0xCommit

Security Audit Report

SOEX - Solana Programs

Treasury , OG and Premium CVT Programs

Version: Final

Date: 2nd November 2024

Table of Contents

License

Disclaimer

Introduction

Purpose of this report

0xCommit has been engaged by **SOEX - Solana Programs** to perform a security audit of several Solana Programs components.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine solana program bugs, which might lead to unexpected behaviour.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebases Submitted for the Audit

The audit has been performed on the following GitHub repositories:

Version	List of programs	Source
1	launchpad , hvt_launchpad and staking_HVT	https://gitlab.dcircle.io/dcchain/soex_solana_x/-/tree/master/programs

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending**, **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Overview

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Summary of Findings

Sr. No.	Program	Description	Severity	Status
1	Launchpad, hvt_launchpad and staking_hvt	Ancillary Initialize function can be called multiple times and are missing access control	Critical	Resolv...
2	Launchpad	Incorrect access control checks	High	Resolv...
3	hvt_launchpad	Incomplete access control checks	Medium	Resolv...
4	hvt_launchpad	Slipage not factored during swap	Medium	Resolv...

Detailed Findings

1. Initialize functions can be called multiple times

Severity: High **Program - Launchpad , HVT_Launchpad**

Description

Following function in the respective contracts can be called multiple times by any one

Contract name	Function Name
Launchpad	init_release_config, init_sol_acct , init_token_acct and init_lp_acct
HVT_launchpad	init_global_account and init_lp_account
Staking_hvt	init_soex_acct (has Access Control in place)

Following controls are missing in these place

- Initialize function can be only called once
- Since these function are not linked to main initialize function they should be called only by admin

Remediation

Ensure all initialize functions can be called once only and can only be initialized by authorized wallet.

Status

Resolved

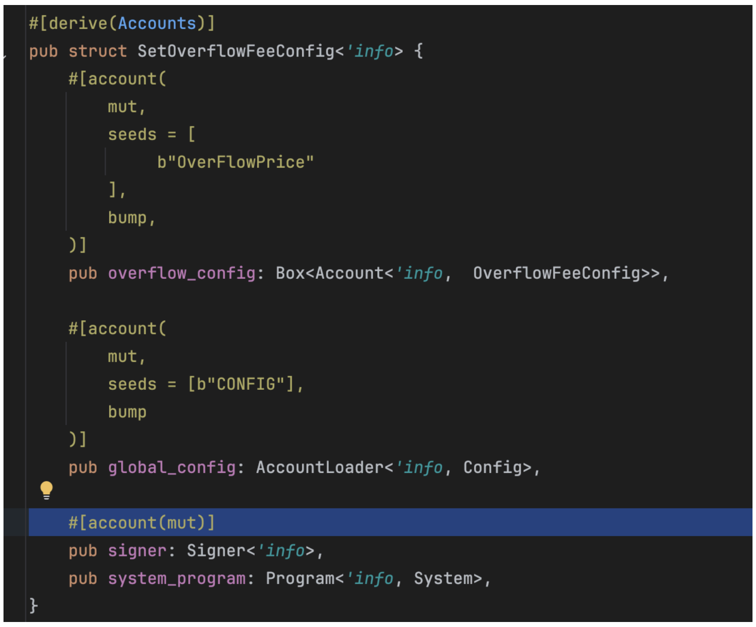
2. Incorrect access control checks

Severity: High ▾

Program - Launchpad

Description

In SetOverFlowFeeConfig struct the signer is not enforced to be admin wallet but the check is kept in function which is not correct way to process access control in anchor



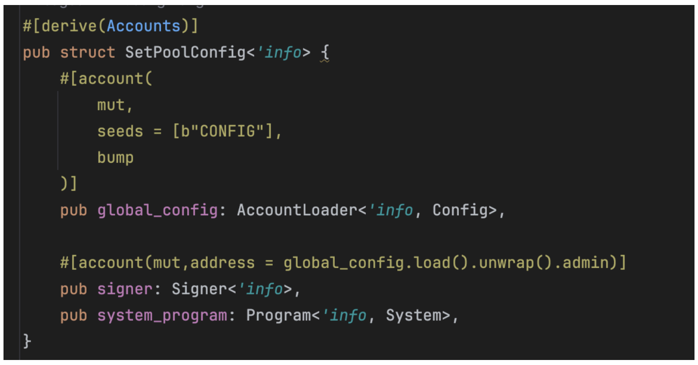
```
#[derive(Accounts)]
pub struct SetOverflowFeeConfig<'info> {
    #[account(
        mut,
        seeds = [
             b"OverFlowPrice"
        ],
        bump,
    )]
    pub overflow_config: Box<Account<'info,  OverflowFeeConfig>>,

    #[account(
        mut,
        seeds = [b"CONFIG"],
        bump
    )]
    pub global_config: AccountLoader<'info, Config>,

    #[account(mut)]
    pub signer: Signer<'info>,
    pub system_program: Program<'info, System>,
}
```

Remediation

Ensure correct access control mechanism is in place which is alignment with anchor, Refer to below mentioned system uniformly across program



```
#[derive(Accounts)]
pub struct SetPoolConfig<'info> {
    #[account(
        mut,
        seeds = [b"CONFIG"],
        bump
    )]
    pub global_config: AccountLoader<'info, Config>,

    #[account(mut,address = global_config.load().unwrap().admin)]
    pub signer: Signer<'info>,
    pub system_program: Program<'info, System>,
}
```

Status

Resolved ▾

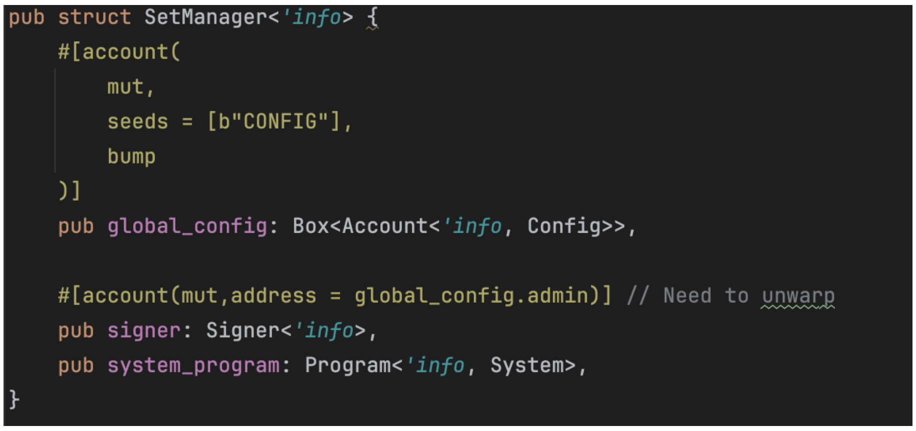
3. Incomplete access control checks.

Severity: Medium

Program - HVT_launchpad

Description

In SetManager struct the signer is not correctly mapped to be admin wallet which is not in alignment with anchor specification.

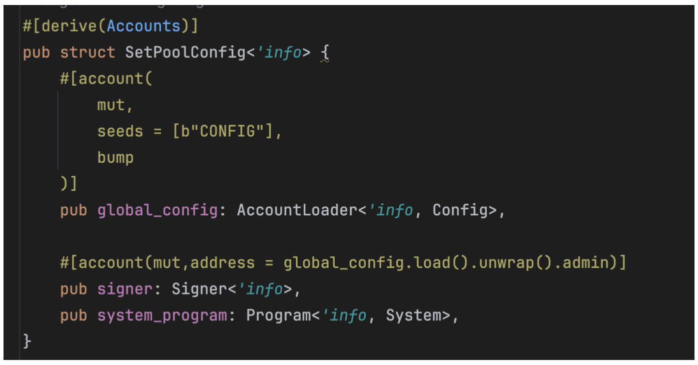


```
pub struct SetManager<'info> {
    #[account(
        mut,
        seeds = [b"CONFIG"],
        bump
    )]
    pub global_config: Box<Account<'info, Config>>,

    #[account(mut,address = global_config.admin)] // Need to unwarp
    pub signer: Signer<'info>,
    pub system_program: Program<'info, System>,
}
```

Remediation

Ensure correct access control mechanism is in place which is alignment with anchor, Refer to below mentioned system uniformly across program where load and unwrap of admin is done in address.



```
#[derive(Accounts)]
pub struct SetPoolConfig<'info> {
    #[account(
        mut,
        seeds = [b"CONFIG"],
        bump
    )]
    pub global_config: AccountLoader<'info, Config>,

    #[account(mut,address = global_config.load().unwrap().admin)]
    pub signer: Signer<'info>,
    pub system_program: Program<'info, System>,
}
```

Status

Resolved

4. Splipage not factored during swaps.

Severity: Medium **Program - hvt_launchpag**

Description

The swap function in handle_deposit does not take into account slippage leading to unforeseen losses for the deposit function

Remediation

Ensure slippage is accounted for during the swap operation and if the transfer value is beyond the accepted threshold then revert the transaction.

Status

Resolved