Eris Protocol amp-compounder contracts Audit Report

Prepared for Eris Protocol, 23rd February 2023

Table of Contents

Introduction

SCV was engaged by Eris Protocol 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/erisprotocol/contracts-terra/tree/develop/contracts/amp-compounder/astroport_farm
- https://github.com/erisprotocol/contracts-terra/tree/develop/contracts/amp-compounder/compound_proxy
- *Code Freeze:* `db2b05a2183d30f9d65790e2179ce79c7bbd56ad`

SCV notes that integrations with Astroport contracts were out-of-scope.

Revisions were applied by the team and revised by SCV up to the following hash:

- https://github.com/erisprotocol/contracts-terra/tree/feature/audit-scv
- *Code Freeze:* `18986f68fb7f77d1c9b61a417df21579d6a45b40`

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 Eris 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**	Documentation was available in the following https://docs.erisprotocol.com
Code Coverage Test	**SUFFICIENT**	N/A
Code Readability	**SUFFICIENT**	N/A
Code Complexity	**SUFFICIENT**	N/A

Threat Modelling

The goal of threat modelling is to identify and evaluate potential threats to a system or application and to develop strategies to mitigate or manage those threats. Threat modelling is an important part of the software development life cycle, as it helps developers and security professionals to proactively identify and address security risks before they can be exploited by attackers.

The main objectives of threat modelling includes (not limited to) the following :

- **Identify threats**: The first objective of threat modelling is to identify potential threats that could affect the security posture of the underlying smart contracts or application. This can include threats from external attackers, internal actors, or even accidental events that could happen.
- **Evaluate risks:** Once potential threats have been identified, the next objective is to evaluate the risks associated with each threat. This involves assessing the likelihood of each threat occurring and the potential impact it could have overall.
- **Mitigation strategies:** After identifying potential threats and evaluating the associated risks, the next objective is to develop strategies to mitigate or reduce the impact of threats. This can include implementing technical controls, such as access controls or further security measures around developing policies and procedures to reduce the likelihood or impact of a threat.
- **Communicate findings:** The final objective of threat modelling is to communicate the findings and recommendations to relevant stakeholders, such as developers, security teams, and management. This helps ensure that everyone involved in the development and maintenance understands the potential risks and the best strategies for addressing them.

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	Malicious actors can skim other user's rewards by staking before the `Compound` message	**Severe**	**Resolved**
2	Sandwich attack in compound proxy contract	**Severe**	**Resolved**
3	Funds inside compound proxy contract can be stolen	**Medium**	**Partially Resolved**
4	Duplicate `LpInit` during `add_lp` causes overwrite operations	**Low**	**Resolved**
5	`wanted_token` is not validated to be one of the pair token's assets	**Low**	**Resolved**
6	Slippage tolerance limit is not enforced for Astroport	**Low**	**Resolved**
7	`query_compound_simulation` does not behave the same during execution mode	**Low**	**Resolved**
8	Outstanding TODO comments in the codebase	**Informational**	**Resolved**
9	Events are not emitted for important executions	**Informational**	**Resolved**
10	Remove unused code	**Informational**	**Resolved**
11	`expires_in` is not validated when proposing a new owner	**Informational**	**Resolved**
12	Inconsistent naming for attribute event	**Informational**	**Resolved**
13	`simulate` function can use pair's `QueryMsg::Simulation` query instead	**Informational**	**Acknowledged**

Detailed Vulnerabilities

1 – Malicious actors can skim other user's rewards by staking before the `Compound` message

Risk Impact: Severe - **Status**: Resolved

Description

In `contracts/amp-compounder/astroport_farm/src/contract.rs:107`, the `Compound` message is configured to be executed by the controller bot. Once the compound is finished, rewards are distributed to all stakers. The number of rewards that can be compounded is generated by the duration of the liquidity pool tokens staked, as seen in line `34`.

However, no limitation ensures the user needs to stake a minimum duration to receive compounded rewards. Since the controller bot is configured to compound the rewards daily, the actor can stake the liquidity pool tokens for a few seconds before executing the compound message, causing them to receive a part of the rewards.

This is unfair toward other users because the rewards come from the liquidity pool tokens staked by them. The actor's tokens staked for a few seconds do not contribute to the overall compounded rewards. After receiving the rewards, the malicious actor can unstake their tokens and wait for the next `Compound` execution.

When the actor's liquidity pool tokens are not staked, they can use them for other profit purposes, such as depositing them into Astroport Generator to receive the rewards individually.

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

Recommendations

Consider enforcing a minimum staking duration before allowing users to receive compounded rewards.

Revision Notes

The `deposit_profit_delay_s` was implemented which set a higher rate than the compounding interval. When depositing, the user will receive relative to the tracked exchange rate shares (`exchange_rate`) .

2 – Sandwich attack in compound proxy contract

Risk Impact: Severe - **Status**: Resolved

Description

The `Compound` and `MultiSwap` messages in the compound proxy contract interact with DEX to swap or provide liquidity. However, the max default spread for both operations is hardcoded to 50%.

As a result, this allows attackers to perform a sandwich attack and cause a loss of funds to the user.

Affected code lines:

- `contracts/amp-compounder/compound_proxy/src/execute.rs:60`
- `contracts/amp-compounder/compound_proxy/src/execute.rs:69`
- `contracts/amp-compounder/compound_proxy/src/execute.rs:154`
- `contracts/amp-compounder/compound_proxy/src/execute.rs:163`
- `contracts/amp-compounder/compound_proxy/src/execute.rs:316`
- `contracts/amp-compounder/compound_proxy/src/execute.rs:345`

Recommendations

Consider setting the slippage to a lower value (e.g., 5% or 10%) or allowing the contract admin to configure default slippage.

3 – Funds inside compound proxy contract can be stolen

Risk Impact: Medium - **Status**: Partially Resolved

Description

In `contracts/amp-compounder/compound_proxy/src/execute.rs:146`, the `MultiSwap` message allows the user to provide a target asset to swap to (see `into` variable). If the deposited asset is the target asset, the `SendSwapResult` callback will be executed to send all funds back to the user. This allows anyone to steal all funds from the contract.

Normally, this is fine as long as the compound proxy contract does not intend to hold any funds. However, this does not seem to be the case.

The `ProvideLiquidity` callback accepts a `prev_balances` parameter which records the previous balance of the assets held by the contract. The excess balance is then calculated in line `381` and used to provide liquidity in line `401`. This means that the contract is intended to hold funds that can be stolen by anyone.

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

Recommendations

Consider following the recommendations below:

- Remove the `prev_balances` asset vector from the `ProvideLiquidity` callback and send all funds to provide liquidity.
- Properly documenting the compound proxy will not hold any funds to educate users.

Revision Notes

The client documented that the compound proxy contract is not intended to hold any funds. The `prev_balances` asset vector remains.

SCV notes that the risks were downgraded from *Severe* to *Medium* as no funds are expected to be available at the proxy contract by design.

4 – Duplicate `LpInit` during `add_lp` causes overwrite operations

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

Description

During the contract instantiation phase in `contracts/amp-compounder/compound_proxy/src/contract.rs:41-43`, if duplicate `lp_init.pair_contract` keys are provided in the `msg.lps` vector, the former configurations would be overwritten, causing only the last key to be saved.

This issue is also present during the configuration update phase in `contracts/amp-compounder/compound_proxy/src/execute.rs:498-502`.

As a result, the contract owner might misconfigure the `LpConfig` struct with different `commission_bps`, `slippage_tolerance`, and `wanted_token` values than intended.

Recommendations

Consider deduping the `msg.lps` vector to prevent misconfiguration.

5 – `wanted_token` is not validated to be one of the pair token's assets

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

Description

When configuring liquidity pools in `contracts/amp-compounder/compound_proxy/src/state.rs:170`, the provided `wanted_token` value is not validated to be one of the assets from `lp_config.pair_info.asset_infos`.

This is problematic since the reward assets sent by the user in the `compound` function in `contracts/amp-compounder/compound_proxy/src/execute.rs:53-80` are needed to be swapped into the `wanted_token`. Subsequently, the assets are utilized to provide liquidity to pair contracts.

If the wanted token is not one of the assets to provide liquidity, the funds will not return to the user, causing a loss of funds.

Recommendations

Consider ensuring that the `wanted_token` is one of the pair contract's asset info during the `add_lp` function.

6 – Slippage tolerance limit is not enforced for Astroport

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

Description

In `contracts/amp-compounder/compound_proxy/src/state.rs:166-169`, when adding a new `LpConfig`, the `slippage_tolerance` value is only validated to be lower than 100%. While Terraswap's max slippage tolerance is 100%, the maximum amount of slippage tolerance for Astroport is only 50%.

As a result, configuring the slippage tolerance value to be higher than 50% would cause Astroport's provide liquidity execution to fail.

Recommendations

We recommend ensuring slippage tolerance is configured to a value lower or equal to 50%.

7 - `query_compound_simulation` does not behave the same during execution mode

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

Description

The `query_compound_simulation` function in `contracts/amp-compounder/compound_proxy/src/simulation.rs:23` is utilized to provide a simulated amount of liquidity pool token based on the given rewards, mimicking the `Compound` message execution. Due to this, ensuring the query and `Compound` message return the same token amount when executed with the same arguments is important. However, this is not the case.

Firstly, in line `36`, the `rewards` assets vector is not deduplicated to ensure that all assets are unique compared to the `assert_uniq_assets` function in `contracts/amp-compounder/compound_proxy/src/execute.rs:34`.

Secondly, the query mentions that `PairType::Custom` is not supported as seen in line `197`. Nonetheless, the `Compound` message allows this as long the Custom pair holds two assets and the contract owner configures them as an allowed liquidity pool.

Lastly, the query does not deduct `MINIMUM_LIQUIDITY_AMOUNT` from the `lp_amount` for `PairType::Xyk` and `PairType::Stable` pair types in lines `109` and `148`. The minimum liquidity amount must be deducted if the pair contract originates from Astroport's `pair` and `pair_stable` contract.

Recommendations

Consider applying the following recommendations:

- Dedupe `rewards` asset vector in line `36`.
- Ensure the `pair_info.pair_type` value is not a `Custom` pair type in the `add_lp` function.
- Deduct `MINIMUM_LIQUIDITY_AMOUNT` from the `lp_amount` for `PairType::Xyk` and `PairType::Stable` pair types.

8 – Outstanding TODO comments in the codebase

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

Description

The codebase contains several outstanding TODO comments, which can signify potential programming or architectural issues that have not yet been resolved.

- `contracts/amp-compounder/astroport_farm/src/bond.rs:121`

Recommendations

Consider resolving or removing the aforementioned comments.

9 – Events are not emitted for important executions

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

Description

During the contract instantiation and configuration update phase, there are no events or attributes are emitted for the compound proxy contract as seen in `contracts/amp-compounder/compound_proxy/src/execute.rs:49` and `520`.

As contract instantiation and updating configurations are important executions, relevant events and attributes should be emitted for off-chain listeners to record and index the configured parameters.

Recommendations

Consider emitting relevant attributes or events when updating the compound proxy contract configuration based on configured parameters.

10 – Remove unused code

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

Description

Several instances of unused code can be found throughout the codebase, which impacts the readability.

- `contracts/amp-compounder/compound_proxy/src/contract.rs:62`
- `contracts/amp-compounder/compound_proxy/src/contract.rs:138-166`
- `contracts/amp-compounder/compound_proxy/src/execute.rs:526-545`

Recommendations

Consider removing the unused code.

11 - `expires_in` is not validated when proposing a new owner

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

Description

When creating a proposal for a new owner in the `propose_new_owner` function in `contracts/amp-compounder/astroport_farm/src/ownership.rs:59`, the `ttl` (time to live) variable lacks input validation.

If the value is set to an extremely small amount, the `claim_ownership` function in `contracts/amp-compounder/astroport_farm/src/ownership.rs:125` will fail. Conversely, if the variable is set to a very high value, the proposal could take a long time to expire or until the `drop_ownership_proposal` function is invoked.

Recommendations

Consider enforcing a minimum and maximum value for the `expires_in` variable.

12 – Inconsistent naming for attribute event

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

Description

In `contracts/amp-compounder/astroport_farm/src/bond.rs:52`, the action attribute emitted is “`bond_assets`,” which is not consistent with its corresponding contract.

Recommendations

Consider modifying “`bond_assets`” to “`ampf/bond_assets`.”

13 – `simulate` function can use pair's `QueryMsg::Simulation` query instead

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

Description

The `simulate` function in `contracts/amp-compounder/compound_proxy/src/execute.rs:436` estimates the swap return amount using custom code.

If there is a calculation modification to the contracts, the `simulate` function will not be updated automatically, requiring a contract migration from the admin.

Recommendations

Consider delegating the `simulate` function to the pair's `Simulation` query instead. For reference, here is Astroport's and Terraswap's simulation query, respectively. This reduces the overall code for the contract, which also increases efficiency.

Document control

Version	Date	Approved by	Changes
0.1	17/02/2023	SCV-Security Team	Document Pre-Release
0.2	22/02/2023	SCV-Security Team	Remediation Revisions
1.0	23/02/2023	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.