Coinsult

Request your audit at coinsult.net

Advanced Manual

Smart Contract Audit

May 22, 2023

Audit requested by

Noahsark

Audit Summary

Project Name	Noahsark
Website	https://www.noahsarkworth.com/
Blockchain	Polygon
Smart Contract Language	Solidity
Audit Method	Static Analysis, Manual Review
Start date of audit	May 22, 2023

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

The audit of the Solidity codebase presented significant challenges due to its disorganized nature and lack of accompanying documentation. The convoluted structure of the code made it increasingly difficult to discern the intended functionality of individual components. Furthermore, the absence of any guiding documentation surrounding the functions made it even more arduous to review and assess the overall quality of the code. Despite these obstacles, the auditing process was diligently carried out to ensure the highest standards of security and stability.

This code base looks to be a potential ponzi-scheme type referral contract. Please DYOR before investing.

Audit Scope

Source Code

Coinsult was comissioned to do a manual code review by Noahsark based on the following code:

https://mumbai.polygonscan.com/address/0xcFA1D3d58E9746bB7Ae49e06463bd4701438C608#code

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Audit Method

Coinsult's manual smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to fix them.

Automated Vulnerability Check

Coinsult uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review

Coinsult's manual code review involves a human looking at source code, line by line, to find vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer's intentions and general business logic into consideration.

Used Tools

- ✓ Slither: Solidity static analysis framework
- ✓ Remix: IDE Developer Tool
- ✓ CWE: Common Weakness Enumeration
- ✓ SWC: Smart Contract Weakness Classification and Test Cases
- ✓ DEX: Testnet Blockchains

Audit Findings

Avatar is a fork of an established and well-functioning smart contract with over 4 million worth of **Matic**. Given the proven reliability and performance of this contract, our audit was primarily focused on the **differences** between the two contracts. By doing so, we aimed to ensure that the modifications introduced in Avatar maintain the same level of security and efficiency as the original contract.

Summary of Findings:

BatchSetReferrerInfo function:

used by admin to set refferers Findings and suggestions:

- **Centralization :** refferer can be set to the user[i] address, this means an address is refferer of itself
- **Centralization** : this function can be used by the owner to give an arbitrary sale level the a refferer
- **Logical** : NewReferrer event is not emitted from this function

Logical – batchClosePositions is not currently protected by **notContract onePosition function :**

this function is used to open a new position in current epoch Findings and suggestions:

1. Logical – lack of validation for **MIN_INVEST** when trying to open a new position suggestion : this opens up the possibility to create a position with 0 ethereum amount.
2. sum of invest – principal – referref -incentivise amounts is not equal to 1,000,000 suggestion: make sure that sum of rates is equal to 1,000,000 (100%)
3. Typo – change “Invalid referrer1” (require error message) to “Invalid referrer”
4. dust ETH is not sent back to the investor in this function

suggestion: calculations in solidity are rounded, hence some ethereum might be stuck in the contract.

_getSalesToLevel function:

used to return a new level based on the sale amount of a user:

Findings and suggestions:

Informational – “**ether**” keyword is used for returning a level based on the sale amount. Since “**ether**” keyword scales the number by 10 ^ 18 this amounts are representing huge very amounts of ethereum that investors might not reach (considerring ethereum mainnet)

Bucket.sol

_getReturnRate:

used to get a return raturn rate after opening a new position based on the daily volume

Findings and suggestions:

Logical - there is a **DAILY_GROWTH** variable set to 20,000 (2%) which is used to determine a target volume for the next day. However the target volume for next day is set to 2% of the last day not 102% (2% more)

Suggestion : change DAILY_GROWTH to 1_020_000

Logical – dailyTradeVolume is never updated in the Avatar.sol contract after opening a new position.

Suggestion: make sure to update dailyTradeVolume by msg.value in the onePosition function

General

Centralization – first refferers can only be set by owner (tempAdmin) after. It means the admin has significant control over the referral structure and may potentially profit from a larger portion of the investments. This can create an uneven distribution of rewards and opportunities among users, leading to a less fair and more centralized system.

Centralization – first refferers can only be set by admin by an arbitrary sale level, this sale level can be set to max level. This creates and unfair distribution of wealth in the system

Optimization – **setPause** is never used in the contract, gamePaused variable is not used anywhere in the code

Logical – lack of escape hatch mechanism : there is currently no way for investors to emergency withdraw their invested ETH.

Logical – lack of withdraw function for ERC20 tokens: there is currently no way for owner of the contract to be able to withdraw ERC20 tokens sent to contract by mistake.

Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

The audit of the Solidity codebase presented significant challenges due to its disorganized nature and lack of accompanying documentation. The convoluted structure of the code made it increasingly difficult to discern the intended functionality of individual components. Furthermore, the absence of any guiding documentation surrounding the functions made it even more arduous to review and assess the overall quality of the code. Despite these obstacles, the auditing process was diligently carried out to ensure the highest standards of security and stability.

This code base looks to be a potential ponzi-scheme type referral contract. Please DYOR before investing.