ECLIPSE FI

SMART CONTRACTS REVIEW

zokyo

December 14th 2023 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed a security audit.

SCORE
90

HYPOTHETICAL SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 0 Critical issue: 0 points deducted
- 1 High issues: 1 resolved = 0 points deducted
- 0 Medium issues: 0 points deducted
- 2 Low issues: = 2 unresolved = -10 points deducted
- 4 Informational issues: 4 resolved = 0 points deducted

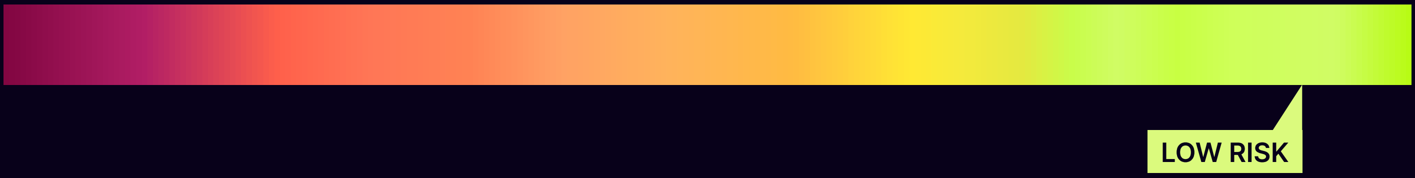
Thus, 100 - 10 = 90

TECHNICAL SUMMARY

This document outlines the overall security of the EclipseFi smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the EclipseFi smart contracts codebase for quality, security, and correctness.

Contract Status



There were 0 critical issues found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contracts but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the NEAR network's fast-paced and rapidly changing environment, we recommend that the EclipseFi team put in place a bug bounty program to encourage further active analysis of the smart contracts.

Table of Contents

Auditing Strategy and Techniques Applied 4

Executive Summary 6

Structure and Organization of the Document 7

Complete Analysis 8

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the EclipseFi repository:
Repo: https://github.com/zokyo-sec/estimation-eclipsepad-cosmwasm-contracts

Last commit - d40c99041db7a6144faae093daf46430681a2d69

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- contracts/presale/src/types.rs
- contracts/presale/src/state.rs
- contracts/presale/src/querier.rs
- contracts/presale/src/msg.rs
- contracts/presale/src/lib.rs
- contracts/presale/src/error.rs
- contracts/presale/src/contract.rs

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most resent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of EclipseFi smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contracts by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo team identified multiple vulnerabilities with different severity levels, encompassing both high and low-risk concerns, along with some informational issues. It's worth noting that the EclipseFi team promptly responded to all the identified issues. For a more detailed breakdown of these findings, please refer to the "Complete Analysis" section.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the EclipseFi team and the EclipseFi team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● **Critical**

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

● **High**

The issue affects the ability of the contract to compile or operate in a significant way.

● **Medium**

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

● **Low**

The issue has minimal impact on the contract’s ability to operate.

● **Informational**

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	DepositPrivateSale could be executed in a Public Sale period	High	Resolved
2	Duplicated functionality	Low	Unresolved
3	Duplicated functionality	Low	Unresolved
4	Unused import	Informational	Resolved
5	Unused import	Informational	Resolved
6	Use of deprecated function	Informational	Resolved
7	Unused variable	Informational	Resolved

HIGH-1 | RESOLVED

DepositPrivateSale could be executed in a Public Sale period

Function: execute_deposit_private_sale
Line: 327
Description:
Function execute_deposit_private_sale has a check that validates that it called not before the private_start_time, but there's nothing that ensures the function is not called after the private sale event

Recommendation:
Add the if-case to check if the current time is past the private sale period.

LOW-1 | UNRESOLVED

Duplicated functionality

Functions: execute_deposit and execute_deposit_private_sale
Description:
Functions execute_deposit and execute_deposit_private_sale have similar functionality of checks and deposit features. This functionality could be combined into one function.

Recommendation:
Combine the deposit functionality into one function.
Comment from a client team: execute_deposit_private_sale is for private sale, and execute_deposit is for the public. While these funcs have similar functionality, we want to split them for more clarity and feature improvements.

LOW-2 | UNRESOLVED

Duplicated functionality

Functions: execute_withdraw_funds and execute_withdraw_unsold_token
Description:
Functions execute_withdraw_funds and execute_withdraw_unsold_token have similar functionality to withdraw tokens by the admin after the sale event. The only difference could be seen is a state.fund_denom vs. state.vesting which could be a parameter to a new function.

Recommendation:
Combine the deposit functionality into one function and call it from those two. Comment from a client team: execute_withdraw_funds, execute_withdraw_unsold_token. these functions also have similar feature but one is for withdrawing fund tokens while another is for withdrawing reward token. Due to the same reason of #2, we will use separate functions.

INFORMATIONAL-1 | RESOLVED

Unused import

Import: cw2::get_contract_version
Line: 9
Description:
The imported function is never used in the code.

Recommendation:
Remove unused import

INFORMATIONAL-2 | RESOLVED

Unused import

Import: semver::Version
Line: 13
Description:
The imported function is never used in the code.

Recommendation:

Remove unused import.

INFORMATIONAL-3 | RESOLVED

Use of deprecated function

Function: cosmwasm_std::to_binary
Lines: 6
Description:
Function cosmwasm_std::to_binary is deprecated

Recommendation:

Use to_json_binary function instead

INFORMATIONAL-4 | RESOLVED

Unused variable

Function: migrate
Variable: deps
Description:
Unused variable in a function.

Recommendation:

Prefix unused argument name with an underscore ("_deps")

	contracts/presale/src/types.rs contracts/presale/src/state.rs contracts/presale/src/querier.rs contracts/presale/src/msg.rs contracts/presale/src/lib.rs contracts/presale/src/error.rs contracts/presale/src/contract.rs
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

We are grateful for the opportunity to work with the EclipseFi team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the EclipseFi team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo