PLEXUS

SMART CONTRACT REVIEW

zokyo

November 24th 2023 | v. 1.0

Security Audit Score

PASS

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

SCORE

99

ZOKYO AUDIT SCORING PLEXUS

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

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:
- 1 Critical issue: 0 points deducted
- 0 High issues: 0 points deducted
- 1 Medium issues: 0 points deducted
- 5 Low issues: = : 0 points deducted
- 6 Informational issues: 5 resolved and 1 unresolved = 1 point deducted

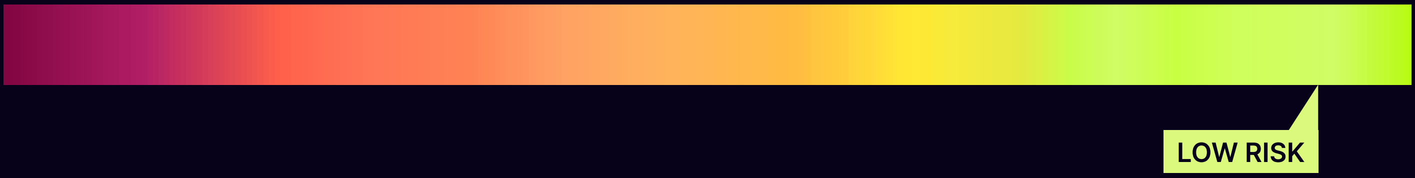
Thus, 100 - 1 = 99

TECHNICAL SUMMARY

This document outlines the overall security of the Plexus smart contract evaluated by the Zokyo Security team.

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

Contract Status



There was 1 critical issue 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 contract but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Plexus team put in place a bug bounty program to encourage further active analysis of the smart contract.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Plexus repository:
Repo: https://github.com/PlexusExchange/PlexusIDO

Last commit - 43d69b3

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

- PlexusIDO.sol

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 Plexus smart contract. 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 contract by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo team identified vulnerabilities of critical, medium, and low severity, along with a few informational issues. For a detailed breakdown of these findings, we recommend consulting 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 Plexus team and the Plexus 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	airdropUpdateLastClaimTime not updated	Critical	Resolved
2	Tautology or contradiction	Medium	Resolved
3	Missing event	Low	Resolved
4	Missing event	Low	Resolved
5	Lack of zero-address check	Low	Resolved
6	Reading storage variable in the loop	Low	Resolved
7	Reading storage variable in the loop	Low	Resolved
8	Boolean equality	Informational	Resolved
9	Boolean equality	Informational	Resolved
10	Cyclomatic complexity	Informational	Unresolved
11	Unused state variable	Informational	Resolved
12	Unused state variable	Informational	Resolved
13	Unused state variable	Informational	Resolved

HIGH-1 | RESOLVED

airdropUpdateLastClaimTime not updated

Contract: PlexusIDO
Function: airdropUserClaimPlx
Details:
The function airdropUserClaimPlx has a check on tine #303 for the airdropUpdateLastClaimTime variable value to be equal to zero. This check will always be true, because the code does never update this variable. Therefore, each time they call, the function will work as the first time, and the correct distribution will not work.

Recommendation:
On lines #303 and #306, use airdropUser.airdropUpdateLastClaimTime instead of airdropUpdateLastClaimTime variable. And then remove the airdropUpdateLastClaimTime declaration from line #48 as obsolete.

MEDIUM-1 | RESOLVED

Tautology or contradiction

Contract: PlexusIDO
Functions: airdropUserClaimPlx, ecosystemClaimPlx, marketingClaimPlx, contributorClaimPlx, advisorClaimPlx, teamClaimPlx
Details:
In all functions above, there is an exact line of code: "if (claimableAmount < 0) revert EmptyTokenClaim();". In all cases variables are declared as: "uint256 claimableAmount". While the Unsigned Integer variable could not be less than zero ever, those checks would never be true, therefore, they are useless.

Recommendation:
Revise the logic here. Maybe you need to check the variable to be equal to zero?

LOW-1 | RESOLVED

Missing event

Contract: PlexusIDO
Function: setOwner
Details:
Functions that drastically change the state should emit events

Recommendation:
Emit an event on minter address change

LOW-2 | RESOLVED

Missing event

Contract: PlexusIDO
Function: setTGEtime
Details:
Functions that drastically change the state should emit events

Recommendation:
Emit an event on minter address change

LOW-3 | RESOLVED

Lack of zero-address check

Contract: PlexusIDO
Function: setOwner
Details:
The function sets the owner of the contract, however, it doesn't check the provided address for a new owner to be a zero address. Once a zero address is provided, either intentionally or accidentally, the owner functions will never be working again.

Recommendation:
Add an address verification

LOW-4 | RESOLVED

Reading storage variable in the loop

Contract: PlexusIDO
Function: emergencyWithdraw
Details:
The for-loop is reading the state variable "signerList.length" in the loop while it never changes, which burns gas.

Recommendation:
Store the variable value to a local variable before the for-loop and use it the loop instead.

LOW-5 | RESOLVED

Reading storage variable in the loop

Contract: PlexusIDO
Function: signerConfirm
Details:
The for-loop is reading the state variable "signerList.length" in the loop while it never changes, which burns gas.

Recommendation:
Store the variable value to a local variable before the for-loop and use it the loop instead.

INFORMATIONAL-1 | RESOLVED

Boolean equality

Contract: PlexusIDO
Function: pauseOnOff
Details:
The function contains the comparison to boolean constants on line #98. Boolean constants can be used directly and do not need to be compared to true or false.

Recommendation:
Remove the equality to the boolean constant.

INFORMATIONAL-2 | RESOLVED

Boolean equality

Contract: PlexusIDO
Function: signerConfirm
Details:
The function contains the comparison to boolean constants on line #476. Boolean constants can be used directly and do not need to be compared to true or false.

Recommendation:

Remove the equality to the boolean constant.

INFORMATIONAL-3 | UNRESOLVED

Cyclomatic complexity

Contract: PlexusIDO
Functions: seedClaimPlx, kolsClaimPlx
Details:
The functions above have cyclomatic complexity 12.

Recommendation:

Reduce cyclomatic complexity by splitting the function into several smaller subroutines.

INFORMATIONAL-4 | RESOLVED

Unused state variable

Contract: PlexusIDO
Variable: airdropUserLastClaimTime
Details:
Variable is not used anywhere in the code

Recommendation:

Remove unused state variables

INFORMATIONAL-5 | RESOLVED

Unused state variable

Contract: PlexusIDO
Variable: liquidityClaimCount
Details:
Variable is not used anywhere in the code

Recommendation:

Remove unused state variables

INFORMATIONAL-6 | RESOLVED

Unused state variable

Contract: PlexusIDO
Variable: liquidityTotalClaimed
Details:
Variable is not used anywhere in the code

Recommendation:

Remove unused state variables

	PlexusIDO.sol
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 Plexus 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 Plexus team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo