Spectrum Protocol
Audit Report

Prepared for Spectrum Protocol, 4th February 2023

Introduction

SCV was engaged by Spectrum to assist in identifying security threats and vulnerabilities that have the potential to affect their security posture. Additionally, SCV will assist the team in understanding the risks and identifying potential mitigations.

Scope

SCV performed the security assessment on the following codebase:

- https://github.com/spectrumprotocol/spectrum-plus
- Code Freeze: `44134c570e72f5da7dfa7c9891ab31eb4860ab6b`

Remediations were reviewed by SCV from the following codebase:

- https://github.com/spectrumprotocol/spectrum-plus/tree/fix/audit
- Code Freeze: `6cf26c222054d242673e056e89ebc28fda54d8d3`

Methodologies

SCV performs a combination of automated and manual security testing based on the scope of testing. The testing performed is based on the extensive experience and knowledge of the auditor to provide the greatest coverage and value to Spectrum Protocol. Testing includes, but is not limited to, the following:

- Understanding the application and its code base purpose;
- Deploying SCV in-house tooling to automate dependency analysis and static code review;
- Analyse each line of the code base and inspect application security perimeter;
- Review underlying infrastructure technologies and supply chain security posture;

Code Criteria and Test Coverage

This section below represents how *SUFFICIENT* or *NOT SUFFICIENT* each code criteria was during the assessment

Criteria	Status	Notes
Provided Documentation	**SUFFICIENT**	High level documentation was provided, but the project lacked technical documentation.
Code Coverage Test	**SUFFICIENT**	79.94% coverage, 3981/4980 lines covered
Code Readability	**SUFFICIENT**	N/A
Code Complexity	**SUFFICIENT**	N/A

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	Attackers can exploit transfer_internal to create bond	**Critical**	**Resolved**
2	Setting minimum and maximum debt ratio to 1 would cause error in the get_target_borrow_value function	**Low**	**Resolved**
3	execute_update_uncollateralized_loan_limit is lacking limit validations	**Low**	**Acknowledged**
4	Deadcode which always fails when called	**Informational**	**Acknowledged**
5	Replace Magic Numbers	**Informational**	**Resolved**

Detailed Vulnerabilities

1 – Attackers can exploit `transfer_internal` to create bond shares

Risk Impact: Critical - **Status**: Resolved

Description

The `transfer_internal` function in `spectrum-plus/contracts/astroport_farm/src/cw20.rs:51-52` does not account for the case when a user specifies themself as a recipient. As a result, the `sender_addr` and `rcpt_addr` are the same. This will cause the `REWARD.save` operation in `line 52` to overwrite the old storage stored in `line 51`, causing the user to end up with more funds than intended.

Please see the test case in the following link to reproduce the issue.

Recommendations

We recommend adding a validation to ensure that the `sender_addr` and the `recipient` are not the same address.

2 – Setting minimum and maximum debt ratio to 1 would cause error in the `get_target_borrow_value` function

Risk Impact: Low - **Status**: Resolved

Description

The minimum and maximum debt ratio in `spectrum-plus/contracts/borrowed_farm/src/bond.rs:53-54` are both deducted by `Decimal::one()`. After that, the remaining value is multiplied and assigned to the `denom` variable, which will be used as the denominator in `line 54` when performing division.

The issue arises where if both minimum and maximum debt ratio is configured to `Decimal::one()`, the resulting value will become `0`, causing a division by zero error in the `get_target_borrow_value` function.

Recommendations

Consider ensuring both minimum and maximum debt ratio is not configured to `Decimal::one()`.

3 - `execute_update_uncollateralized_loan_limit` is lacking limit validations

Risk Impact: Low - **Status**: Acknowledged

Description

The `execute_update_uncollateralized_loan_limit` function in `spectrum-plus/contracts/lending_bank/src/contract.rs:501` allows the owner to update the uncollateralized loan limit by a specified amount. This value is saved before confirming that the contract has enough balance to cover the limit amount.

Recommendations

We recommend adding a validation to ensure that the uncollateralized loan limit specified is within the amount of liquidity the contract has.

Revision Notes

This is by design as the loan limit will be configured beforehand. This number can be higher than the current available liquidity in the lending pool.

4 – Deadcode which always fails when called

Risk Impact: Informational - **Status**: Acknowledged

Description

In `spectrum-plus/contracts/astroport_farm/src/state.rs:101-106`, the `POOL_INFO` is loaded from storage. However, the storage value is never stored. This would cause the transaction to always revert because of storage value not found.

With that said, the current related code is dead code because the if conditions will never be satisfied.

Recommendations

Consider removing the deadcode or perform storage save operation for `POOL_INFO`.

Revision Notes

Team advised that the code must be kept for backward compatibility.

5 – Replace Magic Numbers

Risk Impact: Informational - **Status**: Resolved

Description

Throughout the codebase, magic numbers are used. Magic numbers are hard-coded numbers without context. The use of magic numbers reduces the readability and maintainability of the codebase

Instances of magic numbers are listed below:

- `spectrum-plus/contracts/generator_proxy/src/bond.rs:227`

Recommendations

We recommend replacing the magic numbers mentioned above with a constant that is descriptive of its value and use case.

Document control

Version	Date	Approved by	Changes
0.1	26/01/2023	Vinicius Marino	Document Pre-Release
0.2	30/01 - 04/02	SCV-Team	Revisions
1.0	04/02/2023	Vinicius Marino	Document Release

Appendices

A. Appendix – Risk assessment methodology

A qualitative risk assessment is performed on each vulnerability to determine the impact and likelihood of each.

Risk rate will be calculated on a scale. As per criteria Likelihood vs Impact table below:

	Rare	Unlikely	Possible	Likely
Critical	**Medium**	**Severe**	**Critical**	**Critical**
Severe	**Low**	**Medium**	**Severe**	**Severe**
Moderate	**Low**	**Medium**	**Medium**	**Severe**
Low	**Low**	**Low**	**Low**	**Medium**
Informational	**Informational**	**Informational**	**Informational**	**Informational**

LIKELIHOOD

- Likely: likely a security incident will occur;
- Possible: It is possible a security incident can occur;
- Unlikely: Low probability a security incident will occur;
- Rare: In rare situations, a security incident can occur;

IMPACT

- Critical: May cause a significant and critical impact;
- Severe: May cause a severe impact;
- Moderate: May cause a moderated impact;
- Low: May cause low or none impact;
- Informational: May cause very low impact or none.

B. Appendix – Report Disclaimer

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts SCV-Security to perform a security review. The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The content of this audit report is provided “as is”, without representations and warranties of any kind, and SCV-Security disclaims any liability for damage arising out of, or in connection with, this audit report.