

Secure Code Review

Findings and Recommendations Report Presented to:

Solcial

September 15, 2023
Version: 1.0

Presented by:

Kudelski Security, Inc.
5090 North 40th Street, Suite 450
Phoenix, Arizona 85018

FOR PUBLIC RELEASE

TABLE OF CONTENTS

TABLE OF CONTENTS	2
LIST OF FIGURES.....	2
LIST OF TABLES.....	2
EXECUTIVE SUMMARY	3
Overview	3
Key Findings.....	3
Scope and Rules Of Engagement	4
TECHNICAL ANALYSIS & FINDINGS	5
Findings.....	6
KS-01 Lack of <code>token_program id</code> validation	7
METHODOLOGY	9
Tools	9
Vulnerability Scoring Systems.....	10
KUDELSKI SECURITY CONTACTS	Error! Bookmark not defined.

LIST OF FIGURES

Figure 1: Findings by Severity	5
--------------------------------------	---

LIST OF TABLES

Table 1: Scope	4
Table : Findings Overview	6

EXECUTIVE SUMMARY

Overview

Solcial engaged Kudelski Security to perform a code review of the token-swap program. The assessment was conducted remotely by the Kudelski Security Team. Testing took place between August 22nd 2023 and September 15th 2023, and it was focused on the following objectives:

- Provide the customer with an assessment of the overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the result of our tests.

During the Secure Code Review, Kudelski Security identified **1 Informational** finding according to our Vulnerability Scoring System. This report summarizes the engagement, tests performed, and details of the mentioned findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Teams took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following are the major themes and issues identified during the testing period.

These, along with other items, within the findings section, should be prioritized for remediation to reduce the risk they pose.

- The Kudelski Security team focused this research on reviewing the implementation of mathematics, especially the proper use of decimals and rounding errors. After a thorough review, we verified the correct implementation of the mathematics of this program and did not discover any vulnerabilities.
- The logic implemented in token-swap for automated market makers has been used by many other protocols. All critical points were thoroughly reviewed.
- The token-swap program is part of the Solana program library which has undergone a previous security audit. Consequently, it is not unexpected that this review resulted with only 1 informational finding.

Scope and Rules Of Engagement

Kudelski performed a Secure Code Review on the token-swap program for Solcial. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

Commit Hash
c3430a37df7f38aff3fabf0817dd70376fd289e0
In-Scope Repositories
solcial-solana-client

Table 1: Scope

TECHNICAL ANALYSIS & FINDINGS

During the Secure Code Review, we identified **1 Informational** finding according to our Vulnerability Scoring System.

The following chart displays the findings by severity.

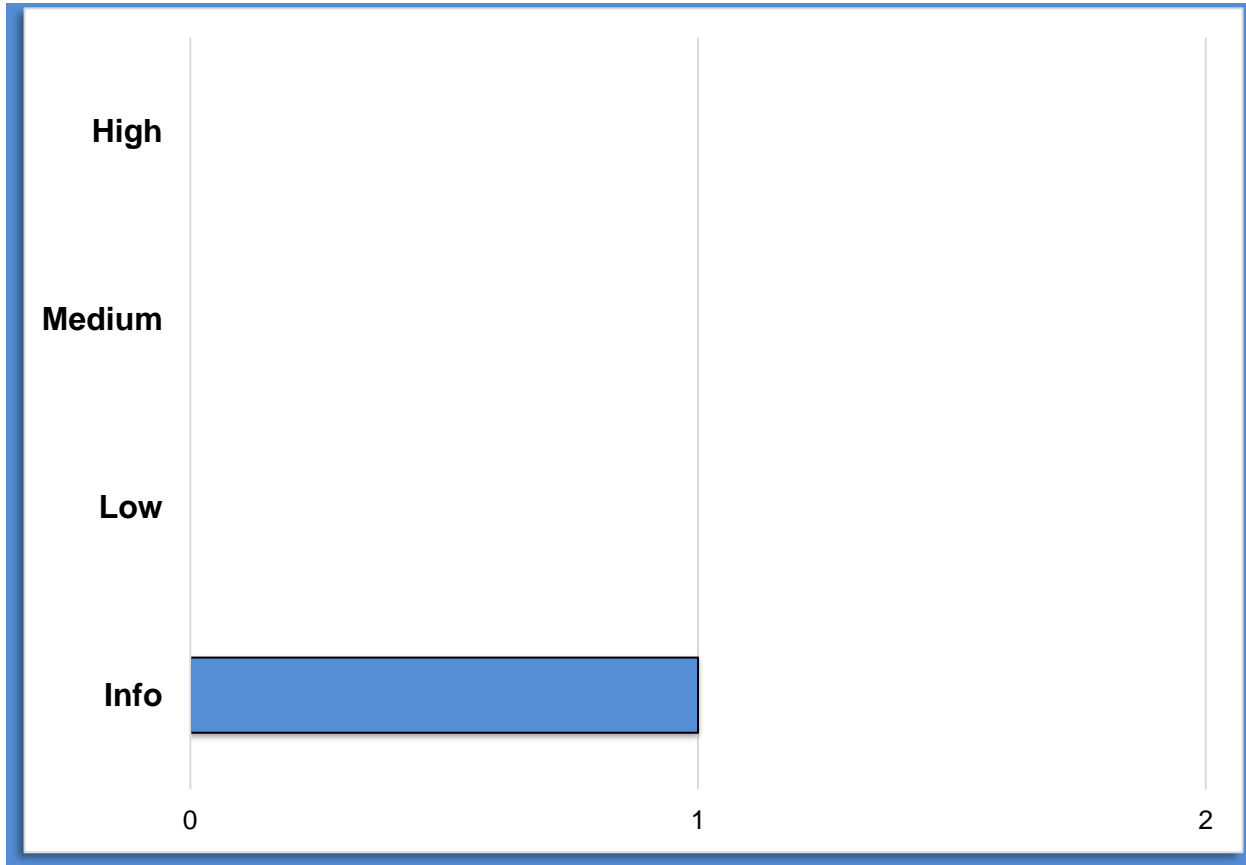


Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description
KS-01	Informational	Lack of <code>token_program id</code> validation

Table 2: Findings Overview

KS-01 Lack of token_program id validation

Severity

Informational

Description

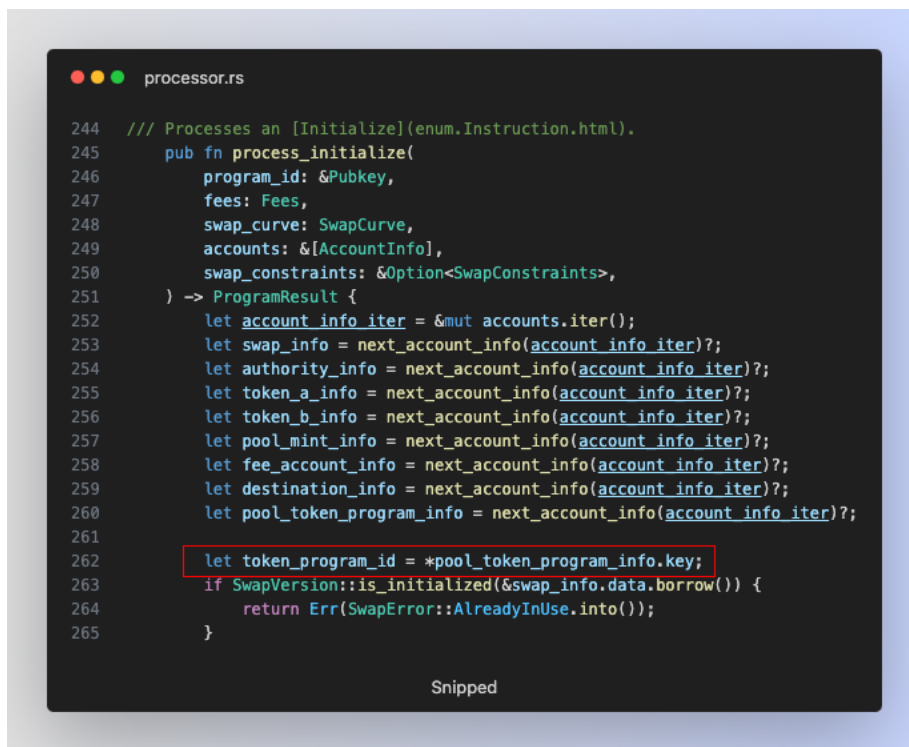
The Kudelski Security team identified a program id that needs to be verified.

In the `process_initialize` instruction, the `token_program` id is not verified to be equal to `spl_token_2022::id()`.

As this id is expected to be provided on the front end, this finding is considered informational since the impact is that a user calling this instruction with an incorrect id is not going to be allowed to call the rest of the functions.

This is because the other instructions use `transfer_checked`, which verifies that the `token_program` id is `spl_token_2022::id()`.

Evidence



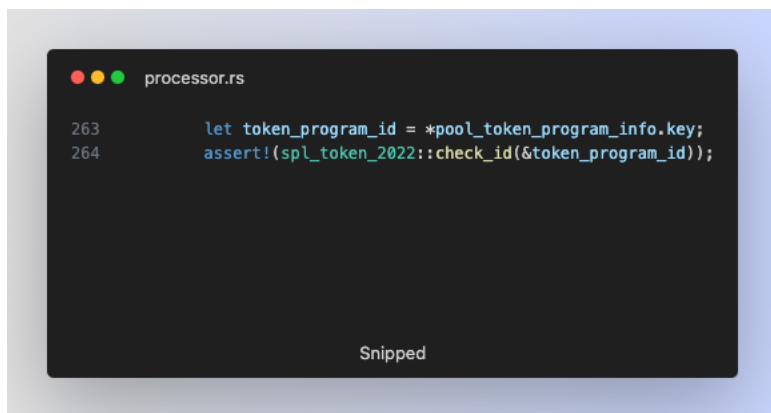
```
processor.rs
244  /// Processes an [Initialize](enum.Instruction.html).
245  pub fn process_initialize(
246      program_id: &Pubkey,
247      fees: Fees,
248      swap_curve: SwapCurve,
249      accounts: &[AccountInfo],
250      swap_constraints: &Option<SwapConstraints>,
251  ) -> ProgramResult {
252      let account_info_iter = &mut accounts.iter();
253      let swap_info = next_account_info(account_info_iter)?;
254      let authority_info = next_account_info(account_info_iter)?;
255      let token_a_info = next_account_info(account_info_iter)?;
256      let token_b_info = next_account_info(account_info_iter)?;
257      let pool_mint_info = next_account_info(account_info_iter)?;
258      let fee_account_info = next_account_info(account_info_iter)?;
259      let destination_info = next_account_info(account_info_iter)?;
260      let pool_token_program_info = next_account_info(account_info_iter)?;
261
262      let token_program_id = *pool_token_program_info.key;
263      if SwapVersion::is_initialized(&swap_info.data.borrow()) {
264          return Err(SwapError::AlreadyInUse.into());
265      }
266  }
```

Snipped

The image above shows the mentioned program id not being validated.

Recommendation

Include a validation to check that the `token_program_id` is correct. Take the image below as an example:



Affected Resource

- `src/processor.rs: L262`

METHODOLOGY

During this source code review, the Kudelski Security Services team reviewed code within the project within an appropriate IDE. During every review, the team spends considerable time working with the client to determine correct and expected functionality, business logic, and content to ensure that findings incorporate this business logic into each description and impact. Following this discovery phase the team works through the following categories:

- Authentication
- Authorization and Access Control
- Injection and Tampering
- Configuration Issues
- Logic Flaws
- Cryptography

Tools

The following tools were used during this portion of the test. A link for more information about the tool is provided as well.

- Visual Studio Code
- Semgrep
- Cargo Audit

Vulnerability Scoring Systems

Kudelski Security utilizes a vulnerability scoring system based on impact of the vulnerability, likelihood of an attack against the vulnerability, and the difficulty of executing an attack against the vulnerability based on a high, medium, and low rating system

Impact

The overall effect of the vulnerability against the system or organization based on the areas of concern or affected components discussed with the client during the scoping of the engagement.

High:

The vulnerability has a severe effect on the company and systems or has an effect within one of the primary areas of concern noted by the client

Medium:

It is reasonable to assume that the vulnerability would have a measurable effect on the company and systems that may cause minor financial or reputational damage.

Low:

There is little to no effect from the vulnerability being compromised. These vulnerabilities could lead to complex attacks or create footholds used in more severe attacks.

Likelihood

The likelihood of an attacker discovering a vulnerability, exploiting it, and obtaining a foothold varies based on a variety of factors including compensating controls, location of the application, availability of commonly used exploits, and institutional knowledge

High:

It is extremely likely that this vulnerability will be discovered and abused

Medium:

It is likely that this vulnerability will be discovered and abused by a skilled attacker

Low:

It is unlikely that this vulnerability will be discovered or abused when discovered.

Difficulty

Difficulty is measured according to the ease of exploit by an attacker based on availability of readily available exploits, knowledge of the system, and complexity of attack. It should be noted that a LOW difficulty results in a HIGHER severity.

Low:

The vulnerability is easy to exploit or has readily available techniques for exploit

Medium:

The vulnerability is partially defended against, difficult to exploit, or requires a skilled attacker to exploit.

High:

The vulnerability is difficult to exploit and requires advanced knowledge from a skilled attacker to write an exploit

Severity

Severity is the overall score of the weakness or vulnerability as it is measured from Impact, Likelihood, and Difficulty