TFM - Limit Order Contract - Audit Report

Prepared for TFM, 1 August 2022

Table of Contents

Introduction

SCV was engaged by TFM 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/tfm-com/audit_limit_order

Code freeze hash: *4218fd74aae15072f72b8ac6e567d712706d60e2*

Remediations were applied into the following commit:

- *2eb49056435535b2412f860ac7da3a0a098a5a4a*

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 TFM. 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**	6-7	N/A
Code Readability	**Sufficient**	7-8	N/A
Code Complexity	**Sufficient**	6-7	N/A

Vulnerabilities Summary

	Title and Summary	Risk	Status
1	Any caller may cancel an order they do not own	**High**	**Remediated**
2	Recipient of withdraw_fees lacks address validation	**Medium**	**Remediated**
3	Multiple Addresses lacking validation	**Low**	**Remediated**
4	Submit_order does not prevent orders with 0 amount	**Low**	**Remediated**
5	Additional funds sent to the contract are lost	**Informational**	**Remediated**
6	Remove unused duplicate struct and enum	**Informational**	**Remediated**
7	info.sender validation is unnecessary and can be removed	**Informational**	**Remediated**

Detailed Vulnerabilities

1. Any caller may cancel an order they do not own

Likelihood	Impact	Risk
Likely	Severe	**High**

Description

The *cancel_order* function currently allows any address to cancel an order with an *expiration_time* of *0*. *contracts/limit_order/src/order.rs:96-98* is a nested conditional that uses the *auth_flag* and sets it to *true*, even though the condition is not auth related. This will effectively override the sender check on lines *93* and *94* and will allow any sender to call this functionality even if they don't own the order so long as the order has an *expiration_time* of *0*.

We classify this issue as severe because it does not result in the loss of user funds, but it exposes functionality that can be used to maliciously gain an advantage by canceling other users' orders.

Recommendations

We recommend updating the validation steps on lines *contracts/limit_order/src/order.rs:93-99* to not be nested and to create a separate boolean value to assess the order's expiration time rather than using the same *auth_flag* which may overwrite the *info.sender* check. Similar to the *expiration_time* checks performed in *submit_order* and *execute_order* that explicitly check to ensure the value is not zero before determining that the order is expired.

2. Recipient of withdraw_fees lacks address validation

Likelihood	Impact	Risk
Possible	Moderate	**Medium**

Description

The *withdraw_fees* function in *contracts/limit_order/src/utils.rs:190* does not perform any address validation on the *to* field. While *to* is defined as an *addr* type, it needs to be validated using by calling *deps.api.addr_validate* to validate that the address is valid, normalized, and in lowercase form. If the contract's owner were to specify an incorrect or invalid address, it may result in the loss of protocol fees or a failed execution of the *WithdrawFees* message. Another way to improve this functionality is to add a validated config address value that is designated to receive fees rather than specifying a unique address for each *WithdrawFees* message.

We classify this finding as major severity because while it can result in the loss of protocol fees, it is only callable by the contract's owner.

Recommendations

We recommend validating the *to* address with *addr_validate* function. In addition, we recommend adding a config value for the fee recipient rather than specifying a *to* address in each *WithdrawFees* message.

3. Multiple Addresses lacking validation

Likelihood	Impact	Risk
Unlikely	Low	**Low**

Description

Throughout the codebase there are addresses that are not validated before being stored or set. In CosmWasm *Addr* must always contain valid addresses, and the caller is responsible for ensuring the input is valid before storing or setting the address value.

The following are instances of the lacking of address validation within the codebase:

- *router (contracts/limit_order/src/utils.rs:152)*
- *proposed_owner (contracts/limit_order/src/utils.rs:23)*

We have separated these address validations from the missing address validation in *withdraw_fees* due to their impact and recoverability. Both of the addresses mentioned above would impact the functionality of the contract but can be easily remedied by the owner by updating the values.

Recommendations

We recommend validating the addresses with *addr_validate* function.

4. Submit_order does not prevent orders with 0 amount

Likelihood	Impact	Risk
Unlikely	Low	**Low**

Description

The *submit_order* function in *contracts/limit_order/src/order.rs:13* does not properly validate that the *offer_asset.amount* is greater than *0*. This value should be checked to prevent spam orders from being added to *ORDERS*.

This issue also exists for native tokens. *assert_sent_native_token_balance* in *packages/tfm/src/asset.rs:89-94* does not return an error when a zero amount is submitted.

Recommendations

We recommend validating that *offer_asset.amount* is greater than zero before storing the order with *store_new_order*.

5. Additional funds sent to the contract are lost

Likelihood	Impact	Risk
Unlikely	Informational	**Informational**

Description

The *assert_sent_native_token_balance* function in *packages/tfm/src/asset.rs:79* does not ensure that multiple funds are not sent, and any additional native tokens are not returned to the user, so they will be stuck in the contract forever.

Recommendations

We recommend checking that the transaction contains only the expected Coin using:

- https://docs.rs/cw-utils/latest/cw_utils/fn.must_pay.html.

6. Remove unused duplicate struct and enum

Likelihood	Impact	Risk
Unlikely	Informational	**Informational**

Description

The file *packages/tfm/src/asset.rs* contains an unused struct and enum that may become confused with *AssetInfo* and *Asset*. *asset* and *assetInfo* appear to implement the same functionality as *Asset* and *AssetInfo* but are not used anywhere within the codebase. This may impact the future readability and maintainability of the codebase if the duplicate data structures were to be utilized unintentionally.

Recommendations

We recommend removing the duplicate data structures or clearly defining their purpose and providing clear guidance on their use in the code comments.

7. info.sender validation is unnecessary and can be removed

Likelihood	Impact	Risk
Unlikely	Informational	**Informational**

Description

In *contracts/limit_order/src/order.rs:48 info.sender* is validated using *addr_validate*. This function call can be removed as *info.sender* does not need the additional validation or normalization provided by the *addr_validate* function.

Recommendations

We recommend removing the validation on *contracts/limit_order/src/order.rs:48*.

Document control

Document changes

Version	Date	Name	Changes
0.1	2022-07-22	Vinicius Marino	Initial report
0.2	2022-07-22	Vinicius Marino	Team communication and Pre-Release
1.0	2022-08-01	Vinicius Marino	Final Revision and Document Release

Document contributors

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

Appendices

Appendix A: Report Disclaimer

The content of this audit report is provided “As is”, without representations and warranties of any kind.

The author and their employer disclaim any liability for damage arising out of, or in connection with, this audit report.

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:

Impact \ Likelihood	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