.

System Overview

Eclipse Equinox marks a significant step in addressing the liquidity problem for new projects. As a governance aggregator, it leverages the eclipASTRO token to transfer the vxASTRO voting rights to ECLIP stakers. It does that by accepting $ASTRO and $xASTRO deposits, which are then bonded on Astroport's platform to receive $vxASTRO in return.

Repository Coverage and Quality

Code	**Tests**	**Documentation**
8 / 10	8 / 10	8 / 10

Resonance's testing team has assessed the Code, Tests, and Documentation coverage and quality of the system and achieved the following results:

- The code follows development best practices and makes use of standard libraries, and language guides but does not make use of some known patterns. It is easily readable and uses the latest stable version of relevant components. Overall, **code quality is good**.
- Unit and integration tests are included. The tests cover both technical and functional requirements. Code coverage is undetermined. Overall, **tests coverage and quality is good**.
- The documentation only includes technical details for the code, relevant explanations of workflows and interactions. Overall, **documentation coverage and quality is good**.

Target

The objective of this project is to conduct a comprehensive review and security analysis of the smart contracts that are contained within the specified repository.

The following items are included as targets of the security assessment:

- Repository: `EclipsePad/equinox-contracts`
- Hash: 388763fbf092b50cfa8cebfb0a1bd73b56138258

- contracts/lp_depositor
- contracts/lp_staking
- contracts/single*sided*staking
- contracts/voter

The following items are excluded:

- External and standard libraries
- Files pertaining to the deployment process
- Financial related attacks

Methodology

In the context of security audits, Resonance's primary objective is to portray the workflow of a real-world cyber attack against an entity or organization, and document in a report the findings, vulnerabilities, and techniques used by malicious actors. While several approaches can be taken into consideration during the assessment, Resonance's core value comes from the ability to correlate automated and manual analysis of system components and reach a comprehensive understanding and awareness with the customer on security-related issues.

Resonance implements several and extensive verifications based off industry's standards, such as, identification and exploitation of security vulnerabilities both public and proprietary, static and dynamic testing of relevant workflows, adherence and knowledge of security best practices, assurance of system specifications and requirements, and more. Resonance's approach is therefore consistent, credible and essential, for customers to maintain a low degree of risk exposure.

Ultimately, product owners are able to analyze the audit from the perspective of a malicious actor and distinguish where, how, and why security gaps exist in their assets, and mitigate them in a timely fashion.

Source Code Review - Rust CosmWasm

During source code reviews for Web3 assets, Resonance includes a specific methodology that better attempts to effectively test the system in check:

1. Review specifications, documentation, and functionalities
2. Assert functionalities work as intended and specified
3. Deploy system in test environment and execute deployment processes and tests
4. Perform automated code review with public and proprietary tools
5. Perform manual code review with several experienced engineers
6. Attempt to discover and exploit security-related findings
7. Examine code quality and adherence to development and security best practices
8. Specify concise recommendations and action items
9. Revise mitigating efforts and validate the security of the system

Additionally and specifically for Rust CosmWasm audits, the following attack scenarios and tests are recreated by Resonance to guarantee the most thorough coverage of the codebase:

- Frontrunning attacks
- Unsafe third party integrations
- Denial of service
- Access control issues
- Inaccurate business logic implementations
- Incorrect gas usage

- Arithmetic issues
- Unsafe callbacks
- Timestamp dependence
- Mishandled panics, errors and exceptions

Severity Rating

Security findings identified by Resonance are rated based on a Severity Rating which is, in turn, calculated off the **impact** and **likelihood** of a related security incident taking place. This rating provides a way to capture the principal characteristics of a finding in these two categories and produce a score reflecting its severity. The score can then be translated into a qualitative representation to help customers properly assess and prioritize their vulnerability management processes.

The **impact** of a finding can be categorized in the following levels:

1. Weak - Inconsequential or minimal damage or loss
2. Medium - Temporary or partial damage or loss
3. Strong - Significant or unrecoverable damage or loss

The **likelihood** of a finding can be categorized in the following levels:

1. Unlikely - Requires substantial knowledge or effort or uncontrollable conditions
2. Likely - Requires technical knowledge or no special conditions
3. Very Likely - Requires trivial knowledge or effort or no conditions

			Likelihood	
		Very Likely	Likely	Unlikely
Impact	Strong	Critical	High	Medium
	Medium	High	Medium	Low
	Weak	Medium	Low	Info

Repository Coverage and Quality Rating

The assessment of Code, Tests, and Documentation coverage and quality is one of many goals of Resonance to maintain a high-level of accountability and excellence in building the Web3 industry. In Resonance it is believed to be paramount that builders start off with a good supporting base, not only development-wise, but also with the different security aspects in mind. A product, well thought out and built right from the start, is inherently a more secure product, and has the potential to be a game-changer for Web3's new generation of blockchains, smart contracts, and dApps.

Accordingly, Resonance implements the evaluation of the code, the tests, and the documentation on a score **from 1 to 10** (1 being the lowest and 10 being the highest) to assess their quality and coverage. In more detail:

- Code should follow development best practices, including usage of known patterns, standard libraries, and language guides. It should be easily readable throughout its structure, completed with relevant comments, and make use of the latest stable version components, which most of the times are naturally more secure.
- Tests should always be included to assess both technical and functional requirements of the system. Unit testing alone does not provide sufficient knowledge about the correct functioning of the code. Integration tests are often where most security issues are found, and should always be included. Furthermore, the tests should cover the entirety of the codebase, making sure no line of code is left unchecked.
- Documentation should provide sufficient knowledge for the users of the system. It is useful for developers and power-users to understand the technical and specification details behind each section of the code, as well as, regular users who need to discern the different functional workflows to interact with the system.

Findings

During the security audit, several findings were identified to possess a certain degree of security-related weaknesses. These findings, represented by unique IDs, are detailed in this section with relevant information including Severity, Category, Status, Code Section, Description, and Recommendation. Further extensive information may be included in corresponding appendices should it be required.

An overview of all the identified findings is outlined in the table below, where they are sorted by Severity and include a **Remediation Priority** metric asserted by Resonance's Testing Team. This metric characterizes findings as follows:

- **"Quick Win"** Requires little work for a high impact on risk reduction.
- **"Standard Fix"** Requires an average amount of work to fully reduce the risk.
- **"Heavy Project"** Requires extensive work for a low impact on risk reduction.

RES-01	🟧	External dependency on incentive distribution in lp_staking	ılıllı	Acknowledged
RES-02	🟧	External Dependency on Incentive Distribution in single_sided_staking contract	ılıllı	Acknowledged
RES-03	🟩	Lack of Two-Step Admin Ownership Accept in lp_staking Contract	ılıllı	Resolved
RES-04	🟩	Lack of parameter validation in try_update_date_config function	ılıllı	Acknowledged
RES-05	🟩	Lack of Two-Step Admin Ownership Accept in single_side_staking contract	ılıllı	Resolved
RES-06	⬜	Presence of unused commented-out code in the codebase	ılıllı	Resolved

External dependency on incentive distribution in lp_staking

Medium **RES-ECLP-EQN01** Business Logic **Acknowledged**

Code Section

- Not Specified

Description

The `lp_staking` contract relies on external sources to send incentives for distribution to stakers, as it lacks the ability to mint rewards internally. This creates a vulnerability where stakers may not receive expected rewards if the external sources fail or are compromised. An attacker or misconfigured system could disrupt reward distribution, leading to loss of user trust and participation in the staking mechanism.

Recommendation

It is recommended to implement redundancy and checks to ensure consistent reward distribution, and consider adding internal minting capabilities or secured oracle mechanisms for better reliability.

Status

The issue was acknowledged by Eclipse team. The development team stated that "Incentives are provided periodically by Eclipsefi DAO".

External Dependency on Incentive Distribution in single_sided_staking contract

Medium **RES-ECLP-EQN02** Business Logic **Acknowledged**

Code Section

- Not Specified

Description

The `single_sided_staking` contract relies on external sources to send incentives for distribution to stakers, as it lacks the ability to mint rewards internally. This creates a vulnerability where stakers may not receive expected rewards if the external sources fail or are compromised. An attacker or misconfigured system could disrupt reward distribution, leading to loss of user trust and participation in the staking mechanism.

Recommendation

It is recommended to implement redundancy and checks to ensure consistent reward distribution, and consider adding internal minting capabilities or secured oracle mechanisms for better reliability.

Status

The issue was acknowledged by Eclipse team. The development team stated that "Incentives are provided periodically by Eclipsefi DAO"*.*

Lack of Two-Step Admin Ownership Accept in lp_staking Contract

Low

RES-ECLP-EQN03 Code Quality **Resolved**

Code Section

- `contracts/lp_staking/src/entry/execute.rs`

Description

The `lp_staking` contract lacks a two-step ownership transfer acceptance mechanism. Currently, ownership of the contract can be transferred in a single transaction without requiring acceptance or verification from the new owner. This practice introduces operational risks and potential vulnerabilities.

Recommendation

It is recommeded to implement a two-step ownership transfer mechanism:

- The current owner initiates the ownership transfer, proposing a new owner.
- The new owner must explicitly accept the ownership transfer through a separate transaction within a predefined timeframe.

Status

The issue has been fixed in 0d4d93e3cc6e35f0fd279177bdf4fa880c31c0e9.

Lack of parameter validation in try_update_date_config function

Low

RES-ECLP-EQN04 Data Validation **Acknowledged**

Code Section

- `contracts/voter/src/entry/execute.rs`

Description

The `try_update_date_config` function in the `voter` contract lacks proper validation for critical parameters such as `genesis_epoch_start_date`, `epoch_length`, and `vote_delay`. While the function allows updating these values, it does not perform any validation checks to ensure that the provided inputs are within acceptable ranges or follow a logical sequence (e.g., future dates, minimum/maximum values for epoch length and vote delay).

Recommendation

It is recommedned to add validation checks to ensure that:

- `genesis_epoch_start_date` is a valid timestamp and represents a future date.
- `epoch_length` is a positive, reasonable value that ensures sufficient time between epochs.
- `vote_delay` is within a sensible range to prevent vote timing manipulation.

Status

The issue was acknowledged by Eclipse team. The development team stated that "These parameters likely will configured single time and we will validate it manually before writing in the contract".

Lack of Two-Step Admin Ownership Accept in single_side_staking contract

Low

RES-ECLP-EQN05 Code Quality **Resolved**

Code Section

- Not Specified

Description

The `single_side_staking` contract lacks a two-step ownership transfer acceptance mechanism. Currently, ownership of the contract can be transferred in a single transaction without requiring acceptance or verification from the new owner. This practice introduces operational risks and potential vulnerabilities.

Recommendation

It is recommeded to implement a two-step ownership transfer mechanism:

- The current owner initiates the ownership transfer, proposing a new owner.
- The new owner must explicitly accept the ownership transfer through a separate transaction within a predefined timeframe.

Status

The issue has been fixed in 046a0f05b95370fade23c200a8c2fe82282c731f.

Presence of unused commented-out code in the codebase

Info **RES-ECLP-EQN06** Code Quality **Resolved**

Code Section

- contracts/single_sided_staking/src/entry/query.rs#L124
- contracts/single_sided_staking/src/config.rs#L29
- contracts/lp_staking/src/entry/query.rs#L73

Description

The codebase contains sections of commented-out code, which are not actively contributing to the functionality of the system. These comments include blocks of code that were likely used during development or testing but were never removed from the final version. While not executed, this code still remains within the repository, serving no functional purpose.

Recommendation

Remove all unnecessary commented-out code to improve code clarity and maintainability. If the commented code serves as a reference or is important for future use, consider moving it to documentation or issue tracking systems. This will keep the codebase clean, easy to maintain, and free from unnecessary clutter.

Status

The issue has been fixed in 3da7be20a338f527a1dc8b3b944855e3097778c2.

Proof of Concepts

No Proof-of-Concept was deemed relevant to describe findings in this engagement.