Coinsult

Request your audit at **coinsult.net**

Advanced Manual

Smart Contract Audit

November 11, 2025

x.com/coinsultaudits

t.me/coinsult_tg

Audit requested by

QuasarChain | PaymentReceiver / PresaleClaim

Global Overview

Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
Informational	1	0	1	0
Low-Risk	1	0	1	0
Medium-Risk	0	0	0	0
Critical-Risk	0	0	0	0

Risk Classification

Coinsult uses certain vulnerability levels, these indicate how bad a certain issue is. The higher the risk, the more strictly it is recommended to correct the error before using the contract.

Vulnerability Level	Description
Informational	Does not compromise the functionality of the contract in any way
Low-Risk	Won't cause any problems, but can be adjusted for improvement
Medium-Risk	Will likely cause problems and it is recommended to adjust
Critical-Risk	Will definitely cause problems, this needs to be adjusted

Audit Summary

Project	
Website	https://quasarchain.io
Blockchain	Ethereum
Source Code	https://etherscan.io/address/0xe380bfd3af966b464b68e045fa11045e0581924c#code
Contract Address	https://etherscan.io/address/0xe380bfd3af966b464b68e045fa11045e0581924c#code

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.

Audit Scope

Coinsult was commissioned to perform an audit based on the provided 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

Table of Contents

Payment Receiver Token Distribution

Informational *- Does not compromise the functionality of the contract in any way*

Error Code	Description
NO-TOKENS	Payment receiver receives payments but never distributes tokens.

Code Snippet

```
function payWithToken(address token, uint256 amount)
    external
    nonReentrant
    whenNotPaused
{
    require(amount > 0, "Zero payment");
    require(supportedTokens[token], "Token not supported");

    uint8 decimals = tokenDecimals[token];
    totalTokenReceived[token] += amount;

    IERC20(token).safeTransferFrom(msg.sender, address(this), amount);

    emit PaymentReceived(msg.sender, token, amount, decimals, block.timestamp);
}
```

Note by QuasarChain

"All payments will be managed on the backend and stored in the database. Once the presale ends, we will use a Merkle Tree to allocate tokens to investors."

Merkle is double hashed

Low-Risk *- Won't cause any problems, but can be adjusted for improvement*

Error Code	Description
MERK-HASH	This is effectively keccak(keccak(abi.encode(...))). Not wrong, but most tooling assumes keccak256(abi.encodePacked(account, amount)). Impact: If your off-chain tree builder doesn't mirror this exactly, every proof will fail.

Code Snippet

```
bytes32 leaf = keccak256(
    bytes.concat(keccak256(abi.encode(msg.sender, totalAllocation)))
);
```

Recommendation

If you keep the double-hash, document the exact off-chain leaf computation.

Centralization

The owner controls setting the token address, Merkle root, and TGE time, can withdraw any tokens via emergencyWithdraw, generates the Merkle data off-chain, funds (or withholds) QC7 tokens, and can transfer or renounce ownership.

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.