Solana Labs - Solana Runtime Commission Update

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **January 4th, 2023 - January 6th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	01/05/2023	Michael Smith
0.2	Document Review	01/06/2023	Piotr Cielas
0.3	Document Review	01/06/2023	Gabi Urrutia
1.0	Remediation Plan	05/19/2023	Piotr Cielas
1.1	Remediation Plan Review	05/19/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Isabel Burruezo	Halborn	Isabel.Burruezo@halborn.com
Michael Smith	Halborn	Michael.Smith@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

The Solana network uses a proof of stake consensus mechanism, allowing validators and stakers to participate in securing the network. Validators are rewarded for their effort and distribute the reward to stakers after charging a commission. Currently, validators can rug pull users by changing the commission right before rewards are distributed, this update aims to mitigate this.

Solana Labs engaged Halborn to conduct a security audit on their Solana programs, beginning on January 4th, 2023 and ending on January 6th, 2023 . The security assessment was scoped to the programs provided in the solana GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

The team at Halborn was provided 3 days for the engagement and assigned one full-time security engineer to audit the security of the programs in scope. The security engineer is a blockchain and Solana program security expert/experts with advanced penetration testing and Solana program hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Identify potential security issues within the programs

Halborn performed a combination of a manual review of the source code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program audit. While manual testing is recommended to uncover flaws in business logic, processes, and implementation; automated testing techniques help enhance coverage of programs and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the audit:

- Research into the architecture, purpose, and use of the platform.
- Manual program source code review to identify business logic issues.
- Mapping out possible attack vectors
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could lead to arithmetic vulnerabilities.
- Scanning dependencies for known vulnerabilities (cargo audit).
- Local runtime testing (solana-test-framework)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.

2 - May cause temporary impact or loss.

1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL

9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.3 SCOPE

Code repositories:

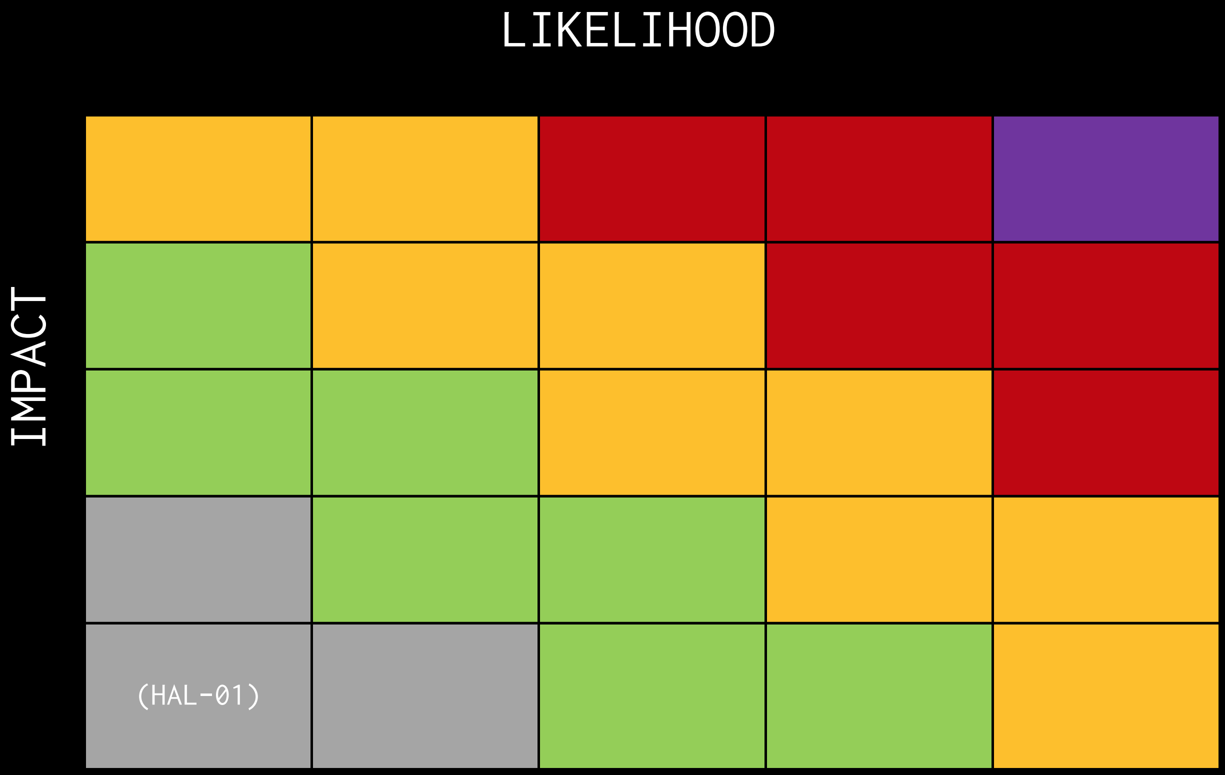
1. Project Name

- Repository: Solana
- Pull Request: 29389
- Programs in scope:
    1. Vote (programs/vote)
    2. SDK (sdk)

Out-of-scope:
- third-party libraries and dependencies
- financial-related attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	1



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) COMMISSION CAN BE UPDATED HALFWAY THROUGH EPOCHS	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) COMMISSION CAN BE UPDATED HALFWAY THROUGH EPOCH - INFORMATIONAL

Description:

Users stake their SOL to validators, who in turn vote on which blocks they believe should be added to the network. Validators are rewarded SOL for their efforts and distribute these rewards to stakers after taking a commission. Prior to this update, validators could change their commission moments before rewards are distributed, now validators can only change their commission in the first half of an epoch or approximately twenty-four hours before rewards are distributed.

Code Location:

Listing 1: programs/vote/src/vote_state/mod.rs (Line 1442)

```rust
/// Given the current slot and epoch schedule, determine if a
  ↳ commission change
/// is allowed
pub fn is_commission_update_allowed(slot: Slot, epoch_schedule: &
  ↳ EpochSchedule) -> bool {
    // always allowed during warmup epochs
    if let Some(relative_slot) = slot
        .saturating_sub(epoch_schedule.first_normal_slot)
        .checked_rem(epoch_schedule.slots_per_epoch)
    {
        // allowed up to the midpoint of the epoch
        relative_slot.saturating_mul(2) <= epoch_schedule.
  ↳ slots_per_epoch
    } else {
        // no slots per epoch, just allow it, even though this
  ↳ should never happen
        true
    }
}
```

Recommendation:

Users should be informed that validators can still change their commission after staking their SOL just that now commissions can’t be changed during the second half of an epoch.

Remediation Plan:

ACKNOWLEDGED: The Solana team acknowledged this issue.

MANUAL TESTING

In the manual testing phase, the following scenarios were simulated. The scenarios listed below were selected based on the severity of the vulnerabilities Halborn was testing the program for.

4.1 UPDATING COMMISSION

Description:

The vote program's UpdateCommission instruction should fail if the vote account attempts to change the commission after the second half of the epoch. Several tests were done to execute UpdateCommission instructions at different slots in the epoch to ensure the code behaves as expected.

Results:

No code vulnerabilities were identified.

AUTOMATED TESTING

5.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with the detection of well-known security issues and vulnerabilities. Among the tools used was cargo-audit, a security scanner for vulnerabilities reported to the Rust-Sec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

Results:

ID	package	Short Description
RUSTSEC-2020-0071	time	Potential segfault, Upgrade to >=0.2.23
RUSTSEC-2021-0139	ansi_term	ansi_term is unmaintained
RUSTSEC-2020-0016	net2	net2 crate has been deprecated; use socket2 instead
RUSTSEC-2021-0127	serde_-cbor	serde_cbor is unmaintained

THANK YOU FOR CHOOSING

// HALBORN