

Secure Code Review

Findings and Recommendations Report Presented to:

Solcial

September 20, 2023
Version: 1.0

Presented by:

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

PUBLIC RELEASE

TABLE OF CONTENTS

TABLE OF CONTENTS	2
LIST OF FIGURES.....	2
LIST OF TABLES.....	2
EXECUTIVE SUMMARY	3
Overview	3
Key Findings	4
Scope and Rules of Engagement.....	5
TECHNICAL ANALYSIS & FINDINGS	6
Findings.....	7
KS-01 Potential Account Confusion	8
KS-02 Missing error handling for data types.....	10
KS-03 Multiple outdated dependencies.....	12
KS-04 Absence of Anchor Framework.....	13
METHODOLOGY	14
Tools.....	14
Vulnerability Scoring Systems.....	14
KUDELSKI SECURITY CONTACTS	Error! Bookmark not defined.

LIST OF FIGURES

Figure 1: Findings by Severity.....	6
-------------------------------------	---

LIST OF TABLES

Table 1: Scope	5
Table : Findings Overview	7

EXECUTIVE SUMMARY

Overview

Solcial engaged Kudelski Security to perform a code review of the solana-contract program.

The assessment was conducted remotely by the Kudelski Security Team. Testing took place between August 21, 2023 and September 15, 2023, and it was focused on the following objectives:

- To provide the customer with an assessment of their 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, we identified 1 **medium**, 1 **low** and 2 **informational** findings 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, as well as any applicable recommendations for remediation.

The review included checks for the following:

- Unchecked math
- Proper error handling
- Validation of function inputs and outputs
- Validation of ownership
- Account creation and usage
- Permissions checks and active/inactive status checks, including permission structures & validations
- Logic flow and sequence
- Proper usage, functionality, and/or validation of instructions
- Sufficient test coverage
- Connections and CPI calls to other programs

Some positive observations include:

- The `solana-contract` codebase presents a well-commented and clean architecture with attention to details as well as potential risks.
- The Solcial team responsible for this codebase were knowledgeable about the programs and provided explanations when we needed them.

Key Findings

The following are 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.

- Account type confusion
- Missing error handling for data types

Scope and Rules of Engagement

Kudelski performed a Secure Code Review 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/programs/solana-contract

Table 1: Scope

TECHNICAL ANALYSIS & FINDINGS

During the Secure Code Review, we identified 1 **medium**, 1 **low** and 2 **informational** findings according to our Vulnerability Scoring System.

The following chart displays the findings by severity.

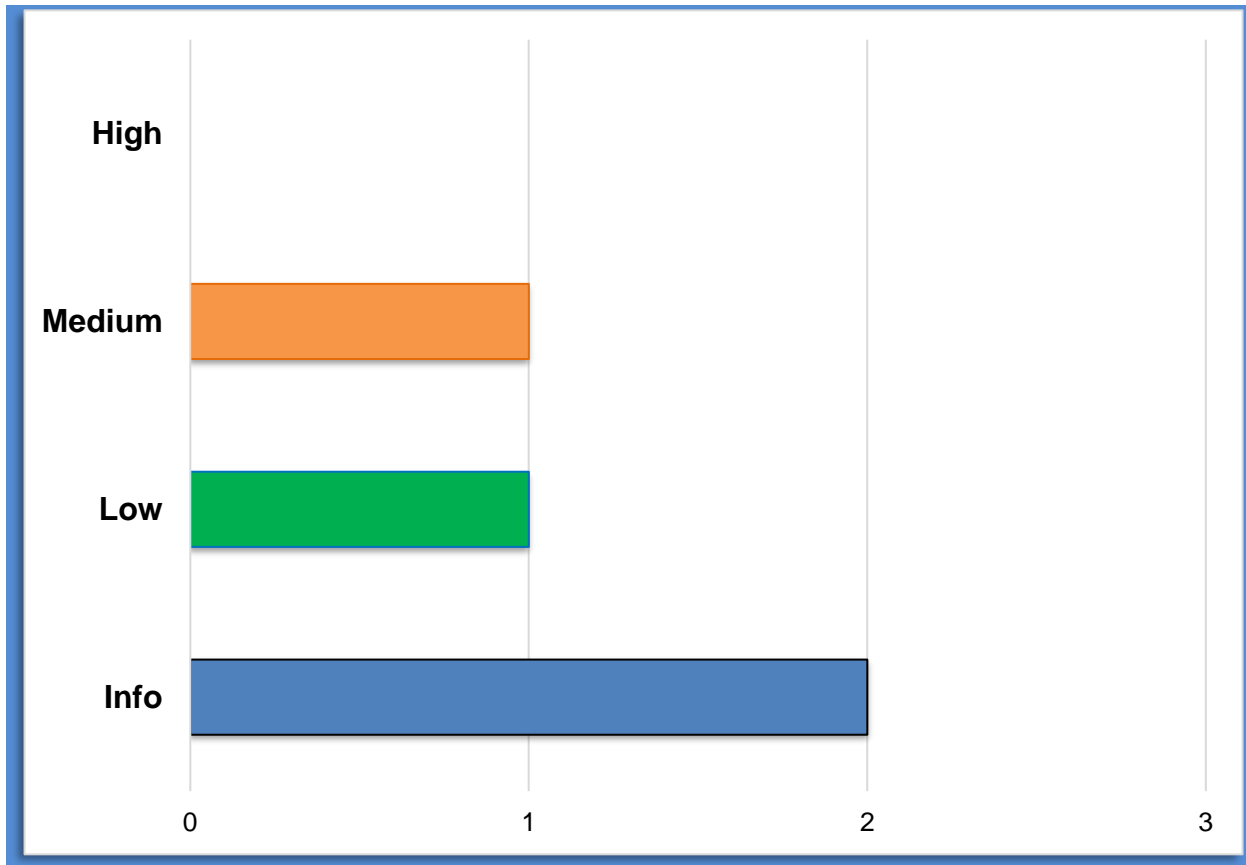


Figure 1: Findings by Severity

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	Medium	Potential account confusion
KS-02	Low	Missing error handling for data types
KS-03	Informational	Multiple outdated dependencies
KS-04	Informational	Absence of Anchor Framework

Table 2: Findings Overview

KS-01 Potential Account Confusion

Severity	Medium
----------	--------

Impact	Likelihood	Difficulty
High	Low	Medium

Description

On line 18 of `from_accounts.rs`, an `accounts` iterator is used that ultimately assigns accounts. There is no account type checking here.

Impact

If these accounts are assigned to similar structs, there is no way to determine if they are of the proper account type.

While it is unlikely, if it does occur, it could lead to tokens withdrawn from or sent to the wrong account.

Evidence

```

5  pub fn do_derive_from_accounts(item: proc_macro::TokenStream) -> proc_macro::TokenStream {
6      let input = parse_macro_input!(item as DeriveInput);
7
8      let span = input.span();
9      let name = input.ident();
10
11     let ty = generate_type(&input.data, span).unwrap_or_else(syn::Error::into_compile_error);
12     let expanded = quote! {
13         impl<'a> ::solana_contract::FromAccounts<'a> for #name<'a> {
14             fn from_accounts(accounts: &[::solana_program::account_info::AccountInfo<'a>]) -> Result<Self, ::solana_program::program_error::ProgramError>
15                 where
16                     Self: Sized,
17                 {
18                     let ai = &mut accounts.iter();
19                     Ok(Self {
20                         #ty
21                     })
22                 }
23     };
24     expanded.into()
25 }

```


Affected Resource

/programs/solana-contract/solana-contract-derive/src/from_accounts.rs line 18

Recommendation

Create a variety of account types using specialized programs, such as Payer, Receiver, etc. This could be enabled with type identifying `discriminator` data. Unique programs identified with `declare_id` can further be used to restrict the account creation and management.

Reference

<https://workshop.neodyme.io/level3-solution.html#5>

KS-02 Missing error handling for data types

Severity	Low
----------	-----

Impact High	Likelihood Low	Difficulty Medium
----------------	-------------------	----------------------

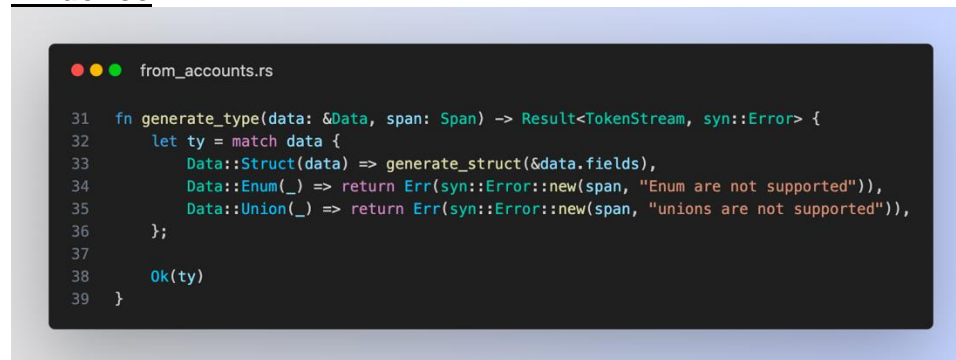
Description

On line 38 of `from_accounts.rs`, the `generate_type` function returns `Ok` while not all data types are caught in the preceding `match` block.

Impact

Usage of improper data passed to the function could return `Ok` but yield run-time errors within any calling functions.

Evidence



```

31 fn generate_type(data: &Data, span: Span) -> Result<TokenStream, syn::Error> {
32     let ty = match data {
33         Data::Struct(data) => generate_struct(&data.fields),
34         Data::Enum(_) => return Err(syn::Error::new(span, "Enum are not supported")),
35         Data::Union(_) => return Err(syn::Error::new(span, "unions are not supported")),
36     };
37
38     Ok(ty)
39 }

```

Affected Resource

`/programs/solana-contract/solana-contract-derive/src/from_accounts.rs` line 18

Recommendation

Create a variety of account types using specialized programs, such as Payer, Receiver, etc. This could be enabled with type identifying `discriminator` data.

Reference

<https://workshop.neodyme.io/level3-solution.html5>

KS-03 Multiple outdated dependencies

Severity

Informational

Description

There are multiple outdated important dependencies present in `solana-contract/Cargo.toml`, including `solana-program` (1.14.12) and `borsh` (0.9.3). Some, like `solana-program`, are outdated by more than two versions.

Impact

While there are no specific vulnerabilities related to these outdated versions that we can discern, outdated dependencies can contain security vulnerabilities that have been patched in updated versions. This can introduce unnecessary vulnerabilities into the system and increase the attack surface down the line.

Affected Resource

`solana-contract/Cargo.toml`

Recommendation

Ensure that all dependencies are continually updated to the latest version. Or, if the outdated version needs to be used, ensure that version notes are continually checked to verify there are no security vulnerabilities present in the outdated version.

Reference

<https://docs.rs/crate/<any crate name>>

KS-04 Absence of Anchor Framework

Severity

Informational

Description

The Anchor framework is not being used in the codebase.

Impact

The Anchor framework is a well-tested and well-respected framework in the Solana ecosystem that not only increases the ease of development in Solana, but it includes multiple built-in features, including automatic serialization/deserialization, type safety checks, ownership checks, signer checks, and, most importantly, security features. Re-implementing these features from scratch opens the codebase to unnecessary security vulnerabilities and potentially a greater attack surface.

Affected Resource

The entire `solana-contract` program is affected.

Recommendation

Use the Anchor framework to avoid potential security issues or errors that would come from doing everything from scratch.

Reference

<https://www.anchor-lang.com/>

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.

- 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.