Shared Memory and Token Swap Code Review

Solana

25 February 2021
Version: 1.2

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

For Public Disclosure

DOCUMENT PROPERTIES

Version:	1.2
File Name:	Research_Report_Shared_Mem_and_Tokenswap_2020_V1.2
Publication Date:	25 February 2021
Confidentiality Level:	For Public Disclosure
Document Owner:	Scott Carlson – Director - Digital Asset Security
Document Recipient:	Solana Foundation
Document Status:	Approved

TABLE OF CONTENTS

TABLE OF FIGURES

EXECUTIVE SUMMARY

Kudelski Security ("Kudelski"), the cybersecurity division of the Kudelski Group, was engaged by the Solana Foundation ("Solana") to conduct an external security assessment in the form of a Shared Memory and Token Swap code review of the Solana Program Library application.

The parts of the Solana Program Library that were to be included in the engagement were

- Share Memory Program – https://github.com/solana-labs/solana-programlibrary/tree/master/shared-memory
- Token-Swap Program – https://github.com/solana-labs/solana-program-library/tree/master/tokenswap

The assessment was conducted remotely by the Kudelski Security. The tests took place from December 1, 2020, to December 23, 2020, and was re-reviewed in late January 2021 to ensure that the remediations have been completed. The review focused on the following objectives:

1. To help the Client to better understand its security posture on the program and identify risks in its application, logic, code, and functionality.
2. To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
3. To identify potential issues and include improvement recommendations based on the result of our tests.

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.

1.1 Engagement Limitations

The engagement was limited to the contents of the Shared Memory and Token-Swap directory in the Solana-Program-Library repository. The final commit included in the audit had the Git identifier: b40e0dd. The engagement was time-boxed to 7 days.

1.2 Engagement Analysis

The Kudelski Security team conducted an audit of Solana Shared-Memory and Token-Swap that consisted of reviewing the logic code.

As a result of our work, we identified **1 High**, **1 Medium**, **1 Low**, and **4 Informational** findings, all which are **remediated** as of the issuance of this report.

Generally, the code base seems to be in good shape, but a lack of documentation for this and related modules makes it harder to understand such things as the assumptions made for cross-program calls.

During the engagement, quick and efficient communication with the development team took place via Slack. The discussions made it possible to verify and verify the various discoveries and quickly determine their severity.

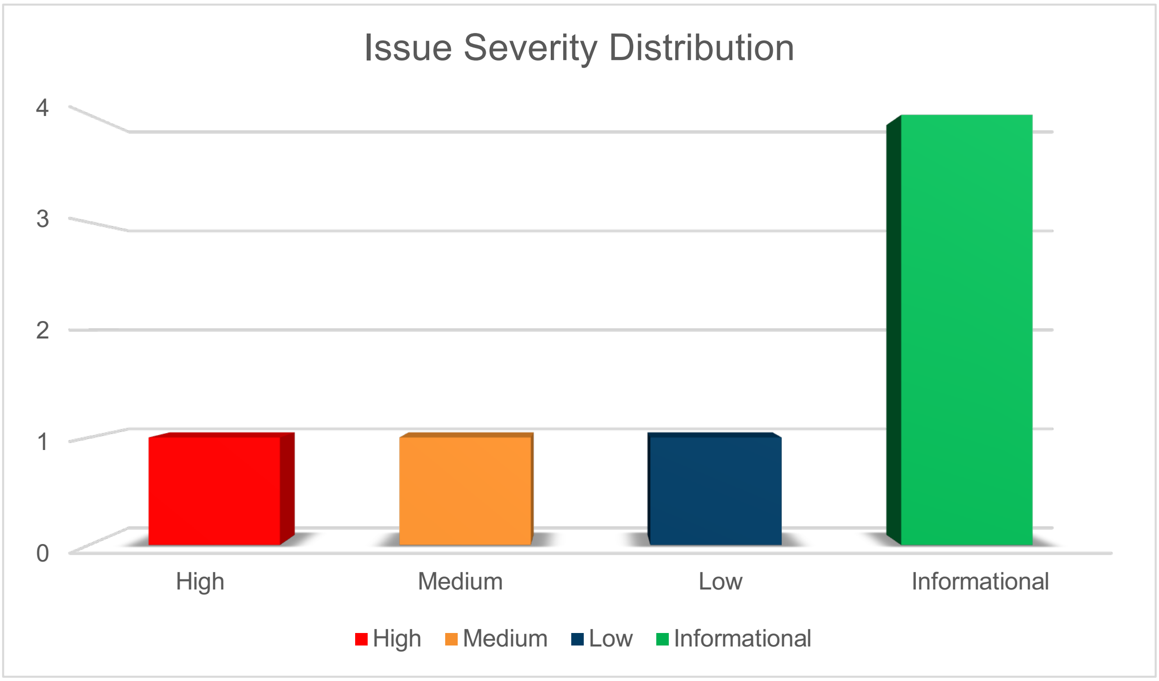

Figure 1 Issue Severity Distribution

1.3 Observations

The code is generally well written and in good shape. The documentation needs to be created and kept up-to-date since comments are sparse.

1.4 Issue Summary List

ID	SEVERITY	FINDING
KS-SHAREDMEM-F-01	Informational	Conversion from u64 to usize
KS-TOKENSWAP-F-01	Informational	Spelling errors.
KS-TOKENSWAP-F-02	Informational	Possible maintainability issues.
KS-TOKENSWAP-F-03	Informational	Duplicate lines in test.
KS-TOKENSWAP-F-04	Low	Missing test.
KS-TOKENSWAP-F-05	High	Attack could possibly fail only half of a swap.
KS-TOKENSWAP-F-06	Medium	Possible fee manipulation by switching curve.

2. METHODOLOGY

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

Kickoff
Ramp-up
Review
Report
Verify

Figure 2 Methodology Flow

2.1 Kickoff

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 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 much 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
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

If present, we analyze the cryptographic primitives and components as well as their implementation. We check 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 remediated.

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 Conversion from u64 to usize

Finding ID: KS-SHAREDMEM-F-01

Severity: **Informational**

Status: **Remediated**

Description

When retrieving the length of the serialized data, several conversions from u64 to usize are made. Working with a 32-bit process would mean that the usize is equal to u32 instead of the expected u64. Since the process only has access to a 32-bit address space, the data object still would not fit in the memory, which renders the issue moot. Regardless the conversion does add unnecessary complexity to the implementation, which reasonably should be remedied.

This issue is only a problem with transactions that are larger than 4 GB in size. The issue would be that the transaction is truncated. If you have a function that is very meticulous in getting all the bits of the transaction, as a hash function, it would be a problem.

Proof of Issue

File path:

shared-memory/program/src/lib.rs line 64 (deserialize_input_parameters)

shared-memory/program/src/lib.rs line 74 (deserialize_input_parameters)

shared-memory/program/src/lib.rs line 102 (entrypoint)

shared-memory/program/src/lib.rs line 106 (entrypoint)

Severity and Impact Summary

Based on further investigation and the Solana development team, we could conclude that this is informational unless the BPF is deployed in a 32-bit environment.

Recommendation

Have the variables be of type u64 instead of usize.

References

Finding ID: KS-TOKENSWAP-F-01

Severity: **Informational**

Status: **Remediated**

There are some misspelled comments in the code.
The following is one example.

Proof of Issue

File path: solana-program-library\token-swap\program\src\curve\calculator.rs

Beginning line number: 11
/// Helper function for calculating swap fee

Severity and Impact Summary

None

Recommendation

Although not a security risk, misspelled or inappropriate words could lead to the wrong advise to the code reader.

References

3.2 Overridden functions

Finding ID: KS-TOKENSWAP-F-02

Severity: **Informational**

Status: **Remediated**

Description

The base CurveCalculator's functions: *owner_withdraw_fee*, *trading_fee*, *owner_trading_fee,* and *host_fee* are overridden in an identical manner in all of its derived implementations.

File path: solana-program-library\token-swap\program\src\curve\constant_product.rs

Beginning line number: 67

File path: solana-program-library\token-swap\program\src\curve\stable.rs

Beginning line number: 166

File path: solana-program-library\token-swap\program\src\curve\flat.rs

Beginning line number: 61

Severity and Impact Summary

Future updates to the code could cause unintended divergence in functionality between the different curves.

Recommendation

Since no other curve type can be used, implementing these functions in the base object would be cleaner and easier to maintain.

3.3 Duplicate test call

Finding ID: KS-TOKENSWAP-F-03

Severity: **Informational**

Status: Status: **Remediated**

Description

Duplicate line in test.

Proof of Issue

Filename:solana-program-library\token-swap\program\src\curve\constant_product.rs

Beginning Line Number: 243

```
check_pool_token_rate(5, 5, 10, Some(2));
check_pool_token_rate(5, 5, 10, Some(2));
```

Severity and Impact Summary

None

Recommendation

Remove line or alter test data.

3.4 Missing tests

Finding ID: KS-TOKENSWAP-F-04

Severity: **Low**

Status: **Remediated**

Description

Curve missing in test.

Proof of Issue

Filename: solana-program-library\token-swap\program\src\processor.rs

Beginning Line Number: 3631

Severity and Impact Summary

Limited impact.

Recommendation

Add test for stable curve.

3.5 Possible partial execution

Finding ID: KS-TOKENSWAP-F-05

Severity: **High**

Status: **Remediated**

Description

It might be possible to set up a contract that will fail a swap or withdraw in the middle of a transaction.

Proof of Issue

Filename: solana-program-library\token-swap\program\src\processor.rs

Beginning Line Number: 317

Severity and Impact Summary

If performed, a withdraw, or swap might end up only doing one transfer.

Recommendation

Make sure that everything is processed in one step on the blockchain.

Remediation: As a remediation to this issue, it appears that the team reworked the whole "nested instruction" idea and now it stops/fails if any of the inner instructions stop/fail and the entire transaction is stopped & rolled-back. This was a major redesign as it now handles all problems from cross-module invocations in the same manner. This code plays an important role and must continue to be managed effectively through updates.

3.6 Curve selection

Finding ID: KS-TOKENSWAP-F-06

Severity: **Medium**

Status: **Remediated**

Description

There are two valid alternatives to the constant product curve that would result in different rates when applied to a swap.

Proof of Issue

File path: solana-program-library\token-swap\program\src\processor.rs

Beginning line number: 253

Severity and Impact Summary

Manipulating the account info in the call and set a flat curve for a large transaction

could be beneficial for the caller.

Recommendation

Remove unused curves.

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 is 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
0.93	Final Draft	8 February 2021	Mikael Bjorn, Scott Carlson	Final Review
1.1	Final	8 February 2021	Mikael Bjorn, Scott Carlson	
1.2	For Public Disclosure	25 Feb 2021	Scott Carlson Scottj.carlson@kudelskisecurity.com	

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.