Treasury & Voting Smart Contracts for DAO Governance × Atala PRISM

— Internal Audit Report —

MuesliSwapTeam

March 26, 2024

1 Purpose & Scope

The purpose of this document is to provide a detailed audit report of a set of treasury and voting/governance smart contracts for the *DAO Governance × Atala PRISM* project supported by Catalyst Fund 10. This includes discovered vulnerabilities, potential security threats, and recommendations for improving the security and reliability of the platform. A draft version of the platform has been developed in OpShin, a domain-specific language for smart contracts on the Cardano blockchain based on Python. This ensure maximum maintainability and readability of the code. Nontheless, before the platform is deployed on the Cardano blockchain, it is crucial to conduct a comprehensive audit to ensure the security, reliability, and efficiency of the platform. In a first step, a seperated part of the team has preformed an independent audit of the code which is presented in this document. The document is made availabile to the public to ensure transparency and to provide a comprehensive overview of the audit process and its outcomes. The document is purely informative and does not constitute financial or investment advice nor does it promise or guarantee the absence of bugs or complete faithfulness of the analyzed code.

The scope of this document entails a list of discovered vulnerabilities and potential security threats in the MuesliSwap On-Chain Governance platform. Moreover it includes recommendations for improving the security and reliability of the platform.

2 Audit Methodology

The audit was conducted by a team of experienced blockchain developers and security experts. The audit process involved a comprehensive review of the provided codebase, focusing on the following key areas:

- Smart contract architecture and design

- Code quality and readability
- Security vulnerabilities and potential attack vectors
- Compliance with best practices and industry standards
- CPU and Mem optimization and efficiency

The audit process was conducted in accordance with industry best practices and security standards. The team utilized a combination of manual code review and automated analysis tools to identify potential vulnerabilities and security threats.

3 Audit Findings

3.1 Excessive Script Output Sizes

Issue Identification It was identified that there was no existing mechanism to ensure the size of assets and the attached data (datum) within script continuations remained within reasonable limits. The absence of such a check posed a potential vulnerability, risking system stability and security.

Issue Description The primary concern revolved around excessively large script outputs, which could lead to inefficient operation of the on-chain governance mechanisms and possibly expose the system to various attacks. The issue is known as *UTXO Value Size Spam AKA Token Dust Attack* [1].

Resolution The resolution to this issue was swiftly implemented by a commit that introduces a simple, yet effective function designed to check the maximum size of script outputs before their execution. This function serves as a preventative measure, ensuring all script continuations adhere to predefined size constraints, thereby mitigating the risk previously identified.

3.2 Inadequate Validation in New Tally Function

Issue Identification It was discovered that the `validate_new_tally` function lacked a crucial validation step to ensure that exactly one output represents a new tally. This oversight could potentially allow the creation of multiple tallies simultaneously, which poses a significant risk to the integrity of the tally creation process.

Issue Description The core issue was identified and concerns the absence of a validation mechanism in the `validate_new_tally` function to verify the creation of only one new tally as an output. The existing implementation did not prevent the possibility of generating multiple tallies in the same transaction. This situation could lead to only the tally specified in the redeemer being validated, thus potentially overlooking additional unauthorized tallies. This flaw was noted as a critical oversight that could undermine the system's security and operational integrity.

Suggested Resolution The suggested resolution, involves incorporating an explicit check within the `validate_new_tally` function. This check would confirm that exactly one output is dedicated to a new tally, thereby safeguarding against the creation of multiple tallies. Furthermore, it was highlighted that a tally's validity is indirectly ensured by the requirement of an accompanying authorization NFT, which guarantees that only one NFT can be minted per transaction. Despite this indirect safeguard, the consensus was that an explicit check would offer a more robust solution. This improvement was promptly addressed by a commit effectively closing the issue and reinforcing the system's validation process.

3.3 Unintended Tally Creation in Governance State Update

Issue Identification A potential vulnerability was identified in the project, related to the `UpdateGovState` redeemer. It was discovered that during the process of updating the government state, there is an opportunity to create arbitrary new tallies. This behavior may not be intentional and could lead to unintended consequences affecting the control and integrity of the tally creation process.

Issue Description The core issue lies in the ability to potentially exploit the `UpdateGovState` redeemer to create new tallies arbitrarily while updating the government state. This functionality raises significant concerns regarding:

- **Integrity of Tally Creation:** Allowing the creation of new tallies in this manner might undermine the controls or checks designed around the tally creation process.
- **Unexpected State Changes:** The unintended functionality could result in unexpected state changes. These changes might not be immediately noticeable but could impact the system's stability and reliability over time.

Resolution Upon the issue being raised it was acknowledged as a vulnerability. The problem was addressed and resolved. The resolution involved:

- **Clarification:** A clear explanation was provided, acknowledging the unintended functionality as a vulnerability.
- **Review and Testing:** The code was reviewed, and additional testing was conducted to ensure that arbitrary tally creation through the `UpdateGovState` redeemer is no longer possible, safeguarding the system's integrity and reliability.

This corrective action ensures the robustness and reliability of the system by eliminating the vulnerability related to unintended tally creation.

3.4 Duplicate Voting Vulnerability

Issue Identification An examination of the voting mechanism within the system raised concerns regarding the prevention of duplicate votes by a single user. Specifically, the focus was on the possibility of a user invoking the `AddVote` function multiple times with the same stake, potentially leading to multiple participations in the same election. This scenario highlights a significant vulnerability that compromises the integrity of the voting process.

Issue Description The core of the issue lies in the lack of mechanisms within both the staking and tally contracts to ensure that a stake is not counted more than once in an election. The current system only ensures that stake is locked in the resulting position after a vote is cast, without validating that a staking update is performed or preventing multiple participations by the same stake in a single vote. This oversight opens the door to the possibility of *Duplicate Voting*, undermining the fairness and integrity of the election process.

Resolution The vulnerability was acknowledged and addressed by a series of comments and a subsequent fix. The proposed solution involves enhancing the staking contract to include the necessary validations for staking updates and to enforce the invariant that prevents multiple participations by the same stake in an election. This approach ensures that each stakeholder is limited to one vote per election, reinstating the intended democratic integrity of the voting mechanism. The issue was marked as resolved upon the implementation of these changes.

3.5 Potential Spam Attack on Tally

Issue Identification An issue was raised regarding a potential vulnerability in the tally system. The concern was that users could add spam tokens to vote updates, which in turn could increase the size of the tally beyond 1000 bytes. This could potentially exploit the system by attaching an excessive number of tokens or reference scripts to manipulate the tally mechanism.

Issue Description The vulnerability identified revolves around the ability to manipulate the tally by adding spam tokens, potentially leading to a situation where the vote integer's size could prevent the system from progressing.

Discussion and Resolution Upon further discussion and analysis it was found that this issue might not be as critical as initially thought. The tally contract is designed to allow the removal of attached funds, meaning that if additional tokens are added, they can be removed in a subsequent transaction. This mechanism not only addresses the potential vulnerability but also provides an incentive for participants to remove spam tokens by obtaining them as a reward. Consequently, this issue was marked as completed and closed.

3.6 Inability to Burn Vote NFTs

Issue Identification It was discovered that the logic for minting vote NFTs lacked a clear pathway for validating the burning process. This oversight could potentially lead to issues with governance token management and the integrity of the voting process, as it prevents the proper removal of vote NFTs from circulation.

Issue Description The core of the problem lies in the absence of a mechanism within the vote NFT minting logic to validate or execute the burning of these tokens. This limitation not only restricts the flexibility needed in the management of vote NFTs but also poses a risk to the governance system's operational efficiency and security. The inability to burn vote NFTs could result in an inflated or inaccurate representation of voting power, undermining the governance framework's reliability.

Resolution The issue was addressed and resolved. The resolution involved implementing a solution that introduced the necessary logic to validate and execute the burning of vote NFTs. This enhancement ensures that the governance framework can accurately reflect the current distribution of voting power by allowing for the removal of vote NFTs from circulation, thereby maintaining the integrity and accuracy of the voting process.

Conclusion The proactive identification and discussion of potential vulnerabilities within the muesliswap-onchain-governance system demonstrate the community's commitment to maintaining the security and efficiency of its operations. The resolution of this issue underscores the importance of having flexible and robust mechanisms in place to mitigate potential attacks and ensure the integrity of the governance process.

3.7 Unrestricted Amount of Licenses Per Winning Vote

Issue Identification An issue was discovered where there was no mechanism to restrict the number of licenses that could be minted from a successful vote tally. This oversight could potentially lead to the unlimited distribution of licenses, raising concerns about system integrity and the possibility of abuse.

Issue Description The issue centers around the governance mechanism's inability to limit the issuance of licenses following a vote. Specifically, there was no validation to ensure that only a predetermined number of licenses could be minted, allowing for an unrestricted amount to be issued. This could undermine the intended scarcity and exclusivity of certain licenses, such as batcher licenses, and possibly lead to unintended distribution or use scenarios.

Discussion and Resolution Approach The discussion on this issue was led by contributors. It was debated whether the lack of restriction was problematic, with considerations on the potential for tracking licenses back to the original recipient and penalizing undesired behavior. The possibility of implementing a minting governor to restrict license issuance was discussed but deemed not necessarily desirable due to the flexibility needed for different batcher implementations and the potential requirement for a new policy ID for changes in license rules. Finally the issue was closed as completed without implementing a direct solution to restrict license minting, focusing instead on oversight and potential penalization for misuse. For example, licenses do now contain a unique identifier that can be used to track the original recipient and revoke access to future licenses in case of misuse.

Resolution The issue was ultimately considered not to be a significant concern, given the control mechanisms in place for managing license misuse through the revocation of access to future licenses. The consensus was that the flexibility for license receivers to manage their licenses, including potential distribution, was acceptable within the governance framework's context. The issue was closed by *pcjordan* as completed without implementing a direct solution to restrict license minting, focusing instead on oversight and potential penalization for misuse.

Comments and Further Actions Further discussions might explore alternative approaches to ensure the governance system's integrity while maintaining flexibility for license distribution and management. Any significant changes to the license issuance rules would require careful consideration of the impact on policy IDs and the adaptability of pool contracts to dynamically update allowed batcher policies based on tally results.

3.8 Potential Tally Spamming

Issue Identification An issue was opened concerning the absence of a fee for creating tally proposals. The current protocol does not require any fee beyond the network fee for proposal creation. This oversight could potentially allow for protocol spamming through the creation of numerous unnecessary proposals.

Issue Description The primary issue lies in the fact that creating a tally proposal is cost-free, aside from the network fee, which does not deter the submission of frivolous or malicious proposals. The suggested resolution involves introducing a proposal creation fee in governance tokens, which would be directed to the treasury. This approach necessitates that the governance contract is aware of the treasury's location, implying the need for an upgrade to the governance thread whenever there is a change in the treasury's address.

Proposed Resolution The proposed solution entails the introduction of a fee for creating tally proposals, payable in governance tokens. This fee structure aims to prevent spam by imposing a cost on the submission of proposals, ensuring that only serious and considered proposals are submitted. Implementing this fee requires the governance contract to be updated to recognize the treasury's location, highlighting the need for flexibility in the governance system's design to accommodate changes in treasury management. A direct implementation has not been performed, yet as it is unclear whether this does actually improve the situation and the upsides outweigh the significant downside of reduced participation in the governance process.

4 Conclusion

The audit has identified several vulnerabilities and potential security threats. The team behind the platform promptly addressed these issues and implemented the necessary changes to mitigate the risks and reinforce the system's security and reliability. The audit process also provided valuable insights and recommendations for improving the platform's security and efficiency. The team's proactive approach to addressing the identified vulnerabilities demonstrates their commitment to maintaining a secure and transparent governance platform. The audit process has been instrumental in identifying and resolving potential security threats, thereby enhancing the overall security and reliability of the proposed platform.

5 References

References

[1] Certification Working Group. *Common Vulnerabilities in Smart Contracts.* URL: `https://github.com/input-output-hk/Certification-working-group/blob/a3b2e1602c0fc52d8d90e008ab421ae9b1e81a2a/Vulnerabilities/full-list-to-be-triaged.md`.