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

SonamiLayer2

-

Global Overview

Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
Informational	0	0	0	0
Low-Risk	0	0	0	0
Medium-Risk	4	4	0	0
Critical-Risk	1	1	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://sonami-so.io
Blockchain	SOL
Source Code	-
Contract Address	-

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

Claim before sale ends

Medium-Risk *- Will likely cause problems and it is recommended to adjust*

Error Code	Description
CLAIM-END	Claims can be enabled while the sale is still ongoing because there's no enforcement that claim_start_ts >= end_ts.

Code Snippet

```
pub fn set_claim_start(ctx: Context<AdminOnly>, claim_start_ts: i64) -> Result<()>
{
   let state = &mut ctx.accounts.presale_state;
   require!(ctx.accounts.admin.key() == state.admin, ErrorCode::NotAdmin);
   require!(claim_start_ts > 0, ErrorCode::InvalidClaimStart);
   // Optionally: require!(claim_start_ts >= state.end_ts,
ErrorCode::ClaimTooEarlyConfig);
   state.claim_start_ts = claim_start_ts;
   Ok(())
}
```

Recommendation

Require claim_start_ts >= end_ts to ensure claims start only after the sale period ends.

Zero-amount or rounding-to-zero purchases

Medium-Risk *- Will likely cause problems and it is recommended to adjust*

Error Code	Description
ZERO-AMT	If a user pays a very small amount, quote_to_tokens may return 0 due to integer division rounding, but the SOL/USDC transfer still happens.

Code Snippet

```
let tokens_out = quote_to_tokens(
    lamports as u128,
    state.sol_price as u128,
    state.token_decimals,
)?;
```

Recommendation

Require amount > 0 and tokens_out > 0 before updating allocations to prevent “paid-but-zero” cases.

Mint authority drift not checked during claim

Medium-Risk *- Will likely cause problems and it is recommended to adjust*

Error Code	Description
MINT-ATH	claim_all assumes mint_auth is still the mint authority. If someone later changes it (e.g. via admin mistake), mint_to could fail or behave incorrectly.

Code Snippet

```
token::mint_to(
    CpiContext::new_with_signer(
        ctx.accounts.token_program.to_account_info(),
        MintTo {
            mint: ctx.accounts.token_mint.to_account_info(),
            to:   ctx.accounts.buyer_token_account.to_account_info(),
            authority: ctx.accounts.mint_auth.to_account_info(),
        },
        signer,
    ),
    remaining,
)?;
```

Recommendation

Add a require! that confirms token_mint.mint_authority == Some(mint_auth) before minting.

Freeze authority

Critical-Risk *- Will definitely cause problems, this needs to be adjusted*

Error Code	Description
FREEZE	You only transfer MintTokens authority to the PDA. If the mint's freeze_authority remains with the admin, they could freeze user ATAs post-claim.

Code Snippet

```
// initialize_presale: also move FreezeAccount authority
token::set_authority(
    CpiContext::new(
        ctx.accounts.token_program.to_account_info(),
        SetAuthority {
            account_or_mint: ctx.accounts.token_mint.to_account_info(),
            current_authority: ctx.accounts.admin.to_account_info(),
        },
    ),
    AuthorityType::FreezeAccount,
    Some(ctx.accounts.mint_auth.key()), // or None to disable freezing
)?;
```

Recommendation

In initialize_presale, also call set_authority(..., AuthorityType::FreezeAccount, Some(mint_auth) or None).

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.