SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Wusle

AUDIT

SECURITY ASSESSMENT

11 Febuary, 2025

for

WUSLE

CONTENTS

Introduction

SolidProof.io is a brand of the officially registered company FutureVisions Deutschland, based in Germany. We're mainly focused on Block-chain Security such as Smart Contract Audits and KYC verification for project teams. Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, Pancake-Swap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	ShibaDino
Website	https://www.wusle.com
About the Project	N/A
Chain	Solana
Language	Rust
Codebase	as File
Commit	N/A
Unit Tests	N/A

Social Medias

Telegram	N/A
Twitter	https://x.com/wusle_official
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
YouTube	N/A
TikTok	N/A
LinkedIn	N/A
CoinMarketCap	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	9 Febuary, 2025	• Layout Project • Automated/Manual-Security Review • Summary
v1.1	11 Febuary, 2024	• Re-Audit

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project. This analysis did not include functional testing (or unit testing) of the contract's logic. We cannot guarantee 100% logical correctness of the contract as it was not functionally tested by us.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with a SHA-1 Hash.

1. lib.rs (376e0ef07eb4000b846b716f64522ce369302028)

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

1. Packagename

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 - 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 - 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 - 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 - 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk.

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security- related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered. We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodolgy

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
    b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
    c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.
2. Testing and automated analysis that includes the following:
    a. Test coverage analysis, which determines whether test cases actually cover code and how much code is executed when those test cases are executed.
    b. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.
4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Medium or higher issues

No critical issues found	✅ Contract is safe to deploy
Description	The contract does not contain issues of high or medium criticality. This means that no known vulnerabilities were found in the source code.
Comment	N/A

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project there are authorities that has the authority over the following functions:

File/Role	Privileges
Main {Owner}	None

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart- contract-based accounts, such as multi-signature wallets.

Here are some suggestions what the client can do.

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness on privileged operations
- Introduce a DAO/Governance/Voting module to increase transperancy and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Results

Critical issues

No critical issues

High issues

#1 | Wrong check for InvalidClaimAmount

File	Severity	Location	Status
Main	high	L314-317	fixed

Description - User_account claimed tokens are set as claimable amount after the claim is successful. For the next time total_tokens are set again by the new buy amount. This claim function will not work correctly after first claim.

```
pub fn claim(ctx: Context<Claim>) -> Result<()> {
    if ctx.accounts.user_account.claimed_tokens >= ctx.accounts.user_account.total_tokens {
        return Err(error!(ErrorCode::InvalidClaimAmount));
    }
```

#2 | Calculation issue with decimal comparsion

File	Severity	Location	Status
Main	high	L97-98	fixed

Description - sold_tokens and tokens divided by decimals cannot be compared with allocation(value in smallest unit) in buy function

```
if
    (ctx.accounts.presale_account.stages[index as usize].sold_tokens + tokens) /
        (10u128).pow(decimals as u32) > allocation
{
    return Err(error!(ErrorCode::AllocationReached));
}
```

#3 | Calculation issue with decimal comparsion

File	Severity	Location	Status
Main	high	L214-215	fixed

Description - sold_tokens and tokens divided by decimals cannot be compared with allocation(value in smallest unit) in buy function

```
if
    (ctx.accounts.presale_account.stages[index as usize].sold_tokens + tokens) /
        (10u128).pow(decimals as u32) > allocation
{
    return Err(error!(ErrorCode::AllocationReached));
}
```

Medium issues

#1 | Presale should not be deleted when it is live

File	Severity	Location	Status
Main	medium	L294-300	fixed

Description - Implement checks for delete presale before start time or after it has ended.

```
pub fn delete_presale(ctx: Context<DeletePresale>) -> Result<()> {
    if ctx.accounts.admin.key.ne(&ctx.accounts.presale_account.admin) {
        return Err(error!(ErrorCode::InvalidOwner));
    }
    ctx.accounts.presale_account.close(ctx.accounts.admin.to_account_info())?;
    Ok(())
}
```

Low issues

#1 | Missing token amout validation

File	Severity	Location	Status
Main	low	L49-53	fixed

Description - Implement a validation check on token amount and decimals in transfer_token function

```
pub fn transfer_token(ctx: Context<TransferTokens>, tokens: u128, decimals: u8) -> Result<()> {
    let final_tokens = tokens * (10u128).pow(decimals as u32);
    token::transfer(ctx.accounts.transfer_token(), final_tokens as u64)?;
    Ok(())
}
```

#2 | Changable active stage

File	Severity	Location	Status
Main	low	L301-313	ack

Description - Active stage should not be modified after the presale starts

```
pub fn change_stage(
    ctx: Context<ChangeStage>,
    active_stage: u8,
    stage_end_time: u64
) -> Result<()> {
    let presale_account = &mut ctx.accounts.presale_account;
    if ctx.accounts.admin.key.ne(&presale_account.admin) {
        return Err(error!(ErrorCode::InvalidOwner));
    }
    presale_account.active_stage = active_stage;
    presale_account.stage_end_time = stage_end_time;
    Ok(())
}
```

Informational issues

#1 | Redundant checks

File	Severity	Location	Status
Main	informational	L66-71	fixed

Description - Redundant check in buy function

```
if ctx.accounts.owner.key.ne(&ctx.accounts.presale_account.owner) {
    return Err(error!(ErrorCode::InvalidOwner));
}
if ctx.accounts.owner.key.ne(&ctx.accounts.presale_account.owner) {
    return Err(error!(ErrorCode::InvalidOwner));
}
```

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY