Inflation Logic Code Review

Solana

19 October 2020
Version: 1.0

Presented by:
Kudelski Security Research Team
Kudelski Security – Nagravision SA

Corporate Headquarters
Kudelski Security – Nagravision SA
Route de Genève, 22-24
1033 Cheseaux sur Lausanne
Switzerland

DOCUMENT PROPERTIES

Version:	1.0
File Name:	Solana Code Inflation Review Report - Final.docx
Publication Date:	19 October 2020
Confidentiality Level:	For Public Distribution
Document Owner:	Kudelski Security
Document Recipient:	Solana Foundation
Document Status:	Approved

Table of Figures

EXECUTIVE SUMMARY

Kudelski Security ("Kudelski"), the cybersecurity division of the Kudelski Group, was engaged by Solana Foundation. ("Solana") client to conduct an external security assessment in the form of a Code Review of the Solana Blockchain implementation specifically focusing on the Inflation capabilities during this review.

The assessment was conducted remotely by the Kudelski Security Team. The tests took place from September 18, 2020 to September 30, 2020 and focused on the following objectives:

1. The system operates in a manner consistent with the white paper or documented objectives.
2. That out of bounds, overflow, and logic conditions are met and are secure
3. That the scenarios discussed follow expected conditions with no ability to improperly manipulate inflation scenarios

This report summarizes the tests performed and findings in terms of strengths and weaknesses. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Teams took to exploit each vulnerability, and recommendations for remediation. As of this final report, all serious vulnerabilities have been resolved to our satisfaction by the Solana development team.

1.1 Engagement Limitations

The project was time-boxed to be finished by September 30, 2020.

During the project the Kudelski teams has focused on the following areas

- Validate technical design claims and cryptographic coding underlying the behavior and intent of the technical systems
- Perform a code-review of provided Rust Code, especially focusing on code written by the internal team, assuming third-party libraries act as expected
- Validate implementation choices, completeness, and assumptions according to the design provisions and deployment
- Validate predictions and behaviors with special attention to decentralized and distributed behaviors for claims with special attention to the defined areas
- Provide recommendations for security related improvements and corrections to the infrastructure and architecture, if found
- Provide recommendations for architectural and implementation related improvements to the infrastructure and architecture, if found

Out of scope for this engagement, which can be included in future engagements include deployment of the infrastructure at-scale to validate findings, operational execution of the code to perform a pen-test of running binaries (memory review, attacks to binaries, theft of secrets), operational assessment of alerting and monitoring when non-ethical behavior is present in the system, or participation in any running test-net environments.

1.2 Engagement Analysis

The engagement was performed by cloning the relevant repositories into the internal GitLab environment utilized by Kudelski Security. The code base cloned was from the Master branch

at 07 September, 2020 with the SHA 9eb10d914419fe448852f1aa1e6d65df9743c1e3 of the last commit. Considerable time and effort of the engagement was spent focusing on the facts deemed most relevant to the client as outlined in the initial document provided by SOLANA.

The code review was then conducted by analyzing the relevant code from different angles with both automated and manual tools.

As a result of our work, we identified **2 High**, **1 Medium**, **2 Low**, and **11 Informational** findings. All of the non-informational findings are remediated at the time of this report.

The high findings are either omissions vulnerabilities in the code or insecure handling of cryptographic components, e.g. private key handling, number over/underflow in calculations and random number generation. It may also be serious discrepancies between design and implementation that severely affect the functionality of the application. In the case of mathematical operations or inflation, a high vulnerability could indicate unintended consequences.

We only found one medium finding that is an issue with selecting a fixed value for the inflation but in the architecture documentation, it is described as a variable value.

Multiple low findings were identified during our assessment. Although these findings do not represent a significant threat, they can increase the attack surface of the system or trigger unexpected conditions.

Several informational findings were found. These are mostly cosmetic or omissions in functionality.

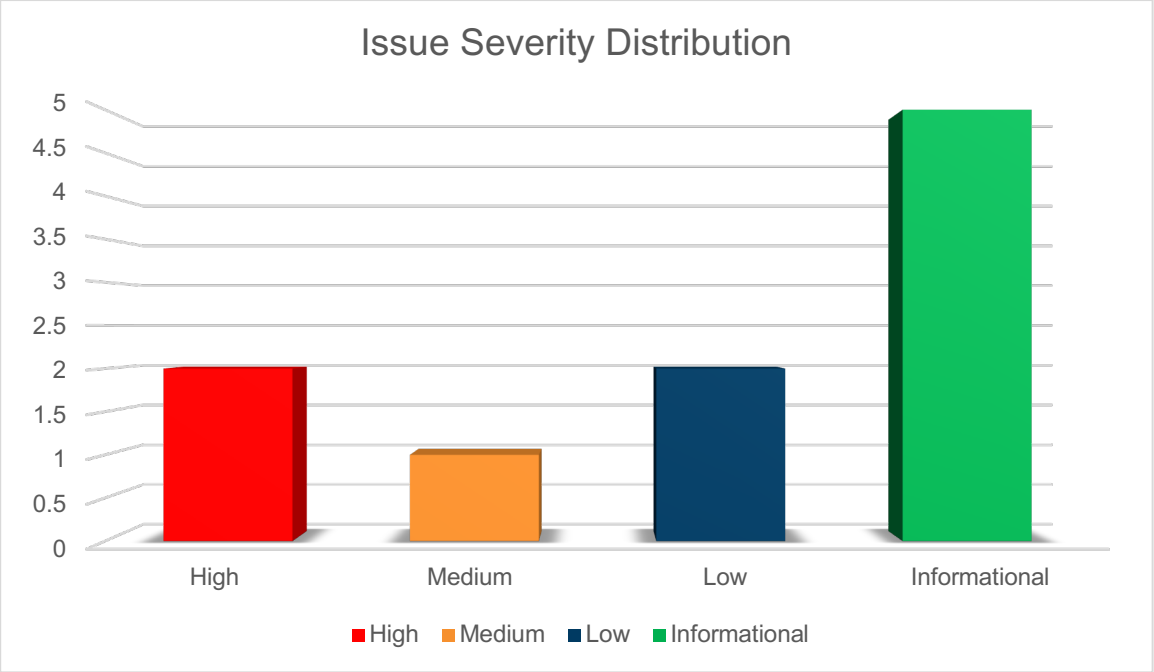


Figure 1 Issue Severity Distribution

1.3 Observations

1.3.1 Code and design documentation

We have performed the following analysis on security of the code

- Code construct and call-trees
- Verification that implementation in core code based on the architecture documentation provided.

The documentation is well written and covers the architecture and the implementation of it as code.

1.3.2 Coding Style

As this is a follow up on the previous code reviews, we must commend the development team on how much of the code quality that has improved. This has made much of the findings being on the bottom half of the risk spectrum, with low and informational taking up more than 90% of the findings.

The codebase analyzed in this assignment was well written and documented to the standard that we would expect from such a high-profile project.

1.4 Issue Summary List

ID	SEVERITY	FINDING
solana_inflation_audit#1	**Informational**	Inflation consequence

ID	SEVERITY	FINDING
solana_inflation_audit#2	Medium	Longterm stable rate is fixed
solana_inflation_audit#3	Informational	Inflation reward inconsistency
solana_inflation_audit#4	Informational	Terminology confusion
solana_inflation_audit#5	High	Negative year allowed in inflation calculation
solana_inflation_audit#6	Informational	Deprecated code
solana_inflation_audit#7	Low	Unclear use of variable epoch
solana_inflation_audit#8	Informational	Incorrect documentation of equation
solana_inflation_audit#9	High	Calculation of inflation uses two conflicting models
solana_inflation_audit#10	Low	Tokens never created
solana_inflation_audit#11	Informational	Retired code still in use
solana_inflation_audit#12	Informational	Docstrings missing
solana_inflation_audit#13	Informational	Code documentation standard
solana_inflation_audit#14	Informational	Incomplete comment
solana_inflation_audit#15	Informational	Spellcheck code
solana_inflation_audit#16	Informational	Unclear naming of function

2. METHODOLOGY

Kudelski Security uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2 Methodology Flow

2.1 Kickoff

The project is kicked all of the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

2.2 Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the particular project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

2.3 Review

The review phase is where a majority of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project

3. Assessment of the cryptographic primitives used
4. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general list and not comprehensive, meant only to give an understanding of the issues we are looking for.

Cryptography

We analyzed the cryptographic primitives and components as well as their implementation. We checked in particular:

- Matching of the proper cryptographic primitives to the desired cryptographic functionality needed
- Security level of cryptographic primitives and their respective parameters (key lengths, etc.)
- Safety of the randomness generation in general as well as in the case of failure
- Safety of key management
- Assessment of proper security definitions and compliance to use cases
- Checking for known vulnerabilities in the primitives used

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

2.4 Reporting

Kudelski Security delivers a preliminary report in PDF format that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project as a whole. We may conclude that the overall risk is low, but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We not only report security issues identified but also informational findings for improvement categorized into several buckets:

- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps, that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

2.5 Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediate.

The output of this phase will be a final report with any mitigated findings noted.

2.6 Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessment the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. These is a solid baseline for severity determination. Information about the severity ratings can be found in **Appendix C** of this document.

3. TECHNICAL DETAILS

This section contains the technical details of our findings as well as recommendations for improvement.

3.1 Inflation consequence

Finding ID: solana_inflation_audit#1

Severity: **Informational**

Status: **Open**

Description

The "Validation-client Economics" page mentions that the inflation at year 0 is 15% with a disinflationary rate of 15%, the provided examples instead use 7,5% respectively 20%.

Proof of Issue

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

Recommendation

Let the examples for the developers reflect the ideas that the code and the documentation inspires to.

References

https://docs.solana.com/implemented-proposals/ed_overview/ed_validation_client_economics/ed_vce_state_validation_protocol_based_rewards

3.2 Longterm stable rate is fixed

Finding ID: solana_inflation_audit#2

Severity: **Medium**

Status: **Remediated**

Description

The long-term stable rate is here specified to be between 1-2%, in the code this value is defined as specifically 1.5%.

Proof of Issue

N/A

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

If the documentation stipulates that, the long-term stable rate will be 1-2% and the source code pin it at 1.5% there is a discrepancy that needs to be addressed.

Recommendation

Correct either the documentation or the code to be coordinated.

References

https://docs.solana.com/implemented-proposals/ed_overview/ed_validation_client_economics/ed_vce_overview

3.3 Inflation reward inconsistency

Finding ID: solana_inflation_audit#3

Severity: **Informational**

Status: **Remediated**

Description

90% of the inflation is supposed to go to the validators, where the other 10% goes is not mentioned. The code at this point however provides a theoretical limit of 95% going to the validators.

Proof of Issue

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

Clarity for the users.

Recommendation

Correct either the documentation or the code to be coordinated.

References

https://docs.solana.com/implemented-proposals/ed_overview/ed_validation_client_economics/ed_vce_overview

3.4 Terminology confusion

Finding ID: solana_inflation_audit#4

Severity: **Informational**

Status: **Remediated**

Description

A node is defined as "a computer participating in a cluster", a node count however is defined as "The number of validators participating in a cluster". This discrepancy between a node being either a computer or a validator could be confusing for some readers.

Proof of Issue

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

Clarity for the users.

Recommendation

Correct the documentation to use the same definition

References

https://docs.solana.com/terminology

3.5 Negative year allowed in inflation calculation

Finding ID: solana_inflation_audit#5

Severity: **High**

Status: **Remediated**

Description

No checks in the inflation.total method confirming that the year is 0 or more.

Proof of Issue

```
/// inflation rate at year
pub fn total(&self, year: f64) -> f64 {
    let tapered = self.initial * ((1.0 - self.taper).powf(year));
    if tapered > self.terminal {
        tapered
    } else {
        self.terminal
    }
}
```

Filename: sdk\src\inflation.rs

Beginning Line Number: 56

Severity and Impact Summary

If you pass a negative number for year you will get a result that is outside the defined parameters and will result in a value that will be used in other calculations erroneously.

Recommendation

Insert a check to require the year parameter to be 0 or more in the code.

References

3.6 Deprecated code

Finding ID: solana_inflation_audit#6

Severity: **Informational**

Status: **Remediated**

Description

The deprecated method "storage" does not seem to serve any purpose in the code.

Proof of Issue

```
    /// DEPRECATED, this field is currently unused
    pub storage: f64,
}
```

Filename: sdk\src\inflation.rs

Beginning Line Number: 21

Severity and Impact Summary

Deprecated code left in the code base. Should be removed unless it is used.

Recommendation

Delete the deprecated code.

References

N/A

3.7 Unclear use of variable epoch

Finding ID: solana_inflation_audit#7

Severity: **Low**

Status: **Remediated**

Description

In the update_reward function it is somewhat unclear that the variable epoch is the previous epoch and that self.epoch is the current one.

Proof of Issue

```
// update reward for previous epoch
fn update_rewards(&mut self, epoch: Epoch) {
    if epoch == self.epoch() {
        return;
    }
```

Filename: runtime/src/bank.rs

Beginning Line Number: 897

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Rename the referenced epoch so that it clearly reflects the use.

References

N/A

3.8 Incorrect documentation of equation

Finding ID: solana_inflation_audit#8

Severity: **Informational**

Status: **Remediated**

Description

Incorrect comment for let period = self.epoch_schedule in the the years_elapsed equation.

Proof of Issue

```
    // period: time that has passed as a fraction of a year, basically the length of
    //  an epoch as a fraction of a year
    //  years_elapsed =   slots_elapsed /  slots/year
    let period = self.epoch_schedule.get_slots_in_epoch(epoch) as f64 / self.slots_per_year;
```

Filename: runtime/src/bank.rs

Beginning Line Number: 907

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Correct the comment so that it reflects the code

References

N/A

3.9 Calculation of inflation uses two conflicting models

Finding ID: solana_inflation_audit#9

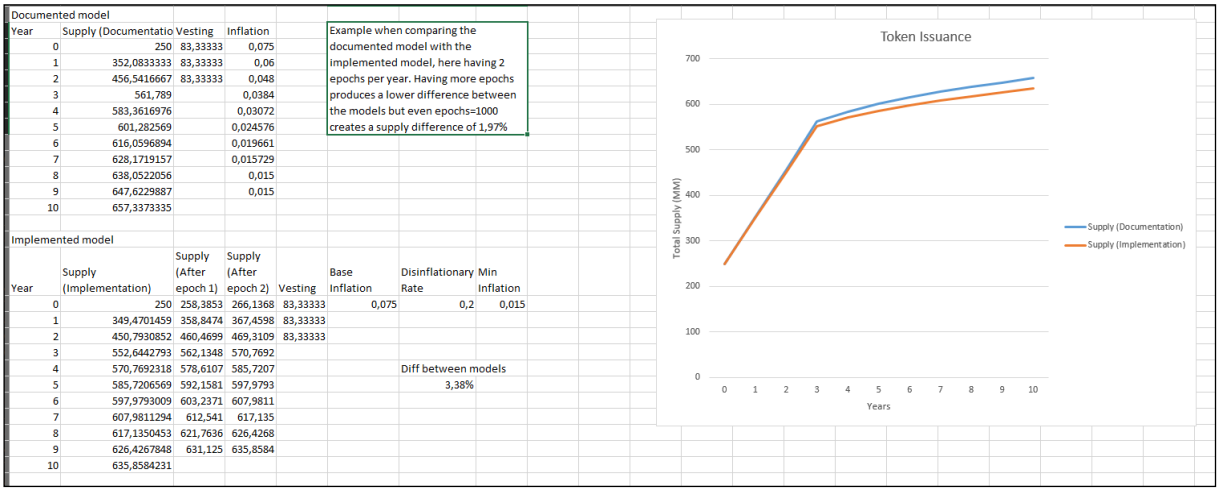
Severity: **High**

Status: **Remediated**

Description

Since the implemented model calculates the inflation impact first at the end of an epoch and the documented model calculates based on the inflation at the start of the year, the total token supply differs between the models

Proof of Issue



Documented model

Year	Supply (Documentatio	Vesting	Inflation
0	250	83,33333	0,075
1	352,0833333	83,33333	0,06
2	456,5416667	83,33333	0,048
3	561,789		0,0384
4	583,3616976		0,03072
5	601,282569		0,024576
6	616,0596894		0,019661
7	628,1719157		0,015729
8	638,0522056		0,015
9	647,6229887		0,015
10	657,3373335		

Example when comparing the documented model with the implemented model, here having 2 epochs per year. Having more epochs produces a lower difference between the models but even epochs=1000 creates a supply difference of 1,97%

Implemented model

Year	Supply (Implementation)	Supply (After epoch 1)	Supply (After epoch 2)	Vesting	Base Inflation	Disinflationary Rate	Min Inflation
0	250	258,3853	266,1368	83,33333	0,075	0,2	0,015
1	349,4701459	358,8474	367,4598	83,33333			
2	450,7930852	460,4699	469,3109	83,33333			
3	552,6442793	562,1348	570,7692				
4	570,7692318	578,6107	585,7207			Diff between models	
5	585,7206569	592,1581	597,9793			3,38%	
6	597,9793009	603,2371	607,9811				
7	607,9811294	612,541	617,135				
8	617,1350453	621,7636	626,4268				
9	626,4267848	631,125	635,8584				
10	635,8584231						

Figure 3 Inflation model consequence

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

There is a discrepancy between the documented way the inflation logic works and the way it is implemented in code. It may influence the complete economic system in a negative way.

Recommendation

Chose one way of calculating the inflation and be sure to use it in all calculations in the code.

References

https://docs.solana.com/implemented-proposals/ed_overview/ed_validation_client_economics/ed_vce_overview

3.10 Tokens never created

Finding ID: solana_inflation_audit#10

Severity: **Low**

Status: **Remediated**

Description

The foundation portion of the inflation does not seem to ever be introduced into the network, thus meaning that the tokens are never created, which makes the actual theoretical maximal inflation rate at 95% of the stated amount.

Proof of Issue

This is part of the architecture and

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

The full potential of the inflation mechanics are not utilized as the 95% cap is in place.

Recommendation

Revisit the architecture design and either correct or document the factual inflation

References

N/A

3.11 Retired code still in use

Finding ID: solana_inflation_audit#11

Severity: **Informational**

Status: **Remediated**

Description

The comment to the call self.update_sysvar_account should explain the actual purpose of the call rather than explaining that it apparently can be "retired".

Proof of Issue

```
// this sysvar could be retired...
self.update_sysvar_account(&sysvar::rewards::id(), |account| {
    sysvar::rewards::create_account(
        self.inherit_sysvar_account_balance(account),
        validator_point_value,
    )
```

Filename: runtime/src/bank.rs

Beginning Line Number: 923

Severity and Impact Summary

If this code is retired, it should be removed. Otherwise the comment should be correctly reflecting the use of the function.

Recommendation

Remove or correct the stated use of the function.

References

N/A

3.12 Docstrings missing

Finding ID: solana_inflation_audit#12

Severity: **Informational**

Status: **Remediated**

Description

Many public methods lack docstrings, and preferably, most if not all private methods should have a comment explaining the purpose of the code.

Proof of Issue

General observation

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Go over the codebase and create a project goal to keep the number of documented functions, both private and public, as high as possible. The number should at least be more than 85%.

References

N/A

3.13 Code documentation standard

Finding ID: solana_inflation_audit#13

Severity: **Informational**

Status: Remediated

Description

Inconsistencies with some comments having an initial uppercase letter and others having an initial lowercase letter.

Proof of Issue

General observation

Filename: N/A

Beginning Line Number: N/A

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Implement a way to incentivize the developers to create good and consistent documentation.

References

N/A

3.14 Incomplete comment

Finding ID: solana_inflation_audit#14

Severity: **Informational**

Status: **Remediated**

Description

Incomplete comment: "Maximum number of credits history… smaller numbers makes" for variable MAX_EPOCH_CREDITS_HISTORY: usize = 64;

Proof of Issue

```
// Maximum number of credits history to keep around
//  smaller numbers makes
pub const MAX_EPOCH_CREDITS_HISTORY: usize = 64;
```

Filename: programs\vote\src\vote_state\mod.rs

Beginning Line Number: 31

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Implement a way to incentivize the developers to create good and consistent documentation.

References

N/A

3.15 Spellcheck code

Finding ID: solana_inflation_audit#15

Severity: **Informational**

Status: **Remediated**

Description

Spelling error in comment "Repeat somewhat large number… randamly…" - randomly

Proof of Issue

```
#[test]
fn test_bank_update_rewards_determinism() {
    // The same reward should be distributed given same credits
    let expected_capitalization = do_test_bank_update_rewards_determinism();
    // Repeat somewhat large number of iterations to expose possible different behavior
    // depending on the randamly-seeded HashMap ordering
    for _ in 0..30 {
        let actual_capitalization = do_test_bank_update_rewards_determinism();
        assert_eq!(actual_capitalization, expected_capitalization);
    }
}
```

Filename: runtime\src\bank.rs

Beginning Line Number: 5115

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Implement a way to incentivize the developers to create good and consistent documentation.

References

N/A

3.16 Unclear naming of function

Finding ID: solana_inflation_audit#16

Severity: **Informational**

Status: **Remediated**

Description

The variable "vote_balance_and_staked" would be easier to understand if renamed to "previous_lamports_balance", indicating that it functions as a snapshot of the vote & stake balance before the rewards are applied.

Proof of Issue

```
let vote_balance_and_staked = self.stakes.read().unwrap().vote_balance_and_staked();
```

Filename: runtime/src/bank.rs

Beginning Line Number: 918

Severity and Impact Summary

Clarity for the developers maintaining and extending the code base.

Recommendation

Rename the function to reflect the intended use

References

N/A

APPENDIX A: ABOUT KUDELSKI SECURITY

Kudelski Security is an innovative, independent Swiss provider of tailored cyber and media security solutions to enterprises and public sector institutions. Our team of security experts delivers end-to-end consulting, technology, managed services, and threat intelligence to help organizations build and run successful security programs. Our global reach and cyber solutions focus are reinforced by key international partnerships.

Kudelski Security is a division of Kudelski Group. For more information, please visit https://www.kudelskisecurity.com.

Kudelski Security

route de Genève, 22-24

1033 Cheseaux-sur-Lausanne

Switzerland

Kudelski Security

5090 North 40th Street

Suite 450

Phoenix, Arizona 85018

APPENDIX B: DOCUMENT HISTORY

VERSION	STATUS	DATE	AUTHOR	COMMENTS
1.0	Final	19 October 2020	Kudelski Security	

APPENDIX C: SEVERITY RATING DEFINITIONS

Kudelski Security uses a custom approach when determining criticality of identified issues. This is meant to be simple and fast, providing customers with a quick at a glance view of the risk an issue poses to the system. As with anything risk related, these findings are situational. We consider multiple factors when assigning a severity level to an identified vulnerability. A few of these include:

- Impact of exploitation
- Ease of exploitation
- Likelihood of attack
- Exposure of attack surface
- Number of instances of identified vulnerability
- Availability of tools and exploits

SEVERITY	DEFINITION
High	The identified issue may be directly exploitable causing an immediate negative impact on the users, data, and availability of the system for multiple users.
Medium	The identified issue is not directly exploitable but combined with other vulnerabilities may allow for exploitation of the system or exploitation may affect singular users. These findings may also increase in severity in the future as techniques evolve.
Low	The identified issue is not directly exploitable but raises the attack surface of the system. This may be through leaking information that an attacker can use to increase the accuracy of their attacks.
Informational	Informational findings are best practice steps that can be used to harden the application and improve processes.