Eris Protocol ampz Audit Report

Prepared for Eris Protocol, 12th 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-terra/tree/develop/contracts/ampz
- Code Freeze: `fb3778ca66aa2b4fb8eeb3b4441df7440db57176`

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

- Code Freeze: `561df3a9f9be17a6da55bc4abfa83ef0e03488b6`

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 were provided and its available at https://docs.erisprotocol.com/products/amp-z/
Code Coverage Test	**SUFFICIENT**	The codebase within scope has sufficient unit testing. Complex authz interactions that cannot be tested with cw-multi-test have been tested dynamically through testnet contract interaction.
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.

Audit observations

The audit observations in the context of a smart contract auditing refer to issues or findings concerns that are identified by an auditor during the audit process. These may be deviations from established smart contract standards, security vulnerabilities, or potential risks that could affect the functionality or accuracy of the smart contract or the environment context it relies on. Audit observations are documented in this report and communicated to the client for corrective action and discussion. The purpose of audit observations is to improve the functionality and security of the smart contract and ensure that it operates as intended.

- The ampz contract does not directly implement the authz grant or revoke, it relies on UI interaction where the user signs an execute message along with the necessary grant messages. This involves a level of trust that the front end is properly assigning the authz grants. The current implementation requires a certain level of reliance on off-chain components, which can be vulnerable to tampering and impersonation attacks by malicious actors. Furthermore, these components may be less auditable overall, which could further increase the risk of potential security breaches. We note this as an observation because this functionality is currently not implemented in the wasm module to allow for grants in this manner.

- The ampz contract is currently migratable. Due to the general message execute authorizations granted to the contract by its users we strongly advise against allowing contract migrations. Users grant a generic authorization for the contract to execute the `MsgExecuteContract` messages on their behalf. While the current version of the contract securely manages these executions, there is no guarantee that an migration bug or a malicious migration executed after an admin compromise could lead to a loss of user owned funds outside Eris protocol.

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	`add_execution` is not resistant to gas griefing and spam attack	**Medium**	**Acknowledged**
2	Farms are not de-duplicated	**Informational**	**Resolved**
3	Introduce minimum `interval_s` to prevent spam	**Informational**	**Resolved**
4	New owner is not validated to ensure it is not the same as current owner	**Informational**	**Resolved**
5	New farms added can be removed in the same execution	**Informational**	**Resolved**
6	Incorrect action emitted for `accept_ownership` function	**Informational**	**Resolved**
7	Successful contract instantiation does not emit attributes or events	**Informational**	**Resolved**
8	Remove unused code	**Informational**	**Resolved**

Detailed Vulnerabilities

1 - `add_execution` is not resistant to gas griefing and spam attack

Risk Impact: Medium - **Status**: Acknowledged

Description

In the `add_execution` function in `contracts/ampz/src/domain/crud.rs:10` the execution's `Source` parameters are unchecked. This can allow for a situation where executions can be created where the user does not have the required staked funds or does not belong to the specified astroport farm.

When these executions are attempted to be executed, they will error and potentially cause enough spam where executors stop performing executions because they only get paid when an execution is successful. This can be used to effectively execute a gas griefing attack because executors will lose spent gas during a contract error. In general, the protocol should prevent potential failing executions before they are created.

Recommendations

We recommend implementing checks to ensure that legitimate executions are being created so that executors experience as few failed executions as possible. This can be accomplished in a number of ways, each with their own benefits and downsides. These controls can also be layered to further limit spam.

The first control is to track the executions that error. For example, if an execution fails a certain number of times, that execution should be removed as it is impacting the executors negatively if they are paying for the gas for a failed execution.

An additional control is to require a small deposit from the execution creator. This deposit can be used to fund failed executions, and when the balance is depleted, then the execution is removed. A production example of this is in CronCats which requires a user deposit equivalent to the estimated gas for a small number of executions.

Additionally, pre-validations can be implemented to further filter out potentially failing validations. While this method is not comprehensive in blocking spam or erroring executions, when combined with the methods mentioned above it will provide additional controls.

Revision Notes

The client has acknowledged that through simulation, executors can determine the associated rewards and assess the viability of the execution in advance. This proactive approach will minimize the occurrence of unsuccessful transactions. Therefore, it is essential for executors to ensure that they only proceed with executions that are likely to succeed.

SCV strongly recommends that executors engage in a simulation process to ascertain the potential outcome of their automations prior to execution. This will enable executors to make informed decisions based on the simulation results, thereby reducing the likelihood of unsuccessful executions.

2 – Farms are not de-duplicated

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

Description

In the ampz’s `exec_instantiate` function in `contracts/ampz/src/instantiate.rs:25` the vector of amp compounder farms is not properly deduplicated. This may potentially introduce unintended states when the farms are interacted with.

Recommendations

We recommend deduplicating the farms in the `exec_instantiate` function.

3 – Introduce minimum `interval_s` to prevent spam

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

Description

In the `add_execution` function in `contracts/ampz/src/domain/crud.rs:10`, the execution is not checked to ensure that its interval is not too frequent. The protocol should implement a minimum `interval_s` to ensure that it is not set too low.

This can create a situation where the reward fee is not higher than the gas prices the operators will pay to execute, thus the execution will not be executed if the compounded amount is too small.

Recommendations

We recommend implementing a validation to check that `interval_s` is not below a specified value.

4 – New owner is not validated to ensure it is not the same as current owner

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

Description

The `transfer_ownership` function in `contracts/ampz/src/domain/ownership.rs:6` does not validate that `new_owner` is not the same address as the current owner. While the situation where an owner updates the address to themselves is not a direct security concern, it should return an error because the ownership transfer may emit events that are contradicting the actual state changes that are occurring.

Recommendations

We recommend implementing a validation to ensure that the new owner is not the same as the current owner.

5 – New farms added can be removed in the same execution

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

Description

When updating the config in `contracts/ampz/src/config.rs:50`, it is possible that the admin can add a farm and unintentionally remove it if the added farm is also one of the farms in `remove_farms`.

Recommendations

We recommend implementing a validation to ensure that the new farms being added are not in `remove_farms` when updating the config.

6 – Incorrect action emitted for `accept_ownership` function

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

Description

In line 41, the value of the “`action`” attribute key emitted is “`ampz/transfer_ownership`” for the `accept_ownership` function. As the “`action`” attribute key normally emits the executed function name, consider modifying the values to the associated function name to prevent confusion.

Recommendations

We recommend modifying the “`ampz/transfer_ownership`” to “`ampz/accept_ownership`”.

7 – Successful contract instantiation does not emit attributes or events

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

Description

When a contract is instantiated, a lack of emitted attributes or events in a successful instantiation on `contracts/ampz/src/instantiate.rs:36` hinders off-chain listeners from indexing the parameters configured by the contract owner.

Recommendations

We recommend emitting relevant events or attributes based on configured parameters.

8 – 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/ampz/src/contract.rs:103`
- `contracts/ampz/src/contract.rs:108-109`

Recommendations

We recommend removing the unused code.

Document control

Version	Date	Approved by	Changes
0.1	05/03/2023	Vinicius Marino	Document Pre-Release
0.2	10/03/2023	SCV Team	Remediation Revisions
1.0	12/03/2023	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.