DcentraLab Diligence

Audit Report

HyperCycle - ShareManager

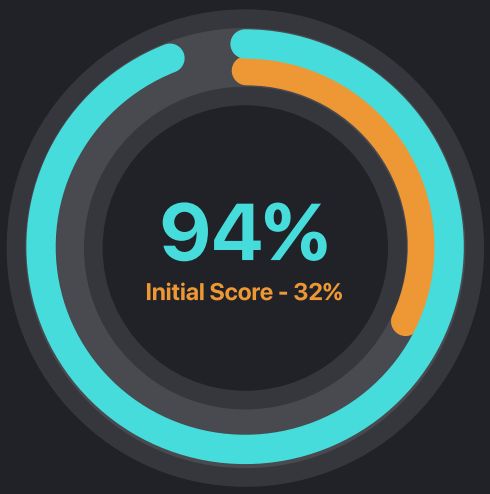
https://www.hypercycle.ai

Provided By DcentraLab Diligence on March 20, 2024

Security Audit Score

Low Risk

DcentraLab Diligence team has conducted an extensive audit on HyperCycle Contracts and has found the code to be in minimal risk level given proper deployment and multi-sig permissioning.



- Minimal Risk
- Low Risk
- Medium Risk
- High Risk
- Critical Risk

Score Guidelines

Scope

Audited Repository:

https://github.com/hypercycle-development/hypercycle-contracts

Audited Branch:

develop

Audited Commit Hash:

08216c7fd9f9244a6d9195b9e6fd10c1c20ec7a2

Fix Branch:

hypersharemanager_development

Fix Commit Hash (Iteration 1):

a8c6f5eabbe365da62d7bdb97e842645431f9055

Fix Commit Hash (Iteration 2):

71c96f413e9527041e960b889a278390a2710c06

Audited Contracts:

HyperCycleShareManager.sol

Reviewed For Context:

- HyperCycleLicense.sol
- HYPCSwapV2.sol
- HYPC.sol
- HyperCycleShareTokensV2.sol
- ShareManagerTypes.sol

Nomenclature Of Issues:

E - Environmental
A - Contract HyperCicleShareTokens.sol

Contracts Architecture Overview

HyperCycleShareManager analysis

HyperCycleShareManager is a governance contract and a management contract for the HyperCycleShareTokens contract. This contract allows the share owner to create a variety of proposals regarding its share management. It allows the shareholders to vote on those proposals on chain. The governance token is the "wealth" token which is part of the HyperCycleShareTokens contract which is a ERC-1155 implementation. The main idea of the Share Manager contract is to extend the Share Tokens contract with a governance mechanism that allows share token holders to execute management functionalities for their share by voting on proposals.
A proposal can be ended and executed by the token holders any time the consensus is reached. The share owner will be able to create votations for the following features:

- Change the hardware operator - required more than 50% consensus
- Change the hardware operator revenue - required more than 50% consensus
- Cancel share tokens - require 100% consensus
- Transfer the share ownership to a new share owner - require 100% consensus

One more important feature of Share Manager contract is the ability to migrate the Share Tokens from the Share Tokens contract to the Share Manager Contract, this will allow the Share Manager Contract to be the one holding the Share Tokens and be able to manage the Share proposals and votations. The Share owner will be able to claim the Hypc tokens based on the amount of the wealth tokens available in case the share proposal is ended and the CHyPC exists.
To migrate the Share Tokens to the Share Manager, the Share Token owner needs to call `startShareProposalMigration` function, this function will start the migration and the Share Manager will be able to finish the migration only if the ownership of the share token is changed to the Share Manager, and the Share Proposal is pending.

Roles:

- Voter: The user who have the right to vote on various proposals
- cHyPC Owner: The owner of the cHyPC token (stored on the cHyPC contract)
- License Holder: This is the user who holds the cHyPC token.

External Functions:

- **createShareProposal(bytes memory proposalData):**

Allows a cHyPC owner to create a new revenue sharing proposal. It encodes various parameters related to the share proposal, including cHyPC ID, revenue, wealth to assign, and hardware operator information.

Contracts Architecture Overview

- **cancelPendingShareProposal(uint256 shareProposalId):**

Allows the owner of a license to cancel a pending share proposal by specifying the share proposal ID. It's meant to be used before the proposal has been completed and started.

- **completeShareProposal(uint256 shareProposalId, uint256 licenseNumber):**

Enables the license holder to complete a pending share proposal, essentially starting the revenue sharing process by linking a license to the proposal.

- **claimChypcPortion(uint256 shareProposalId):**

Allows participants of a concluded share proposal to claim their portion of the revenue generated, based on the number of wealth tokens they hold.

- **claimSurplus(uint256 shareProposalId):**

Enables participants to claim any surplus revenue that was not distributed during the share proposal's active phase, after it has ended.

- **startShareProposalMigration(uint256 shareTokenNumber, address hardwareOperator):**

Initiates the migration of a share proposal from the Share Tokens contract to the Share Manager, specifying the share token number and new hardware operator.

- **finishShareTokenMigration(uint256 shareProposalId):**

Concludes the migration process of a share proposal to the Share Manager, ensuring that the Share Manager now holds the relevant share tokens.

- **proposeNewHardwareOperatorAddress(uint256 shareProposalId, uint256 deadline, string memory newProposedString, address newHardwareOperator):**

Allows share token holders to propose a new hardware operator for a specific share proposal, specifying the proposal ID, deadline, and details of the new operator.

- **proposeNewHardwareOperatorRevenue(uint256 shareProposalId, uint256 deadline, uint256 newRevenue):**

Share token holders can propose a change in the revenue allocation for the hardware operator of a given share proposal.

- **proposeNewManager(uint256 shareProposalId, uint256 deadline, address newShareManager):**

Enables share token holders to propose changing the Share Manager to a new contract address for managing the share proposal.

- **vote(uint256 shareProposalId, uint256 votationIndex, bool voteFor):**

Allows share token holders to vote on active proposals related to the share proposal they are part of.

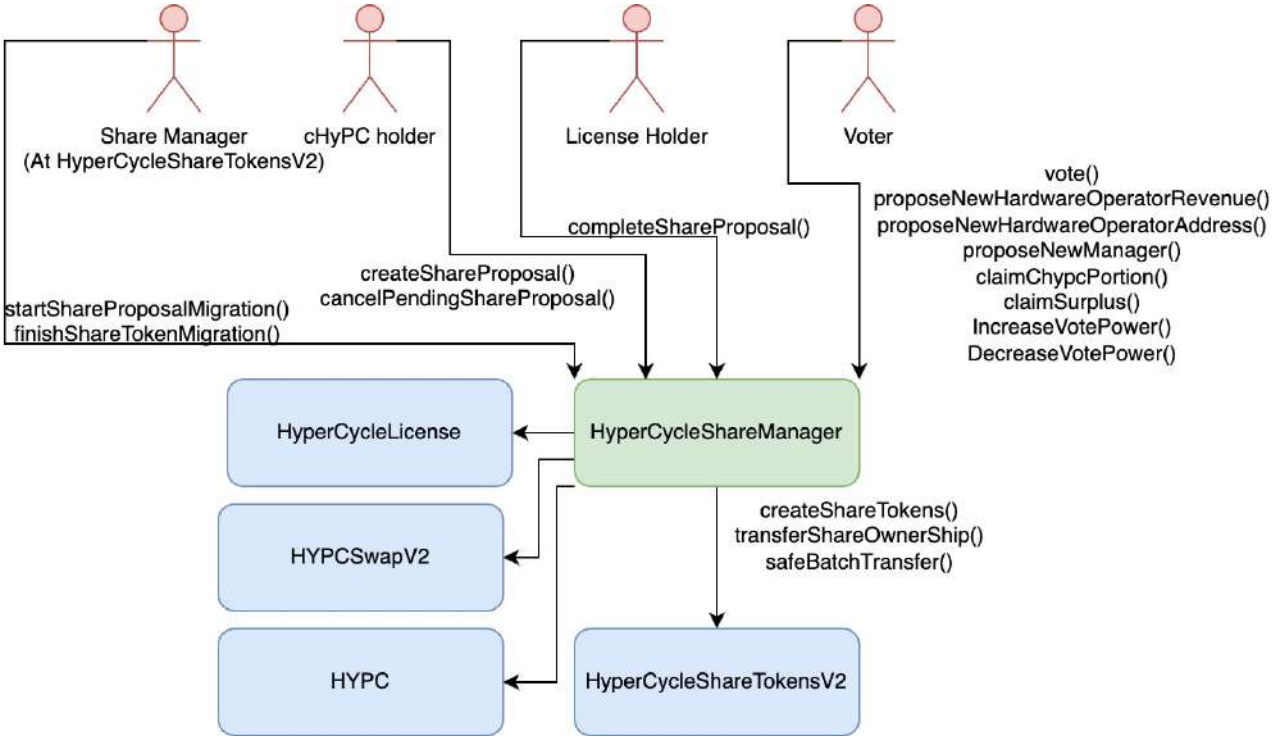
Contracts Architecture Overview

- **increaseVotePower(uint256 shareProposalId):**

Share token holders can increase their voting power in a proposal by locking more wealth tokens into the contract.

- **decreaseVotePower(uint256 shareProposalId):**

Allows participants to decrease their voting power after a proposal has ended, withdrawing their locked wealth tokens.



Issues Severity Reference Table

Type

Discussion

The issue severity is dependent on design, centralization, and product specifications of the project.

Informational

This issue is not critical and does not pose an immediate threat to the functionality or security of the smart contract. It is simply an informational item that the auditors have identified and recommends addressing for best practices or to improve the overall performance of the contract.

Low

This issue is relatively minor and does not pose a significant risk to the functionality or security of the smart contract. While it is recommended to address these issues to ensure the highest level of quality and security, they are not likely to cause significant problems if left unaddressed.

Medium

This issue poses a moderate risk to the functionality or security of the smart contract. While it may not be immediately exploitable, it has the potential to cause problems in the future if left unaddressed. It is recommended to address these issues as soon as possible to prevent any potential negative impact on the contract.

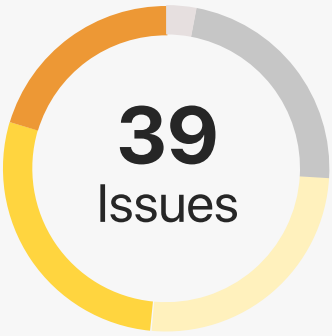
High

This issue poses a significant risk to the functionality or security of the smart contract. Addressing these issues as soon as possible is recommended to prevent any potential negative impact on the contract. Failure to address these issues could result in significant problems and potential loss of funds or other assets.

Critical

This issue poses an immediate and severe risk to the functionality or security of the smart contract. It is recommended to address these issues immediately to prevent any potential negative impact on the contract. Failure to address these issues could result in catastrophic problems and significant loss of funds or other assets.

Findings Summary



- Discussion
- Informational
- Low Risk
- Medium Risk
- High Risk
- Critical Risk

ID	Title	Severity	Status	Risk Points
A.1	Consistent Access Control Mechanism	Low	Resolved	-1
A.2	Documentation Mismatch	Informational	Resolved	
A.3	Documentation Mismatch	Informational	Resolved	
A.4	Unnecessary Function Invocation	Medium	Resolved	-5
A.5	Lack of Balance Verification Prior to Transfer	Medium	Resolved	-5
A.6	Function documentation and implementation mismatch	Discussion	Resolved	
A.7	Missing Event For Unsuccessful Votations	Low	Resolved	-1
A.8	Inconsistency in Coding Practices	Low	Resolved	-1
A.9	Inability to Precisely Adjust Vote Power	Low	Acknowledged	-1
A.10	Typographical Errors in Comments	Informational	Resolved	

Findings Summary

ID	Title	Severity	Status	Risk Points
A.11	Inconsistent In Contract Documentation	Informational	Resolved	
A.12	Missing proposal mechanism for 'changePendingRevenueDelay'	High	Resolved	-15
A.13	Missing proposal mechanism for 'setShareMessage'	High	Acknowledged	-5
A.14	Absence of Validation Check	Medium	Resolved	-5
A.15	Missing Validation Check On Share Proposals	High	Acknowledged	-10
A.16	Incomplete Implementation of 'getVotationConsensus' Function	Medium	Resolved	-5
A.17	Inefficient Management of Consensus Percentages	Medium	Partially Resolved	-5
A.18	Inaccurate Parameter Description in 'proposeNewHardwareOperatorRevenue' Function	Informational	Resolved	-1
A.19	Unnecessary Import of Ownable Contract	Low	Acknowledged	-1
A.20	Lack of Migration Reversion Mechanism	Medium	Resolved	-5
A.21	Inaccurate Token Calculations Due to Burnable Tokens	High	Resolved	-20

Findings Summary

ID	Title	Severity	Status	Risk Points
A.22	Inaccurate Consensus Calculations Due to Burned Tokens	High	Resolved	-20
A.23	Inaccurate Reward Distribution Due to Unadjusted Token Supply	High	Resolved	-20
A.24	Inaccurate Revenue Proposal Limits Due to Static Token Supply Assumptions	High	Resolved	-20
A.25	Risk of Asset Lock due to 100% Consensus Requirement	High	Resolved	-10
A.26	Revenue Claim Limitation Due to Voting Power Allocation in 'claimChypcPortion'	Medium	Resolved	-5
A.27	Misalignment in Total Supply Calculation for Revenue and Wealth Tokens	Medium	Acknowledged	-5
A.28	Irreversible Voting Mechanism - Resolved	Medium	Resolved	-5
A.29	Absence of Validation for Matching CHyPC and License Levels	Medium	Resolved	-5
A.30	Simplify Process for Holders of Both cHyPC and License	Informational	Resolved	
A.31	Lack of Support for Creating Share Proposals with Assigned Licenses	Informational	Acknowledged	

Findings Summary

ID	Title	Severity	Status	Risk Points
A.32	Absence of Validation for Matching CHyPC and License Levels	Low	Resolved	-1
A.33	Potential Misalignment of Assigned Operator Message during Migration	Low	Resolved	-1
A.34	Simplifying HYPC Token Claims with Voting Power	Low	Resolved	-1
A.35	Misleading Function Naming	Informational	Resolved	
A.36	Potential Override of Existing License Assignments	Medium	Acknowledged	-5
A.37	Potential Front Running of Assignment	Low	Acknowledged	-1
A.38	Potential Token Rounding Issue in claimSurplus Function	Low	Resolved	-1
A.39	Typographical Error	Informational	Resolved	

Quality Score Card

Factor	Positive Score Points rage	Final Score
Presence of up to date documentation	1-5	4
Code readability	1-5	4
Operational security of the team	1-5	5
Code reliance on third-party software	1-10	10
Test Coverage	1-10	8

Total quality score: 32/35

Total Score

	Risk Score	Quality Score	Total Score
Pre-Audit	0 (*0.65)	32	**32**
Final Report	95 (*0.65)	32	**94**

Complete Analysis

Local Contract Findings:
Contract: HyperCicleShareManager.Sol

ID A.1: **Status: Resolved**

Low | Consistent Access Control Mechanism

Present at: 'cancelPendingShareProposal', 'completeShareProposal'@ L211, L224

Description: The access control checks for the token owner and the license owner are currently implemented using 'require' statements within the function bodies.

Recommendation: Consider using a modifier, same way as done with 'onlyVoter'. Using modifiers for Access Control is a better practice to prevent errors and mistakes in implementation.

Recommendation 2: Modifier wasn't added for the liecense owner check in 'completeShareProposal'

ID A.2: **Status: Resolved**

Informational | Documentation Mismatch

Present at: 'cancelPendingShareProposal'@ L211

Description: The function's @notice comment incorrectly identifies the license owner as the authorized caller, which contradicts the actual functionality.

Recommendation: Adopt a unified approach to access control by introducing custom modifiers, similar to the onlyVoter modifier used elsewhere in the contract. Modifiers enhance readability, maintainability, and consistency across the contract's access control logic, reducing the likelihood of errors or oversight in implementation.

Complete Analysis

ID A.3: **Status: Resolved**

Informational | Documentation Mismatch

Present at: 'completeShareProposal'@ L224

Description: The '@notice' documentation inaccurately states that the owner of the cHyPC NFT can call this function, which does not align with the function's implementation.

Recommendation: Correct the documentation to accurately represent the function's authorization. Amend the @notice comment to: "Allows the owner of the license to complete a pending share proposal,".

ID A.4: **Status: Resolved**

Medium | Unnecessary Function Invocation

Present at: '_completeProposal''@ L283

Description: The setShareMessage function call in the HyperCycleShareTokens contract is redundant, as the initial message is already established during share creation at line 279.

Recommendation: Eliminate the redundant setShareMessage invocation to streamline the contract's execution and optimize gas usage.

Complete Analysis

ID A.5: **Status: Resolved**

Medium | Lack of Balance Verification Prior to Transfer

Present at: 'claimChypcPortion''@ L457

Description: The function does not verify if the contract's HYPC token balance is sufficient to fulfill a user's claim based on their wealth token balance. Insufficient contract balance will cause the transfer to fail and revert the transaction.

Recommendation: To mitigate this issue, a balance check should be added before attempting the transfer, ensuring the contract has enough hypcToken to cover the claim.

Recommendation 2: Notice for an error in function documentation. It should be Hypc instead of Chypc in like 476

ID A.6: **Status: Resolved**

Discussion | Function documentation and implementation mismatch

Present at: 'claimSurplus'@ L470

Description: The @dev documentation mentions a scenario where if the cHyPC does not exist, the entire surplus is sent to the license owner. This logic isn't reflected in the implementation. It's unclear under which conditions the cHyPC would be nonexistent.

Recommendation: Clarify within the implementation whether and how the contract addresses the absence of a cHyPC, as described. If this scenario is not applicable, the documentation should be updated to match the actual function behavior.

Complete Analysis

ID A.7: **Status: Resolved**

Low | Missing Event For Unsuccessful Votations

Present at: 'vote'@ L674

Description: The current implementation of the vote function is designed to emit the VotationEnded event exclusively when the voteFor condition is true, leading to the successful conclusion of a votation. This approach does not account for scenarios where the votation concludes unsuccessfully, such as when a proposal fails to achieve the required consensus. Examples include a vote not passing because it did not receive 100% of "For" votes when such unanimity is required, or when the majority votes against in situations where more than 50% of "For" votes are necessary for passage. This omission could hinder transparency and comprehensive tracking of votation outcomes.

Recommendation: To enhance the contract's transparency and provide complete feedback on all votation outcomes, it is recommended to introduce event emission for cases where a votation ends without execution due to insufficient "For" votes.

ID A.8: **Status: Resolved**

Low | Inconsistency in Coding Practices

Present at: 'vote'@ L658

Description: The implementation exhibits inconsistent coding practices within the vote function. Specifically, the function uses 'getWealthTokenTotalSupply' to determine the total supply of wealth tokens at one point (line 658), but then switches to directly accessing '_shareProposals[shareProposalId].chypcData.tokenLevel - 1' for a related calculation in another (line 666). This inconsistency can lead to confusion and potentially introduce errors if the underlying logic for these values diverges in the future.

Recommendation: To maintain consistency and ensure future code maintainability, it is advisable to standardize the approach by using 'getWealthTokenTotalSupply' for both calculations.

Complete Analysis

ID A.9: **Status:** Acknowledged

Low | Inability to Precisely Adjust Vote Power

Present at: 'increaseVotePower', 'decreaseVotePower' @ L682, L698

Description: The functions 'increaseVotePower' and 'decreaseVotePower' do not allow users to specify the amount by which they wish to adjust their voting power. Instead, these functions operate on an all-or-nothing basis: 'increaseVotePower' uses the user's entire wealth token balance to increase voting power, and 'decreaseVotePower' removes all of a user's voting power upon execution. This approach limits user flexibility in managing their vote power.

Recommendation: To enhance user experience and provide greater control, consider adding an amount parameter for both functions.

ID A.10: **Status:** Resolved

Informational | Typographical Errors in Comments

Present at: 'getVotePower' , 'getUserVote' @L1013 L1019 L1021 L1022

Description: The words "addres" and "powe" are misspelled

Recommendation: Replace "addres" with "address" and "powe" with "power".

Complete Analysis

ID A.11: **Status: Resolved**

Informational | Inconsistent In Contract Documentation

Present at: docs at the begging of the contract @L73

Description: The contract's documentation on the begging of it are causing confusing between the Share Owner and the Share Manager. At some places the docs are referring to the "Share Owner", The address who defined as a share owner on the HyperCycleShareContract as "Share Manager". This is inconsistent with other places where the docs refer to that owner as the "Share Owner". Also the contract itself called "ShareManager" so it is not clear when the docs creator is meaning to the contract it self or the "Share Owner"

Recommendation: For more consistent and understandable docs please refer to the Share Owner only as "Share Owner" as he defined in the HyperCycleShareTokens contract. In order to refer to the share manager contract refer to it as "Share Manager Contract"

ID A.12: **Status: Resolved**

High | Missing proposal mechanism for 'changePendingRevenueDelay'

Description: The 'changePendingRevenueDelay' function in the HyperCycleSharedTokens contract enables the share owner to modify the pending revenue delay. However, when the share owner role is assigned to the manager contract, there is no mechanism to propose changes to the pending revenue delay. This omission restricts the functionality from being utilized through the manager contract for managing share tokens.

Recommendation: Introduce a new proposal type that permits users to vote on and subsequently execute the 'changePendingRevenueDelay' function. This enhancement will ensure comprehensive use of all contract features, even when shares are managed via the manager contract.

Complete Analysis

ID A.13: **Status:** Acknowledged

High | Missing proposal mechanism for 'setShareMessage'

Description: The setShareMessage function within the HyperCycleSharedTokens contract is designed to allow the share owner to update the share message. However, similar to the previously noted issue, if the manager contract acts as the share owner, there's no provided method for initiating a proposal to update the share message. This gap effectively prevents the modification of share messages when shares are administered through the manager contract, limiting operational flexibility.

Recommendation: Implement an additional proposal mechanism that enables users to cast votes on changes to the share message, leveraging the setShareMessage function. By facilitating this through a voting process, the contract can maintain decentralized governance while expanding the functionalities accessible under management by the manager contract.

ID A.14: **Status:** Resolved

Medium | Absence of Validation Check

Present at: 'proposeNewHardwareOperatorRevenue', @L573

Description: The 'proposeNewManager' function lacks a verification step to ensure the proposed new share manager address differs from the existing one.

Recommendation: It is advisable to integrate a 'require' statement to confirm that the new manager's address is not identical to the current manager's address, thereby preventing redundant or unnecessary proposals.

Complete Analysis

ID A.15: **Status:** Acknowledged

High | Missing Validation Check On Share Proposals

Present at: 'proposeNewHardwareOperatorAddress', 'proposeNewManager', 'proposeNewManager', @L544, L606, L606

Description: The functions for proposing new hardware operators and new managers lack crucial validation checks. Specifically, they do not verify whether the 'ShareProposalStatus' for a given share is in the 'STARTED' state or confirm the existence of the share. This omission could lead to proposals being made inappropriately for shares that are either non-existent or not in a state that should allow proposal submission.

Recommendation: Consider adding validations to make sure that the status is STARTED and that the share indeed exists as indicated by the _shareTokenExists mapping.

ID A.16: **Status:** Resolved

Medium | Incomplete Implementation of 'getVotationConsensus' Function

Present at: 'getVotationConsensus' @L995

Description: The getVotationConsensus function currently only accounts for votation scenarios requiring a consensus of more than 50% of votes to pass. However, the governance mechanism includes four types of votations, some of which, like changing the manager, necessitate a 100% consensus. The function's current implementation does not reflect this diversity in consensus requirements.

Recommendation: Modify the getVotationConsensus function to dynamically return the required consensus percentage based on the specific votation type.

Complete Analysis

ID A.17: **Status: Partially Resolved**

Medium | Inefficient Management of Consensus Percentages

Description: The consensus percentages for various votation types are currently not centralized within the contract, leading to potential inconsistencies and duplication in the codebase. This scattered approach to defining consensus requirements can result in a higher risk of errors during updates or modifications and complicates the process of maintaining the contract over time.

Recommendation: Introduce a mapping that consolidates the required consensus percentages for each votation type. This mapping should serve as the single source of truth for determining the consensus needed for votations to pass.

Recommendation 2: the recommendation was to introduce a mapping containing percentage for each vote type for example:

```
mapping(Types.VotationOptions => uint) public voteTypeToPercentage;

constructor() {

    voteTypeToPercentage[Types.VotationOptions.CANCEL_SHARE] = 100;
    voteTypeToPercentage[Types.VotationOptions.CHANGE_HARDWARE_OPERATOR_ADDR
    ESS] = 50;
    voteTypeToPercentage[Types.VotationOptions.CHANGE_HARDWARE_OPERATOR_REVE
    NUE] = 50;
    voteTypeToPercentage[Types.VotationOptions.CHANGE_MANAGER_CONTRACT] =
    100;
    voteTypeToPercentage[Types.VotationOptions.CHANGE_DEPOSIT_REVENUE_DELAY]
    = 100;

    //rest of your code…
    }
}
```

Complete Analysis

ID A.18: **Status: Resolved**

Informational | Inaccurate Parameter Description in 'proposeNewHardwareOperatorRevenue' Function

Present at: 'proposeNewHardwareOperatorRevenue' @L572

Description: The documentation for the newRevenue parameter within the 'proposeNewHardwareOperatorRevenue' function incorrectly specifies that 'newRevenue' should be "greater or equal than 2**20 and lower or equal than 30". This description does not accurately reflect the implemented validation logic, which checks if newRevenue falls within a dynamic range based on the 'tokenLevel' of a share proposal rather than static numerical bounds.

Recommendation: Revise the '@param newRevenue' documentation to correctly describe the validation criteria. A more accurate description might be: "The proposed new revenue for the hardware operator must fall within a range dynamically calculated as a fraction of the share's token level, specifically between 10% and 30% of the value derived from 1 << tokenLevel." This adjustment will ensure the function's documentation accurately communicates the expectations and constraints for the newRevenue parameter.

ID A.19: **Status: Resolved**

Informational | Unnecessary Import of Ownable Contract

Present at: imports at the top of the contract @L8

Description: The contract imports 'Ownable' from OpenZeppelin's contracts library but does not utilize any of its functionality within the contract. This inclusion of an unnecessary import does not pose a direct risk to the contract's security or functionality but contributes to code clutter and can potentially lead to confusion about the contract's access control mechanisms.

Recommendation: Review the contract's access control requirements to determine if Ownable or its functionality is needed. If not, consider removing this import to streamline the contract code and reduce potential confusion.

Complete Analysis

ID A.20: **Status: Resolved**

Medium | Lack of Migration Reversion Mechanism

Present at: 'startShareProposalMigration', 'finishShareTokenMigration' @L332, L401

Description: The current implementation does not allow a user to revert or cancel a migration once it has been initiated but not completed. This lack of flexibility could lead to scenarios where users, due to changing circumstances or considerations, might want to cancel their migration process but are unable to do so. This could potentially result in undesired commitments or asset management issues.

Recommendation: Consider implementing a cancellation function that allows users to cancel their migration process before it is completed. This function should ensure that any assets transferred to the contract in anticipation of migration are safely returned to the user and that the migration process is cleanly aborted without leaving any residual state or commitments. Additionally, consider adding checks within the 'finishShareTokenMigration' function to prevent completion if the user has already canceled the migration.

Recommendation 2: the 'cancelShareTokenMigration' that was introduced is not returning the revenue tokens back to the shareOwner. Make sure the whole process that happened on startShareMigration was reversed.

Complete Analysis

ID A.21: **Status: Resolved**

High | Inaccurate Token Calculations Due to Burnable Tokens

Present at: 'startShareProposalMigration' @L332

Description: The 'startShareProposalMigration' function assumes that the total supply of wealth and revenue tokens directly correlates with the tokenLevel through the calculation (1 << tokenLevel) * 7 / 10 for initial revenue tokens and (1 << tokenLevel) for initial wealth tokens. This approach does not account for the possibility that tokens may have been burned prior to the migration. Since both wealth and revenue tokens are burnable, any tokens that have been burned reduce the actual total supply, potentially leading to inaccurate calculations within the migration logic. This discrepancy can affect the distribution of wealth and revenue tokens, the consensus mechanism for votations, and ultimately the fairness and functionality of the share proposal migration process.

Recommendation: Consider adding a mechanism to fetch the actual current total supply of wealth and revenue tokens.

ID A.22: **Status: Resolved**

High | Inaccurate Consensus Calculations Due to Burned Tokens

Present at: 'vote', 'getVotationConsensus' @L666, L995

Description: The contract uses initial token supply assumptions for consensus calculations in votations. However, it does not account for changes in supply due to tokens being burned.

Recommendation: Consider adding a mechanism to fetch the actual current total supply of wealth and revenue tokens.

Complete Analysis

ID A.23: **Status: Resolved**

High | Inaccurate Reward Distribution Due to Unadjusted Token Supply

Present at: 'claimChypcPortion', 'claimSurplus' @L437, L470

Description: While these functions may not directly use total supply for calculations, any logic that depends on the proportion of holdings for determining rewards could yield inaccurate distributions if it fails to consider current token supply. For 'claimSurplus' make sure the 'hypcSurplus' param is current (check '_sendRevenueToHardWareOperator' line 530).

Recommendation: Ensure reward distribution logic queries the current token supplies to accurately calculate payouts based on the actual proportion of holdings.

ID A.24: **Status: Resolved**

High | Inaccurate Revenue Proposal Limits Due to Static Token Supply Assumptions

Present at: 'proposeNewHardwareOperatorRevenue' @L606

Description: The function for proposing new hardware operator revenue uses static assumptions about token levels to set bounds on acceptable revenue proposals. This approach doesn't account for potential changes in the total supply of tokens due to burns, which could misalign the revenue proposal limits with the current economic reality of the token ecosystem.

Recommendation: Consider adding a mechanism to fetch the actual current total supply of wealth and revenue tokens.

Complete Analysis

ID A.25: **Status: Resolved**

High | Risk of Asset Lock due to 100% Consensus Requirement

Present at: 'vote' @L656

Description: The current voting mechanism requires 100% consensus for critical actions such as changing the manager contract or canceling a share. This stringent requirement poses a significant risk of assets being permanently locked in the ShareTokensV2 contract if even a single wealth token is lost or sent to an inaccessible address.

Recommendation: To mitigate this risk, consider implementing one or more of the following solutions:

1. Deadlock Resolution Mechanism: Introduce a special resolution process for situations where achieving 100% consensus is impossible due to lost or inaccessible tokens. This could involve a time-bound override by a multi-sig governance council or another form of decentralized decision-making authority.
2. Consensus Threshold Adjustment: Reevaluate the necessity of a 100% consensus requirement. Lowering the threshold to a supermajority (e.g., 90% or 95%) could reduce the risk of asset lock while still ensuring that any action has broad support among token holder
3. Emergency Escape Hatch: Provide an emergency mechanism, controlled by a decentralized governance process, that allows for the recovery or reallocation of assets in extreme scenarios. This should be designed with strict safeguards to prevent abuse.

Addressing this risk is crucial to ensure the long-term viability and user trust in the ShareTokensV2 and associated contracts.

Complete Analysis

ID A.26: **Status: Resolved**

Medium | Revenue Claim Limitation Due to Voting Power Allocation in 'claimChypcPortion'

Present at: 'claimChypcPortion' @L442

Description: Users must decrease their voting power by transferring wealth tokens back from the manager contract to claim their due revenue in the claimChypcPortion function. This process adds an unnecessary step for users wishing to claim revenue, potentially leading to confusion or reduced participation in votations due to the inconvenience of managing wealth tokens between voting and revenue claiming.

Recommendation: Simplify the process by allowing users to claim revenue without the need to adjust their voting power. This could involve automatically accounting for wealth tokens locked as voting power in the revenue claim calculation, ensuring users receive revenue proportionate to their total wealth token holdings without additional steps. Additionally, consider updating the user interface and documentation to clearly communicate how wealth tokens used as voting power are included in revenue calculations, enhancing transparency and user experience.

ID A.27: **Status: Acknowledged**

Medium | Misalignment in Total Supply Calculation for Revenue and Wealth Tokens

Present at: 'createShareProposal', '_completeProposal' @L164, L250

Description: In 'createShareProposal' you are using '(1 << chypcLevel)' to describe the total supply and in '_completeProposal' you are using the '(1 << licenseLevel)' to describe the total supply. There might be a mismatch between those numbers

Recommendation: Ensure the total supply for both revenue and wealth tokens is derived consistently across the contract. It is recommended to fetch the total supply directly from the ShareTokens contract to maintain consistency and accuracy.

Recommendation 2: why not fetching total supply directly from the ShareTokens contract?

Complete Analysis

ID A.28: **Status: Resolved**

Medium | Irreversible Voting Mechanism

Present at: 'vote' @L636

Description: Once a user increases their vote power and casts a vote, for instance, to cancel a share, they are unable to retrieve their wealth tokens until the share proposal reaches an ENDED status. This mechanism effectively locks users' wealth tokens indefinitely in scenarios where a proposal does not conclude, potentially due to a lack of consensus or other factors.

Recommendation: Implement a mechanism that allows users to retract their votes under certain conditions, enabling them to subsequently decrease their vote power and reclaim their wealth tokens. This change would enhance the flexibility of the voting process and prevent the indefinite locking of users' assets.

ID A.29: **Status: Resolved**

Medium | Absence of Validation for Matching CHyPC and License Levels

Present at: '_completeProposal' @L288

Description: There is no validation to ensure the CHyPC and license levels are matched. This could result sending transaction with 0 as amount on '_completeProposal'

Recommendation: Consider adding a require to make sure there is no sending of 0 tokens. Also make sure they are on the same level with a validation or describe in the docs how it would not happen.

Complete Analysis

ID A.30: **Status: Resolved**

Informational | Simplify Process for Holders of Both cHyPC and License

Description: When the holder of both a cHyPC and a License intends to create share tokens, they are required to execute 'createShareProposal' followed by 'completeShareProposal', which can be redundant since the same user possesses both tokens. This two-step process could be streamlined to enhance user experience and efficiency.

Recommendation: Consider implementing a wrapper function that combines 'createShareProposal' and 'completeShareProposal' into a single transaction for users who own both a cHyPC and a License, simplifying the process and improving usability.

ID A.31: **Status: Acknowledged**

Informational | Lack of Support for Creating Share Proposals with Assigned Licenses

Present at: '_completeProposal' @L278

Description: The process of creating share tokens, as triggered by 'createShareTokens' in the ShareTokens contract, assumes the presence of a cHyPC token by passing 'true' for the 'chypcTokenHeld' parameter. This approach does not accommodate scenarios where a license is already assigned without a corresponding cHyPC token, potentially limiting the flexibility and utility of share proposals.

Recommendation: consider implementing a mechanism or function that allows for the initiation of share proposals even when the license is already assigned.

Complete Analysis

ID A.32: **Status: Resolved**

Low | Absence of Validation for Matching CHyPC and License Levels

Present at: 'startShareProposalMigration' @L322

Description: The token being used for total supply in that function is the chypc token level. There is a missing validation to make sure it is the same as the license level.

Recommendation: Consider adding a validation to make sure they are on the same level or get the total supply from the ShareTokens contract.

ID A.33: **Status: Resolved**

Low | Potential Misalignment of Assigned Operator Message during Migration

Present at: 'startShareProposalMigration' @L380

Description: The migration process involves setting the 'operatorAssignedString' based on the current 'getShareMessage' output from the ShareTokens contract. This approach does not guarantee that the retrieved message accurately reflects the intended operator information, as it may contain outdated or irrelevant content (when initializing the shareToken the message is defined as startingMessage and there is no guarantee it was started with the hardwareOperator, but on migration its value is ingested expecting as such).

Recommendation: It is advisable to verify the relevancy and accuracy of the 'operatorAssignedString' during migration, ensuring it pertains to the current hardware operator. Alternatively, initializing the operator message with a standard default value could be a good solution.

Complete Analysis

ID A.34: **Status: Resolved**

Low | Simplifying HYPC Token Claims with Voting Power

Present at: 'claimChypcPortion' @L438

Description: Currently, voters must undertake a two-step process to claim their HYPC tokens from the ShareManager contract: first, decrease their voting power to retrieve their wealth tokens, and then execute 'claimChypcPortion' to return these tokens to the contract for claiming their HYPC portion. This process could be streamlined to enhance user convenience.

Recommendation: It may be beneficial to integrate the calculation of a user's voting power with their wealth token balance, enabling HYPC token claims in a single transaction. This approach would eliminate the need for voters to separately reduce their voting power, simplifying the claim process.

ID A.35: **Status: Resolved**

Informational | Misleading Function Naming

Present at: 'claimChypcPortion' @L438

Description: The function name 'claimChypcPortion' suggests it is related to claiming portions of CHyPC tokens, which might be misleading. Given the functionality focuses on claiming HYPC tokens

Recommendation: Consider changing function name to 'claimHypcPortion'

Complete Analysis

ID A.36: **Status:** Acknowledged

Medium | Potential Override of Existing License Assignments

Present at: '_completeProposal' @L275

Description: When creating a share proposal by calling the ShareTokens contract's 'createShareTokens' function, if the cHyPC already has an assigned license, the createShareTokens might unintentionally override that assignment. There is no validation to check if an assignment already exists.

Recommendation: Consider adding a validation step to ensure that no existing assignment is overridden, maintaining the integrity of license assignments within the system.

ID A.37: **Status:** Acknowledged

Low | Potential Front Running of Assignment

Present at: 'createShareProposal' @L145

Description: There is no mechanism for the cHyPC owner to specify the specific license they wish to use to complete the proposal. This lack of specificity could lead to issues if someone front-runs the '_completeProposal' transaction in cases where the proposal was not intended for them.

Recommendation: Consider adding a parameter to 'createShareProposal' allowing the cHyPC owner to specify the exact license or the specific address authorized to complete the proposal, thereby reducing the risk of front-running.

Complete Analysis

ID A.38: **Status: Resolved**

Low | Potential Token Rounding Issue in claimSurplus Function

Present at: 'claimSurplus' @L527

Description: The claimSurplus function evenly splits the surplus tokens between the cHyPC owner and the license owner. However, due to the integer division used, when the surplus amount is an odd number, it may result in a rounding issue

Recommendation: Consider first transferring half of the surplus amount to the cHyPC owner, and then transferring the remaining balance to the license owner. This approach ensures no surplus tokens are left in the contract.

ID A.39: **Status: Resolved**

Informational | Typographical Error

Present at: 'ONE_HUNDRED_PORCENT' , 'SELECTED_VOTATION_PORCENT' @L89

Description: typographical error in the word "porcent"

Recommendation: Replace "PORCENT" with "PERCENT".

Disclaimer:

DcentraLab Diligence (DD) has provided the code to the client as is and assumes no responsibility nor legal liability for any use client may do with the code. Any and all usage and/or deployment of the code provided by DcentraLab Diligence will be done solely by the client, at the sole discretion, responsibility, risk, and legal liability of the Client, and DD will not be held accountable or liable for any loss of funds, security exploits or incidents, or any other unintended or negative outcome that may occur in relation to the code provided by DD.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 DD to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This report and the provided code or services as part of the SOW pertaining to this report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should it be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. DD's position is that each company and individual are responsible for their own due diligence and continuous security. DD's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by DD are subject to dependencies and are under continuing development. You agree that your access and/or use, including but not limited to any services, code, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, DcentraLab Diligence (DD) HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, DD SPECIFICALLY DISCLAIMS

ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, DD MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT / VERIFICATION REPORT, WORK PRODUCT, CODE OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

WITHOUT LIMITATION TO THE DISCLAIMER HyperCycle Contracts FOREGOING, DD PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET THE CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR-FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER DD NOR ANY OF DD'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION, CODE OR CONTENT PROVIDED THROUGH THE SERVICE. DD WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT OR CODE, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, CODE, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS," AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN THE CUSTOMER AND THE THIRD-PARTY
OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS. THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO THE CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT DD'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST DD WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS. THE REPRESENTATIONS AND WARRANTIES OF DD CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF THE CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST DD WITH RESPECT TO SUCH
REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.
FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS, CODE, OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL,
REGULATORY, OR OTHER ADVICE.

dcentralab.com/diligence

DcentraLab Diligence

Provided By DcentraLab Diligence on March 20, 2024