Eris Protocol Tokenfactory Audit Report

Prepared for Eris Protocol, 28th March 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-tokenfactory
- Code Freeze: *b84a480a82a5214a17949801da171b2b071692a2*

Remediations were applied by Eris team and reviewed by SCV on the following hash:

- *9fbfea5489410730052543f8b6a557a1564e86f6*

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.
- Analyze 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**	N/A
Code Coverage Test	**SUFFICIENT**	Testing Coverage and integration tests were sufficient during the audit
Code Readability	**SUFFICIENT**	The codebase had good readability and utilized many Rust and CosmWasm best practices.
Code Complexity	**SUFFICIENT**	N/A

Threat Modeling

The goal of threat modeling is to identify and evaluate potential threats to a system or application and to develop strategies to mitigate or manage those threats. Threat modeling 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 modeling includes (not limited to) the following :

- **Identify threats**: The first objective of threat modeling 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 modeling 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	Inconsistent time condition will cause `QueueUnbond` to error	**Medium**	**Resolved**
2	Hub vulnerable to share inflation attack if donations are enabled	**Low**	**Acknowledged**
3	Finding a new delegation can cause panic when using the defined delegation strategy	**Low**	**Resolved**
4	`update_config` does not emit updated values as attributes	**Informational**	**Acknowledged**
5	Update chain specific token reference	**Informational**	**Resolved**
6	Large list of validators could prevent instantiating the contract	**Informational**	**Acknowledged**
7	Removing validators can lead to divide by zero error	**Informational**	**Resolved**
8	Missing event logging when failing to reconcile batches	**Informational**	**Resolved**
9	Migrating to a previous version of the contract is possible	**Informational**	**Acknowledged**
10	Misspelling on `UpdateConfig` parameter	**Informational**	**Resolved**
11	Remove commented code	**Informational**	**Acknowledged**

Detailed Vulnerabilities

1 – Inconsistent time condition will cause QueueUnbond to error

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

Description

In the `queue_unbond` function in `contracts/hub/src/execute.rs:611` if the function is called on or after `pending_batch.est_unbond_start_time` the `ExecuteMsg::SubmitBatch` message is passed. If the call is made exactly at the `pending_batch.est_unbond_start_time`, when the contract enters the `submit_batch` function it will error since the condition within on `line 640` is not inclusive. This will roll-back not only the submitted batch execution, but also the original user's call `queue_unbond` even though the caller made a valid call.

Recommendations

We recommend updating these conditionals to ensure that they are reflecting the same time condition so that they can be called within the same atomic transaction without throwing an error.

2 − Hub vulnerable to share inflation attack if donations are enabled

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

Description

The `compute_mint_amount` function in `contracts/hub/src/math.rs:30` does not properly handle the initial share allocation. This can allow for a malicious actor to manipulate the initial bonding transactions. In the early stages, the mint amount can be susceptible to an inflation attack, resulting in initial bonders losing some or all of their funds to either the pool or a hacker.

The vulnerability arises from a rounding issue in `compute_mint_amount`. While the `compute_mint_amount` function does use `multiply_ratio` from cosmwasm-std which is more resistant to rounding issues, the issue is still present because the denominator of `utoken_bonded` can be manipulated through the `Donate` entrypoint.

It is also important to note that donation is currently disabled initially after instantiation.

There are three main scenarios where this can be manipulated by a malicious actor:

> **Scenario 1:** Attacker backruns the instantiation. Makes a deposit of `1` and thus gets one share. Then frontruns the first user to bond with a transaction that donates `(victim_bond / 2) +1`. This will mean that the victim will also get one share. At this point the victim and the attacker will both have `1ustake` even though the attacker has contributed just over half of the amount of `utoken`. Unlike a normal liquidity pool or token vault the attacker's ability to profit is limited by the unbonding process so it would be harder for them to directly profit a meaningful amount from orchestrating this share inflation attack.
>
> **Impact:** Unlike scenario 2 which would likely result in errors but no loss of user funds. Scenario 1 would result in the loss of user funds.
>
> **Scenario 2:** Attacker backruns the instantiation and with a transaction that provides an initial donation. This will block any bonding because

`compute_mint_amount` will return `0`. This is because `ustake_supply` in `line 39` will be 0 which will mean that the returned mint amount is always `0`.

Impact: Griefing attack mainly. The tokenfactory module won't mint a `0` amount so this would cause all other deposits to error. If subsequent transactions were to silently pass with no error this would be a critical vulnerability.

Scenario 3: Attacker backruns the instantiation. Makes a deposit of `1` and thus gets one share. The attacker then frontruns the first user bond with a donation of `bond_amount +1`. This will cause the mint amount to be `0`.

Impact: Griefing attack mainly. The tokenfactory module wont mint a `0` amount so this would cause all other deposits to error. Additionally this would not be profitable for the attacker. If subsequent transactions were to silently pass with no error this would be a critical vulnerability.

Recommendations

The most straightforward remediation that fits within the current design is to restrict donations while the `ustake_supply` is below a specific value. Ensuring that the `ustake_supply` is adequately high will ensure that the economic cost of this attack is high. Another more centralized but more comprehensive approach is to only allow donations from whitelisted addresses.

Revision Notes

This issue is only possible with donations enabled. The client acknowledges this finding and will not allow donations to be enabled until enough deposits are made. Since the donations are disabled by default, SCV has downgraded the severity of this finding from Severe, to Low based on the likelihood being decreased.

3 – Finding a new delegation can cause panic when using the defined delegation strategy

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

Description

When finding a new delegation with `DelegationStrategy::Defined`, if the delegation's query returns empty, the first validator stored in state is used. There are no validations on the validators vector done previously. If the validators vector stored in state is empty, unwrapping the option `first` will cause the contract to panic in `contracts/hub/src/execute.rs:554`.

Recommendations

Consider validating the validators vector not being empty during instantiation, alternatively avoid using `unwrap` and get the option value safely.

4 – `update_config` does not emit updated values as attributes

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

Description

The `update_config` function in `contracts/hub/src/execute.rs:1037` does not emit the newly updated attribute values. While this is not a security concern, it is best practice to emit specific attributes describing the state changes that have occurred during the contract call. This is ultimately a best practice suggestion as any user can simply query the contract to retrieve these values.

This is also present in `contracts/hub/src/execute.rs:41` in the `instantiate` function.

Recommendations

We recommend emitting detailed attributes that describe the actions taken and the parameters updated within the above functions.

5 – Update chain specific token reference

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

Description

In the most recent update for the Eris hub, there has been an effort made to ensure that there is a generalized approach when it comes to implementation. Throughout the codebase there is specific parameter naming specific to the terra chain:

- `contracts/hub/src/execute.rs:528`
- `contracts/hub/src/execute.rs:574`
- `contracts/hub/src/helpers.rs:16`
- `contracts/hub/src/helpers.rs:37`

Recommendations

We recommend updating the parameters mentioned above.

6 – Large list of validators could prevent instantiating the contract

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

Description

The set of validators is included as a vector in the `InstantiateMsg` without any limitation or restriction. When calling `assert_validators_exists` it loops and performs a staking query on every validator in the vector. If the vector is large enough, the contract could hit gas limits.

This issue propagates into other functions such as `add_validator`, which uses the `contains` function on the vector, which is `O(n)`.

Similarly, harvesting user's delegations could face the same situation.

Recommendations

We recommend setting a cap to the amount of validators that can be passed on instantiation and checking this number is not exceeded to prevent gas issues.

7 – Removing validators can lead to divide by zero error

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

Description

The `remove_validator` function in `contracts/hub/src/execute.rs:931` allows the admin of the contract to remove all the validators. When removing the last registered validator, if the delegation strategy is `DelegationStrategy::Uniform`, computing delegations will panic with a divide by zero exception, specifically at `contracts/hub/src/math.rs:336`.

Recommendations

We recommend handling this error gracefully, or setting a cap on the minimum number of validators to prevent such issues. Also, doing all mathematical operations with their `checked` counterparts would be preferable.

8 – Missing event logging when failing to reconcile batches

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

Description

The `reconcile_batches` function will silently fail reconciling batches when it cannot resolve the underflow distribution problem in `contracts/hub/src/math.rs:418`.

Recommendations

Even though failing to reconcile batches doesn't affect the functioning of the protocol, we recommend logging an event to inform the user or developer about this unresolved state.

9 – Migrating to a previous version of the contract is possible

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

Description

There is no version validation during contract migration in `contracts/hub/src/contract.rs:203`. This can lead to migrating to an older contract version, which might lead to potential errors if in future versions of the contract the state structures are changed.

Recommendations

Unless this behavior is desired, we recommend making proper validations on migration so such a situation cannot happen.

Revision Notes

The client states that it is a requirement that they will never implement backward breaking changes. They must support the option to rollback to a previous contract version.

10 – Misspelling on `UpdateConfig` parameter

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

Description

There is a misspelling in the parameter `withdrawls_preset` used in `ExecuteMsg::UpdateConfig`, found in `packages/eris/src/hub.rs:213`.

Recommendations

Consider renaming the parameter to the correct spelling.

11 – Remove commented code

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

Description

There are multiple snippets of code that are commented out across the code, namely:

- `contracts/hub/src/helpers.rs:180`
- `contracts/hub/src/queries.rs:52`
- `contracts/hub/src/types/gauges.rs`
- `contracts/hub/src/contract.rs:201`
- `contracts/hub/src/contract.rs:206`
- `contracts/hub/src/execute.rs:970`

Recommendations

We recommend removing all unused code from the project to make it more readable and maintainable.

Revision Notes

The client states that all comments related to gauges are out of the scope of this audit and will be reviewed at a later date then the functionality is fully implemented.

Document control

Version	Date	Approved by	Changes
0.1	05/03/2023	Vinicius Marino	Document Pre-Release
0.2	27/03/2023	SCV Team	Remediation Revisions
1.0	28/03/2023	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.