Alpha Homora V2 Report

Prepared for Alpha Venture DAO, 2 May 2022

Table of Contents

Introduction

SCV was engaged by Alpha Venture DAO 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/alpha-contracts *e6f0d850cec624a6b20649cdcb175c9d8737ea94*

Vulnerabilities were remediated by Alpha Venture DAO in several commits hashes.

Additonal information details can be found in each findings technical notes section.

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 Alpha Venture DAO. 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

SCV used a scale from **0** to **10** that represents how **SUFFICIENT(6-10)** or **NOT SUFFICIENT(0-5)** each code criteria was during the assessment:

Criteria	Status	Scale Range	Notes
Provided Documentation	**Sufficient**	6-7	N\A
Code Coverage Test	**Sufficient**	8-9	N\A
Code Readability	**Sufficient**	7-8	N\A
Code Complexity	**Sufficient**	6-7	N\A

Vulnerabilities Summary

	Title and Summary	Risk	Status
1	Spot price feed allows potential manipulation and inconsistent results for low liquidity pairs	**Medium**	**Remediated**
2	Extra funds sent are lost	**Low**	**Remediated**
3	execute_liquidate will panic if liquidator pays in excess of debt amount	**Low**	**Remediated**
4	Use of common variable name may impact maintainability and readability	**Informational**	**Remediated**
5	UserPositionResponse contains misleading health_status field	**Informational**	**Remediated**

Detailed Vulnerabilities

1. Spot price feed allows potential manipulation and inconsistent results for low liquidity pairs

Likelihood	Impact	Risk
Unlikely	Moderate	**Medium**

Notes

Feature was removed. Remediation was applied in the *4f4bbd162253717bb84088a6ce06ce93788ab3d4* git commit. Alpha Venture DAO team advises that Spot Price would only be used for testing and consequentially, this issue were downgrade from *Severe* to *Moderate*.

Description

Low liquidity assets with spot price sources may be subject to manipulation and/or miscalculations. The spot price feed uses a Terraswap simulation to determine the spot price. This is problematic because some Terraswap assets have low liquidity and are subject to manipulation. An attacker could cause an imbalance in the pool either with a flash loan or a large sum of funds.

In addition, the use of a `DEFAULT_PROBE_AMOUNT` is problematic. The probe amount is 1,000,000 which may introduce a large spread into the terraswap SimulationResponse depending on the amount of assets in the pool. The spread amount is not checked before returning the spot price result. This would result in a lower return_amount that does not correctly represent the asset's true spot price.

Recommendations

We recommend removing the spot price feed or restricting it's use to very specific pairs with deep liquidity.

2. Extra funds sent are lost

Likelihood	Impact	Risk
Possible	Low	**Low**

Notes

Remediation was applied in the *856b59ae552a0940904ba358fc74888a4da3dab4* git commit hash.

Description

The function `get_denom_amount_from_coins` in `alpha-contracts-master/contracts/lending-bank/src/contract.rs:1262` does not properly handle the case where a user may send multiple funds which may lead to a loss of additional fund that are not the specified denom. `info.funds` is a vector of Coin, but the current implementation only accounts for the specific denom used by the contract. It is best practice to provide an error if funds are sent that the contract does not handle. This function is called for both DepositNative and RepayNative messages, both of which should explicitly accept one native token.

Recommendations

We recommend updating the previously mentioned occurrences to ensure that the length of info.funds is equal to one and returning an error if this condition is not true.

3. execute_liquidate will panic if liquidator pays in excess of debt amount

Likelihood	Impact	Risk
Possible	Low	**Low**

Notes

Remediation was applied in the *3a55d249628963e23bd80d107c9fcd95d28b5978* git commit and severity downgraded to *Low*.

Description

The `execute_liquidate` function in `alpha-contracts-master/contracts/leveraged-farm/src/health.rs:192` does not properly handle liquidation events where the liquidator sends funds in excess of the debt position they are attempting to liquidate. More importantly, excess repayment will actually result in an underflow due to an unchecked subtraction when deducting excess debt share.

It is important to note that while compiling in release mode, rust does not include checks for integer overflow/underflow. However, the Alpha team has explicitly added overflow-checks to the release profile which will result in a panic rather than a wrapping integer overflow/underflow. Excess payment is a relatively common occurrence, so this will ultimately provide poor user experience.

While the this issue currently results in a panic, if the release profile were to be updated and overflow-checks was removed, it would result in state inconsistencies pertaining to the user's debt share.

Recommendations

We recommend including functionality to calculate the excess liquidation amount and refunding that amount to the liquidator. This can be accomplished in a similar manner to how excess repayment is handled in the lending-bank.

4. Use of common variable name may impact maintainability and readability

Likelihood	Impact	Risk
Possible	Informational	**Informational**

Notes

Remediation was applied in the *2ec1cc38ac4c2eedd01cb254f1da3a87b96108f1* git commit.

Description

In Homora-bank, there are multiple occasions where the variable name state is used to represent `ASSET_STATE`. While this is valid, it is best practice to reserve the use of state to only be used to represent contract state. This may impact the maintainability and readability of the contract.

Recommendations

We recommend using a more descriptive variable name to represent `ASSET_STATE`.

5. UserPositionResponse contains misleading health_status field

Likelihood	Impact	Risk
Possible	Informational	**Informational**

Notes

Remediation was applied in the *df4ccb8ee7da3cfb92fde8e7de1c1145bf4861d6* git commit.

Description

The UserPositionResponse contains a misleading `health_status` field. If the user has debt positions the UserHealthStatus will always return `Borrowing(Decimal::one())`. While this does not impact the functionality of the lending bank, in the future the caller may inadvertently mistake this health status value for a true health status value. It seems as though this field could be more accurately labeled as debt_status or something similar.

Recommendations

We recommend updating the UserHealthStatus, UserPosition, UserPositionResponse in the Lending Bank to a more accurate name to describe the position's debt/borrowing status rather that using the word health which is not entirely accurate and mislead users.

Document control

Document changes

Version	Date	Name	Changes
0.1	2022-04-18	Vinicius Marino	Initial report
0.2	2022-04-19	Vinicius Marino	Team communication and Pre-Release
1.0	2022-05-02	Vinicius Marino	Final Report Release

Document contributors

Name	Role	Email address
Vinicius Marino	Security Specialist	vini@scv.services

Appendices

Appendix A: Report Disclaimer

Appendix B: 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:

Likelihood / Impact	Rare	Unlikely	Possible	Likely
Critical	**Medium**	**High**	**Critical**	**Critical**
Severe	**Low**	**Medium**	**High**	**High**
Moderate	**Low**	**Medium**	**Medium**	**High**
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.

SCV