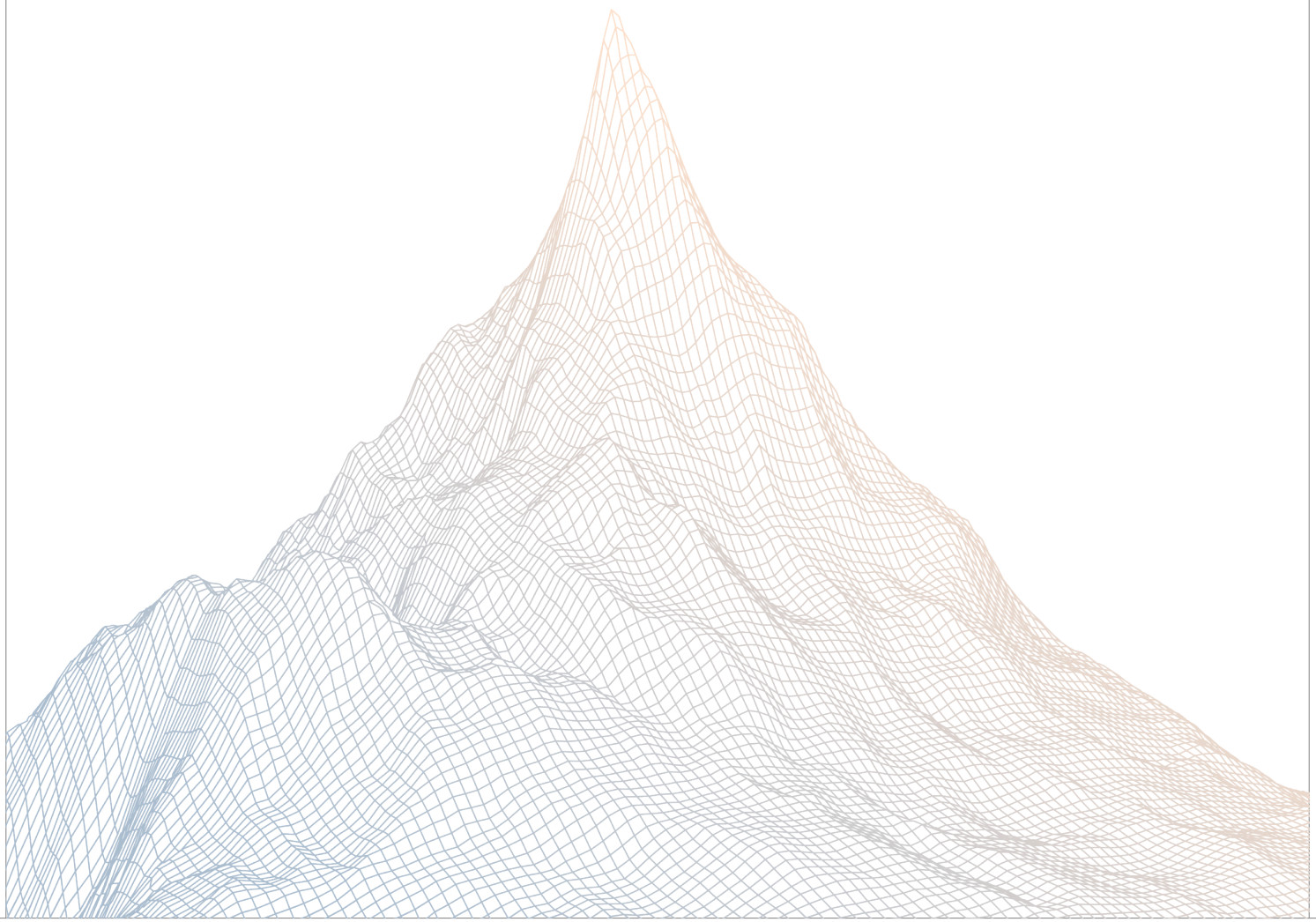
Zenith

Legion Solana

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: March 31st to April 7th, 2025

AUDITED BY: Jakub Heba

Contents

1 Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2

Executive Summary

2.1 About Legion

The goal of Legion is to create a network where anyone can freely chat and socialize without compromising their privacy, using the hashgraph consensus.

2.2 Scope

The engagement involved a review of the following targets:

Target	solana-contracts
Repository	https://github.com/Legion-Team/solana-contracts
Commit Hash	`36603af3290837c08db8cc7948a221d05d90083e`
Files	`refactor(anchor): allow 10 decimal points for claiming`

Target	solana-contracts
Repository	https://github.com/Legion-Team/solana-contracts
Commit Hash	`197b937fb9001d13aa13f1a4d04be2cb28d90c83`
Files	`Update user accepted deposit to be performed by the user with signature`

Target	solana-contracts
Repository	https://github.com/Legion-Team/solana-contracts
Commit Hash	`26c74e797c0a5e035ba8c806e4615d7f59da3f89`
Files	`Fix to change the user validation on accepted deposit`

Target	solana-contracts
Repository	https://github.com/Legion-Team/solana-contracts
Commit Hash	`8a7b2e5f9c0d4e3a1b6f8d7c9e0a2b4d5c6f8a9b`
Files	`Added Sale distribution`

2.3 Audit Timeline

March 31, 2025	Audit start
April 1, 2025	Audit end
April 7, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	2
Informational	0
Total Issues	**2**

3

Findings Summary

ID	Description	Status
L-1	Sequenced version may prevent messages processing	Acknowledged
L-2	The find_ed25519_instruction limits protocol's functionality	Acknowledged

4

Findings

4.1 Low Risk

A total of 2 low risk findings were identified.

[L-1] Sequenced `version` may prevent messages processing

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- update_user_account_accepted_deposit_and_claim_percentage.rs

Description:

In current system `version` works as a nonce:

```
pub fn _update_user_account_accepted_deposit_and_claim_percentage(
    ctx: Context<UpdateUserAccountAcceptedDepositAndClaimPercentage>,
    accepted_deposit_amount: u64,
    claim_total_supply_percentage: u64,
    version: u8,
    signature: [u8; 64],
) -> Result<()> {
    // Emit UserAccountUpdated event
    emit!(UserAccountAcceptedDepositAndClaimPercentageUpdated {
        accepted_deposit_amount,
        claim_total_supply_percentage,
    });

[..]

    // Validate that the version is greater than current version
    //  avoids replay attacks
    require!(
        version > ctx.accounts.user_account.version,
        TokenSaleError::UserAccountVersionMustBeGreaterThanCurrentVersion
    );
```

It's in the type of `u8`, so maximum is `255`. It means that only 255 messages can be processed for specific `user_account`. That limit should be keep in mind, as it might be problematic, when more than 255 messages is expected to be processed.

What is more, there is assumption that version gradually increases (`version > ctx.accounts.user_account.version`), however, message signer can control version parametr, meaning that 1st message can have version of 255 (as a result of backend mistake), then no more messages can be processed by the program for this `user_account`.

Recommendations:

We recommend considering if limit of 255 messages per user is acceptable and rethinking if version should not be provided in sequence, to avoid unexpected limitations.

Legion: Acknowledged. Likely we would have a maximum of 5 updates per user account, so it's good to leave only 255 for now.

[L-2] The `find_ed25519_instruction` limits protocol's functionality

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- ed25519.rs

Description:

With the `find_ed25519_instruction` taking 1st instruction that addressing ED25519 program ID, there is a limitation that only deposit action (one message) can be done in transaction.

There is common pattern to take `current_ix - 1` as a signature instruction. This approach would allow to squeeze two deposit messages in one transcation (Solana has limitation of 5 instructions per transaction)

```
pub fn find_ed25519_instruction(ix_sysvar: &AccountInfo) →
    Result<Instruction> {
    let num_instructions = ix_sysvar.data_len();

    for i in 0..num_instructions {
        if let Ok(ix) = load_instruction_at_checked(i, ix_sysvar) {
            if ix.program_id == ED25519_ID {
                // Return the Ed25519Program instruction
                return Ok(ix);
            }
        }
    }
```

Recommendations:

We recommend analyzing whether the current implementation of `find_ed25519_instruction` is not too strict and adapting the code to a commonly used pattern.

Legion: Acknowledged. It is a good point, but the methods that use this signature, are

triggered individually by each user, so currently there is no scenario that requires multiple signings.