AUDIT REPORT

ERIS

Contracts ve3

Prepared by SCV-Security

On 1st August 2024

Table of Contents

Table of Contents **2**
Introduction **3**
Scope Functionality 3
Submitted Codebase 4
Revisions Codebase 4
Methodologies 5
Code Criteria 6
Findings Summary **8**
Findings Technical Details **9**
1. The user index is not updated if the computed reward is zero 9
2. Users might receive more funds due to rounding issue 11
3. Potential incorrect accounting for ASSET_BRIBES 12
4. Updating asset staking configuration may cause stuck funds 13
5. User assets and voting power are lost if merge_lock is called with the same token ID 14
6. Unfair reward distribution to legitimate stakers 15
7. Users can prevent themselves from being blacklisted 16
8. Potential out-of-gas error when checking the user is not blacklisted 17
9. Locks can be created for blacklisted recipients 18
10. Incorrect MIN_LOCK_PERIODS value 19
11. Deposit assets' contract address is not validated 20
12. Messages that do not expect funds to be sent are not validated 21
13. Important event attributes are not emitted 22
14. Crate dependencies do not contain full semver version 23
15. Incorrect NFT name and symbol in voting-escrow 24
Document Control **25**
Appendices **26**
A. Appendix - Risk assessment methodology 26
B. Appendix - Report Disclaimer 27

Introduction

SCV has been engaged by *ERIS* to conduct a comprehensive security review with the goal of identifying potential security threats and vulnerabilities within the codebase. The purpose of this audit is to evaluate the security posture of the codebase and provide actionable recommendations to mitigate any identified risks. This report presents an overview of the findings from our security audit, outlining areas of concern and proposing effective measures to enhance the codebase's security.

Scope Functionality

The `asset-staking` contract implements a subset of the Alliance Protocol functionality and allows users to stake tokens and claim rewards based on their stake. The rewards that users can claim are distributed through take rates and bribes.

The `voting-escrow` contract allows ampLP token holders to stake their tokens to boost their governance and delegation power. Voting power is boosted according to how long someone locks their ampLP. The contract utilizes an NFT collection to manage user lock positions.

The `global-config` contract manages global configuration settings across the protocol for authentication purposes.

Submitted Codebase

asset-staking	
Repository	https://github.com/erisprotocol/contracts-ve3
Commit	6d66c97a7e07ca3f52b1c341b42d5bdb0b1f8161
Contract	asset-staking
Branch	main
voting-escrow	
Repository	https://github.com/erisprotocol/contracts-ve3
Commit	1ac07a5df7814aebfb10f71e6d28434a4bfa0581
Contract	voting-escrow
Branch	main
global-config	
Repository	https://github.com/erisprotocol/contracts-ve3
Commit	acebda438b3847431daf8eb65b55ba6c4a258693
Contract	global-config
Branch	main

Revisions Codebase

asset-staking	
Repository	https://github.com/erisprotocol/contracts-ve3
Commit	c17ed1579299295901ed8fa8dd69d2a2a1dfda2a
Contract	asset-staking
Branch	main
voting-escrow	
Repository	https://github.com/erisprotocol/contracts-ve3

Commit	c17ed1579299295901ed8fa8dd69d2a2a1dfda2a
Contract	voting-escrow
Branch	main
global-config	
Repository	https://github.com/erisprotocol/contracts-ve3
Commit	7ac081ae31b8370997932834313c5e9c91c6f240
Contract	global-config
Branch	main

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*. Testing includes, but is not limited to, the following:

- Understanding the application and its functionality 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

This section provides an evaluation of specific criteria aspects as described below:

- **Documentation:** Evaluating the presence and comprehensiveness of publicly available or provided explanatory information, diagram flowcharts, comments, and supporting documents to enhance code understanding.
- **Coverage:** Evaluating whether the code adequately addresses all necessary cases and scenarios, ensuring that the intended functionality or requirements are sufficiently covered.
- **Readability:** Assessing how easily the code can be understood and maintained, considering factors such as code structure, naming conventions, and overall organisation.
- **Complexity:** Evaluating the complexity of the code, including factors such as, number of lines, conditional statements, and nested structures.

The status of each criteria is categorised as either **SUFFICIENT** or **NOT-SUFFICIENT** based on the audit assessment. This categorisation provides insights to identify areas that may require further attention and improvement.

Criteria	Status	Notes
Documentation	**SUFFICIENT**	The `asset-staking` contract had sufficient inline comments but used the default CosmWasm README. The `voting-escrow` contract had sufficient inline comments and a comprehensive README describing the contract behavior. The `global-config` contract had sufficient inline comments, but the README file is empty.
Coverage	**SUFFICIENT**	Test coverage is considered sufficient, with 88.65% of the code currently covered. The coverage can be improved by adding tests for the `Ve3` stake config in `asset-staking`, as the current tests only include the `Default` and

		`Astroport` stake config scenarios.
Readability	**SUFFICIENT**	The codebase had good readability overall and utilised many Rust and CosmWasm best practices.
Complexity	**SUFFICIENT**	N/A

Findings Summary

Summary Title	Risk Impact	Status
The user index is not updated if the computed reward is zero	SEVERE	RESOLVED
Users might receive more funds due to rounding issue	SEVERE	RESOLVED
Potential incorrect accounting for `ASSET_BRIBES`	SEVERE	RESOLVED
Updating asset staking configuration may cause stuck funds	SEVERE	RESOLVED
User assets and voting power are lost if `merge_lock` is called with the same token ID	SEVERE	RESOLVED
Unfair reward distribution to legitimate stakers	MODERATE	ACKNOWLEDGED
Users can prevent themselves from being blacklisted	MODERATE	ACKNOWLEDGED
Potential out-of-gas error when checking the user is not blacklisted	MODERATE	ACKNOWLEDGED
Locks can be created for blacklisted recipients	MODERATE	RESOLVED
Incorrect `MIN_LOCK_PERIODS` value	MODERATE	RESOLVED
Deposit assets' contract address is not validated	LOW	RESOLVED
Messages that do not expect funds to be sent are not validated	LOW	ACKNOWLEDGED
Important event attributes are not emitted	LOW	ACKNOWLEDGED
Crate dependencies do not contain full semver version	INFO	ACKNOWLEDGED
Incorrect NFT name and symbol in `voting-escrow`	INFO	RESOLVED

Findings Technical Details

1. The user index is not updated if the computed reward is zero

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `_calc_reward_share` function in `contracts/asset-staking/src/contract.rs:528-529` does not set the user index (`USER_ASSET_REWARD_RATE`) to the global index (`ASSET_REWARD_RATE`) if the computed rewards for the user is zero. This is problematic because if the user staked a small amount (1 share) and the difference between `ASSET_REWARD_RATE` and `USER_ASSET_REWARD_RATE` is less than `Decimal::one()` (e.g., a small amount of harvested rewards), the user index will not be updated, allowing the user to stake a huge amount of funds to steal rewards from other users because their reward index uses the old value.

Recommendation

Consider setting the user index to the global index in `contracts/asset-staking/src/contract.rs:529`.

2. Users might receive more funds due to rounding issue

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `unstake` function in `contracts/asset-staking/src/contract.rs:417` computes the number of shares to deduct from the user and returns them the `withdraw_amount` funds amount. If a rounding issue occurred during the calculation in `compute_share_amount`, the amount of the shares returned may be lesser than intended, allowing the user to withdraw more funds at the expense of other users.

Recommendation

Consider applying the `ceil()` function when computing the shares to deduct so users do not receive more funds than intended.

3. Potential incorrect accounting for ASSET_BRIBES

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `_update_asset_config` function in `contracts/asset-staking/src/contract.rs:228` calls `unstake_check_received_msg` to unstake the funds from the old staking contract. If the old staking contract is `Astroport` or `Ve3`, the `track_bribes_callback_msg` function will be called in `packages/ve3-shared/src/stake_config.rs:121-138` to record the contract balance and set the increased them to the `ASSET_BRIBES` state in `contracts/asset-staking/src/contract.rs:760-768`.

If any rewards from `Astroport` or `Ve3` are one of the stake denom (i.e., `AssetInfoWithConfig.info`), the balance will be recorded before the unstake. Once the unstake is completed and funds are sent back, the `track_bribes_callback` function will incorrectly set the unstaked funds as the received rewards, therefore inflating the `ASSET_BRIBES` state.

Additionally, this also causes rewards to not being accounting in `contracts/asset-staking/src/contract.rs:332-346` because the sent amount already increased the contract balance, and was not deducted before querying it. If rewards are being accrued, the `ASSET_BRIBES` state will not take it into account.

Recommendation

Consider validating the reward information from `Astroport` and `Ve3` is not the stake denom (`AssetInfoWithConfig.info`).

4. Updating asset staking configuration may cause stuck funds

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `_update_asset_config` function in `contracts/asset-staking/src/contract.rs:221-229` unstakes the available funds (computed as `TOTAL - AssetConfigRuntime.taken`) and stakes them on the new staking contract. The `AssetConfigRuntime.taken` value represents the fee charged by the protocol and is only distributed when the `distribute_take_rate` function is called to harvest the rewards.

This means that when the available funds are unstaked from the old staking contract, some portion of `AssetConfigRuntime.taken` may be ignored, which makes them remain in the old staking contract. For example, if `AssetConfigRuntime.taken` increases and the `distribute_take_rate` function is not called in time to withdraw them, the increased amount will be stuck in the old staking contract due to the formula of (`balance - current.taken`).

As a result, the `distribute_take_rate` and `unstake` functions may fail due to an insufficient funds error, preventing withdrawal and harvest operations from working correctly.

Recommendation

Consider unstaking the balance and unharvested fees when updating the stake configuration. This can be accomplished by computing the available amount as (`balance - current.harvested`) in `contracts/asset-staking/src/contract.rs:224`.

5. User assets and voting power are lost if `merge_lock` is called with the same token ID

RISK IMPACT: SEVERE	STATUS: RESOLVED

Description

The `merge_lock` function in `contracts/voting-escrow/src/contract.rs:557` allows users to combine two of their vAMP lock positions into a single position. However, no validation prevents a user from passing through the same token ID for the `token_id` and `token_id_add` fields.

As a result, the user's lock position will be erased, removing their voting power and preventing them from withdrawing their assets.

This linked PoC highlights how the user's lock position, voting power, and corresponding NFT are all erased if the same token ID is passed through in a `MergeLock` message.

Recommendation

Consider validating the `token_id` and `token_id_add` fields as not equal when calling `merge_lock`.

6. Unfair reward distribution to legitimate stakers

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that `UpdateRewards` is a gas heavy operation and will be called every couple of hours to keep rewards at a sane amount. They will also be monitoring the txs if this is happening and will make appropriate changes.

Description

The `_calc_reward_share` function in `contracts/asset-staking/src/contract.rs:510` is responsible for computing the user index. However, the `update_rewards` function is not called first to increase the global index. This is important because if there are any existing rewards to be harvested, the global index will increase and distribute rewards to existing stakers who staked across the reward accrual period.

A user can perform a JIT (Just-In-Time) liquidity attack by staking only when there are rewards to be claimed and unstake after claiming them. This is unfair to legitimate stakers because the user can utilize their funds on other DeFi applications for profit instead of staking with other users, resulting in them having an unfair advantage. Ideally, the rewards should be distributed to the existing stakers and not new stakers.

Recommendation

Consider calling `CallbackMsg::UpdateRewards` with the `config.reward_info` balance in `_calc_reward_share` so the global index increases before computing the user index.

7. Users can prevent themselves from being blacklisted

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that the contract will be controlled by a multi-sig, in case of someone wanting to exploit this fact, the contract can be updated to iteratively blacklist tokens of a user.

Description

The `update_blacklist` function in `contracts/voting-escrow/src/contract.rs:1204-1242` iterates over all tokens owned by the user to update the voting powers when blacklisting them. If a user creates many locks via the `create_lock` function, the `update_blacklist` function will fail due to an out-of-gas error when iterating over all the user's locks. As a result, the owner is unable to blacklist the user as part of the protocol's intended behavior.

Recommendation

Consider implementing a maximum amount of locks that can be created for a user.

8. Potential out-of-gas error when checking the user is not blacklisted

RISK IMPACT: MODERATE	STATUS: ACKNOWLEDGED

Revision Notes

The team mentions that the blacklist is expected to be unused or very small.

Description

The `assert_not_blacklisted` function in `contracts/voting-escrow/src/utils.rs:57-63` loads the `BLACKLIST` state and calls the `contains()` function to check whether the user is inside the vector of address. Since the `contains()` function's operation is `O(n)`, an out-of-gas error may occur if there are too many blacklisted users.

Recommendation

Consider modifying the `BLACKLIST` state to use a map instead of a vector so the operation is `O(1)` when checking the user is not blacklisted.

9. Locks can be created for blacklisted recipients

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

The `split_lock` function in `contracts/voting-escrow/src/contract.rs:655` does not check the `recipient` is a blacklisted address. This is required because blacklisted users cannot create locks in the protocol, which is shown in `contracts/voting-escrow/src/contract.rs:483`.

Recommendation

Consider checking the recipient is not a blacklisted user.

10. Incorrect MIN_LOCK_PERIODS value

RISK IMPACT: MODERATE	STATUS: RESOLVED

Description

The `MIN_LOCK_PERIODS` constant in `packages/ve3-shared/src/constants.rs:42` is set to one. This is incorrect because it should be three based on the comment in line 41, which indicates that "Funds need to be at least locked for 3 weeks.".

Recommendation

Consider setting the `MIN_LOCK_PERIODS` constant value to 3 or updating the `MIN_LOCK_PERIODS` docs to reflect that the minimum lock period is 1 week.

11. Deposit assets' contract address is not validated

RISK IMPACT: LOW	STATUS: RESOLVED

Description

The `instantiate` function in `contracts/voting-escrow/src/contract.rs:48-54` does not validate the `AssetInfoConfig::ExchangeRate` contract address in `msg.deposit_assets` is valid. This validation should be performed like `contracts/voting-escrow/src/contract.rs:1343-1347` in `execute_update_config`.

Recommendation

Consider implementing the contract address validation.

12. Messages that do not expect funds to be sent are not validated

RISK IMPACT: LOW	STATUS: ACKNOWLEDGED

Description

Most entry-point messages in the `asset-staking` and `voting-escrow` contracts do not expect funds to be sent along with them. However, only a few of the messages perform validation to ensure that the user only sends the expected number of assets, if any. A user may accidentally send funds along with an executed message that does not expect them and thus lose their funds.

Listed below are the messages where validations should be added to ensure that no unexpected funds are sent:

- `asset-staking`
  - `Unstake`
  - `ClaimReward`
  - `ClaimRewards`
  - `DistributeTakeRate`
  - `DistributeBribes`
  - `WhitelistAssets`
  - `RemoveAssets`
  - `UpdateAssetConfig`
  - `SetAssetRewardDistribution`
- `voting-escrow`
  - `Withdraw`
  - `ExtendLockTime`
  - `LockPermanent`
  - `UnlockPermanent`
  - `MergeLock`
  - `SplitLock`
  - `TransferNft`
  - `SendNft`
  - `Burn`

Recommendation

Consider implementing validation to ensure that no user funds are sent with messages that do not expect them.

13. Important event attributes are not emitted

RISK IMPACT: LOW	STATUS: ACKNOWLEDGED

Description

Some important event attributes are not emitted in the `voting-escrow` and `asset-staking` contracts. Emitting relevant attributes is important in ensuring that the contract is transparent and monitorable.

- `voting-escrow`
  - `instantiate` emits default attributes and does not contain the global config address or deposit assets.
  - `execute_update_config` does not emit any attributes containing the updated fields.
- `asset-staking`
  - `instantiate` does not emit attributes containing the `global_config_addr`, `reward_info`, `default_yearly_take_rate`, or `gauge`.
  - `set_asset_reward_distribution` does not contain the new asset reward distribution.

Recommendation

Consider implementing more descriptive event attributes for the above cases to ensure that the contract is transparent, monitorable, and auditable.

14. Crate dependencies do not contain full semver version

RISK IMPACT: INFORMATIONAL	STATUS: ACKNOWLEDGED

Description

Some crate dependencies in the `contracts-ve3:Cargo.toml` file are only locked to a major version and are not locked to minor or patch versions as well. It is best practice to lock a minor and patch version as well in case any breaking changes are accidentally introduced in any future minor or patch version if the contracts are compiled after the dependencies are updated.

Recommendation

Consider adding a minor and patch version for all crate dependencies in the `contracts-ve3:Cargo.toml` file.

15. Incorrect NFT name and symbol in `voting-escrow`

RISK IMPACT: INFORMATIONAL	STATUS: RESOLVED

Description

The NFT contract symbol and name created when instantiating the `voting-escrow` contract are “veLUNA” and “Vote Escrowed LUNA”. However, based on the `voting-escrow` README file, the NFT contract symbol should be “vAMP”, and the contract name should be “Vote Escrowed (LUNA-ampLUNA) ampLP”.

Recommendation

Consider updating the NFT contract name and symbol to the name and symbol described in the README to minimize confusion.

Document Control

Version	Date	Notes
-	10th July 2024	Security audit commencement date.
0.1	25th July 2024	Initial report with identified findings delivered.
0.5	31st July 2024	Fixes remediations implemented and reviewed.
1.0	1st August 2024	Audit completed, final report delivered.

Appendices

A. Appendix – Risk assessment methodology

SCV-Security employs a risk assessment methodology to evaluate vulnerabilities and identified issues. This approach involves the analysis of both the LIKELIHOOD of a security incident occurring and the potential IMPACT if such an incident were to happen. For each vulnerability, SCV-Security calculates a risk level on a scale of 5 to 1, where 5 denotes the highest likelihood or impact. Consequently, an overall risk level is derived from combining these two factors, resulting in a value from 10 to 1, with 10 signifying the most elevated level of security risk

Risk Level	Range
CRITICAL	10
SEVERE	From 9 to 8
MODERATE	From 7 to 6
LOW	From 5 to 4
INFORMATIONAL	From 3 to 1

LIKELIHOOD and **IMPACT** would be individually assessed based on the below:

Rate	LIKELIHOOD	IMPACT
5	**Extremely Likely**	Could result in severe and irreparable consequences.
4	**Likely**	May lead to substantial impact or loss.
3	**Possible**	Could cause partial impact or loss on a wide scale.
2	**Unlikely**	Might cause temporary disruptions or losses.
1	**Rare**	Could have minimal or negligible impact.

B. Appendix – Report Disclaimer

This report should not be regarded as an "endorsement" or "disapproval" of any specific project or team. These reports do not indicate the economics or value of any "product" or "asset" created by a team or project that engages SCV-Security for a security review. The audit report does not make any statements or warranties about the code's utility, safety, suitability of the business model, regulatory compliance of the business model, or any other claims regarding the fitness of the implementation for its purpose or its bug-free status. The audit documentation is intended 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 damages arising from or in connection with this audit report. Copyright of this report remains with SCV-Security.

THANK YOU FOR CHOOSING

SCV SECURITY

scv.services

contact@scv.services