verichains

SECURITY AUDIT OF

RENEC LEND

RENEC

Public Report

Jan 22, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Solana	A decentralized blockchain built to enable scalable, user-friendly apps for the world.
SOL	A cryptocurrency whose blockchain is generated by the Solana platform.
Anchor	A framework for Solana's Sealevel runtime providing several convenient developer tools for writing smart contracts.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Jan 22, 2024. We would like to thank the Renec for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Renec Lend. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had suggested some best practices for the code in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Renec Lend

The Relend lending protocol is based on the token-lending program.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Renec Lend.It was conducted on the code changed after cloning from the original repository: *https://github.com/solendprotocol/solana-program-library* at the commit 86746738e3c8289db6c18d2a0f2f6b421c74a2f1

SHA256 Sum	File
c133f5d8c832d7abce1da27da09942d0373733a6	cli/src/lending_state.rs
cd830789605443539b0bbe81d1fb761b347846d0	cli/src/main.rs
0eef15a6ef9c765606f6018bb3c59d9bf4d74887	sdk/src/error.rs
f75a5987a9a7a5389a32a70fec9989d2648f7c53	sdk/src/lib.rs
1be64b4daeac4574199aec1876cc010fd31574db	sdk/src/oracles.rs
540f7acde7aaba1f36cd94e1423b06927839c8b3	sdk/src/state/rate_limiter.rs
318e27d63efa2a9b7849fe07d5ed060a23131cc1	sdk/src/state/obligation.rs
aa7d8039e81177189ae0b7a640a268a2b8497bd4	sdk/src/state/last_update.rs
4792fbc2310528e4a85c3422602114e9370fa386	sdk/src/state/mod.rs
4c744332314869578fe284375a105a2656019153	sdk/src/state/price.rs
10005e77961db5ddc71ae786a12399c75d225250	sdk/src/state/lending_market_metadata.rs
138c5700dde58b647a18c9755ca2200c38eda755	sdk/src/state/lending_market.rs
9c8ca6f9527897b2e39c72372a99198c57143ab5	sdk/src/state/reserve.rs
eff10e1f88c6648e5c6bb9007de0e4df4b36a2c4	sdk/src/math/decimal.rs
8e6e470ab5f1af3b99b97d06926b947aff6f3d04	sdk/src/math/rate.rs
f3bc358a22f0cd8ffb9e2a07e03ad9c62bae4480	sdk/src/math/mod.rs
a938c03b087283e5741ff2b9a13044e0ef1f58f4	sdk/src/math/common.rs

575e9dd1429c34a66946bff55d8eb92632320ea9	sdk/src/instruction.rs
732a92214029bddca1ae24e0e22784908c663460	brick/src/entrypoint.rs
f55563fb4b47f77084ba8ffd5f856a67e141c78e	brick/src/lib.rs
73a964c6f8bbbcba8be0313edf41a77259bc4eba	program/src/processor.rs
0675d3a4cde50892a49109c1effbbc733af91bdd	program/src/entrypoint.rs
5ffe3271f1c9de1db90957d6f46e0596c71d231e	program/src/lib.rs

Table 1. Hash value of files in the latest version

1.3. Audit methodology

Our security audit process for Solana smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the Solana smart contract:

- Arithmetic Overflow and Underflow
- Signer checks
- Ownership checks
- Rent exemption checks
- Account confusions
- Bump seed canonicalization
- Closing account
- Signed invocation of unverified programs
- Numerical precision errors
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.

SEVERITY LEVEL	DESCRIPTION
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 2. Severity levels

1.4. Disclaimer

Renec acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Renec understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Renec agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Renec will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Renec, the final report will be considered fully accepted by the Renec without the signature.

2. AUDIT RESULT

2.1. Overview

The Renec Lend was written in `Rust` language. The source code was written based on the library of `Anchor`, `solana-program`.

The Renec Lend uses two oracles to fetch token prices. These oracles are passed as parameters on the instruction processing.

The `get_oracle_price` function also supports the `REUSD` stable coin and pairs `REUSD_REVND` and `REUSD_RENGN`.

2.2. Findings

During the audit process, the audit team had suggested some best practices for the code in the contract code.

2.2.1. Best practices to avoid potential vulnerabilities on dependencies - INFORMATIVE

Affected folders:

- token-lending/brick
- token-lending/cli
- token-lending/program
- token-lending/sdk

The program uses outdated versions of dependencies. The audit team recommends to update to the latest version of dependencies.

#	Dependency	Current Version	Latest Version
1	token-lending/brick	1.13.6	1.16.23
2	token-lending/cli/solana-clap-utils	1.13.3	1.17.9
3	token-lending/cli/solana-cli-config	1.13.3	1.17.9
4	token-lending/cli/solana-client	1.13.3	1.17.9
5	token-lending/cli/solana-logger	1.13.3	1.17.9
6	token-lending/cli/solana-sdk	1.13.3	1.17.9
7	token-lending/cli/solana-program	1.13.3	1.17.9

#	Dependency	Current Version	Latest Version
8	token-lending/cli/solana-account-decoder	1.13.3	1.17.9
9	token-lending/cli/spl-token	3.3.0	4.0.0
10	token-lending/cli/spl-associated-token-account	1.0	2.2.0
12	token-lending/program/solana-program	1.13.3	1.17.9
13	token-lending/program/spl-token	3.2.0	4.0.0
14	token-lending/program/bytesmuck	1.5.1	1.14.0
15	token-lending/sdk/solana-program	1.13.3	1.17.9
16	token-lending/sdk/anchor-lang	0.20.1	0.29.0
17	token-lending/sdk/borsh	0.9.3	1.2.1
18	token-lending/sdk/borsh-derive	0.9.3	1.2.1

UPDATES

Jan 22, 2024: The Renec team has been acknowledged the issue.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 15, 2023*	Public Report	Verichains Lab
1.1	*Jan 22, 2024*	Public Report	Verichains Lab

Table 3. Report versions history