ackee blockchain

EverSOL Stake Pool Audit

25. 2. 2022

by Ackee Blockchain

Contents

1. Document Revisions

1.0	Final report	Feb 25, 2022

2. Overview

This document presents our findings in reviewed program.

2.1. Ackee Blockchain

Ackee Blockchain is an auditing company based in Prague, Czech Republic, specialized in audits and security assessments. Our mission is to build a stronger blockchain community by sharing knowledge – we run a free certification courses Winter School of Solana, Summer School of Solidity and teach at the Czech Technical University in Prague. Ackee Blockchain is backed by the largest VC fund focused on blockchain and DeFi in Europe, Rockaway Blockchain Fund.

2.2. Audit Methodology

The Ackee Blockchain auditing process follows a routine series of steps:

1. Code review
    a. High-level review of the specifications, sources, and instructions provided to us to make sure we understand the project's size, scope, and functionality.
    b. Detailed manual code review, which is the process of reading the source code line-by-line to identify potential vulnerabilities. We focus mainly on common classes of Solana program vulnerabilities, such as:

    missing ownership checks, missing signer authorization, signed CPI of unverified programs, cosplay of Solana accounts, missing rent exemption assertion, bump seed canonicalization, incorrect accounts closing, casting truncation, numerical precision errors, arithmetic overflows or underflows, ...

   c. Comparison of the code and given specifications, ensuring that the program logic correctly implements everything intended.
   d. Review of best practices, to improve efficiency, clarity and maintainability.

2. Testing and automated analysis
   a. Run client's tests to ensure that the system works as expected, potentially write missing unit or fuzzy tests using our own testing framework Trdelnik.

3. Local deployment + hacking
   a. The programs are deployed locally and we try to attack the system and break it. There is no specific strategy here, and each project's attack attempts are characteristic of each program audited. However, when trying to attack, we rely on the information gained from previous steps and our rich experience.

2.3. Review team

Member's Name	Position
Tibor Tribus	Lead Auditor
Vladimír Marcin	Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

2.4. Disclaimer

We've put our best effort to find all vulnerabilities in the system, however our findings shouldn't be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

3. Executive Summary

Everstake engaged Ackee Blockchain to conduct a security review of their Stake Pool program with a total time donation of 5 engineering days. The review took place between 17. February and 25. February 2022.

The scope included the following repository with a given commit:

- **Repository**: stake-pool
- **Commit**: `43342a929872f2191634cfa0b355856aa1482a26`

The repository presented is a fork of the Solana stake-pool program, which Everstake subsequently expanded with new functionality. As several companies have already audited the Solana stake-pool program, our review aimed to verify only the safety of the newly added functionality. Specifically, it was about functionality related to instant unstaking and the addition of a treasury fee. Instant unstaking is implemented by instructions `deposit_liquidity_sol()` and `withdraw_liquidity_sol()`. Scope of treasury fee functionality cannot be defined precisely. Still, we have examined all parts of the code related to it in detail. Along with the newly added functionality, we tried to identify all system parts affected and focused on them during the audit.

The beginning of the audit was dedicated to understanding the entire Stake pool program as it was necessary to understand the audited part. During the review, we paid special attention to the findings from previous audits of the Stake pool program (whether it was correctly addressed) and the newly added functionality.

Our review resulted in only two informational errors, suggesting that the audited code is secure and ready for production deployment. Overall code quality is high, as it is a program from the SPL library and the newly added

functionality also copies this high standard.

Supporting documentation in RustDoc and the documentation on the Solana website and the Everstake website sufficiently describes the system and has significantly helped us understand the system overview.

The tests sufficiently verified the basic functionality and helped us understand the system. They were also a good starting point in our attempts to attack the audited program.

Ackee Blockchain recommends Everstake to:

- address reported issues,
- monitor the SPL stake-pool program and apply major changes in the future, as the program is still in active development.

4. System Overview

This section contains an outline of the audited contracts. Note that this is meant for understandibility purposes and does not constitute a formal specification.

Staking is the process by which a SOL token holder assigns some or all of their tokens to a particular validator or validators, which helps increase those validators' voting weight. Assigning your tokens to add to a validator's stake-weight is known as "delegating" your tokens.

On Solana, stake pools provide a way to delegate SOL to a validator delegation strategy rather than directly to an individual validator. Deciding which validators to delegate to requires technical skill and ongoing maintenance. With a stake pool, delegators can task a "stake pool manager" with determining how to best delegate SOL. Typically, the delegation strategy will delegate SOL across more than one validator, according to some delegation criteria (strategy).

4.1. Everstake Staking process

1. Delegator (user) deposits SOL to EverSOL Stake Pool.
2. Delegator receives eSOL tokens equivalent to the size of their stake in the pool. The pool uses rate of exchange from previous epoch to calculate the amount of eSOL to be minted.
3. EverSOL Stake Pool utilizes an automated delegation strategy to spread the delegator's stake across multiple validators to maximize rewards (this takes ~1 epoch).
4. As new SOL is minted each epoch, the total amount of staked SOL in the pool increases, while the amount of eSOL tokens remains consistent with the amount of SOL deposited. This increases the value of the stake pool

tokens. Note: the rewards from the stake pool come in the form of newly minted SOL which is automatically staked to the pool.

5. When a delegator comes to withdraw their staked SOL, they exchange their eSOL back for SOL (receiving more tokens than they staked initially).
6. The withdrawn SOL is placed in a new stake account that belongs to the delegator. They can then withdraw tokens from the stake account back to their primary wallet.

4.2. Actors

This part describes actors of the system, their roles and permissions.

Manager

Creates and manages the stake pool, earns fees, can update the fee, staker, and manager.

Staker

Adds and removes validators to the pool, rebalances stake among validators.

User

Provides staked SOL into an existing stake pool.

4.3. Trust model

The deposit from the user is entirely under the control of the program, so the user does not have to trust any third party.

5. Vulnerabilities risk methodology

Each finding contains an *Impact* and *Likelihood* ratings.

If we have found a scenario in which the issue is exploitable, it will be assigned an impact of *Critical*, *High*, *Medium*, or *Low*, based on the direness of the consequences it has on the system. If we haven't found a way, or the issue is only exploitable given a change in configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.) or given a change in the codebase, then it will be assigned an impact rating of *Warning* or *Informational*.

Low to *Critical* impact issues also have a *Likelihood* which measures the probability of exploitability during runtime.

5.1. Finding classification

The full definitions are as follows:

Impact

- **High** - Code that activates the issue will lead to undefined or catastrophic consequences for the system.
- **Medium** - Code that activates the issue will result in consequences of serious substance.
- **Low** - Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.
- **Warning** - The issue cannot be exploited given the current code and/or configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.), but could be a security vulnerability if these were to change slightly. If we haven't found a way to exploit the issue given the time constraints, it might be marked as

"Warning" or higher, based on our best estimate of whether it is currently exploitable.

- **Informational** - The issue is on the border-line between code quality and security. Examples include insufficient logging for critical operations. Another example is that the issue would be security-related if code or configuration (see above) was to change.

Likelihood

- **High** - The issue is exploitable by virtually anyone under virtually any circumstance.
- **Medium** - Exploiting the issue currently requires non-trivial preconditions.
- **Low** - Exploiting the issue requires strict preconditions.

6. Findings

This section contains the list of discovered findings. Unless overriden for purposes of readability, each finding contains:

- a *Description*,
- an *Exploit scenario*, and
- a *Recommendation*

Many times, there might be multiple ways to solve or alleviate the issue, with varying requirements in terms of the necessary changes to the codebase. In that case, we will try to enumerate them all, making clear which solve the underlying issue better (albeit possibly only with architectural changes) than others.

Summary of Findings

Id		Type	Impact	Likelihood	Status
I1	Documentation and code mismatch.	N/A	Informational	N/A	Reported
I2	TODOs in production code.	N/A	Informational	N/A	Reported

Table 1. Table of Findings with High Impact

6.2. Documentation and code mismatch.

Impact:	Informational	Likelihood:	N/A
Target:	instructions.rs	Type:	N/A

Description

In the documentation, there is an account for validator fee (see docs), but in the code, there is no such account (see `instruction.rs` and `processor.rs`).

Recommendation

Update RustDoc to match the code.

Go back to Findings Summary

6.3. TODOs in production code.

Impact:	Informational	Likelihood:	N/A
Target:	processor.rs	Type:	N/A

Description

TODO is forgotten in the `processor.rs` file (see comment).

Recommendation

Delete or address all TODOs. TODOs should not be in the production code.

Go back to Findings Summary

6.4. Appendix A

How to cite

Please cite this document as:

> Ackee Blockchain, "EverSOL Stake Pool Audit", January 22, 2022.

If an individual issue is referenced, please use the following identifier:

`ABCH-{project_identifer}-{finding_id}`,

where `{project_identifier}` for this project is `EVERSOL-STAKE-POOL` and `{finding_id}` is the id which can be found in Summary of Findings. For example, to cite I1 issue, we would use `ABCH-EVERSOL-STAKE-POOL-I1`.

ackee blockchain

Thank You

Ackee Blockchain a.s.

Prague, Czech Republic

hello@ackeeblockchain.com

https://discord.gg/wpM77gR7en